BAB 2
LANDASAN TEORI

2.1 Definisi Tanda Tangan

Tanda tangan adalah hasil proses menulis seseorang yang bersifat khusus sebagai substansi simbolik. Tanda tangan merupakan salah satu bentuk yang digunakan untuk identifikasi seseorang.

Contoh-contoh tanda tangan setiap orang umumnya identik namun tidak sama. Artinya tanda tangan seseorang sering berubah-ubah setiap waktu. Perubahan ini menyangkut posisi, ukuran maupun faktor tekanan tanda tangan. Pada kenyataannya, perubahan-perubahan tersebut dipengaruhi oleh waktu, umur, kebiasaan dan keadaan mental tertentu (Abbas, 1994).



Gambar 2.1 Contoh Citra Tanda Tangan

2.2 Kecerdasan Buatan

Kecerdasan Buatan adalah salah satu cabang ilmu pengetahuan berhubungan dengan pemanfaatan mesin untuk memecahkan persoalan yang rumit dengan cara yang lebih manusiawi. Hal Ini biasanya dilakukan dengan mengikuti karakteristik dan analogi berpikir dari kecerdasan/Inteligensia manusia, dan menerapkannya sebagai algoritma yang dikenal oleh komputer. Dengan suatu pendekatan yang kurang lebih fleksibel dan efisien dapat diambil tergantung dari keperluan, yang mempengaruhi bagaimana wujud dari perilaku kecerdasan buatan. AI biasanya dihubungkan dengan Ilmu Komputer, akan tetapi juga terkait erat dengan bidang lain seperti Matematika, Psikologi, Pengamatan, Biologi, Filosofi, dan yang lainnya. Kemampuan untuk mengkombinasikan pengetahuan

dari semua bidang ini pada akhirnya akan bermanfaat bagi kemajuan dalam upaya menciptakan suatu kecerdasan buatan (Suyanto, 2014).

Pengertian lain dari kecerdasan buatan adalah bagian ilmu komputer yang membuat agar mesin komputer dapat melakukan pekerjaan seperti dan sebaik yang dilakukan manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kehidupan manusia. Komputer tidak lagi hanya digunakan sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia.

Manusia bisa menjadi pandai dalam menyelesaikan segala permasalahan di dunia ini karena manusia mempunyai pengetahuan dan pengalaman yang diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki oleh seseorang tentu saja diharapkan akan lebih mampu dalam menyelesaikan permasalahan. Namu bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran, mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang mereka miliki. Tanpa memiliki kemampuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik. Demikian juga dengan kemampuan menalar yang sangat baik, namun tanpa bekal pengetahuan dan pengalaman yang memadai, manusia juga tidak akan bisa menyelesaikan masalah dengan baik. Agar komputer bisa bertindak seperti dan sebaik manusia, maka komputer juga harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Untuk itu AI akan mencoba untuk memberikan beberapa metoda untuk membekali komputer dengan kedua komponen tersebut agar komputer bisa menjadi mesin pintar.

2.2.1 Keuntungan Kecerdasan Buatan

Keuntungan kecerdasan buatan antara lain :

1. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang

pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer dan program tidak mengubahnya.

2. Kecerdasan buatan lebih mudah diduplikasi dan disebarkan. Mentransfer pengetahuan manusia dari satu orang ke orang lain butuh proses dan waktu lama. Disamping itu suatu keahlian tidak akan pernah bisa diduplikasi secara lengkap. Sedangkan jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebuat dapat ditransfer atau disalin dengan mudah dan cepat dari satu komputer ke komputer lain
3. Kecerdasan buatan lebih murah dibanding dengan kecerdasan alami. Menyediakan layanan komputer akan lebih mudah dan lebih murah dibanding dengan harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
4. Kecerdasan buatan bersifat konsisten. Hal ini disebabkan karena kecerdasan busatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami senantiasa berubah-ubah.
5. Kecerdasan buatan dapat didokumentasikan. Keputusan yang dibuat komputer dapat didokumentasikan dengan mudah dengan melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
6. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibanding dengan kecerdasan alami
7. Kecerdasan buatan dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami.

2.2.2 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 sidik jari (*finger recognition*), pengenalan suara (*voice recognition*), pengenalan tanda tangan (*signature recognition*), atau pengenalan wajah (*face recognition*). Selain itu juga dapat digunakan sebagai mesin pencarian dengan gambar atau suara sebagai kata kuncinya dengan menggunakan pengenalan gambar (*image recognition*) dan pengenalan suara (*speech recognition*).

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 lain sebagainya 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.

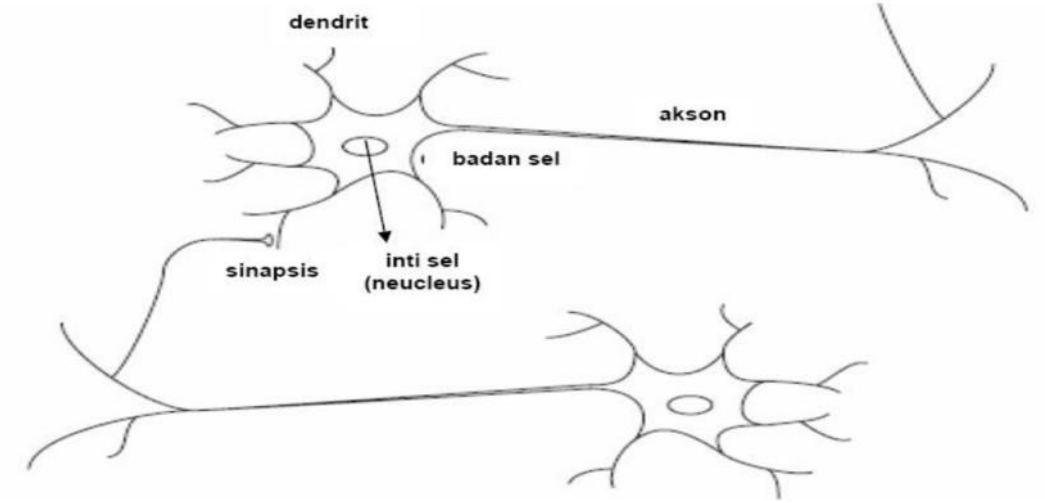
2.3 Jaringan Syaraf Tiruan dan Biologi

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. Jaringan syaraf biologi merupakan jaringan syaraf yang terdapat pada manusia, sedangkan jaringan syaraf tiruan merupakan jaringan syaraf yang dibuat dan dijalankan pada komputer (Muis, 2006).

2.3.1 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 atau 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.



Gambar 2.2 Neuron

Komponen utama neuron dapat dikelompokan 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.

2.3.2 Jaringan Syaraf Tiruan (JST)

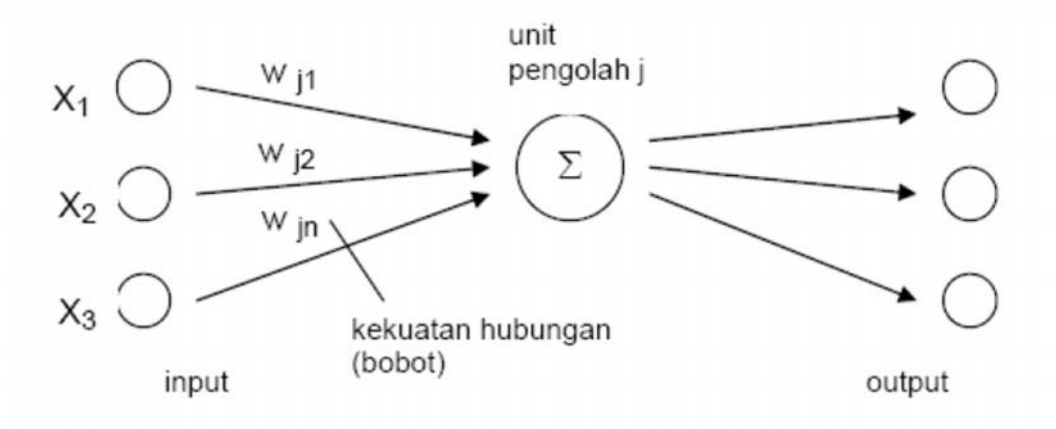
Jaringan syaraf tiruan adalah paradigma pemrosesan suatu informasi yang terinspirasi oleh sistem sel syaraf biologi, sama seperti otak yang memproses suatu informasi. Elemen mendasar dari paradigma tersebut adalah struktur yang baru dari sistem pemrosesan informasi. Jaringan syaraf tiruan, seperti manusia, belajar dari suatu contoh. Jaringan syaraf tiruan dibentuk untuk memecahkan suatu masalah tertentu seperti pengenalan pola atau klasifikasi. Jaringan syaraf tiruan berkembang secara pesat pada beberapa tahun terakhir. Jaringan syaraf tiruan dikembangkan sebelum adanya komputer konvensial yang canggih dan terus berkembang walaupun pernah mengalami masa vakum selama beberapa tahun.

Jaringan syaraf tiruan 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 atau neuron melalui suatu sambungan penghubung.
3. Setiap sambungan penghubung memiliki bobot yang bersesuaian. Bobot ini digunakan untuk menggandakan atau mengalikan sinyal yang dikirim melaluinya.
4. Setiap sel syaraf akan menerapkan fungsi aktivasi terhadap sinyal hasil penjumlahan berbobot yang masuk kepadanya untuk menentukan sinyal keluarannya.

Tabel 2.1 Analogi JST dan JSB

JST	JSB
Node/masukan	Badan sel (soma)
Masukan	Dendrit
Keluaran	Akson
bobot	Sinapsis



Gambar 2.3 Model Struktur JST

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 dasarnya pembelajaran kedalam 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

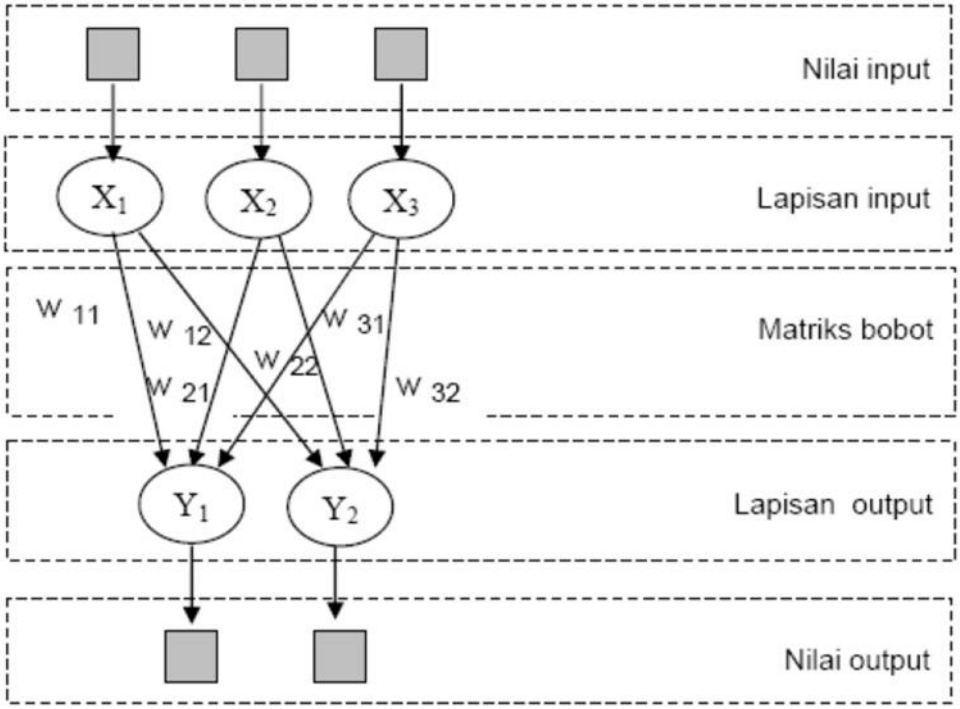
2.3.3 Arsitektur Jaringan Syaraf Tiruan

Pada jaringan syaraf tiruan, neuron-neuron akan dikumpulkan dalam lapisan-lapisan (*layer*) yang disebut dengan lapisan neuron. Neuron-neuron pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya. Informasi yang diberikan pada jaringan syaraf tiruan akan dirambatkan lapisan ke lapisan, mulai dari lapisan masukan sampai lapisan keluaran melalui 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 akan memiliki fungsi aktivasi yang sama. Bila neuron-neuron pada suatu lapisan (misal lapisan tersembunyi) akan dihubungkan dengan neuron-neuron 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 jaringan syaraf tiruan, yaitu :

1. Jaringan Dengan Lapisan Tunggal (*Single Layer Net*)

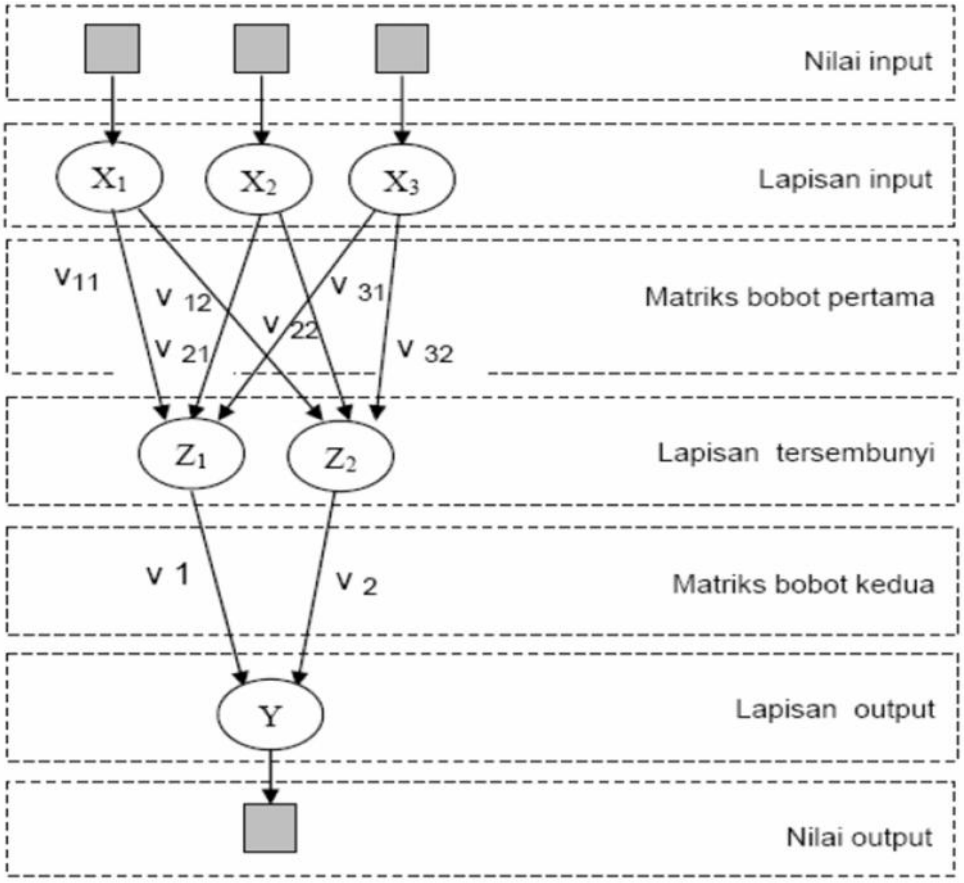
Jaringan ini hanya memiliki 1 lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima masukan kemudian secara langsung akan mengolahnya menjadi keluaran tanpa harus melalui lapisan tersembunyi. Seberapa besar hubungan antara 2 neuron ditentukan oleh bobot yang bersesuaian. Semua unit masukan akan dihubungkan dengan setiap unit keluaran, seperti terlihat pada Gambar 2.4.



Gambar 2.4 Jaringan Dengan Lapisan Tunggal

2. Jaringan Dengan Banyak Lapisan (*Multilayer Net*)

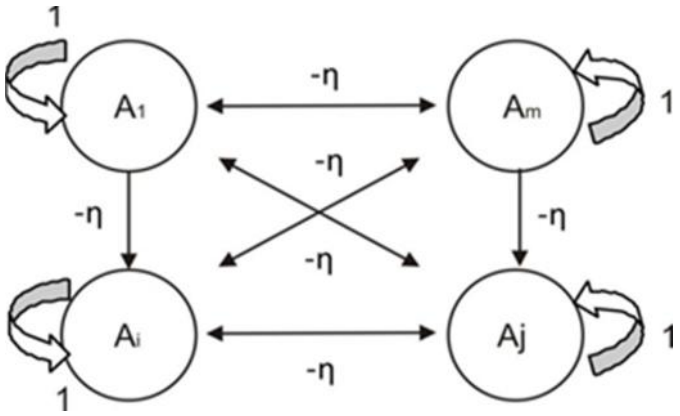
Jaringan ini memiliki 1 atau lebih lapisan yang terletak diantara lapisan masukan dan lapisan keluaran. Umumnya ada lapisan bobot-bobot yang terletak antara 2 lapisan yang bersebelahan. Jaringan dengan banyak jaringan lapisan dapat dilihat pada Gambar 2.5.



Gambar 2.5 Jaringan Dengan Banyak Lapisan

3. Jaringan Dengan Lapisan Kompetitif (*Competitive Layer Net*)

Pada jaringan ini sekumpulan neuron bersaing untuk mendapatkan hak menjadi aktif. Umumnya berhubungan antar neuron pada lapisan kompetitif ini tidak diperlihatkan pada diagram arsitektur. Gambar 2.6 Menunjukan salah satu contoh arsitektur jaringan dengan lapisan kompetitif.



Gambar 2.6 Jaringan Dengan Lapisan Kompetitif

Berdasarkan dari arsitektur (pola koneksi) jaringan syaraf tiruan dapat dibagi kedalam dua kategori :

2.3.3.1 Struktur *feedforward*

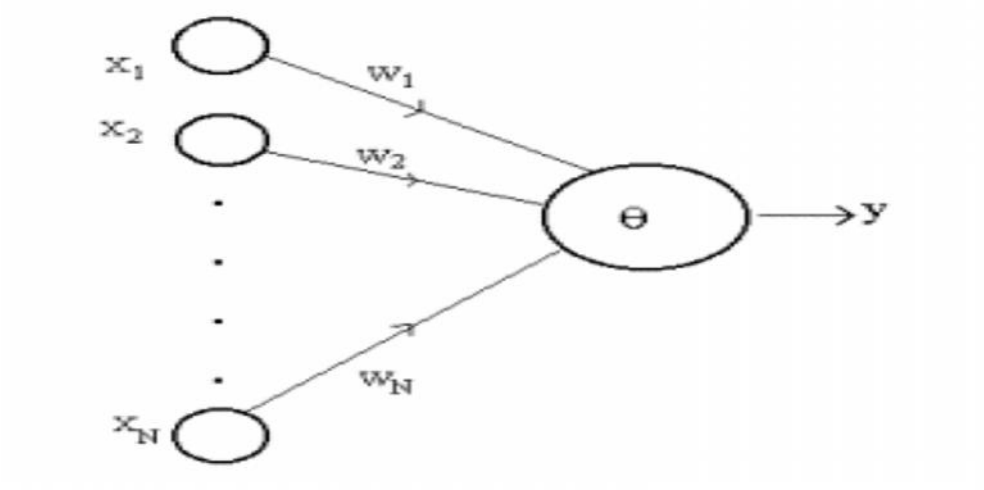
Sebuah jaringan yang sederhana mempunyai struktur *feedforward* dimana signal bergerak dari input kemudian melewati lapisan tersembunyi dan akhirnya mencapai unit output (memiliki struktur perilaku yang stabil).

Tipe jaringan *feedforward* mempunyai sel syaraf yang tersusun dari beberapa lapisan. Lapisan input bukan merupakan sel syaraf. Lapisan ini hanya memberi pelayanan dengan mengenalkan suatu nilai dari suatu variabel. Lapisan tersembunyi dan lapian output sel syaraf terhubung satu sama lain dengan lapisan sebelumnya. Kemungkinan yang timbul adalah adanya hubungan dengan beberapa unit dari lapisan sebelumnya atau terhubung semuanya (lebih baik).

Metode-metode yang termasuk kedalam struktur *feedforward* antara lain :

a. Hebb Rule

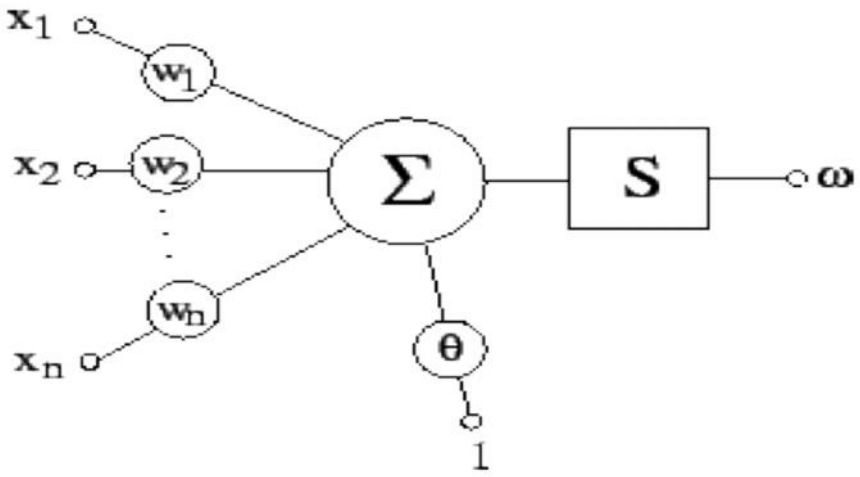
Hebb rule adalah metode pembelajaran yang paling sederhana. Pada metode ini pembelajaran dilakukan dengan secara memperbaiki nilai bobot sedemikian rupa sehingga jika ada 2 neuron yang terhubung, dan keduanya pada kondisi “hidup” pada saat yang sama, maka bobot antara keduanya dinaikkan.



Gambar 2.7 Jaringan Hebb Rule

b. Perceptron

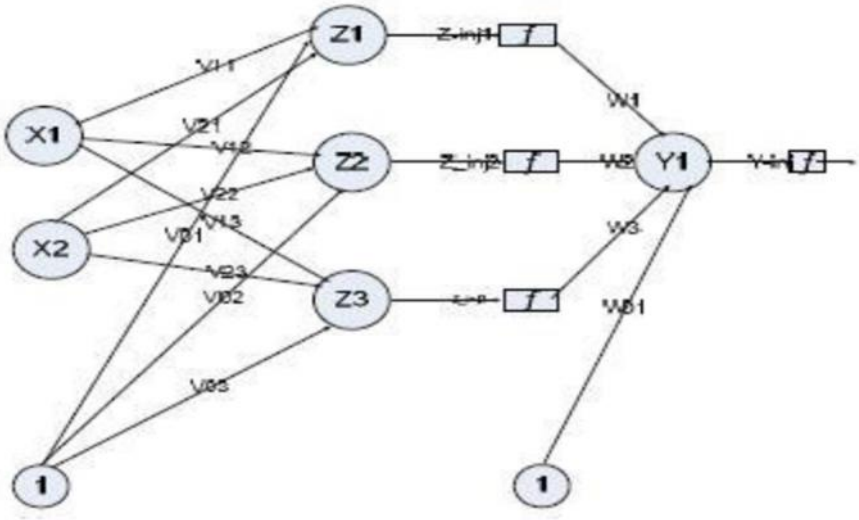
Jaringan jenis ini hanya terdiri dari *layer* masukan dan *layer* keluaran saja. Data masukan yang masuk melalui vektor masukan akan langsung diproses dan kemudian ditentukan target keluaran dari hasil pengolahan.



Gambar 2.8 Jaringan Perceptron

c. Backpropagation

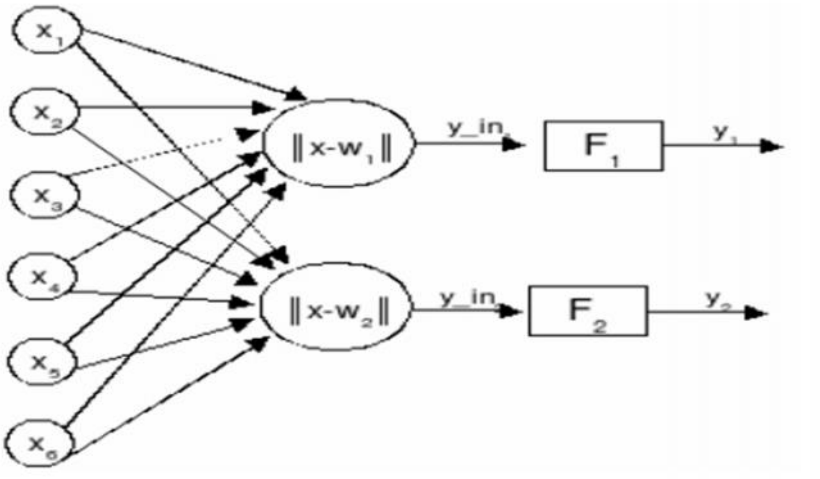
Salah satu arsitektur jaringan syaraf tiruan yang sangat populer adalah *multilayer feedforward networks.* Jaringan *backpropagation* terdiri dari sejumlah unit neuron sebagai lapisan masukan, satu atau lebih lapisan simpul-simpul neuron komputasi keluaran. Sinyal masukan di propagasikan ke arah depan (arah lapisan keluaran), lapisan demi lapisan.



Gambar 2.9 Jaringan *Backpropagation*

d. *Learning Vector Quantization* (*LVQ*)

LVQ merupakan jaringan syaraf tiruan 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 kompetitif ini hanya tergantung pada jarak antara vektor-vektor masukan. Jika 2 vektor masukan mendekati sama, maka lapisan kompetitif akan meletakan kedua vektor masukan tersebut kedalam kelas yang sama.



Gambar 2.10 Jaringan LVQ

Pada Gambar 2.10 memperlihatkan bahwa yang bertindak sebagai dendrit atau data masukan adalah x1-x6, yang bertindak sebagai sinapsis atau bobot adalah W, 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 LVQ :

Langkah 0 : Inisialisasi 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.

$$C_j = \sqrt{\sum_{i=1}^{n}(x_i - w_{ij})^2} \quad (1)$$

b. Perbaikan Wj dengan :

i. Jika T = Cj maka

$$Wj(baru) = Wj + \alpha\,[x - Wj(lama)] \quad (2)$$

ii. Jika T ≠ Cj maka

$$Wj(baru) = Wj - \alpha\,[x - Wj(lama)] \quad (3)$$

Langkah 3 : kurangi *learning rate*

$$\alpha = \alpha - pengurangan\,rasio \quad (4)$$

Langkah 4 : tes kondisi berhenti.

Keterangan rumus :

T : Target

J : Jumlah selisih data dan bobot

C : Kelas selisih bobot terkecil

W : Bobot

α : Rasio pembelajaran

x : Data

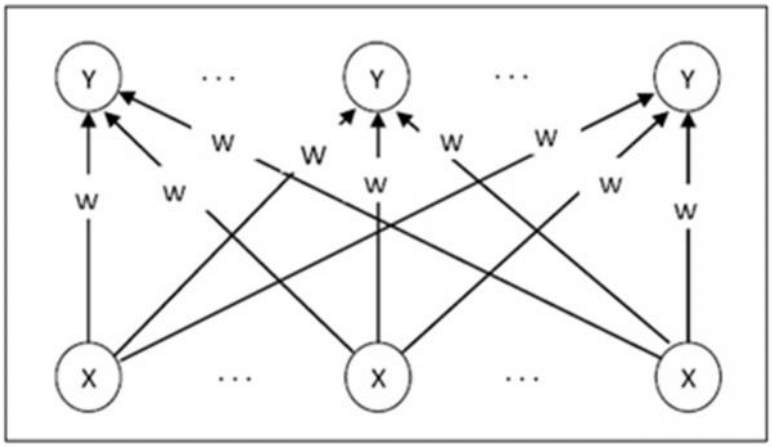
2.3.3.2 Struktur *recurrent* (*feedback*)

Jika suatu jaringan berulang (mempunyai koneksi kembali dari output ke input) akan menimbulkan ketidakstabilan dan akan menghasilkan dinamika yang sangat kompleks. Jaringan yang berulang sangat menarik untuk diteliti dalam jaringan syaraf tiruan, namun sejauh ini struktur *feedforward* sangat berguna untuk memecahkan masalah.

Metode-metode yang termasuk kedalam struktur *feedback* antara lain :

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 dengan pola masukan (memiliki jarak yang paling dekat) akan terpilih sebagai pemenang beserta neuron-neuron tetangganya akan memperbaiki bobotnya.

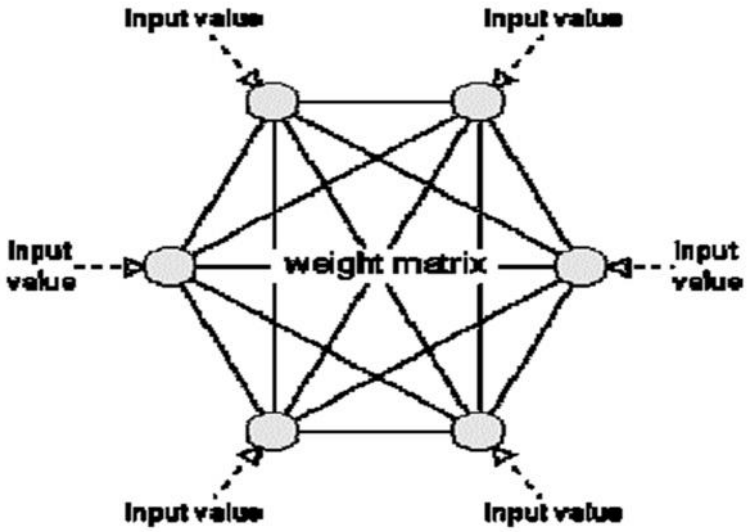


Gambar 2.11 Jaringan Kohonen

b. Jaringan Hopfield

Pada tahun 1982, John Hopfield dari *California Institute of Technology* merancang sebuah jaringan saraf tiruan yang kemudian dikenal dengan nama jaringan hopfield. Dalam jaringan hopfield, semua neuron saling berhubungan penuh. Neuron yang satu mengeluarkan output dan

kemudian menjadi input bagi semua neuron yang lain. Proses pengiriman dan penerimaan sinyal antar neuron ini secara *feedback* tertutup dan terus menerus sampai dicapai kondisi stabil.



Gambar 2.12 Jaringan Hopfield

2.4 Pengolahan Citra

Pengolahan citra merupakan suatu proses pengolahan data dan analisis citra yang banyak melibatkan persepsi visual. Proses ini mempunyai ciri data masukan dan informasi keluaran berbentuk citra. Istilah pengolahan citra digital umum didefinisikan sebagai pemrosesan citra dua dimensi dengan komputer. Dalam definisi yang lebih luas, pengolahan citra digital juga mencakup semua data dua dimensi. Citra digital adalah barisan bilangan nyata maupun kompleks yang diwakili oleh bit-bit tertentu (Ahmad, 2005).

Citra merupakan istilah lain dari gambar yang merupakan komponen multimedia yang memegang peranan sangat penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu kaya akan informasi. Citra digital adalah citra hasil digitalisasi citra kontinu (analog). Tujuan dibuatnya citra digital adalah agar citra tersebut dapat diolah menggunakan komputer atau piranti digital dan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (komputer). Teknik-teknik

pengolahan citra mentransformasikan citra menjadi citra lain yang mempunyai kualitas lebih baik.

Beberapa jenis operasi pengolahan citra adalah sebagai berikut :

1. Modifikasi Kecermelangan (*Brightness Modification*)

   Mengubah nilai keabuan/warna dari gelap menuju terang atau sebaliknya emngubah citra yang terlalu cemerlang/terang menajdi gelap.

2. Peningkatan Kontras (*Contrast Enchancement*)

   Dengan peningkatan kontras maka titik yang cenderung gelap menjadi lebih gelap dan yang cenderung terang menjadi lebih terang.

3. Negasi

   Operasi untuk mendapatkan citra negatif (*negative image*)

4. Pengabuan (*Grayscale*)

   Grayscale merupakan proses untuk mengubah warna gambar menjadi kabu-abuan. Proses *grayscale* dilakukan dengan mengubah nilai RGB setiap pixel gambar menjadi 1 nilai yang sama. Sehingga setiap pixel memiliki nilai yang sama untuk ketiga unsur warnanya. Salah satu prosentase yang digunakan dalam fungsi *image processing toolbox* adalah 29,9% dari warna merah (*Red*), 58,7% dari warna hijau (*Green*), dan 11,4% dari warna biru (*Blue*).

   Rumus untuk menghitung *grayscale* :

   $$grayscale = 0.3 * R + 0.59 * G + 0.11 * B \quad (5)$$

5. *Thresholding*

   Operasi ini digunakan untuk mengubah citra menjadi memiliki 2 buah nilai yaitu 255 dan 0. Tentukan ambang batas nilai RGB. Dalam hal ini diambil batas 128. Baca setiap pixel gambar, jika nilai rata-rata RGBnya lebih besar dari 128, maka matriks pada kordinat yang sama dengan pixel yang sedang dibaca diisi dengan nilai 255, dan jika nilainya lebih kecil atau sama dengan 128, matriksnya diisi dengan 0.

6. Ekstraksi Ciri

   operasi ini mengubah citra menjadi nilai biner yang memiliki 2 buah nilai yaitu 1 dan 0. Tentukan ambang batas nilai RGB. Dalam hal ini diambil

batas 128. Baca setiap pixel gambar, jika nilai rata-rata RGBnya lebih besar dari 128, maka matriks pada kordinat yang sama dengan pixel yang sedang dibaca diisi dengan nilai 1, dan jika nilainya lebih kecil atau sama dengan 128, matriksnya diisi dengan 0.

7. Pencerminan (*Flippng*)

   Pencerminan merupakan proses menggambar citra ke bentuk kebalikannya seperti ketika sedang bercermin.

8. Rotasi (*Rotating*)

   Rotasi yaitu proses memutar koordinat citra sesuai derajat yang ditentukannya.

9. Pemotongan (*Cropping*)

   Memotong saru bagian dari citra sesuai kebutuhan.

10. Pengskalaan (*Scaling*)

    Mengubah ukuran citra menjadi lebih besar atau lebih kecil.

11. Deteksi Tepi (*Edge Detection*)

    Deteksi tepi pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari objek-objek citra.

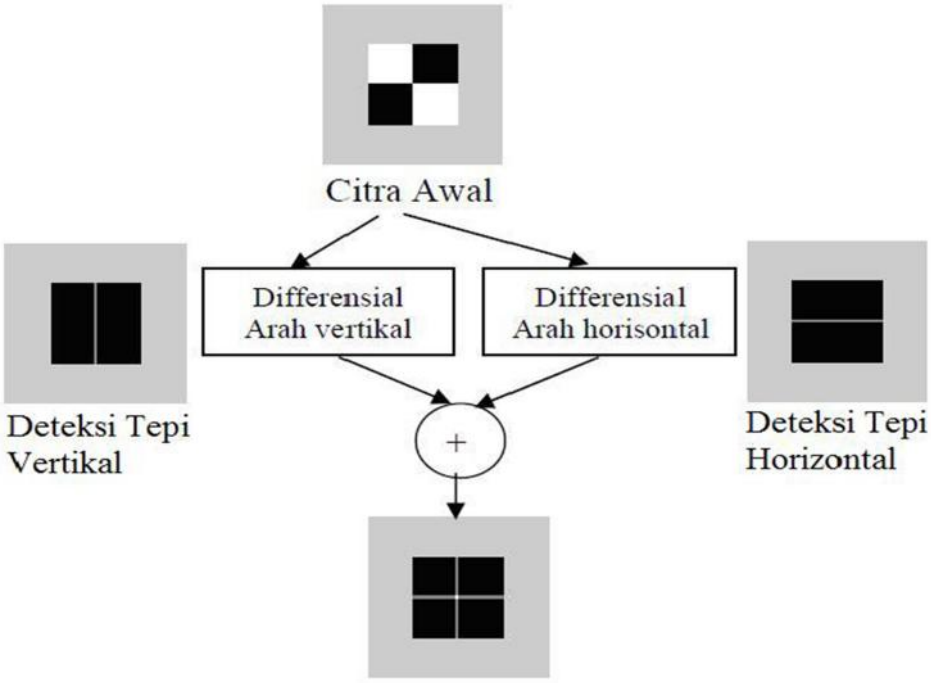
2.4.1 Deteksi Tepi

Tepi adalah perubahan nilai intensitas derajat keabuan yang mendadak (besar) dalam jarak yang sangat singkat. Perbedaan intensitas inilah yang memperlihatkan rincian pada gambar. Tepi dapat diorientasikan dengan suatu arah, dan arah ini berbeda-beda, tergantung pada perubahan intensitas. Deteksi tepi merupakan langkah pertama untuk melingkupi informasi didalam citra. Tepi mencirikan batas-batas objek dan karena itu tepi berguna untuk proses segmentasi dan identifikasi objek di dalam citra. Suatu titik (x,y) dikatakan sebgai tepi (*edge*) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya.

Deteksi tepi (*Edge Detection*) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari objek citra, tujuannya adalah :

a. Untuk menandai bagian yang menjadi detail citra.

b. Untuk memeperbaiki detail dari citra yang kabur, yang terjadi karena *error* atau adanya efek dari proses akuisisi citra.
c. Serta untuk mengubah citra 2D menjadi bentuk kurva



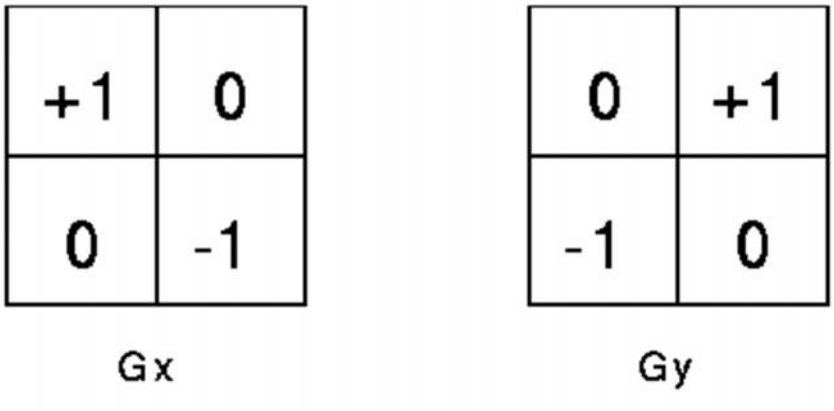
Gambar 2.13 Deteksi Tepi

Macam-macam metode untuk deteksi tepi antara lain :

1. Metode Robert

Metode pendeteksian tepi dengan mencari perbedaan (*differensial*) pada arah horisontal dan perbedaan pada arah vertikal, dengan ditambahkan proses 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 ayang 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 keluaran mutlak mewakili besarnya estimasi gardien spasial citra masukan pada saat itu. Operator robert terdiri

dari sepasang kernel 2x2 konvolusi. Satu kernel yang lain hanya diputar oleh 90̊.



Gambar 2.14 Matriks Metode Robert

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. Filter HPF menciptakan sebuah gambar dimana tepi (perubahan tajam nilai tingkat keabuan) yang akan ditampilkan. Hanya dilter dengan ukuran 3x3 yang dapat digunakan dengan filter ini. Filter ini menggunakan dua template 3x3 untuk menghitung nilai prewitt.

$s_x =$

-1	0	1
-1	0	1
-1	0	1

$s_y =$

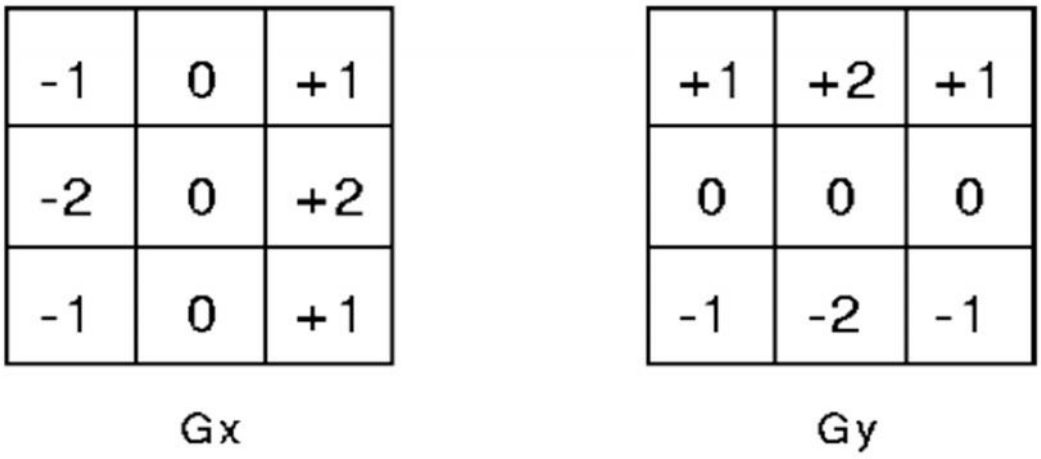
1	1	1
0	0	0
-1	-1	-1

Gambar 2.15 Matriks Deteksi Prewitt

3. Deteksi Tepi 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 metode sebelumnya. Operator sobel terdiri dari sepasang kernel 3x3 konvolusi seperti yang ditunjukan pada Gambar 2.16. satu kernel yang lain hanya diputar oleh 90̊. Deteksi tepi sobel adalah salah satu deteksi tepi yang menghindari adanya perhitungan gradien dititik interpolasi. Operator ini menggunakan kernel ukuran 3x3 piksel untuk perhitungan gradien sehingga perkiraan gradien berada tepat ditengah jendela.



Gambar 2.16 Matriks Deteksi Sobel

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 disetiap orientasi (menyebut Gx dan Gy). Perhitungan pada deteksi tepi ini, semua pixel dihitung kecuali pixel yang berada pada tepi gambar. Hal ini dilakukan karena didalam perhitungan nilai tepi sebuah pixel, harus menggunakan nilai pixel dari pixel-pixel disekitar pixel yang sedang dihitung, sedangkan pixel pada tepi gambar tidak memiliki beberapa pixel disekitarnya yang dibutuhkan dalam perhitungan.

Langkah perhitungan sobel adalah sebagai berikut :

a. Lakukan perhitungan pada pixel yang akan dihitung sisi x dan y nya. Masing-masing x dan y dimulai dari -1 hingga 1. Perulangan pada sisi

x diwakili oleh variabel wi dan pada sumbu y diwakili oleh variabel hw. Pada setiap perulangan dilakukan perhitungan :

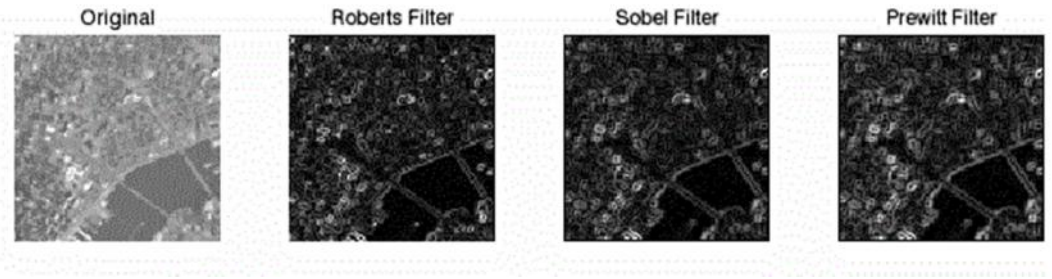
C = (p(j + wi,i + hw).R*0.3 + p(j + wi,i + hw).G*0.59 + p(j + wi,I + hw).B*0.11) (6)

$$Xbaru += gx(hw + 1, wi + 1) * C \quad (7)$$

$$Ybaru += gy(hw + 1, wi + 1) * C \quad (8)$$

b. Jika $\sqrt{Xbaru^2 + Ybaru^2}$ lebih besar dari 255, maka nilai RGB pixel pada titik yang sedang dihitung diubah menjadi 255, sedangkan jika hasilnya lebih kecil atau sama dengan dari 255 maka nilai RGB pixel yang sedang dihitung sesuai nilai hasil akhir xbaru dan ybaru.

c. Langkah a. dan b. diulang terhadap semua pixel kecuali pixel tepi gambar.



Gambar 2.17 Keluaran Hasil Metode Robert, Sobel , Prewitt

2.5 OOP (*Object Oriented Programming*)

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 sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek (Nugroho, 2005)

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat

mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan saat ini sangat beragam (aplikasi bisnis, real-time, utility, dan sebagainya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut :

a. Meningkatkan produktivitas

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

b. Kecepatan pengembangan

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

c. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering diubah-ubah.

d. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

e. Meningkatkan kualitas perangkat lunak

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

2.5.1 Konsep Dasar Berorientasi Objek

Berikut adalah konsep dasar Permrograman Berorientasi Objek :

a. Objek (*Object*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur,status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi(status) dan mempunyai operasi yang dapat diterapkan atau dapat berpengaruh pada status objeknya.

b. Kelas (*class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dalam kelas tersebut .

c. Pembungkusan (*Encapsulation*)

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

d. Pewarisan (*Inheritance*) dan Generalisasi/Spesialisasi

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

e. Metode

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural .

f. *Polimorfisme*

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

2.6 UML (*Unified Modeling Language*)

Model ini digunakan oleh para pengembang professional untuk berkomunikasi tentang proyek mereka kepada pembuat aplikasi mereka atau untuk pengembang aplikasi yang lain. UML telah menjadi bagian penting dari pengembangan perangkat lunak sejak diperkenalkan pada tahun 1997. UML telah

berkembang selama bertahun-tahun. Sebuah keuntungan penting dari UML adalah konsistensi dan bagian dari konsistensi tersebut adalah penerapan model umum elemen-elemen di diagram yang berbeda. UML yang digunakkan adalah UML 2.3 yang terdiri dari 13 macam diagram yang dikelompokan pada 3 kategori (Widodo, 2011), yaitu :

A. ***Structure Diagram***

Yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

Diagram-diagram yang termasuk kedalam tipe ini adalah sebagai berikut :

1. *Class Diagram*

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

2. *Object Diagram*

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

3. *Component Diagram*

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

4. *Composite Structure Diagram*

   Composite structure diagram baru mulai ada pada UML versi 2.0. Diagram ini dapat digunakan untuk menggambarkan struktur dari bagian-bagian yang saling terhubung maupun mendeskripsikan struktur pada saat berjalan (*runtime*).

5. *Package Diagram*

   Package diagram menyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam diagram UML. Hampir semua diagram dalam UML dapat dikelompokkan menggunakan *package diagram.*

6. *Deployment Diagram*

   Deployment menunjukan konfigurasi komponen dalam proses eksekusi aplikasi.

B. ***Behavior Diagram***

   Yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

   Diagram-diagram yang termasuk kedalam tipe ini adalah sebagai berikut :

1. *Use Case Diagram*

   Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

2. *Activity Diagram*

   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.

3. *State Machine Diagram*

   State machine diagram digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek.

C. ***Interactions Diagram***

   Yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi antar subsistem pada suatu sistem.

   Diagram-diagram yang termasuk kedalam tipe ini adalah sebagai berikut :

1. *Sequence Diagram*

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

2. *Communication Diagram*

   Communication Diagram menggambarkan interaksi antar objek/bagian dalam bentuk urutan pengiriman pesan. Diagram komunikasi merepresentasikan

informasi yang diperoleh dari diagram kelas, diagram sekuen, dan diagram *use case* untuk mendeskripsikan gabungan antara struktur statis dan tingkah laku dinamis dari suatu sistem.

3. *Timing Diagram*

   Timing diagram merupakan diagram yang fokus pada penggambaran terkait batasan waktu.

4. *Interaction overview diagram*

   Interaction overview diagram mirip dengan diagram aktivitas yang berfungsi untuk menggambarkan sekumpulan urutan aktivitas, diagram ini adalah bentuk aktivitas diagram yang setiap titik merepresentasikan diagram interaksi.

2.7 Visual Studio

Microsoft visual studio merupakan sebuah perangkat lunak (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, atau komponen aplikasinya, dalam bentuk *console*, aplikasi wondows, atau aplikasi web. Visual studio mencakup *compiler*, SDK, *Intergrated Development Environment* (IDE) dan dokumentasinya (umumnya berupa MSDN Library), *compiler* yang dimasukkan ke dalam paket visual studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual FoxPro, dan Visual CourceSafe.

Microsoft visual studio dapat digunakan untuk mengembangkan aplikasi *native code* (dalam bentuk bahasa mesin yang berjalan di atas wondows) atau *managed code* (dalam bentuk Microsoft Intermediete Language diatas .NET Framework). Selain itu visual studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan diatas .NET Compact Framework).

2.8 Bahasa Pemrograman C# (C Sharp)

C# (tanda ‘#’ dibaca “Sharp”) merupakan bahasa pemograman baru yang diciptakan Microsoft secara khusus sebagai salah satu bahasa pemrograman dalam

teknologi .Net sebagai bahasa baru, C# tidak berevolusi dari bahasa C# versi bukan teknologi .Net. dengan demikian C# dapat memaksimalkan kemampuannya tanpa khawatir dengan masalah kompatibilitas dengan versi-versi sebelumnya. Keharusan sebuah perangkat lunak untuk tetap dapat kompatibel dengan versi-versi sebelumnya sebagaimana yang terjadi pada Visual Basic (VB) maupun C++ biasanya menghambat optimalitas kemampuan dari perangkat lunak tersebut (Hartanto, 2008).

Sejak diluncurkan pada tahun 2000, C# dengan cepat merebut hati progammer C++ bahkan VB. Dengan tata cara penulisan yang mirip C++ dan *interface* mirip VB 6.0 menurut wikipedia, sebuah ensiklopedia gratis di internet pengguna C# .Net pada saat ini sudah melebihi pengguna VB.Net. sementara itu jumlah pengguna bahasa pemrograman lain masih berada dibawah jumlah pengguna VB .Net. masih menurut wikipedia, jumlah buku C# yang terjual pun berada dikisaran 2 hingga 3 kali lebih banyak dari jumlah buku VB yang terjual.

Dari informasi ini dapat disimpulkan bahwa C# merupakan bahasa pemrogrman baru yang sedang berkembang dan dapat diterima dengan baik oleh kebanyakan progammer dan kalangan industri. Di Microsoft sendiri, C# merupakan bahasa pemrograman yang digunakan untuk membuat perangkat lunak yang berteknologi .Net dengan demikian dapat diperkirakan bahwa C# akan menjadi bahasa pemrograman yang akan banyak digunakan di masa-masa mendatang.