# BAB 2
# LANDASAN TEORI

## 2.1 Definisi jamur

Jamur adalah tubuh buah yang tampak di permukaan media tumbuh dari sekelompok fungi (Basidiomycota) yang berbentuk seperti payung: terdiri dari bagian yang tegak ("batang") dan bagian yang mendatar atau membulat. Secara teknis biologis, tubuh buah ini disebut basidium. Beberapa jamur aman dimakan manusia bahkan beberapa dianggap berkhasiat obat, dan beberapa yang lain beracun. Contoh jamur yang bisa dimakan: jamur merang (Volvariela volvacea), jamur tiram (Pleurotus), jamur kuping (Auricularia polytricha), jamur kancing atau champignon (Agaricus campestris), dan jamur shiitake (Lentinus edulis). [7]




Gambar 2.1 Jenis – jenis jamur

Berdasarkan jenis jamur tersebut , penulis lebih tertarik membahas jenis jamur tiram, karena paling banyak ditemui di daerah cisarua bandung

### 2.1.1 Jamur Tiram

Jamur tiram (*Pleurotus Ostreatus*) adalah jamur pangan dari kelompok basidiomycota dan termasuk kelas homobasidiomycetes dengan ciri-ciri umum tubuh buah berwarna putih hingga krem dan tudungnya berbentuk setengah lingkaran mirip cangkang tiram dengan bagian tengah agak cekung. Jamur tiram masih satu kerabat dengan pleurotus eryngii dan sering dikenal dengan sebutan *King Oyster Mushroom*.

Jamur tiram yang biasa dijadikan sebagai bahan makanan yang sarat nutrisi ini, selain mengandung protein, vitamin dan mineral yang cukup tinggi ternyata juga memiliki kandungan jenis asam amino esensial yang lengkap. Terdapat sembilan jenis asam amino yang diketahui terkandung pada jamur tiram, yaitu lysine, methionin, trypthofan, theonin, valin, leusin, isoleusin, histidin dan fenilalanin. Keunggulan dari jamur tiram adalah tergolong ke dalam makanan atau penganan yang sehat karena bebas dari pestisida, pupuk dan bahan-bahan kimia lainnya.

Ciri –ciri jamur yang siap panen memiliki pertumbuhan yang optimal, yakni ukurannya cukup besar, tetapi tudungnya belum mekar penuh (ditandai pada bagian pinggir tudung jamur masih terlihat utuh/belum pecah-pecah). Ukuran diameter jamur yang siap dipanen rata-rata mencapai 5-10 cm.[5]





a. Jamur siap panen b. Jamur belum siap panen

Gambar 2.2 Jamur siap panen dan belum siap panen

## 2.2 Kecerdasan Buatan (Artificial Intelligence)

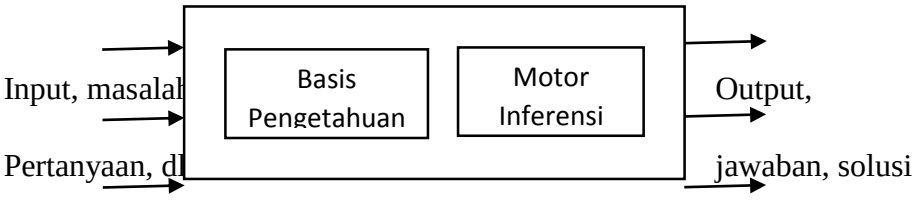
Kecerdasan buatan (Artificial Intelligence) adalah bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia bahkan bisa lebih baik daripada yang dilakukan manusia. [10]

Menurut John McCarthy, 1956, AI :

"Untuk mengetahui dan memodelkan proses – proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia."

Cerdas adalah memiliki pengetahuan dan pengalaman, penalaran yaitu bagaimana membuat keputusan dan mengambil tindakan, serta moral yang baik. Agar mesin bisa cerdas atau bertindak seperti dan sebaik manusia, maka harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Penerapan kecerdasan buatan digambarkan pada Gambar 2.3. Dua bagian utama yg dibutuhkan untuk aplikasi kecerdasan buatan adalah [10]:

a. Basis pengetahuan (knowledge base): berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
b. Motor inferensi (inference engine) : kemampuan menarik kesimpulan berdasarkan pengetahuan.




Gambar 2.3 Penerapan Kecerdasan Buatan[1]

### 2.2.1 Perbedaan Kecerdasan Buatan dan Kecerdasan Alami

Berikut ini merupakan Perbedaan antara kecerdasan buatan dan kecerdasan alami :

1. Kelebihan kecerdasan buatan
    a. Lebih bersifat permanen. Kecerdasan alami bisa berubah karena sifat manusia pelupa. Kecerdasan buatan tidak berubah selama sistem komputer dan program tidak mengubahnya.
    b. Lebih mudah diduplikasi dan disebarkan. Memindahkan pengetahuan manusia dari 1 orang ke orang lain membutuhkan proses yang sangat lama dan keahlian tidak akan pernah dapat diduplikasi dengan lengkap.Jadi jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebut dapat disalin dari komputer tersebut dan dapat dipindahkan dengan mudah ke komputer yang lain.
    c. Lebih murah. Menyediakan layanan komputer akan lebih mudah dan murah dibandingkan mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
    d. Bersifat konsisten dan teliti karena kecerdasan buatan adalah bagian dari teknologi komputer sedangkan kecerdasan alami senantiasa berubah-ubah.
    e. Dapat didokumentasikan. Keputusan yang dibuat komputer dapat didokumentasikan dengan mudah dengan cara melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
    f. Dapat mengerjakan beberapa tugas dengan lebih cepat dan lebih baik dibanding manusia.
2. Kelebihan kecerdasan alami
    a. Kreatif : manusia memiliki kemampuan untuk menambah pengetahuan, sedangkan pada kecerdasan buatan untuk menambah pengetahuan harus dilakukan melalui sistem yang dibangun.

b. Memungkinkan orang untuk menggunakan pengalaman atau pembelajaran secara langsung. Sedangkan pada kecerdasan buatan harus mendapat masukan berupa data-data simbolik.
c. Pemikiran manusia dapat digunakan secara luas, sedangkan kecerdasan buatan sangat terbatas.

## 2.3 Jaringan Syaraf Tiruan (Artificial Neural Network)

Jaringan syaraf merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran.

### 2.3.1 Penerapan Kecerdasan Buatan

Kecerdasan buatan banyak diterapkan pada berbagai bidang, misalnya pada bidang ilmu komputer (informatika), teknik elektro, kesehatan, manajemen dan lain sebagainya. Beberapa contoh penerapan kecerdasan buatan adalah sebagai berikut:

1. **Pengenalan Pola**

   Pengenalan pola dilakukan untuk mengenali suatu objek tertentu, misalnya untuk keperluan absensi menggunakan pengenalan sidik jari (finger recognition), pengenalan suara (voice recognition) atau pengenalan wajah (face recognition). Selain itu juga dapat digunakan sebagai mesin pencarian dengan gambar atau suara sebagai kata kunci(keyword)nya dengan menggunakan pengenalan gambar (image recognition) dan pengenalan ucapan (speech recognition).[9]
2. **Sistem Pakar**

   Sistem pakar digunakan sebagai sarana untuk menyimpan pengetahuan para pakar, sehingga komputer dapat menyelesaikan permasalahan dengan

meniru keahlian yang dimiliki oleh pakar. Misalnya analisis penyakit, kerusakan komputer, penasihat keuangan dan lain sebagainya.

3. **Game Playing**

   Game merupakan fasilitas menarik yang terdapat dalam komputer. Game dalam komputer seperti catur, kartu, dan lainnya menggunakan kecerdasan buatan agar dapat bermain selayaknya manusia, memiliki strategi langkah selanjutnya, pengetahuan peraturan permainan dan dapat mengetahui pemenangnya.

4. **Robotika dan Sistem Sensor**

   Salah satu contoh sistem sensor adalah pada mesin cuci yaitu menggunakan sensor optik, mengeluarkan cahaya ke air dan mengukur bagaimana cahaya tersebut sampai ke ujung lainnya. Makin kotor, maka sinar yang sampai makin redup. Sistem juga mampu menentukan jenis kotoran tersebut daki/minyak. Sistem juga bisa menentukan putaran yang tepat secara otomatis berdasarkan jenis dan banyaknya kotoran serta jumlah yang akan dicuci.

## 2.4 Jaringan Syaraf Tiruan ( JST )

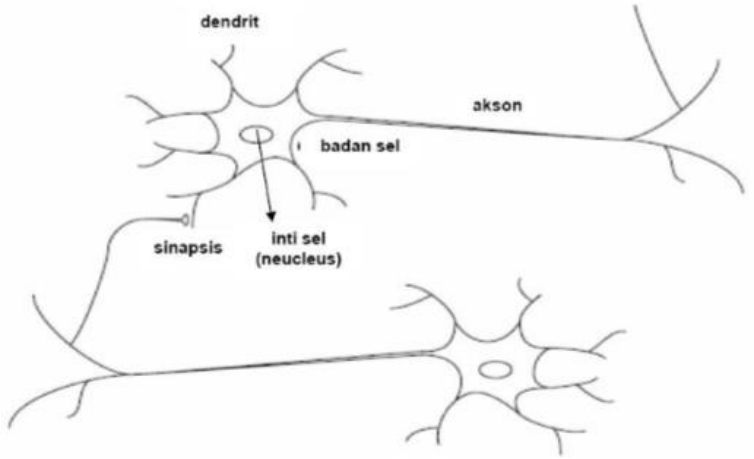
Jaringan syaraf adalah merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran[10].

### 2.4.1 Jaringan Syaraf Biologi dan Tiruan

Jaringan syaraf biologi merupakan jaringan syaraf yang terdapat pada manusia, sedangkan jaringan syaraf tiruan merupakan jaringan syaraf yang dibuat dan dijalankan pada komputer.

## A. Jaringan Syaraf Biologi (JSB)

Otak manusia memiliki struktur yang sangat kompleks dan memiliki kemampuan yang luar biasa. Otak terdiri dari neuron-neuron dan penghubung yang disebut sinapsis. Neuron bekerja berdasarkan impuls/sinyal yang diberikan pada neuron. Neuron meneruskannya pada neuron lain. Diperkirakan manusia memiliki $10^{12}$ neuron dan $6.10^{8}$ sinapsis. Dengan jumlah yang begitu banyak, otak mampu mengenali pola, melakukan perhitungan, dan mengontrol organ-organ dengan kecepatan yang lebih tinggi dibandingkan komputer digital. Sebagai perbandingan, pengenalan wajah seseorang yang sedikit berubah (misal memakai topi, memiliki jenggot tambahan, dan lain-lain) akan lebih cepat dilakukan manusia dibandingkan komputer. Gambar neuron seperti dilihat pada Gambar 2.3




Gambar 2.4 Neuron [11]

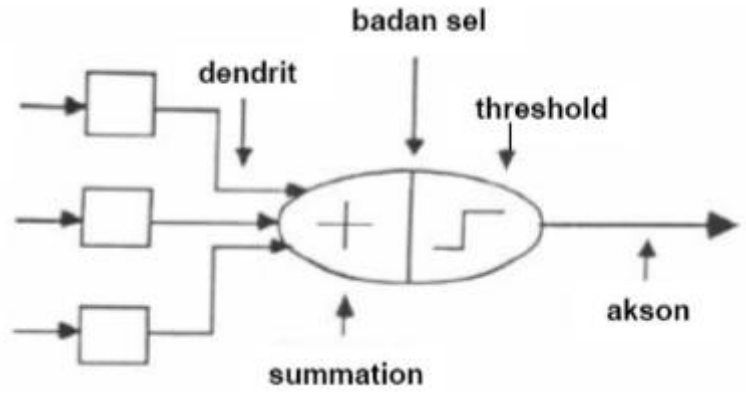
Komponen utama neuron dapat dikelompokkan menjadi 3 bagian, yaitu:

1. Dendrit = bertugas menerima informasi = jalur masukan bagi soma
2. Badan sel (soma) = tempat pengolahan informasi
3. Akson = bertugas mengirimkan impuls-impuls sinyal ke sel syaraf lain = jalur keluaran bagi soma.

Dari Gambar 2.4 didapat kesimpulan sebagai berikut:

1. Sebuah neuron menerima impuls-impuls sinyal (informasi) dari neuron lain melalui dendrit dan mengirimkan sinyal yang dibangkitkan (hasil penjumlahan) oleh badan sel melalui akson.
2. Akson dari sel syaraf ini bercabang-cabang dan berhubungan dengan dendrit dari sel syaraf lain dengan cara mengirimkan impuls melalui sinapsis.
3. Sinapsis adalah unit fungsional antara 2 buah sel syaraf, misal A dan B, dimana yang satu adalah serabut akson dari neuron A dan satunya lagi adalah dendrit dari neuron B.Kekuatan sinapsis bisa menurun / meningkat tergantung seberapa besar tingkat propagasi (penyiaran) sinyal yang diterimanya.
4. Kekuatan sinapsis bisa menurun / meningkat tergantung seberapa besar tingkat propagasi (penyiaran) sinyal yang diterimanya.
5. Impuls-impuls sinyal (informasi) akan diterima oleh neuron lain jika memenuhi batasan tertentu, yang sering disebut dengan nilai ambang (threshold).

Model struktur neuron jaringan syaraf biologi ditunjukkan pada Gambar 2.5.




Gambar 2.5 Model Struktur Neuron JSB[11]

## B. Jaringan Syaraf Tiruan (JST)

Jaringan Syaraf Tiruan (JST) merupakan suatu sistem pemrosesan informasi yang mempunyai karakteristik menyerupai jaringan syaraf biologi (JSB). JST tercipta sebagai suatu generalisasi model matematis dari pemahaman manusia (human cognition) yang didasarkan atas asumsi sebagai berikut :

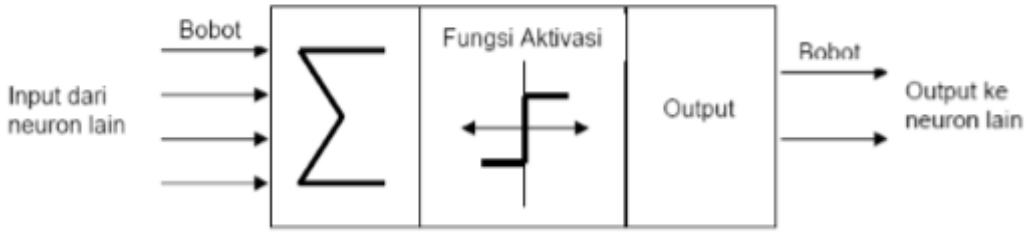
1. Pemrosesan informasi terjadi pada elemen sederhana yang disebut neuron
2. Sinyal mengalir diantara sel saraf/neuron melalui suatu sambungan penghubung
3. Setiap sambungan penghubung memiliki bobot yang bersesuaian. Bobot ini akan digunakan untuk menggandakan / mengalikan sinyal yang dikirim melaluinya.
4. Setiap sel syaraf akan menerapkan fungsi aktivasi terhadap sinyal hasil penjumlahan yang masuk kepadanya untuk menentukan sinyal keluarannya.

Analogi jaringan syaraf tiruan dengan jaringan syaraf buatan dijelaskan pada Tabel 2.1 berikut ini:

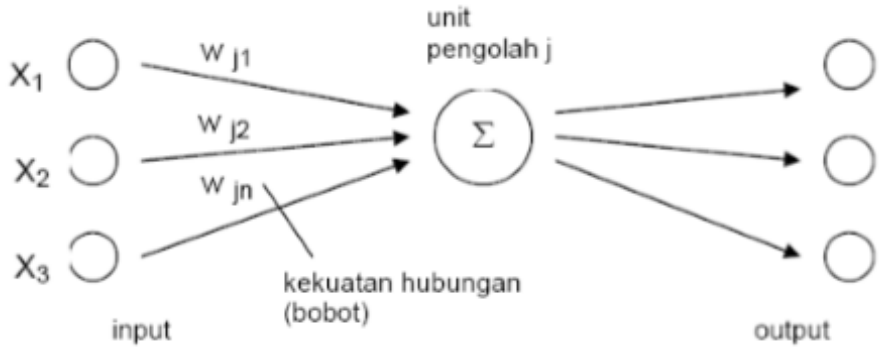
Tabel 2.1 Analogi JST dengan JSB

| JST | JSB |
|---|---|
| Node/masukan | Badan sel (soma) |
| Masukan | Dendrit |
| Keluaran | Akson |
| Bobot | Sinapsis |

Model struktur neuron jaringan syaraf tiruan dijelaskan pada Gambar 2.6 dan Gambar 2.7.




Gambar 2.6 Model Struktur JST[11]




Gambar 2.7 Model Struktur JST[11]

Jaringan syaraf tiruan dapat belajar dari pengalaman, melakukan generalisasi atas contoh-contoh yang diperolehnya dan mengabstraksi karakteristik esensial masukan bahkan untuk data yang tidak relevan. Algoritma untuk JST beroperasi secara langsung dengan angka sehingga data yang tidak numerik harus diubah menjadi data numerik. JST tidak diprogram untuk menghasilkan keluaran tertentu. Semua keluaran atau kesimpulan yang ditarik oleh jaringan didasarkan pada pengalamannya selama mengikuti proses pembelajaran. Pada proses pembelajaran, ke dalam JST dimasukkan pola-pola masukan (dan keluaran) lalu jaringan akan diajari untuk memberikan jawaban yang bisa diterima. Pada dasarnya karakteristik JST ditentukan oleh :

1. Pola hubungan antar neuron (disebut arsitektur jaringan)

2. Metode penentuan bobot-bobot sambungan (disebut dengan pelatihan atau proses belajar jaringan)
3. Fungsi aktivasi

### 2.4.2 Arsitektur JST

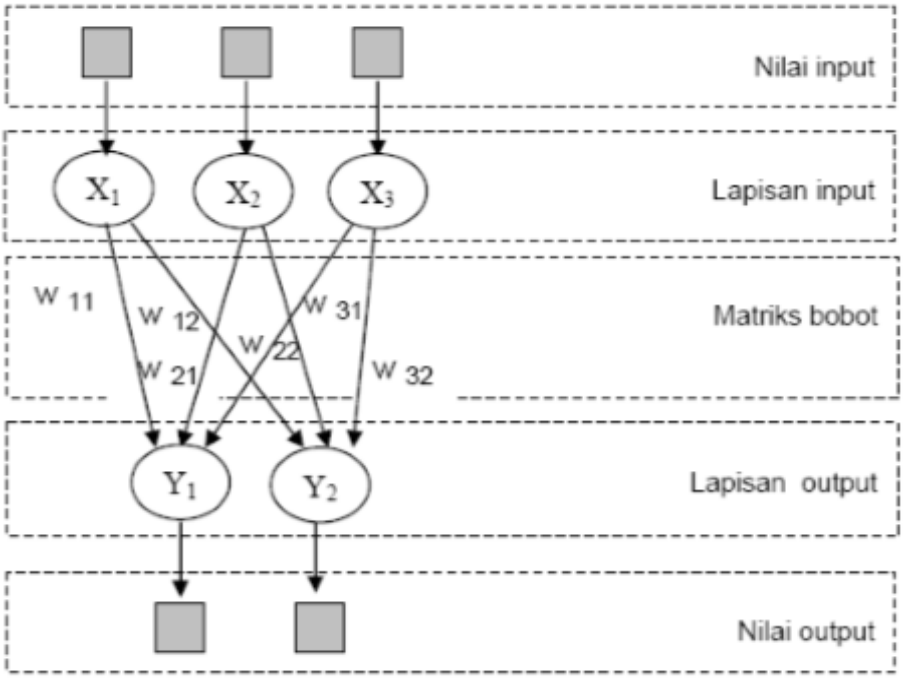
Pada JST, neuron-neuron akan dikumpulkan dalam lapisan-lapisan (layer) yang disebut dengan lapisan neuron (neuron layers). Neuron-neuron pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya. Informasi yang diberikan pada jaringan syaraf akan dirambatkan lapisan ke lapisan, mulai dari lapisan masukan sampai ke lapisan keluaran melalui lapisan tersembunyi (hidden layer). Gambar berikut ini jaringan syaraf dengan 3 lapisan dan bukanlah struktur umum jaringan syaraf karena beberapa jaringan syaraf ada yang tidak memiliki lapisan tersembunyi.

Faktor terpenting dalam menentukan kelakuan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Umumnya neuron-neuron yang terletak pada lapisan yang sama akan memiliki keadaan yang sama sehingga pada setiap lapisan yang sama neuron-neuron memiliki fungsi aktivasi yang sama. Bila neuron-neuron pada suatu lapisan (misal lapisan tersembunyi) akan dihubungkan dengan neuron-neuron pada lapisan lain (misal lapisan keluaran) maka setiap neuron pada lapisan tersebut (lapisan tersembunyi) juga harus dihubungkan dengan setiap neuron pada lapisan lainnya (lapisan keluaran). Terdapat 3 macam arsitektur JST, yaitu:

1. Jaringan dengan lapisan tunggal (single layer net)

   Jaringan ini hanya memiliki 1 lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima masukan kemudian secara langsung akan mengolahnya menjadi keluaran tanpa harus melalui lapisan tersembunyi. Pada gambar berikut neuron-neuron pada kedua lapisan saling berhubungan. Seberapa besar hubungan antara 2 neuron ditentukan oleh

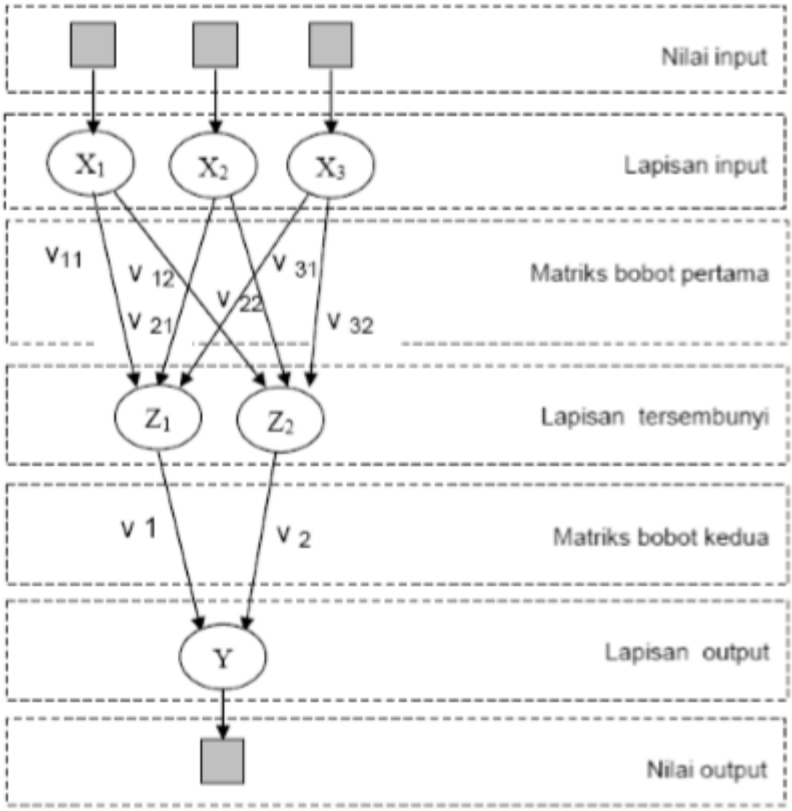
bobot yang bersesuaian. Semua unit masukan akan dihubungkan dengan setiap unit keluaran seperti terlihat pada Gambar 2.8.




Gambar 2.8 Jaringan Dengan Lapisan Tunggal [11]

2. Jaringan dengan banyak lapisan (multilayer net)

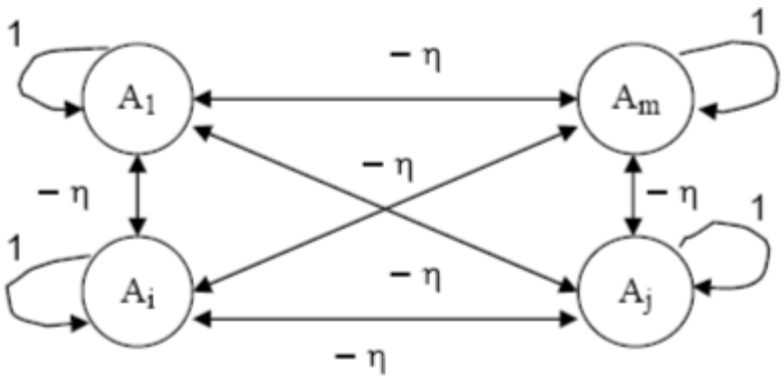
   aringan ini memiliki 1 atau lebih lapisan yang terletak diantara lapisan masukan dan lapisan keluaran. Umumnya ada lapisan bobot-bobot yang terletak antara 2 lapisan yang bersebelahan seperti terlihat pada Gambar 2.9. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan tunggal, tentu saja dengan pembelajaran yang lebih rumit. Pada banyak kasus, pembelajaran pada jaringan dengan banyak lapisan ini lebih sukses dalam menyelesaikan masalah.




Gambar 2.9 Jaringan Dengan Banyak Lapisan[11]

3. Jaringan dengan lapisan kompetitif (competitive layer net)

   Pada jaringan ini sekumpulan neuron bersaing untuk mendapatkan hak menjadi aktif. Umumnya hubungan antar neuron pada lapisan kompetitif ini tidak diperlihatkan pada diagram arsitektur. Gambar 2.9 menunjukkan salah satu contoh arsitektur jaringan dengan lapisan kompetitif yang memiliki bobot $-\eta$.




Gambar 2.10 Jaringan dengan lapisan kompetitif[11]

### 2.4.3 Proses Pembelajaran Jaringan

Cara belajar JST adalah sebagai berikut:

a. Ke dalam JST dimasukkan informasi yang sebelumnya telah diketahui hasil keluarannya.
b. Proses memasukkan informasi ini dilakukan lewat node-node atau unit-unit masukan. Bobot-bobot antarkoneksi dalam suatu arsitektur diberi nilai awal dan kemudian JST dijalankan. Bobot-bobot ini bagi jaringan digunakan untuk belajar dan mengingat suatu informasi. Pengaturan bobot dilakukan secara terus-menerus dan dengan menggunakan kriteria tertentu sampai diperoleh keluaran yang diharapkan.

Hal yang ingin dicapai dengan melatih/mengajari JST adalah untuk mencapai keseimbangan antara kemampuan memorisasi dan generalisasi. Kemampuan memorisasi adalah kemampuan JST untuk memanggil kembali secara sempurna sebuah pola yang telah dipelajari. Kemampuan generalisasi adalah kemampuan JST untuk menghasilkan respon yang bisa diterima terhadap pola-pola masukan yang serupa (namun tidak identik) dengan pola-pola yang sebelumnya telah dipelajari.

Hal ini sangat bermanfaat bila pada suatu saat ke dalam JST dimasukkan informasi baru yang belum pernah dipelajari, maka JST masih akan tetap dapat memberikan tanggapan yang baik, memberikan keluaran yang paling mendekati. Paradigma/metode pembelajaran/pelatihan JST adalah sebagai berikut:

a. Pembelajaran terawasi (supervised learning)
   Pada pembelajaran ini kumpulan data masukan yang digunakan, data keluarannya telah diketahui. Perbedaan antara keluaran-keluaran aktual dengan data keluaran yang diinginkan digunakan untuk mengoreksi bobot JST agar JST dapat menghasilkan jawaban sedekat (semirip) mungkin dengan jawaban yang benar yang telah diketahui oleh JST.
b. Pembelajaran tak terawasi (unsupervised learning)
   Pada pembelajaran ini, JST mengorganisasi dirinya sendiri untuk membentuk vektor-vektor masukan yang serupa, tanpa menggunakan data atau contoh-contoh pelatihan. Struktur menggunakan dasar data atau korelasi antara pola-pola data yang dieksplorasi. Paradigma pembelajaran ini mengorganisasi pola-pola ke dalam kategori-kategori berdasarkan korelasi yang ada.
c. Gabungan pembelajaran terawasi dan tak terawasi (hybrid)
   Merupakan kombinasi dari kedua pembelajaran tersebut. Sebagian dari bobot-bobotnya ditentukan melalui pembelajaran terawasi dan sebagian lainnya melalui pembelajaran tak terawasi.

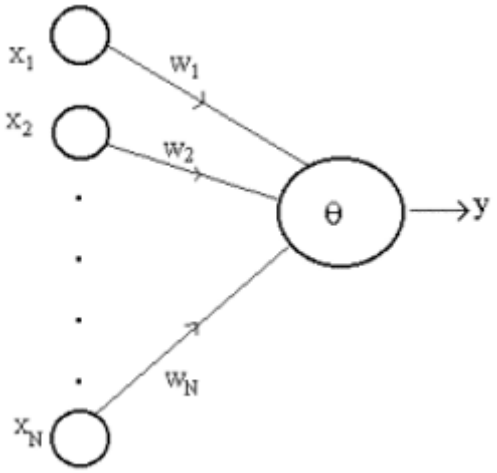
### 2.4.3.1 Pembelajaran Terawasi (Supervised Learning)

Metode pembelajaran pada jaringan syaraf disebut terawasi jika keluaran yang diharapkan telah diketahui sebelumnya, misalkan seperti pada proses AND. Pada proses pembelajaran, satu pola masukan akan diberikan ke satu neuron pada lapisan masukan . Pola ini akan dirambatkan di sepanjang jaringan syaraf hingga sampai ke neuron pada lapisan keluaran. Lapisan keluaran ini akan membangkitkan pola keluaran

yang nantinya akan dicocokkan dengan pola keluaran targetnya. Apabila terjadi perbedaan antara pola keluaran hasil pembelajaran dengan pola target, maka akan muncul error. Apabila nilai error masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi. Metode-metode yang termasuk dalam pembelajaran terawasi antara lain[6]:

**A. Hebb Rule**

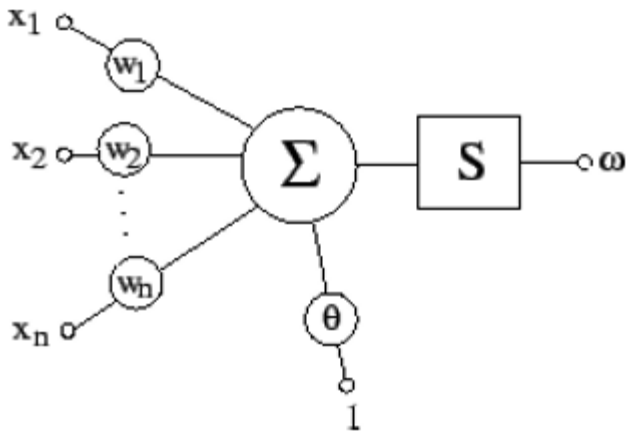
Hebb rule adalah metode pembelajaran yang paling sederhana. Pada metode ini pembelajaran dilakukan dengan cara memperbaiki nilai bobot sedemikian rupa sehingga jika ada 2 neuron yang terhubung, dan keduanya pada kondisi „hidup" (on) pada saat yang sama, maka bobot antara keduanya dinaikkan. Gambar 2.11 merupakan gambar jaringan hebb rule.




Gambar 2.11 Hebb Rule[12]

**B. Perceptron**

Jaringan jenis ini hanya terdiri dari layer masukan dan layer keluaran saja. Data masukan yang masuk melalui vector masukan akan langsung diproses dan kemudian ditentukan target keluaran dari hasil pengolahan. Jaringan perceptron dapat dilihat pada Gambar 2.12.

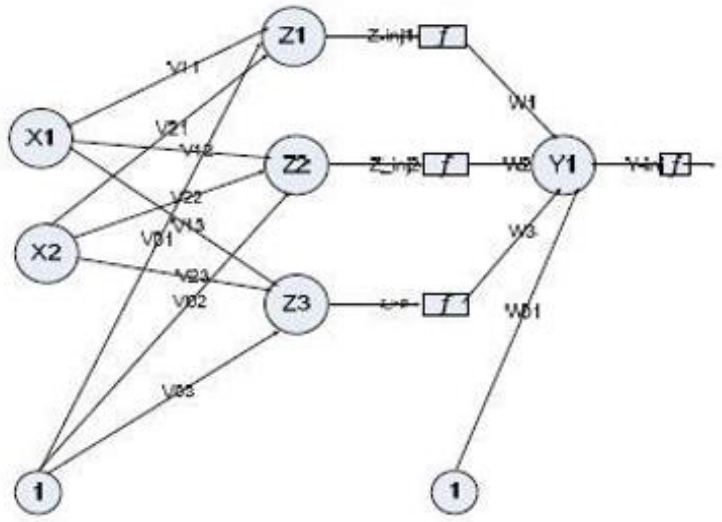



Gambar 2.12 Perceptron [13]

**C. Backpropagation**

Salah satu arsitektur jaringan saraf tiruan yang sangat populer adalah multilayer feedforward networks. Secara umum, jaringan seperti ini terdiri dari sejumlah unit neuron sebagai lapisan masukan, satu atau lebih lapisan simpul-simpul neuron komputasi lapisan tersembunyi, dan sebuah lapisan simpul-simpul neuron komputasi keluaran. Sinyal masukan dipropagasikan ke arah depan (arah lapisan keluaran), 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 pembelajaran terawasi. Propagasi balik (ke arah lapisan masukan) terjadi setelah jaringan menghasilkan keluaran yang mengandung error. Pada fase ini seluruh bobot synaptic (yang tidak memiliki aktivasi nol) dalam jaringan akan disesuaikan untuk mengkoreksi/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 lewatan untuk seluruh data latihan dalam sebuah proses pelatihan jaringan) sampai error yang terjadi mencapai batas

kecil toleransi tertentu atau nol. Jaringan backpropagation dapat dilihat pada Gambar 2.13.[18]




Gambar 2.13 Backpropagation[14]

Dari Gambar 2.12 dapat dilihat bahwa yang bertindak sebagai dendrit adalah X1 dan X2, yaitu data masukan pada jaringan tersebut. Terdapat 2 sinapsis atau bobot yaitu V dan W, sedangkan Z dan Y merupakan bagian dari soma atau badal sel dari jaringan tersebut. Dan yang bertindak sebagai akson atau data keluaran adalah Y. Berikut ini algoritma pembelajaran metode backpropagation.

Langkah 0 : Inisialisasi bobot. (sebaiknya diatur pada nilai acak yang kecil),

Langkah 1 : Jika kondisi berhenti tidak tercapai, lakukan langkah 2-9, (kondisi berhenti jika perulangan mencapai maksimal perulangan atau mean square error (MSE) sudah lebih kecil dari rasio pembelajaran).

Langkah 2 : Untuk setiap pasangan pelatihan, lakukan langkah 3-8,

**Perambatan Maju :**

Langkah 3 : Tiap unit masukan (xi, i = 1,…, n) menerima sinyal xi dan menghantarkan sinyal ini ke semua unit lapisan di atasnya (unit tersembunyi),

$$z_in_j = v_{oj} + \sum_{i=1}^{n} x_i v_{ij} \quad \text{……(2.1)}$$

Langkah 4 : Setiap unit tersembunyi (xi, i = 1,…, p) jumlahkan bobot

sinyal masukannya, $v_{oj}$ = bias pada unit tersembunyi j aplikasikan fungsi aktivasinya untuk menghitung sinyal keluarannya, $zj = f\ (z - in_j$ , dan kirimkan sinyal ini keseluruh unit pada lapisan diatasnya (unit keluaran).

Langkah 5: Tiap unit keluaran (yk, k = 1,…, m) jumlahkan bobot sinyal masukannya,

$$y_in_k = w_{ok} + \sum_{j=1}^{n} z_j w_{jk} \quad \text{……(2.2)}$$

$w_{ok}$ = bias pada unit keluaran k dan aplikasikan fungsi aktivasinya untuk menghitung sinyal keluarannya, yk = f (y_ink).

**Perambatan Mundur :**

Langkah 6 : Tiap unit keluaran (yk , k = 1,…, m) menerima pola target

yang saling berhubungan pada masukan pola pelatihan, hitung kesalahan informasinya,

$$\delta_k = (t_k - y_k) f'(y_in_k) \quad \text{……(2.3)}$$

hitung koreksi bobotnya (digunakan untuk memperbaharui wjk nantinya),

$\Delta w_{jk} = \alpha\, \delta_k z_j$ …………………………………………… (2.4)

hitung koreksi biasnya (digunakan untuk memperbaharui wok nantinya), dan kirimkan δk ke unit-unit pada lapisan dibawahnya,

Langkah 7 : Setiap unit lapisan tersembunyi (zj, j = 1,…, p) jumlahkan hasil perubahan masukannya (dari unit-unit lapisan diatasnya),

$\Delta_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$ ……………………………………………(2.5)

kalikan dengan turunan fungsi aktivasinya untuk menghitung informasi kesalahannya,

$\delta_j = \delta_in_j\, f^1 (z_in_j)$ …………………………………………(2.6)

hitung koreksi bobotnya (digunakan untuk memperbaharui voj nanti),

Langkah 8 : Tiap unit keluaran (yk, k = 1,…, m) update bias dan bobotnya (j = 0,…, p) :

$w_{jk}$ (baru) = $w_{jk}$ (lama) + $\Delta\, w_{jk}$ ………………………………(2.7)

Tiap unit lapisan tersembunyi (zj, j = 1,…, p) update bias dan bobotnya (I = 0,…,n) :

$v_{ij}$ (baru) = $v_{ij}$ (lama) + $\Delta\, v_{ij}$ ……………………………………(2.8)

Langkah 9 : Test kondisi berhenti.

Keterangan Rumus:

$z_in$ : Bobot sinyal unit tersembunyi

$z$ : Keluaran unit tersembunyi

$y_in$ : Bobot sinyal lapisan keluaran

$y$ : Keluaran unit keluaran

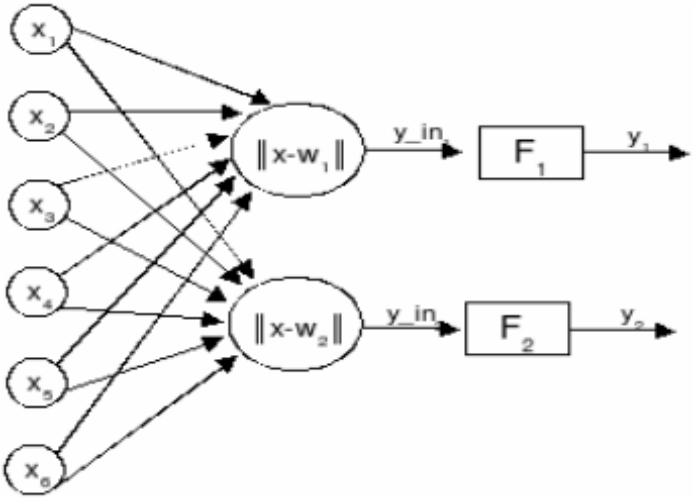
$\delta$ : Kesalahan

$\Delta w$ : Koreksi bobot unit keluaran

$\Delta v$ : Koreksi bobot unit tersembunyi

**D. Learning vector quantization (LVQ)**

LVQ merupakan jaringan syaraf dengan tipe arsitektur jaringan lapis-tunggal umpan-maju (Single Layer Feedforward) yang terdiri atas unit masukan dan unit keluaran. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor masukan . Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitifini hanya tergantung pada jarak antara vektor vektor masukan. Jika 2 vektor masukan mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor masukan tersebut ke dalam kelas yang sama. Arsitektur jaringan learning vector quantization seperti terlihat pada Gambar 2.13.[1]




Gambar 2.14 Learning vector quantization[1]

Gambar 2.14 memperlihatkan bahwa yang bertindak sebagai dendrit atau data masukan adalah X1-X6, yang bertindak sebagai sinapsis atau bobot adalahW, sedangkan soma atau badan sel dari jaringan ini adalah perhitungan $X - W$ . Dan yang bertindak sebagai akson atau data keluaran adalah Y. Berikut ini algoritma pembelajaran metode learning vector quantization:

Langkah 0 : Inisialisasikan vektor referensi dan learning rate(α).

Langkah 1 : Selama kondisi berhenti bernilai salah, kerjakan langkah 2 dan 3 (Kondisi berhenti jika perulangan mencapai maksimal perulangan dan perubahan rasio lebih kecil dari minimal eror)

Langkah 2 : Untuk masing-masing pelatihan vektor masukan , kerjakan:

a. Temukan J sehingga $\|x - Wj\|$ bernilai minimum.

$$c_j = \sqrt{\sum_{i=1}^{n}(x_{i-}\, w_{ij})^2} \quad \text{.....(2.9)}$$

b. Perbaiki Wj dengan :

i. Jika T=Cj maka

Wj(baru) = Wj(lama) + α [x - Wj(lama)] ……....(2.10)

ii. Jika T≠Cj maka

Wj(baru) = Wj(lama) - α [x - Wj(lama)]……...(2.11)

Langkah 3 : Kurangi learning rate.

$$\alpha = \alpha * pengurang\ rasio$$

Langkah 4: Tes kondisi berhenti.

Keterangan rumus:

T : Target

J : Jumlah selisih data dan bobot

C : Kelas selisih bobot terkecil

W : Bobot

: Rasio Pembelajaran

x : data

### 2.4.3.2 Pembelajaran Tak Terawasi (Unsupervised Learning)

Metode pembelajaran tak terawasi merupakan metode pembelajaran yang tidak memerlukan target keluaran. Pada metode ini, tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu range tertentu tergantung pada nilai masukan yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya sangat cocok untuk pengelompokan pola.[1]

**A. Jaringan Kohonen**

Jaringan kohonen pertama kali diperkenalkan oleh Prof. Teuvo Kohonen pada tahun 1982. Pada jaringan ini, suatu lapisan yang berisi neuron-neuron akan menyusun dirinya sendiri berdasarkan masukan nilai tertentu dalam suatu kelompok yang dikenal dengan istilah pengelompokkan (cluster). Selama proses penyusunan diri, cluster yang memiliki vektor bobot paling cocok denga pola masukan (memiliki jarak yang paling dekat) akan terpilih sebagai pemenang. Neuron yang menjadi pemenang beserta neuron-neuron tetangganya akan memperbaiki bobotnya.

## 2.5 Pengenalan Pola (Pattern Recognition)

Pola(pattern) merupakan contoh model sesuatu yang dapat disalin. Tetapi pada saat menjelaskan berbagai jenis objek, baik untuk masalah yang tampak secara fisik ataupun abstrak, setiap hubungan antar data (analog/digital),konsep/kejadian, merupakan pola. Contoh masalah pengenalan pola adalah :pengenalan wajah, sebait musik, lukisan, perkataan yang direkam, tulisan tangan dan lain sebagainya. Secara umum teknik pengenalan pola bertujuan mengklasifikasikan dan mendeskripsikan pola atau objek yang kompleks melalui pengukuran sifat atau ciri-ciri objek yang bersangkutan.[9]

### 2.5.1 Pengenalan Citra (Image Recognition)

Citra (image) adalah gambar pada bidang dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinyu menjadi gambar diskrit melalui proses digitasi. Citra sebagai keluaran dari suatu sistem perekam data dapat bersifat :

1. Optik berupa foto
2. Analog berupa sinyal video seperti gambar pada TV
3. Digital yang dapat langsung disimpan pada storage device

Sebagai hasil keluaran, citra dapat berupa :

1. Citra diam (still image)

2. Citra bergerak (moving image)

Contoh citra diam dapat dilihat pada Gambar 2.15.

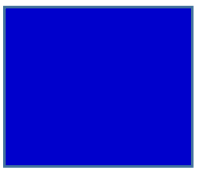


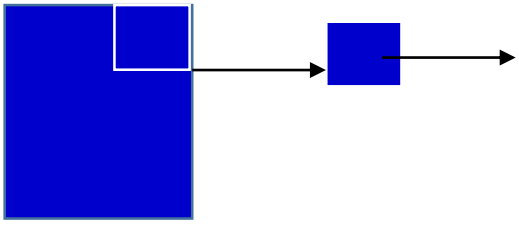
Gambar 2.15 Gambar Diam (Still Image)[15]

Pengenalan(recognition) yaitu suatu tindakan untuk mengenali sesuatu, baik benda maupun lainnya. Pengenalan Citra(image recognition) adalah kemampuan mesin atau program untuk mengidentifikasi pola dan bentuk dalam suatu gambar dan mengkonversikannya ke format yang dapat dibaca oleh mesin. Pengembangan dari pengenalan citra antara lain pengenalan wajah, pengenalan sidik jari, pengenalan tulisan tangan, pengenalan tanda rangan dan lain sebagainya.

Sebuah citra digital memiliki tingkat kedalaman warna 24 bit, pada setiap pixelnya terdiri dari 3 unsur warna yaitu Red (R), Green (G) dan Blue (B) yang masing-masing memiliki kedalaman warna 8 bit (0-255).[6] Berikut ini merupakan ilustrasinya:

1. Sebagai contoh, berikut ini gambar sebuah persegi dengan ukuran 3x3 px (kotak berwarna biru).



2. Setiap pixel terdiri dari 3 unsur warna yaitu red, green dan blue yang masing-masing memiliki kedalaman warna 8 bit (0-255).
3. 

Red = 0

Green = 0

Blue = 255

Nilai pada ketiga unsur warna tersebut (RGB) merupakan kode yang ditentukan oleh suatu badan yang disebut International Electrotechnical Commision (IEC) untuk mewakili setiap kombinasi warna pada komputer.

Untuk dapat digunakan pada sistem pengenalan citra, citra digital harus diubah formatnya kedalam bentuk yang dapat dibaca oleh mesin, yaitu bentuk biner, atau dengan kata lain menjadi citra dengan kedalaman warna 2 bit (0 dan 1). Untuk mengubah citra 24 bit menjadi 2 bit, citra perlu melewati tahap preprocessing, tahap ini berfungsi untuk menyederhanakan citra sehingga dapat digunakan untuk proses pengenalan citra.

Mengubah citra dari 24 bit menjadi 2 bit tidak dapat dilakukan secara langsung karena hal tersebut akan menghilangkan informasi-informasi penting yang terdapat dalam citra tersebut. Untuk itu pertama-tama kedalaman warna perlu disederhanakan menjadi keabu-abuan (grayscale) atau dengan kata lain 8 bit. Setelah warna disederhanakan menjadi keabu-abuan, dilakukan proses untuk mendapatkan warna hitam dan putih, yaitu melalui proses deteksi tepi. Hasil dari proses deteksi tepi berupa citra dengan warna hitam dan putih (0 dan 255), setelah itu citra melalui proses thresholding untuk mengubah nilai pixel menjadi biner (0 dan 1). Setelah citra sudah merubah menjadi 2 bit/biner, maka citra sudah dapat digunakan untuk perhitungan-perhitungan lain seperti pengenalan citra.

## 2.6 Pengolahan Citra

Pengolahan citra merupakan kegiatan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia/mesin (komputer). masukannya adalah citra dan keluarannya juga citra tapi dengan kualitas lebih baik daripada citra masukan, misal citra warnanya kurang tajam, kabur (blurring), mengandung noise (misal bintik-bintik putih), sehingga perlu ada pemrosesan untuk memperbaiki citra karena citra tersebut menjadi sulit diinterpretasikan karena informasi yang disampaikan menjadi berkurang. Operasi-operasi pada pengolahan citra diterapkan pada citra bila[6] :

1. Perbaikan atau memodifikasi citra dilakukan untuk meningkatkan kualitas penampakan citra/menonjolkan beberapa aspek informasi yang terkandung dalam citra (image enhancement). Contoh : perbaikan kontras gelap/terang, perbaikan tepian objek, penajaman, pemberian warna semu, dll
2. Adanya cacat pada citra sehingga perlu dihilangkan/diminimumkan (image restoration). Contoh : penghilangan kesamaran (debluring) citra tampak kabur karena pengaturan fokus lensa tidak tepat / kamera goyang, penghilangan noise.
3. Elemen dalam citra perlu dikelompokkan, dicocokan atau diukur (image segmentation). Operasi ini berkaitan erat dengan pengenalan pola.
4. Diperlukannya ekstraksi ciri-ciri tertentu yang dimiliki citra untuk membantu dalam pengidentifikasian objek (image analysis). Proses segementasi kadangkala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya. Contoh : pendeteksian tepi objek
5. Sebagian citra perlu digabung dengan bagian citra yang lain (image reconstruction). Contoh : beberapa foto rontgen digunakan untuk membentuk ulang gambar organ tubuh
6. Citra perlu dimampatkan (image compression) Contoh : suatu file citra berbentuk BMP berukuran 258 KB dimampatkan dengan metode JPEG menjadi berukuran 49 KB.

7. Menyembunyikan data rahasia (berupa teks/citra) pada citra sehingga keberadaan data rahasia tersebut tidak diketahui orang (steganografi dan watermarking)

Beberapa jenis operasi pengolahan citra adalah sebagai berikut:

1. Modifikasi Kecemerlangan(Brightness Modification)
   Mengubah nilai keabuan/warna dari gelap menuju terang atau sebaliknya mengubah citra yang terlalu cemerlang/pucat menjadi gelap.
2. Peningkatan Kontras (Contast Enhancement)
   Dengan peningkatan kontras maka titik yang cenderung gelap menjadi lebih gelap dan yang cenderung terang menjadi lebih cemerlang.
3. Negasi
   Operasi untuk mendapatkan citra negatif (negative image)
4. Pengabuan (grayscale)
   Merupakan proses konversi citra dengan warna sebenarnya (true color) menjadi citra keabuan (grayscale).
5. Pengambangan (Thresholding)
   Operasi pengambangan digunakan untuk mengubah citra dengan format skala keabuan, yang mempunyai kemungkinan nilai lebih dari 2 ke citra biner yang memiliki 2 buah nilai (yaitu 0 dan 1)
6. Pencerminan (Flipping)
   Pencerminan merupakan proses menggambar citra ke bentuk kebalikannya seperti ketika sedang bercermin.
7. Rotasi (Rotating)
   Rotasi yaitu proses memutar koordinat citra sesuai derajat yang ditentukan.
8. Pemotongan (Cropping)
   Memotong satu bagian dari citra sesuai kebutuhan.
9. Pengskalaan (Scaling)
   Mengubah ukuran citra menjadi lebih besar atau lebih kecil.

10. Deteksi Tepi (Edge Detection)

    Deteksi tepi (edge detection) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra.

## 2.7 Citra Digital

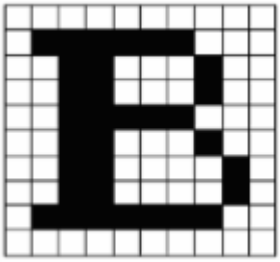
Citra digital adalah gambar dua dimensi yang dapat ditampilkan pada layar monitor komputer sebagai himpunan berhingga (diskrit) nilai digital yang disebut pixel (picture elements). Piksel adalah elemen citra yang memiliki nilai yang menunjukkan intensitas warna.

Berdasarkan cara penyimpanan atau pembentukannya, citra digital dapat dibagi menjadi dua jenis. Jenis pertama adalah citra digital yang dibentuk oleh kumpulan pixel dalam array dua dimensi. Citra jenis ini disebut citra bitmap (bitmap image) atau citra raster (raster image). Jenis citra yang kedua adalah citra yang dibentuk oleh fungsi-fungsi geometri dan matematika. Jenis citra ini disebut grafik vektor (vector graphics).

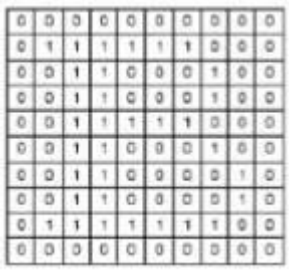
Dalam pembahasan tugas akhir ini, yang dimaksud citra digital adalah citra bitmap. Citra digital (diskrit) dihasilkan dari citra analog (kontinu) melalui digitalisasi. Digitalisasi citra analog terdiri atas sampling dan kuantisasi (quantization) Sampling adalah pembagian citra ke dalam elemen-elemen diskrit (pixel), sedangkan kuantisasi adalah pemberian nilai intensitas warna pada setiap pixel dengan nilai yang berupa bilangan bulat (G.W. Awcock, 1996).

## 2.8 Citra Biner

Citra biner adalah gambar pada bidang dua dimensi. Dalam tinjauan matematis, citra yang hanya terdiri atas dua warna, yaitu hitam dan putih. Oleh karena itu, setiap pixel pada citra biner cukup dipresentasikan dengan satu bit. Pada gambar 2.16 merupakan citra biner, sedangkan pada gambar 2.17 merupakan representasi dari citra biner, dimana citra yang berwarna putih memiliki nilai satu, sedangkan citra yang berwarna hitam memiliki nilai nol.



Gambar 2.16 Citra biner



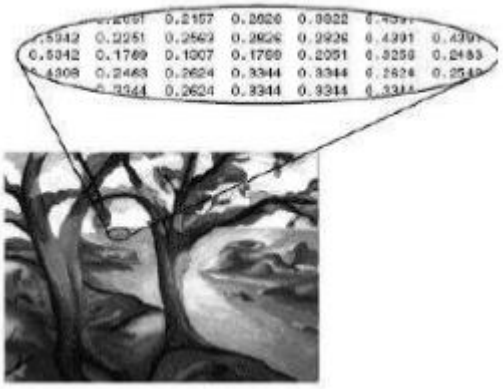
Gambar 2.17 Representasi citra biner

Meskipun saat ini citra berwarna lebih disukai karena memberi kesan yang lebih kaya daripada citra biner, namun tidak membuat citra biner tidak digunakan lagi. Pada beberapa aplikasi citra biner masih tetap dibutuhkan, misalkan citra logo instansi (yang hanya terdiri dari warna hitam dan putih), citra kode barang (barcode) yang tertera pada label barang, citra hasil pemindaian dokumen teks, dan sebagainya. Seperti yang sudah disebutkan diatas, citra biner hanya mempunyai dua nilai derajat keabuan : hitam dan putih. Pixel-pixel objek bernilai satu dan pixel-pixel latar belakang bernilai nol. Pada waktu menampilkan gambar, adalah putih dan satu adalah hitam. Jadi pada citra biner, latar belakang berwarna putih sedangkan objek berwarna hitam seperti tampak pada gambar 2.16 diatas. Meskipun komputer saat ini dapat memproses citra putih-abu abu (grayscale) maupun citra berwarna, namun citra biner masih tetap di pertahankan keberadaannya [7]

## 2.9 Citra Grayscale

Dalam komputasi, suatu citra digital grayscale adalah suatu citra dimana nilai dari setiap pixel merupakan sampel tunggal. Citra yang ditampilkan dari citra jenis ini terdiri atas warna abu-abu, bervariasi pada warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat. Citra grayscale berbeda dengan citra hitam-putih (black dan write), dimana pada konteks komputer, citra hitam putih hanya terdiri atas dua warna saja yaitu hitam dan putih saja. Pada citra grayscale warna bervariasi antara putih dan abu-abu, tetapi variasi warna diantaranya sangat banyak. Citra grayscale seringkali merupakan perhitungan dari intensitas cahaya pada setiap pixel pada spektrum elektromagnetik.

Citra grayscale disimpan dalam format delapan bit untuk setiap sampel piksel, yang memungkinkan sebanyak 256 intensitas. Format ini sangat membantu dalam pemrograman karena manipulasi bit yang tidak terlalu banyak. Pada aplikasi lain seperti pada aplikasi medical imaging dan remote sensing biasa juga digunakan format 10 bit, 12 bit dan 16 bit yaitu citra yang nilai pixel-nya merepresentasikan derajat keabuan atau intensitas warna putih. Nilai intensitas paling rendah merepresentasikan warna hitam dan nilai intensitas paling tinggi merepresentasikan warna putih. Pada umumnya citra grayscale memiliki kedalaman pixel delapan bit (256 derajat keabuan), tetapi ada juga citra grayscale yang kedalaman pixel-nya bukan delapan bit, misalnya 16 bit untuk penggunaan yang memerlukan ketelitian tinggi. Pada gambar 2.17 merupakan contoh citra grayscale.



Gambar 2.18 Citra grayscale

Citra grayscale merupakan citra satu kanal, dimana citra f(x,y) merupakan fungsi tingkat keabuan dari hitam keputih, x menyatakan variable kolom atau posisi pixel di garis jelajah dan y menyatakan variable kolom atau posisi pixel di garis jelajah. Intensitas f dari gambar hitam putih pada titik (x,y) disebut derajat keabuan (grey level), yang dalam hal ini derajat keabuannya bergerak dari hitam keputih. Derajat keabuan memiliki rentang nilai dari I min sampai I max, atau Imin < f < Imax, selang (Imin, Imax) disebut skala keabuan. Biasanya selang (Imin, Imax) sering digeser untuk alasan-alasan praktis menjadi selang [0,L], yang dalam hal ini nilai intensitas nol meyatakan hitam, nilai intensitas L meyatakan putih, sedangkan nilai intensitas antara nol sampai L bergeser dari hitam ke putih. Sebagai contoh citra grayscale dengan 256 level artinya mempunyai skala abu dari nol sampai 255 atau [0,255], yang dalam hal ini intensitas nol menyatakan hitam, intensitas 255 menyataka putih, dan nilai antara nol sampai 255 menyatakan warna keabuan yang terletak antara hitam dan putih. [7]

## 2.10 Citra Asli

Citra asli yaitu citra yang nilai pixel-nya merepresentasikan warna tertentu. Banyaknya warna yang mungkin digunakan bergantung kepada kedalaman pixel citra yang bersangkutan. Citra berwarna direpresentasikan dalam beberapa kanal yang

menyatakan komponen-komponen warna penyusunnya. Banyaknya kanal yang digunakan bergantung pada model warna yang digunakan pada citra tersebut. Pada gambar 2.19 merupakan contoh citra RGB.

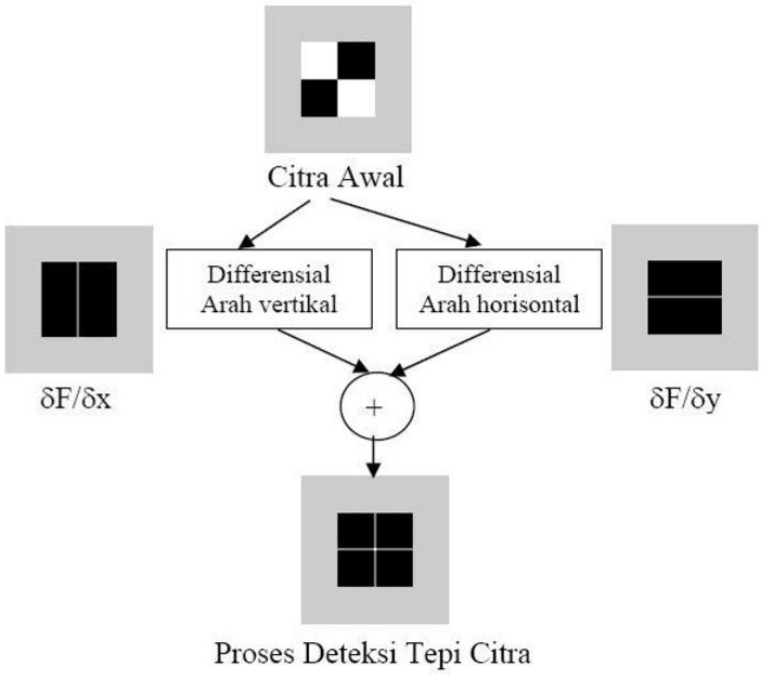


Gambar 2.19 Citra RGB

Intensitas suatu pada titik pada citra berwarna merupakan kombinasi dari tiga intensitas : derajat keabuan merah (fmerah(x,y)), hijau (fhijau(x,y)), dan biru (fbiru(x,y)). Persepsi visual citra berwarna umumnya lebih kaya di bandingkan dengan citra hitam putih. Citra berwarna menampilkan objek seperti warna aslinya meskipun tidak selalu tepat demikian. Warna-warna yang diterima oleh mata manusia merupakan hasil kombinasi cahaya dengan panjang gelombang berbeda. [7]

## 2.11 Deteksi Tepi (Edge Detection)

Deteksi tepi (edge detection) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, Tepi-tepi ini akan menandai bagian detail citra. Suatu titik (x,y) dikatakan sebagai tepi (edge) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Gambar 2.20 menggambarkan bagaimana tepi suatu gambar diperoleh.



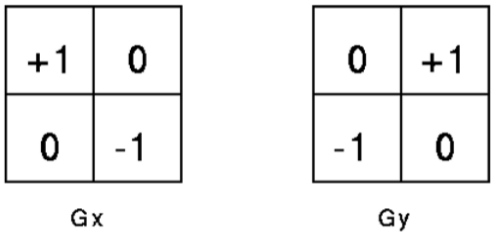

Gambar 2.20 Edge Detection [7]

Macam-macam metode untuk proses deteksi tepi, antara lain:

**1. Metode Robert**

Metode pendektesian tepi dengan mencari perbedaan (differensial ) pada arah horisontal dan perbedaan pada arah vertikal, dengan ditambahkan prosesi konversi biner setelah dilakukan perbedaan. Agar mendapatkan tepi-tepi yang lebih baik, maka konversi biner dilakukan dengan meratakan distribusi warna hitam dan putih atau dengan kata lain objek gambar yang akan digunakan untuk metode ini sebaiknya adalah gambar hitam putih. Kelebihan operator ini yaitu cepat dan mudah untuk menghitung pengukuran gradien 2-D spasial pada gambar. Pixel nilai di setiap titik di output mutlak mewakili besarnya estimasi gradien spasial citra input pada saat itu. Operator terdiri dari

sepasang kernel 2 × 2 konvolusi. Satu kernel yang lain hanya diputar oleh 90 °. Ini sangat mirip dengan operator Sobel.




Gambar 2.21 Robert's Method [7]

Kernel ini dirancang untuk merespon secara maksimal untuk tepi berjalan pada 45 ° ke jaringan pixel, satu kernel untuk masing-masing dua orientasi tegak lurus. Kernel dapat diterapkan secara terpisah dengan citra input, 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. Besarnya gradien diberikan oleh:

$|G| = \sqrt{Gx^2 + Gy^2}$ ……(2.12)

walaupun biasanya, sebuah besaran perkiraan dihitung dengan menggunakan:

|G| = |Gx| + |Gy| ……..(2.13)

yang lebih cepat untuk menghitung.

Sudut orientasi tepi menimbulkan gradien spasial (relatif terhadap orientasi grid pixel) diberikan oleh:

θ = arctan (Gy / Gx) - 3π/4 ……..(2.14)

**2. Metode Prewitt**

Metode ini adalah pengembangan metode robert dengan menggunakan

filter HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian yang dikenal sebagai fungsi untuk membangkitkan HPF. Sehingga tepi-tepi yang dihasilkan lebih banyak dari metode Robert.

$S_x =$

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

$S_y =$

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

Gambar 2.22 Prewitt Method [7]

Filter HPF menciptakan sebuah gambar di mana tepi (perubahan tajam nilai gray level) yang akan ditampilkan. Hanya ukuran 3x3 filter dapat digunakan dengan filter ini. Filter ini menggunakan dua 3x3 template untuk menghitung nilai Prewitt gradien seperti yang ditunjukkan di bawah ini:

Template:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$ Sebagai X …….....(2.15)

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$ Sebagai Y ……...(2.16)

Gunakan template untuk jendela filter 3x3

$\begin{bmatrix} a1\ a2\ a3 \\ a4\ a5\ a6 \\ a7\ a8\ a9 \end{bmatrix}$ 3x3 filter jendela ……(2.17)

dimana a1 .. a9 adalah tingkat keabuan dari setiap piksel di jendela filter.

X = -1 * a1 + 1 * a3 - 1 * a4 + 1 * A6 - 1 * A7 + 1 * A9.....(2.18)

Y = 1 * a1 + 1 * a2 + 1 * a3 - 1 * A7 - 1 * A8 - 1 * A9.....(2.19)

PrewittGradient = sqrt (X * X + Y * Y).....(2.20)

**3. Metode Sobel**

Metode ini juga 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 2 metode sebelumnya. Operator terdiri dari sepasang kernel 3 × 3 konvolusi seperti yang ditunjukkan pada Gambar 2.22. Satu kernel yang lain hanya diputar oleh 90 °.

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

Gx

| | | |
|---|---|---|
| +1 | +2 | +1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

Gy

Gambar 2.23 Sobel Method [7]

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 input, 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. Besarnya gradien diberikan oleh:

$|G| = \sqrt{Gx^2 + Gy^2}$ ……(2.21)

walaupun biasanya, sebuah besaran perkiraan dihitung dengan menggunakan:

$|G| = |Gx| + |Gy|$ ……..(2.22)

yang lebih cepat untuk menghitung.

Sudut orientasi tepi (relatif ke grid pixel) menimbulkan gradien spasial diberikan oleh:

$\theta = \arctan(Gy / Gx)$ ……(2.23)

**4. Metode Canny**

Deteksi tepi Canny dapat mendeteksi tepian yang sebenarnya dengan tingkat eror yang minimum dengan kata lain, operator Canny di desain untuk menghasilkan citra tepian yang optimal. Langkah-langkah Deteksi Tepi Canny:

Berikut adalah langkah-langkah dalam melakukan deteksi tepi Canny.

Langkah I : Menghilangkan Noise yang ada pada citra dengan mengimplementasikan Filter Gaussian. Hasilnya citra akan tampak sedikit buram. Hal ini dimaksudkan untuk mendapatkan tepian citra yang sebenarnya. Bila tidak dilakukan maka garis-garis halus juga akan dideteksi sebagai tepian. Berikut ini adalah salah satu contoh filter gaussian dengan $\sigma = 1.4$ :

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Langkah II : melakukan deteksi tepi dengan salah satu operator deteksi tepi seperti Roberts, Perwit atau Sobel dengan melakukan pencarian secara horizontal (Gx) dan secara vertikal (Gy). Berikut ini salah satu contoh operator deteksi tepi (Operator Sobel):

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$G_x$ $G_y$

Hasil dari kedua operator digabungkan untuk mendapatkan hasil gabungan tepi vertikal dan horizontal dengan rumus:

$|G|=|G_X| + |G_y|$ ……(2.24)

Langkah III : Menentukan Arah tepian yang ditemukan dengan menggunakan rumus:

$\theta = \arctan\left(\frac{G_y}{G_x}\right)$ ……(2.25)

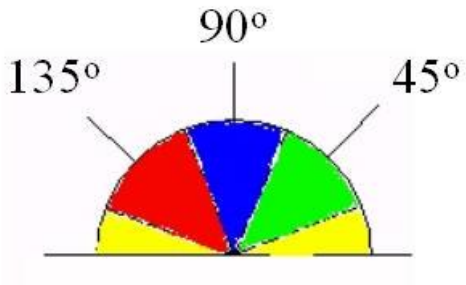
Dan selanjutnya membagi ke dalam 4 warna sehingga garis dengan arah yang berbeda memiliki warna yang berbeda. Pembagiannya adalah :

Derajat 0 – 22,5 dan 157,5 – 180 berwarna Kuning.

Derajat 22,5 – 67,5 Berwarna Hijau, dan

Derajat67,5 – 157,5 Berwarna Merah.

Berikut ini adalah bagan pembagian warna berdasarkan arah tepian yang dilakukan oleh Canny:




Gambar 2.24 Pembagaian warna yang di lakukan Canny [7]

## 2.12 Pengertian Eclipse

Eclipse adalah sebuah IDE (Integrated Development Environment) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (platform-independent). Berikut ini adalah sifat dari Eclipse:

a. Multi-platform: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
b. Mulit-language: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
c. Multi-role: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan open source, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan plug-in. Eclipse dibuat dari kerja sama antara perusahaan-perusahaan anggota 'Eclipse Foundation' (beserta individu-individu lain). Banyak nama besar yang ikut dalam 'Eclipse Foundation', termasuk IBM, BEA, Intel, Nokia, Borland. Eclipse bersaing langsung dengan Netbeans IDE. Plugin tambahan pada Eclipse jauh lebih banyak dan bervariasi dibandingkan IDE lainnya.

## 2.13 UML (Unified Modeling Language)

a. Pengertian UML

UML adalah sebuah bahasa untuk menetukan, visualisasi, kontruksi, dan mendokumentasikan artifact bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. Artifact dapat berupa model, deskripsi atau perangkat lunak dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

b. Bagian-bagian UML

Bagian-bagian utama dari UML adalah view, diagram, model element, dan general mechanism.

1. View

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. View bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis view dalam UML antara lain: use case view, logical view, component view, concurrency view, dan deployment view.

a) Use case view

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan external actors. Actor yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya. View ini digambarkan dalam use case diagrams dan kadang-kadang dengan activity diagrams. View ini digunakan terutama untuk pelanggan, perancang (designer), pengembang (developer), dan penguji sistem (tester).

b) Logical view

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (class, object,dan relationship) dan kolaborasi dinamis yang terjadi ketika object mengirim pesan ke object lain dalam suatu fungsi tertentu. View ini digambarkan dalam class diagrams untuk struktur statis dan dalam state, sequence, collaboration, dan activity diagram untuk model dinamisnya. View ini digunakan untuk perancang (designer) dan pengembang (developer).

c) Component view

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari code module diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. View ini digambarkan dalam component view dan digunakan untuk pengembang (developer).

d) Concurrency view

Membagi sistem ke dalam proses dan prosesor. View ini digambarkan dalam diagram dinamis (state, sequence, collaboration, dan activity diagrams) dan diagram implementasi (component dan deployment diagrams) serta digunakan untuk pengembang (developer), pengintegrasi (integrator), dan penguji (tester).

e) Deployment view

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (nodes) dan bagaimana hubungannya dengan lainnya. View ini digambarkan dalam deployment diagrams dan digunakan untuk pengembang (developer), pengintegrasi (integrator), dan penguji (tester).

2.Diagram

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu view tertentu dan ketika digambarkan biasanya dialokasikan untuk view tertentu. Adapun jenis diagram antara lain :

a) Use Case Diagram

Use case adalah abstraksi dari interaksi antara sistem dan actor. Use case bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Use case merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata user. Sedangkan use case diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan client.

b) Class Diagram

Class adalah dekripsi kelompok obyek-obyek dengan property, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya class diagram dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari class- class yang ada dan relasinya satu dengan yang lainnya.Sebuah sistem biasanya mempunyai beberapa class diagram.Class diagram sangat membantu dalam visualisasi struktur kelas dari suatu sistem.

c) Component Diagram

Component software merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. Komponen merupakan implementasi software dari sebuah atau lebih class. Komponen dapat berupa source code, komponen biner, atau executable component. Sebuah komponen berisi informasi tentang logic class atau class yang diimplementasikan sehingga membuat pemetaan dari logical view ke component view. Sehingga component diagram merepresentasikan dunia nyata yaitu component software yang mengandung component, interface dan relationship.

d) Deployment Diagram

Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian software yang berjalan pada bagian-bagian hardware, menunjukkan hubungan komputer dengan perangkat (nodes) satu sama lain dan jenis hubungannya. Di dalam nodes, executeable component dan object yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh node tertentu dan ketergantungan komponen.

e) State Diagram

Menggambarkan semua state (kondisi) yang dimiliki oleh suatu object dari suatu class dan keadaan yang menyebabkan state berubah. Kejadian dapat berupa object lain yang mengirim pesan. State class tidak

digambarkan untuk semua class, hanya yang mempunyai sejumlah state yang terdefinisi dengan baik dan kondisi class berubah oleh state yang berbeda.

f) Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara object juga interaksi antara object, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

g) Collaboration Diagram

Menggambarkan kolaborasi dinamis seperti sequence diagrams. Dalam menunjukkan pertukaran pesan, collaboration diagrams menggambarkan object dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan sequence diagrams, tapi jika penekanannya pada konteks gunakan collaboration diagram.

h) Activity Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti use case atau interaksi. [8]

## 2.14 Android

Android adalah sistem operasi *mobile* yang dimodifikasi dari sistem operasi Linux. Android bersifat *open source* yang berarti setiap pengembang *software* yang ingin menciptakan aplikasi Android dapat mengunduh *source code* Android secara gratis. Dengan menggunakan Android, pengembang perangkat lunak bebas merancang aplikasi yang bisa dijalankan pada berbagai perangkat yang berbeda. Beberapa perangkat yang dapat menggunakan sistem operasi Android adalah *smartphone, tablet*, perangkat e-*reader*, *netbooks*, MP4 *players*, dan TV internet.

## 2.15 Arsitektur Android

Sistem operasi Android memiliki arsitektur yang terdiri dari 5 lapisan yakni :

a. *Applications*

*Applications* adalah lapisan paling atas dari arsitertur Android yang terdiri dari aplikasi inti dari Android seperti email *client*, program SMS , kalender, peta, *browser*, kontak, dan lain-lain. Semua aplikasi ini ditulis menggunakan bahasa pemrograman Java.

b. *Application Framework*

*Application framework* adalah lapisan dimana terdapat program untuk mengatur fungsi-fungsi dasar dari aplikasi. Pengembang aplikasi bisa menggunakan fungsi-fungsi dasar tersebut untuk mengembangkan aplikasi Android.

c. *Libraries*

Android mengandung sekumpulan *library* C/C++ yang digunakan oleh berbagai komponen dalam sistem android. Berikut ini adalah beberapa *library* inti dari Android:

- *System* C *library* – implementasi BSD yang berasal dari sistem *library* C standar yang disesuaikan untuk dilekatkan pada perangkat berbasis Linux.
- Media *Libraries* – *Library* yang mendukung pemutaran dan perekaman berbagai format audio, video, dan gambar seperti MPEG4, H.264, MP3, AAC, AMR, JPG, dan PNG.
- *Surface Manager* – mengelola akses ke tampilan subsistem dan memuluskan gabungan layar grafik 2D dan 3D dari beberapa aplikasi.
- LibWebCore – sebuah mesin web *browser* modern yang memiliki kekuatan *browser* Android dan tampilan *web embeddable*.
- SGL – *library* yang mendasari mesin grafik 2D.

- 3D *libraries – library* yang merupakan implementasi dari OpenGL ES 1.0 APIs yang digunakan untuk rasterisasi objek 3D.
- *FreeType – library* untuk me-*render font bitmap* dan *vector*
- SQLite – *library* yang merupakan mesin *database* relasional yang kuat, ringan dan tersedia untuk semua aplikasi.

d. Android *Runtime*

Android *runtime* terdiri dari sekumpulan *library* inti yang menyediakan fungsi *library* inti pada pemrograman bahasa java. Android *runtime* juga memiliki mesin virtual Dalvik yang membuat setiap aplikasi android dapan berjalan sesuai dengan prosesnya sendiri. Mesin virtual Dalvik dibuat khusus untuk android dan dioptimalkan pada perangkat *mobile* yang menggunakan baterai, memori, dan CPU yang terbatas.

e. Linux Kernel

Linux kernel Android menggunakan Linux versi 2.6 untuk melakukan layanan sistem inti seperti keamanan, manajemen memori, manajemen proses, *network stack* dan *driver model*. Pada lapisan ini mengandung semua *driver* perangkat keras dari Android.