**BAB II**

**TINJAUAN PUSTAKA**

* 1. **Tinjauan Studi**

Penelitian tentang penempatan lokasi telah banyak dilakukan oleh penulis terdahulu antara lain :

1. Penelitian yang dilakukan oleh Reni Resita (2016) tentang “Identifikasi buah jeruk menggunakan metode jaringan saraf tiruan berdasarkan tekstur kulit” Pada penelitian tersebut berisi Hasil pengujian yang diperoleh, JST jarak 8 cm memberikan 60% dengan batasan MSE 1e-10 dan JST 10 cm memberikan 56% dengan batasan MSE 1e-8 dan MSE 1e-10, sedangkan JST jarak 6 cm tidak memberikan dapat memberikan hasil. Dari 3 jenis JST didapatkan bahwa JST jarak 8 cm memberikan tingkat akurasi lebih tinggi pada identifikasi jenis buah jeruk berdasarkan tekstur kulit dibandingkan dengan JST jarak 6 cm dan 10 cm.
2. Penelitian yang dilakukan oleh Karsadi Warman (2015) tentang “Identifikasi Kematangan Buah Jeruk Berdasarkan Citra Warna RGB Dan Jaringan Saraf Tiruan”. Identifikasi jeruk yang diproses proses berdasarkan citra jeruk didapatkan hasil untuk latihan sebanyak 12 buah citra, presentase sistem untuk melakukan pengenalan 97,16% ini menunjukan sistem identifikasi yang di buat sudah berhasil mengenali objek data. Proses identifikasi dengan data latih untuk data citra jeruk sebanyak 28 buah citra keberhasilannya 89,28%. Persentase hasil identifikasi tersebut menunjukan bahwa sistem yang di buat menggunakan RGB dan Jaringan Saraf Tiruan sudah berhasil.
3. Penelitian yang dilakukan oleh Muhammad Asyhar Agmalaro (2013) tentang “Identifikasi Tanaman Buah Tropika Berdasarkan Tekstur Permukaan Daun Menggunakan Jaringan Saraf Tiruan” Pada penelitian terebut dijelakan metode ini sangat baik untuk diimplementasikan dalam melakukan klasifikasi permukaan daun tanaman buah tropika berdasarkan pola tektur dari tanaman buah tropika menggunakan JST. Akurasi tertinggi pengenalan buah tropika berdasarkan testur daun dengan menggunkan JST untuk semua hidden neuro yang telah dirata-ratakan pada buah Belimbing yaitu sebesar 94%, dan akurasi terendah terdapat pada pengujian terhadap tanaman buah Nangka dengan rata-rata akurasi setiap hidden neuronnya sebesar 66%. Secara keseluruhan, hasil pengujin utntuk seluruh tanaman buah topika dengan menggunakan hidden neuron sebanyak 7 menghailkan akurasi terbaik yaitu 90%. Penambahan jumlah hidden neuron setelahnya tidak meningkatkan akurasi JST dalam mengenali citra daun.
   1. **Tinjauan Pustaka**
      1. **Kesuburan Tanaman Padi Melaluai Warna Daun**

Padi merupakan tanaman dari daerah tropis dan subtropis. Tanaman padi terbagi atas bagian batang, dan daun; sedangkan bagian *generative* adalah mulai, gabah, dan bunga. Padi merupakan tanaman yang paling penting di negeri kita Indonesia ini. Betapa tidak karena makanan pokok di Indonesia adalah nasi dari beras yang tentunya dihasilkan oleh tanaman padi. Selain di Indonesia padi juga menjadi makanan pokok negara-negara di benua Asia lainnya seperti China, India, Thailand, Vietnam dan lain-lain (Balai Besar Penelitian Tanaman Padi).

* + 1. **Kecerdasan Buatan**

McCarthy mendefinisikan kecerdasan buatan atau *Artificial Intellegence* (AI) sebagai berikut.(*Artificial Intelegent*. Konsep dan penerapannya. Widodo Budiharto, Derwin Suhartono, 2014):

“AI merupakan cabang dari ilmu komputer yang berfokus pada pengembangan komputer untuk dapat memiliki kemampuan dan berperilaku seperti manusia.”

Manusia pandai dalam menyelesaikan permasalahan Karena manusia mempunyai pengetahuan dan pengalaman agar komputer bertindak seperti sebaik manusia, maka komputer juga harus di bekali bekal pengetahuan dan mempunyai kemampuan untuk menalar.

Dua bagian utama konsep kecerdasan buatan, yaitu (*Artificial Intellegence*. Konsep dan penerapannya Widodo Budiharto, Derwin Suhartono, 2014):

1. Basis pengetahuan. Merupakan represetasi pengetahuan yang diperlukan untuk memahami memformulasikan, dan memecahkan masalah. Terdiri dari dua elemen dasar yaitu:
2. Fakta berupa informasi tentang situasi permasalahan, teori dari area permasalahan atau informasi tentang objek.
3. *Spatial heuristic*, merupakan informasi tentang cara membangkitkan fakta.
4. Mesin Inferensi. Membuat *Inferensi* yang memutuskan aturan – aturan mana yang akan digunakan



**Gambar 2.1:** Konsep Kecerdasan Buatan

Adanya irisan penggunaan kecerdasan buatan di berbagai disiplin ilmu menyebabkan sulitnya pengklasifikasian berdasarkan disiplin ilmu. Oleh Karena itu, pengklasifikasian kecerdasan buatan dibuat berdasarkan keluaran yang dihasilkan. Ruang lingkup kecerdasan buatan adalah :

1. Sistem pakar (*expert system*) sistem pakar menggunakan penalaran dengan meniru atau mengadopsi keahlian yang dimili oleh pakar sehingga komputer dapat menyelesaikan permasalahan layaknya seseorang pakar.
2. Pengolahan Bahasa alami (*natural language processing*). Merupakan proses komputer yang melakukan ekstraksi informasi dari input yang berupa natural language dan /atau menghasilkan output yang juga berupa *natural language*.
3. Robotika dan sistem navigasi bidang ilmu yang mempelajari bagaimana membuat sistem robot yang dapat menyelesaikan tugas - tugas nya
4. Visi Komputer (*computer vision*). Cabang ilmu yang erat kaitannya dengan pembangunan arti atau makna dari image ke objek fisik. Hal yang di butuhkan adalah metode untuk memperoleh, melakukan proses menganalisa, dan memahami *image*.
5. Permainan (*game playing*). Konsep kecerdasan buatan dapat di terapkan pada permainan. Tujuan intinya adalah membuat *non player* memiliki strategi yang cerdas untuk mengalahkan *player*.
   * 1. **Komputer Visi**

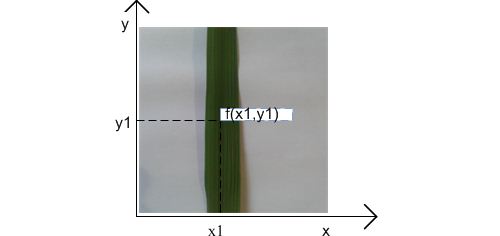
Salah satu pendekatan untuk mengimplementasikan sistem visi komputer adalah dengan berusaha meniru sistem visi manusia. Namun, permasalahan yang ada pada pendekatan ini adalah sistem visi manusia sangat kompleks dan sulit di mengerti. Tidaklah mungkin untuk dapat meniru sistem visi manusia secara sempurna, walaupun demikian studi terhadap sistem *biologis* memberikan petunjuk untuk membangun sistem visi komputer tentunya ini bukanlah hal yang mustahil.



**Gambar 2.2:** Sistem Komputer Visi

* + 1. **Pengolahan Citra Digital**

Citra adalah representasi informasi dua dimensi yang di ciptakan atau dibuat dengan melihat atau lebih tepatnya merasakan sebuah gambar atau pemandangan (Tabratas T., Onno W. Purbo, 2000). Dalam tinjauan matematis, citra merupakan fungsi kontinyu dari intensitas cahaya pada bidang dua dimensi. Ketika sumber cahaya menerangi objek, objek memantulkan kembali cahaya tersebut. Pantulan ini ditangkap oleh alat pengindra optik. Misalnya mata manusia, kamera, *scanner*, dan sebagainya. Bayangan objek tersebut akan terekam sesuai itensitas pantulan cahaya. Ketika alat optik yang merekam pantulan cahaya itu merupakan mesin digital, misalnya kamera digital, maka citra yang dihasilkan merupakan citra digital (Murni, 2004).



**Gambar 2.3:** Ilustrasi Citra Gambar

Sebuah citra digital dapat di anggap sebagai sebuah larik besar yang berisi sample *point* dari sebuah gambar kontinyu yang masing – masing *point* mempunyai kualitas kecerahan. Pengertian tersebut lebih di kenal dengan istilah *pixsel* yang sering digunakan pada citra digital.

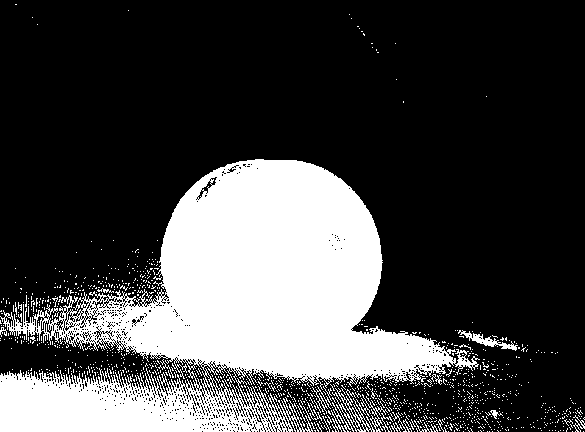
*Pixsel* mempuyai dua parameter, yaitu koordinat, dan intentitas atau warna, nilai yang terdapat pada koordinat *(x,y)* adalah *f(x,y),* yaitu besar intensitas atau warna dari titik piksel itu. Oleh sebab itu, sebuah citra digital dapat ditulis dalam bentuk matriks berikut.

Berdasarkan gambaran tersebut, secara matematis citra digital dapat dituliskan sebagai fungsi *f(x,y),* dimana harga x (baris) dan y (kolom) merupakan koordinat posisi dan *f(x,y)* adalah nilai fungsi pada setiap titik *(x,y)* yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari *pixel* di titik tersebut (sutoyo, et al, 2009).

Suatu *pixsel* memiliki nilai dalam rentang tertentu, dari nilai minimum hingga maksimum. Jangkauan yang digunakan berbeda - beda tergantung dari jenis warnanya. Namun secara umum jangkauannya adalah 0 (nol) sampai dengan 255. Citra dengan penggambaran seperti ini di golongkan kedalam citra integer. Citra digital secara umum dikelompokan menjadi 3 macam, yaitu :

* Citra Biner

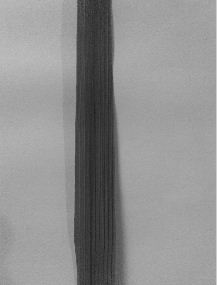
Dimana masing – masing *pixel* hanya berwarna hitam dan putih, maka hanya di perlukan satu bit per – *pixel*. Citra yang di representasikan sebagai citra biner sangat cocok digunakan untuk citra biner sangat cocok digunakan untuk citra yang menggambarkan *text, finger print*, atau rencana *arsitektural.*



**Gambar 2.4:** Citra Biner

* Citra abu – abu *( Grayscale ).*

Dimana masing – masing pixel berisikan warna abu – abu dengan nilai normal antara 0 (hitam) sampai 255 (putih). Range tersebut berarti masing – masing pixel dapat direpresentasikan oleh nilai 8 bit atau satu byte. Citra *Grayscale* dengan range nilai yang lain juga digunakan. Akan tetapi pada umumnya citra tersebut memiliki *range* pangkat dua dari 8 bit.

****

**Gambar 2.5:** Citra Skala keabuan (*Gray Scale*)

* Citra RGB atau True Color.

Untuk citra RGB masing – masing piksel mempunyai sebuah warna khusus. Warna dideskripsikan oleh kombinasi warna merah *(Red),* hijau *(Green)*, dan biru *(Blue)*. Jika masing - masing komponen mempunyai range antara 0-255. Maka total range yang digunakan untuk citra RGB adalah 155 pangkat 3 atau 16.777.216 kemungkinan warna. Range tersebut merupakan range yang mencukupi untuk membuat sebuah citra yang penuh dengan warna. Oleh Karena jumlah bit yang di butuhkan oleh masing – masing *pixsel* adalah 24 maka citra RGB biasa di sebut dengan 24-bit color image. Sebagai sebuah citra yang terdiri dari tumpukan *(stuck)* 3 matriks yang mewakili nilai merah, hijau, dan biru. Dengan kata lain untuk setiap piksel memiliki 3 buah nilai.

****

**Gambar 2.6:** Citra Warna (True Color)

Sedangkan pada citra digital terdapat beberapa elemen dasar pembentuk citra digital, diantaranya :

1. Kecerahan *(brightness)*

Kecerahan intesitas cahaya rata – rata dari suatu area yang melingkupinya.

1. Kontras *(contrast)*

Sebaran terang *(lightness)* dan sebaran gelap *(darkness)* di dalam suatu citra. Citra dengan kontras rendah komposisi citranya sebagian besar terang atau sebagian besar gelap. Citra dengan kontras yang baik , komposisi gelap dan terangnya tersebar merata.

1. Kontur *(Contour)*

Merupakan keadaan yang ditimbulkan oleh perubahan intensitas pada piksel – piksel tetangga, sehingga kita dapat mendeteksi tepi objek di dalam citra.

1. Warna *(Color)*

Adalah persepsi yang dirasakan sistem visual manusia terhadap panjang gelombang cahaya yang di pantulkan objek. Warna – warna yang dapat ditangkap oleh mata manusia merupakan kombinasi cahaya dengan panjang berbeda. Kombinasi yang memberikan rentang warna paling lebar adalah *Red* (R), *Green* (G), *Blue* (B).

1. Bentuk *(shape)*

Adalah properti intrinsic dari objek 3 (tiga) dimensi, dengan pengertian bahwa bentuk merupakan property intrinsik utama untuk visual manusia. Umumnya citra yang di bentuk oleh manusia merupakan 2D, sedangkan objek yang dilihat adalah 3D.

1. Tekstur *(texture)*

Walaupun ada definisi secara formal tentang tekstur, secara intuitif tekstur menyatakan ciri dari permukaan objek yang menggambarkan pola visual. Ciri ini berisi informasi tentang komposisi struktur permukaan, seperi misalnya awan, daun, batu bata, dan kain. Selain itu juga menjelaskan hubungan antara permukaan untuk lingkungan sekitarnya (Al-Teyeche, 2003). Selain itu menurut (Kusao, 2009) Tekstur adalah sifat – sifat atau karakteristik yang memiliki oleh suatu daerah yang cukup besar sehingga secara alami sifat tersebut dapat berulang dalam daerah tersebut. Sehingga tekstur menjadi salah satu fitur yang penting. Ciri tekstur antara lain meliputi kehalusan *(smoothless)*, kekerasan *(coarseness),* dan keteraturan *(regularity)*.

Penggunaan fitur tekstur telah banyak digunakan secara luas oleh peneliti dalam menyelesaikan masalah pengenalan pola *(patern recognition)* dan *computer vision.* Secara umum, representasi tekstur dapat diklasifikasikan menjadi dua, yaitu : structural dan statistic (Gonzales, et al, 2002).

Untuk representasi tekstur secara statistic, dilakukan denganmenganalisa distribusi statistik dari intensitas citra termasuk diantaranya adalah teknik *multiresolution filtering seperti Glem Filter*.

Meskipun sebuah citra kaya akan informasi, namun seringkali citra yang kita miliki mengalami penurunan mutu (*degradasi*), misalnya mengandung cacat atau noise, warnanya terlalu kontras, kurang tajam, kabur *(blur)*, dan sebagainya tentu saja citra semacam ini menjadi lebih sulit diinterpretasi Karena informasi yang disampaikan oleh citra tersebut menjadi berkurang (Munir, 2004).

Agar citra yang mengalami gangguan mudah di interpretasi, maka citra tersebut perlu di manipulasi menjadi citra lain yang memiliki kualitas lebih baik. Bidang studi yang menyangkut hal ini adalah pengolahan citra *(image prosessing)* (Munir, 2004).

Konsep dasar pemrosesan suatu objek pada gambar menggunakan pengolahan citra diambil dari kemampuan indra pengelihatan manusia yang selanjutnya di hubungkan dengan kemampuan otak manusia. Dalam sejarahnya pengolahan citra telah di aplikasikan dalam berbagai bentuk, dengan tinggkat kesuksesan yang cukup besar. Seperti berbagai cabang ilmu lainnya, pengolahan citra menyangkut pula berbagai cabang ilmu lainnya, pengolahan citra menyangkut pula berbagai gabungan cabang – cabang ilmu. Diantaranya adalah optic, *electronic*, matematika, *fotografi,* dan teknologi computer (Gonzalez, 2002).

*Image prosessing* (pengolahan citra) mencakup dua aspek proses pengubah sebuah citra seperti sebagai berikut :

1. Meningkatkan kualitas informasi dari sebuah citra (gambar) yang digunakan untuk kepentingan interpretasi manusia.
2. Mengubah citra dari sebuah gambar yang digunakan untuk mempermudah pemrosesan persepsi mesin *autonomous* agar lebih mudah dalam mengambil keputusan.

Pada umumnya, objektifitas dari pengolahan citra adalah mentransformasi atau menganalisa suatu gambar sehingga informasi baru tentang gambar dibuat lebih jelas, Citra dalam pengolahannya dibagi menjadi tiga *level* proses, yaitu:

1. *Low\_level process* : proses-proses yang berhubungan dengan operasi primitif seperti image pre-processing untuk mengurangi *noise, input* dan *output-nya* berupa gambar.
2. *Mid-level process* : proses-proses yang berhubungan dengan tugas-tugas seperti segmentasi gambar (membagi gambar menjadi objek-objek), pengenalan *(recoqnition)* suatu objek individu. Pada *mid-level process, input* pada umumnya berupa gambar tetapi *output-nya* berupa atribut yang dihasilkan dari proses yang dilakukan gambar tersebut seperti garis, garis *contour*, dan objek-objek individu.
3. *High-level process* : proses-proses yang berhubungan dengan hasil dari *mid-level proces* (Gonzalez, 2002).

Umumnya tahap-tahapan dalam pemrosesan citra untuk mendapatkan informasi, yaitu :

* Menentukan citra. Pertama kali harus menentukan objek citra yang akan diolah. Untuk kepentingan ini dapat menggunakan kamera atau scanner.
* Pemrosesan awal. Pemrosesan awal dijalakan sebelum proses utama dijalankan, bagian penting pada tahap ini adalah untuk menjalankan beberapa operasi dasar untuk menghasilkan gambar akhir sehingga lebih mudah dikerjakan pada tahap selanjutnya, Pada kasus ini contoh operasi yang biasa dilakukan adalah mengkonversi citra ke skala keabuan *(gray scalling)*, meningkatkan kontras, dan lainnya.
* Representasikan dan deskripsi. Pada tahap ini merujuk pada fitur khusus yang memungkinkan pengguna untuk membedakan objek.
* Pengenalan, klasifikasi, dan interpretasi, pada tahap ini telah didapatkan kode pos dan merepresentasikannya pada sebuah string.
  + 1. **Akuisisi Citra**

Tujuan akuisisi citra adalah menentukan data yang diperlukan dan memilih metode perekaman *digital*, Tahap ini dimulai dari objek yang akan diambil gambarnya, persiapan alat-alat sampai pada percitraan. Pencitraan adalah kegiatan transformasi dari citra tempak menjadi citra digital. Beberapa alat yang dapat digunakan seperti : kamera *digital, scanner*, foto sinar-x atau sinar infra merah, dan alat penangkapan bayang lainnya. Hasil dari akuisisi citra ini ditentukan oleh kemampuan *sensor* atau alat untuk mendigitalisasikan sinyal yang terkumpul pada sensor atau alat tersebut.

* + 1. **Pra-Pengolahan Data Citra**

Tehnik pra-pemrosesan digunakan untuk mempersiapkan citra agar dapat menghasilkan ciri yang lebih baik pada pada tahap pemisahan cirri (ekstraksi ciri) terhadap proses pengenalan pola. Pra-pemrosesan adalah pengubahan data mentah untuk membantu kemampuan komputasional dan pencari ciri. Pada tahap pra-pemrosesan, kita cukup mengkaji metode atau pendekatan fungsi yang tepat untuk melakukan pemrosesan awal (pra-pemrosesan).

* + 1. **Gray Scalling**

*Gray scalling* adalah tehnik yang digunakan untuk mengubah citra berwarna menjadi bentuk tingkat aras keabuan. Jumlah warna citra *grey* adalah 256, karena citra *grey* jumlah *bit*-nya adalah 8, sehingga jumlah warnanya adalah =256, nilainya berada pada jangkauan 0-255. Pengubahan citra warna ke citra skala keabuan dapat menggunakan beberapa algoritma diantaranya : *averaging, Luminosity, Desaturation,* citra minimal dan *Maximal decamposition, single color channel.*

Metode *Luminosity* ini menggunakan perhitungan nilai berdasarkan tingkat luminositas. Metode ini juga bekerja dengan cara merata-ratakan nilai, tetapi metode ini membentuk rata-rata dengan bobot tertentu berdasarkan persepsi mata manusia. Mata manusia lebih sensitif terhadap warna hijau, oleh karena itu warna hijau diberi bobot yang lebih besar dari pada warna lainnya, Metode *luminosity* yang bekerja menggunakan persepsi mata manusia menghasilkan citra yang paling baik dari beberapa metode yang digunakan untuk konversi citra ke skala keabuan.

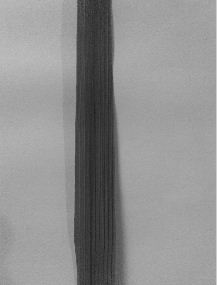
Adapun beberapa persamaan dari metode luminosity sebagai berikut :

P1= (0,21R)+(0.71G)+(0.07B)

P2=(0.2125R)+(0.7154G)+(0.0721B)

P3=(0.50R)+(0.419G)+(0.081B)

P4=(0.229R)+(0.587G)+(0.114B)



**Gambar 2.7:** Hasil mengubah citra RGB ke citra grayscale. (a) Citra RGB/Asli, dan (b) Citra Grayscale/hasil pengubahan

* + 1. **Ekualisasi histogram *(histogram aqualization)***

Metode ekualisasi histogram merupakan teknik penyesuaian nilai piksel sehingga menghasilkan citra dengan kontras yang lebih baik. Metode ini menggunakan perataan histogram agar derajat keabuan dari yang paling rendah (nol) hingga ke yang paling tinggi (255) memiliki kemunculan yang rata. Dengan ekualisasi histogram, hasil gambar yang memiliki histogram yang tidak merata atau distributif kualitatif yang memiliki banyak loncatan gradasinya akan menjadi gambar yang lebih jelas karena derajat keabuan tidak domain gelap atau domain terang.

**2.2.9 Ekstraksi fitur *(festure extraction).***

Ekstraksi fitur (ciri) adalah proses untuk mendapatkan ciri-ciri pembela yang membedakan suatu sampel dari sampel lainnya. Bagi sebagian besar aplikasi pengenalan pola *(pattern recognition),* teknik ekstraksi fitur yang handal merupakan kunci utama dalam penyelesaian masalah pengenalan pola. Dengan adanya ekstraksi fiur, informasi penting yang berada dalam citra tersebut dapat diambil dan disimpan kedalam vektor fitur *(feature vektor)*.

Fitur-fitur yang dapat diekstrak pada citra dapat berdasarkan elemen warna, bentuk, tekstur, kecerahan, kontur , dan kontras dengan menggunakan teknik ekstraksi fitur tertentu.

Tekstur merupakan karakteristik instrinsik dari suatu citra yang terkait dengan tingkat kekerasan *(reughness)*, granularitas *(gramulation)*, dan keteraturan *(regularity)* susunan struktur *pixsel* : Tekstur dicirikan sebagai distribusi spasial dari derajat keabuan didalam sekumpulan *pixsel-pixsel* yang bertetangga. Jadi tekstur, tidak dapat diindefikasikan untuk sebuah piksel, melainkan suatu citra dianggap sebagai suatu kesatuan.

Informasi tekstur dapat digunakan untuk membedakan sifat permukaan tersebut, sehingga analisis tekstur lazim dimanfaatkan sebagai proses antara untuk melakukan klasifikasi dan interpretasi citra.

Ekstraksi fitur citra berdasarkan tekstur pada orde pertama dapat menggunakan mode statik, yaitu dengan melihat statistik distribusi derajat keabuan pada histogram citra tersebut (Wong & Zrimec, 2006). Dalam penelitian ini nilai-nilai pada histogram yang dihasilkan dihitung dari parameter ciri (fitur) *Entropy*. *Entropy* (H) menunjukan ukuran tingkat keacakan (ketidakteraturan) bentuk dari suatu citra. *Entropy* dirumuskan sebagai berikut:

Dimana :

*fn*=Nilai intensitas keabuan

*p (fn)*= Nilai histogram

* + 1. **Gray Level Co-Occurrence Matrix**

Fitur Ekstraksi *Grey Level Co-occurrence Matrix* (GLCM) merupakan matriks yang menggambarkan frekuensi munculnya pasangan dua piksel dengan intensitas tertentu dalam jarak d dan orientasi arah sudut tertentu dalam citra, jarak dinyatakan dalam *pixsel* dan orientasi sudut dinyatakan dalam derajat

**Table 2.1:** Contoh Matrix Asal

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

Hubungan spesial untuk =0o dan d=1 pada matriks diatas dapat dituliskan dalam matriks:

**Tabel 2.2:** *Matrix Co-Occurrence* dari *Matrix* Asal

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

Matriks *Co-occurrence* yang dapat kemudian di tambahkan dengan matriks transposenya untuk menjadikanya simetris terhadap sumbu diagonal

**Tabel 2.3:** GLCM Simetris Ternormalisasi Dari Matriks Asal

|  |  |  |  |
| --- | --- | --- | --- |
| 0.250 | 0 | 0.083 | 0 |
| 0 | 0.167 | 0.083 | 0 |
| 0.083 | 0.083 | 0.083 | 0.083 |
| 0 | 0 | 0.083 | 0 |

Langkah – langkah membuat GLCM simetris ternormalisasi adalah sebagai berikut :

* Membuat Framework Matriks
* Menentukan hubungan spasial antara piksel refrensi dengan piksel tetangga, berupa sudut dan jarak.
* Menghitung jumlah co-occurrence dan mengisikanya pada framework.
* Menjumlahkan matriks Co-occurrence transposnya untuk menjadikannya simetris.
* Normalisasi matriks untuk mengubahnya ke bentuk probabilitas

Dalam penelitian ini Ciri – ciri fitur di ekstraksi GLCM yang di gunakan berjumlah enam, yaitu:

* *Angular second moment*

Menunjukan sifat kehomogenitas citra.

* *Contrast*

Menunjukan ukuran penyebaran elemen – elemen matriks citra.

* *Corelation*

Menunjukan ukuran ketergantungan linear derajat keabuan citra sehingga dapat memberikan petunjuk adanya struktur linear dalam citra.

* *Variance*

Menunjukan variasi elemen – elemen matriks kookurensi. Citra dengan transisi derajat keabuan kecil kan memiliki variasi yang kecil pula.

* *Inverse Difference Moment*

Menunjukan kehomogenan citra yang berderajat keabuan sejenis citra homogeny memiliki harga IDM yang besar.

* *Entrophy*

Menunjukan ukuran ketidak teraturan bentuk, harga ENT besar untuk citra dengan transisi derajat keabuan merata dan bernilai kecil jika struktur citra dapat diatur.

**2.2.11 Classification**

Klasifikasi adalah penemuan sebuah model (atau fungsi) yang menggambarkan dan membedakan kelas data atau konsep yang bertujuan agar bias digunakan untuk memprediksi kelas dari objek yang label kelasnya (Han, 2006). Sedangkan klasifikasi menurut serapannya yang berasal dari belanda, classificatie, yang sendirinya berasal dari Bahasa perancis *classification*. Istilah ini menunjuk kepada sebuah metode untuk menyusun data secara sistematis atau menurut beberapa aturan atau kaidah yang telah di tetapkan.

Kelasifikasi data terdiri dari dua tahap proses. Pertama adalah Learning (fase training), dimana algoritma klasifikasi dibuat untuk menganalisa data training lalu diintrepresentasikan dalam bentuk rule klasifikasi. Proses kedua adalah klasifikasi, dimana data tes digunanakan untuk memperkirakan akurasi dari rule klasifikasi (Han, 2006).

Model dalam klasifikasi sama artinya dengan suatu model yang menerima masukan kemudian mampu melakukan pemikiran terhadap masukan dan memberikan jawaban sebagai keluaran dari hasil pemikiran terhadap masuk dan memberikan jawaban sebagai keluaran hasil pemikiran, Proses klasifikasi didasarkan pada empat komponen (Gurunescu, 2011) :

1. Kelas adalah variable dependen yang berupa kategorikal yang mempresentasikan “label” yang terdapat pada objek.
2. Predictor, merupakan variable independen yang direpresentasikan oleh karakteristik (attribute) data.
3. Training dataset, Satu set data yang berisi nilai dari dua komponan diatas yang digunakan untuk menentukan kelas yang cocok berdasarkan predictor.
4. Testing Dataset, Berisi data baru yang akan di klasifikasikan oleh model yang telah di buat dan akurasi klasifikasi dievaluasi.

**2.2.12 ANN – Backpropagation**

*Artificial Neural Network* (ANN) adalah prosessor yang terdistribusi paralel, terbuat dari unit – unit yang sederhana, dan memiliki kemampuan untuk menyimpan pengetahuan yang di peroleh secara eksperimental dan siap pakai untuk berbagai tujuan (Sutujo, Mulyanto, & S, 2011). ANN ini meniru otak manusia dari sudut : (1) Pengetahuan diperoleh oleh *network* dari lingkungan, melalui suatu proses pembelajaran; (2) kekuatan koneksi antar unit yang disebut *synaptic weights*, berfungsi untuk menyimpan pengetahuan yang telah di peroleh oleh *network* tersebut. Pada tahun 1943. Mc. Culloch dan Pitts memperkenalkan model matematika yang merupakan penyerhanaan dari struktur sel saraf yang sebenarnya (Sutojo, Mulyanto, & S, 2011).

Korelasi antara ketiga komponen pada persamaan di atas yaitu : Signal x berupa *vector* berdimensi n (x1,x2,…..,xn)T akan mengalami penguatan oleh *synapse* w (w1, w2,…,wn)T. Selanjutnya akumulasi dari penguatan tersebut akan mengalami trasnsformasi oleh fungsi aktifasi f. fungsi f ini akan memonitor, bila akumulasi penguatan signal itu telah melebihi batas tertentu, maka sel neuron yang berada dalam kondisi “0”, akan mengeluarkan signal “1”. Berdasarkan nilai output tersebut (y), sebuah neuron dapat berada dalam dua status : “0” atau “1”. Neuron disebut dalam kondisi firing bila menghasilkan output bernilai “1”.



(Sutojo, Mulyanto, & S, 2011):

**Gambar 2.8:**  *Artificial Neural Network* (ANN**)**

Gambar di atas memperlihatkan bahwa sebuah neuron memiliki tiga komponen:

1. Synapse (w1, w2,…,wn) T;
2. Alat penambah (adder);
3. Fungsi aktifasi (f).

Sebuah neural network dapat dianalisa dari dua sisi:

1. Bagaimana neuron – neuron tersebut dirangkaikan dalam suatu jaringan (arsitektur);
2. Bagaimana jaringan tersebut dilatih agar memberikan output sesuai dengan yang dikehendaki (algoritma pembelajaran).

Algoritma pembelajaran ini menentukan cara bagaimana nilai penguatan yang optimal di peroleh secara otomatis. Berdasarkan arsitekturnya, neural network dapat dikategorikan, antara lain, *single layer neural network, multiplayer neural network, dan recurrent neural network*. Berbagai algoritma pembelajaran antara lain *Hebb’s law, delta rule, Backpropagadation algorithm,* dan *self Organizing Feature Map*. Berawal dari perkenalannya model matematika nauro oleh Mc. Culloch dan Pitts. Penelitian di bidang neural network berkembang cukup pesat, dan mencapai puncak keemasan pertama pada era tahun 60, dan puncak kedua pada pertengahan tahun 80-an. (Sutojo, Mulyanto, & S, 2011)

Dewasa ini*, neural network* telah di publikasikan di berbagai bidang. Hal ini dikarenakan *neural network* memiliki kelebihan – kelebihan sebagai berikut:

1. Dapat memecahkan problema non-linear yang umum di jumpai di aplikasi;
2. Kemampuan memberikan jawaban terhadap pattern yang belum pernah di pelajari *(generalization);*

Dapat secara otomatis mempelajari data numerik yang diajarkan pada jaringan tersebut.

Salah satu metode pelatihan terawasi pada neural network adalah metode backpropogation, dimana ciri dari metode ini adalah meminimalkan error pada output yang dihasilkan oleh jaringan. Pada gambar di bawah ini, unit input dilambangkan dengan X, hidden unit dilambangkan dengan Z, dan unit output dilambangkan dengan Y. bobo tantara X dan Z dilambangkan dengan V sedangkan bobo tantara Zdan Y dilambangkan sengan W. (Sutojo, Mulyanto, & S, 2011)



(Sutojo, Mulyanto, & S, 2011):

**Gambar 2.9:** Arsitektur Jaringan Backpropagation

Penerapan backpropagation neural network terdiri dari 2 tahap (Sutojo, Mulyanto, & S, 2011):

1. Tahap pelatihan, di mana pada tahapan ini diberikan sejumlah data pelatihan dan target;
2. Tahapan pengujian atau evaluasi, dilakukan setelah selesai tahap penelitian.

Pada intinya, pelatihan dengan metode backpropagation terdiri dari tiga langkah, yaitu (Sutojo, Mulyanto, & S, 2011):

1. Data dimasukan ke input jaringan *(feedforward)*;
2. Perhitungan dalam propagandasi balik dari error yang bersangkutan;
3. Pembaharuan (adjustment) bobot dan bias.

Saat umpan maju *(feedforward)*, setiap unit input 9Xi) akan menerima sinyal input dan akan menyebarkan sinyal tersebut pada tiap hiden unit (), setiap hidden unit kemudian akan menghitung aktivasinya dalam mengirimkan sinyal () ketiap unit output. Kemudian setiap unit output () juga akan menghitung aktivasinya () untuk menghasilkan respons terhadap input yang diberikan jaringan. Saat proses pelatihan (training), setiap unit output membandingkan aktivasinya () dengan nilai target () untuk menentukan besarnya error. Berdasarkan error dihitung factor k, dimana factor inidigunakan untuk mendistribusikan error dari output ke layer sebelumnya. Dengan cara yang sama, factor •j juga dihitung pada hidden unit Zj, di mana factor ini digunakan untuk memperbaharui bobo tantara hidden layer dan input layer. Setelah semua factor ditentukan, bobot untuk semua layer di perbaharui. Notasi yang digunakan dalam algoritma pelatihan (Sutojo, Mulyanto, & S, 2011):

= Data *training input* x= (z1,…,xi,…,xn)

= Data *training* untuk target *output* t=(t1,…,tk,…,tm)

= *Learning rate* yaitu parameter untuk mengontrol perubahan bobot selama pelatihan.

= Unit *input* ke – i

= *Hidden* unit ke – j

= Unit *Outpu*t ke -k

V0j = bias *hidden* unit ke-j

Vij = bobot antara unit *input* ke-I dengan *hidden* unit ke-j

W0k = bias untuk unit *output* ke-k

Wjk = bobot antara *hidden* unit ke-j dengan unit *output* ke-k

δk = factor koreksi *error* untuk bobot Wjk

δj = factor koreksi *error* untuk bobot Vij

= momentum

Tahap – tahap pelatihan (Sutojo, Mulyanto, & S, 2011):

1. Tahap 0: Inisialisasi bobot dan bias. Baik bobot maupun bias dapat siset dengan sembarangan angka (acak) dan biasanya angka sekitar 0 dan 1 atau -1 (bias *positif* atau *negative*).
2. Tahap 1: Jika *stooping condition* masih belum terpenuhi, jalankan tahap 2-9
3. Tahap 2: Untuk setiap data training, lakukan tahap 3-8
4. Tahap 3: Setiap unit input (Xi,i=1,…,n) menerima sinyal input xi dan menyebarkan sinyal tersebut pada seluruh unit *hidden layer*. Perlu diketahui bahwa *input* xi yang dipakai di sini adalah *input training* data yang sudah di sahkan.
5. Tahap 4: Setiap hidden unit (Zj,j-1,…,p) akan menjumlahkan sinyal – sinyal input yang sudah berbobot, trmasuk biasnya

Dan memakai fungsi aktivasi yang telah di tentukan untuk menghitung sinyal *output* dari *hidden* unit yang bersangkutan.

Lalu mengirim sinyal output ini ke seluruh unit pada unit *output*

1. Tahap 5: Setiap unit output (Yk,K=1,…,m) akan menjumlahkan sinyal – sinyal input yang sudah berbobot termasuk biasnya,

Dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dari unit *output* yang bersangkutan

1. Tahap 6: *Propagasi* balik *error (backpropagation of error).* Setiap unit output (Yk,K=1,…,m) menerima suatu target (output yang diharapkan) yang akan dibandingkan dengan output yang dihasilkan.

Faktor • k ini digunakan untuk menghitung kereksi *error* (•Wjk) yang nantinya akan dipakai untuk memperbaharui Wjk, Dimana:

Selain itu juga dihitung kereksi bias •W0k yang nantinya akan di pakai untuk memperbaharui Wjk, di mana:

Faktor δk ini kemudian di kirim ke layer di depannya.

1. Tahap 7: Setiap hidden unit (Zj,j=1,…,p) menjumlahkan input delta (yang dikirim dari layer pada tahap 6 yang sudah berbobot.

Kemudian hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan factor koreksi error • j, di mana:

Faktor • j ini digunakan untuk menghitung kereksi error (• vij) yang nantinya akan dipakai untuk memperbaharui vij, dimana:

Selain itu juga dihitung koreksi bias • Voj yang nantinya akan di pakai untuk memperbaharui Voj, di mana:

1. Tahap 8: Pemperbaharuan bobot dan bias: setiap unit output (Yk,k=1,…,m) akan memperbaharui bias dan bobotnya dengan setiap hidden unit.
2. Demikian pula untuk setiap hidden unit akan memperbaharui bias dan bobotnya dengan setiap unit input.
3. Tahap 9: Memeriksa *stopping condition*, jika stop condition telah di perbaharui, maka pelatihan jaringan dapat dihentikan. Untuk menentukan stopping condition terdapat dua acara yang bias di pakai, yaitu: (1) membatasi iterasi yang ingin di lakukan. Misalnya jaringan akan di latih sampai iterasi yang ke-500. Yang di maksud dengan satu iterasi adalah perulangan tahap 3 sampai tahap 8 untuk semua training data yang ada; (2) membatasi error.

Misalnya menentukan besar antara output yang dikehendaki dan output yang di hasilkan oleh jaringan. Jika terdapat banyak m training data, maka untuk menghitung mean square error digunakan persamaan berikut:

MSE = 0,5 {(tk1 – yk1)2 + (tk2 – yk2)2+…+(tkm – ykm)2}

Setelah pelatihan selesai, *backpropagation network* (BPN) dianggap telah pintar sehingga apabila jaringan diberi input tertentu, jaringan akan menghasilkan output seperti yang diharapkan. Cara mendapatkan output tersebut adalah dengan mengimplementasikan *backpropagation* yang sama seperti proses pelatihan, tetapi hanya pada bagian umpan majunya saja. Notasi yang digunakan dalam algoritma pengujian.

= Unit input ke-i

= Hidden Unit ke-j

= Unit output ke-k

V0j = Bias untuk hidden unit ke-j

Vij = bobo tantara unit input ke – I dengan hidden unit ke-j

W0k = Bias untuk unit output ke-k

Wjk = Bobo tantara hidden unit ke-j dengan unit output ke-k

Tahap-tahap pengujian pada model ANN, sbb:

1. Tahap 0: Inisialisasi bobot sesuai dengan bobot yang telah dihasilkan pada proses pelatihan diatas;
2. Tahap 1: Untuk setiap input, lakukan tahap 2-4;
3. Tahap 2: Untuk setiap input i=1,…,n skalakan bilangan dalam range fungsi aktivasi seperti yang dilakukan pada proses pelatihan di atas
4. Tahap 3: untuk j=1,…,p:
5. Tahap 4: untuk k=1,…,m:

Variable adalah output yang masih dalam skala menurut range fungsi aktivasi. Untuk mendapatkan nilai output yang sesungguhnya, harus dikembalikan seperti semula.

Contoh:

Buatlah jaringan saraf tiruan untuk fungsi logika XOR-2 input sesuai table berikut.

**Tabel 2.4:**  XOR-2 Input

|  |  |  |
| --- | --- | --- |
| X1 | X2 | Target |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

* 1 input layer, dengan 2 neuron (X1 dan X2)
* 1 hidden layermdengan 4 neuron (Z1, Z2, Z3, Dan Z4) Gunakan fungsi aktivasi sigmoid
* 1 output layer dengan 1 neuron Gunakan fungsi aktivasi sigmoid



**Gambar 2.10:** Arsitektur ANN. Fungsi XOR-2 input

Jawab:

Inisialisasi bobot

Nilai-nilai bobot dan bias awal di tentukan sebagai berikut:

Bobot awal input ke *hidden layer*:

Bobo awal hidden layer ke output layer

Bobo awal hidden layer ke output layer

Bobot awal bias ke output layer.

Untuk kebutuhan pelatihan jaringan di tentukan pula.

tearning rate () = 1

Maksimum Epoch = 1500

Target Error =0,02

Pelatihan jaringan :

Epoch ke -1:

----------------------------------------------------------

Data ke = 1 (X1=0,X2=0, target T=0)

Tahap Perambanan Maju (*Forward propagation*)

Operasi pada hidden layer Layer:

Z\_in1= V01 + V11 \* X1 + V21 \* X2

= 0,1 + 0,1 \* 0 + 0,1 \* 0

= 0,1

Z\_in2= V02 + V12 \* X1 + V22 \* X2

= 0,2+ 0,2 \* 0 + 0,2 \* 0

= 0,2

Z\_in3= V03 + V13 \* X1 + V23 \* X2

= 0,3 + 0,3 \* 0 + 0,3 \* 0

= 0,3

Z\_in4= V04 + V14 \* X1 + V24 \* X2

= 0,4 + 0,4 \* 0 + 0,4 \* 0

= 0,4

Fungsi aktivasi pada hidden layer :

Operasi pada output layer :

Y\_in = W0 + W1 \*Z1 + W2 \* Z2 + W3 \* Z3 + W4 \*Z4

= 0,5 + 0,1 \*0,5250 + 0,2 \* 0,5498 + 0,3 \* 0,5744 + 0,4 \* 0,5987

= 1,0743

Fungsi aktivasi pada output layer :

Check error (iterasi berhenti bila

Error = 0-0.7454 = -0,7454

Jumlah Kuadrat eror = ( -0,7454

**Tahap Perambatan – Balik *(Backpropagation)***

Tahap perubahan bobot dan bias

Untuk data kedua dilakukan operasi yang sama dengan data pertama, hanya saja nilai-nilaibobot dan bias awal yang digunakan adalah nilai - nilai bobot dan bias baru dari hasil perhitungan dxa pertama. Demikian seterusnya sampai data keempat (l epoch).

Proses ini diteruskan hingga maksimum epoch ke- 1500 atau kuadrat eror 0,02 (target eror) baru berhenti.

Misalkan setelah akhir literasi di peroleh nilai bobot dan akhir berikut.

Bobot awal input ke hidden layer :

Bobot awal bias ke hidden layer:

Bobot awal hidden layer ke output layer:

Bobot awal bias ke output layer:

Sekarang kita uji jaringan tersebut untuk setiap data.

-------------------------------------------------------------------

Pengujian terhadap data ke – 1 (X1=0 X2=0 target T=0)

Operasi pada hidden layer :

Z\_in1 = V01 + (V11\*X1) + (V24\*X2)

= 2,4618 + (5,8716\*0) + (-4,8532\*0)

= 2,4618

Z\_in2 = V02 + (V12\*X1) + (V22\*X2)

= -0,3844 + (3,6067\*0) + (-2,8028\*0)

= -0,3884

Z\_in3 = V03 + (V13\*X1) + (V23\*X2)

= -1,4258 + (3,4877\*0) + (-5,1943\*0)

= -1,4258

Z\_in4 = V04 + (V14\*X1) + (V24\*X2)

= -0,6994 + (0,0704\*0) + (-7,636\*0)

= -0,6994

Fungsi pada hidden layer :

Operasi pada output layer:

+(0,1938\*6,9212)+(0,3319\*-0,7503

=-3,3468

Fungsi aktivasi pada output layer :

Fungsi aktivasi :T =

Hasil aktivasi ; T = 0 (Sama dengan target)

-----------------------------------------------------------------

Pengujianterhadap datake-2 (X1=0 X2=1 Target T=1)

Operasi pada hidden layer :

Z\_in1 = v01 + (V11\* X1) + (V21 \* X2)

=2,4618 + (5,8716\*0) + (-4,8532\*0)

=-2,3914

Z\_in2 = v02 + (V12\* X1) + (V22 \* X2)

=-0,3884 + (3,6067\*0) + (2,8028\*0)

=2,4144

Z\_in3 = v03 + (V13\* X1) + (V23 \* X2)

=-2,4258 + (3,4877\*0) + (-5,1943\*0)

=-6,6201

Z\_in4 = v04 + (V14\* X1) + (V24 \* X2)

=-0,6994+ (0,0704\*0) + (0,7636\*0)

=0,0642

Fungsi aktivasi pada hidden layer :

Operasi pada output layer:

+(0,0013\*6,9212)+(0,5160\*-0,7503

=2,9684

Fungsi aktivasi pada output layer:

Fungsi aktivasi :T =

Hasil aktivasi ; T = 1 (Sama dengan target)

-----------------------------------------------------------------

Pengujianterhadap datake-3 (X1=1 X2=0 Target T=1)

Operasi pada hidden layer :

Z\_in1 = v01 + (V11\* X1) + (V21 \* X2)

=2,4618 + (5,8716\*1) + (-4,8532\*0)

=8,3334

Z\_in2 = v02 + (V12\* X1) + (V22 \* X2)

=-0,3884 + (3,6067\*1) + (2,8028\*0)

=3,2183

Z\_in3 = v03 + (V13\* X1) + (V23 \* X2)

=-1,4258+ (3,4877\*1) + (-5,1943\*0)

=2,0619

Z\_in4 = v04 + (V14\* X1) + (V24 \* X2)

=-0,6994+ (0,0704\*1) + (0,7636\*0)

=0,7064

Fungsi aktivasi pada hidden layer:

Operasi Pada output layer:

+(0,8871\*6,9212)+(0,3304\*-0,7503

=2,8918

Fungsi aktivasi pada output layer :

Fungsi aktivasi :T =

Hasil aktivasi ; T = 1 (Sama dengan target)

----------------------------------------------------------------

Pengujianterhadap datake-4 (X1=1 X2=1 Target T=0)

Operasi pada hidden layer :

Z\_in1 = v01 + (V11\* X1) + (V21 \* X2)

=2,4618 + (5,8716\*1) + (-4,8532\*1)

=3,4802

Z\_in2 = v02 + (V12\* X1) + (V22 \* X2)

=-0,3884 + (3,6067\*1) + (2,8028\*1)

=6,0211

Z\_in3 = v03 + (V13\* X1) + (V23 \* X2)

=-1,4258+ (3,4877\*1) + (-5,1943\*1)

=-3,1324

Z\_in4 = v04 + (V14\* X1) + (V24 \* X2)

=-0,6994+ (0,0704\*1) + (0,7636\*1)

=-0,0062

Fungsi aktivasi pada hidden layer:

Operasi pada output layer:

+(0,0418\*6,9212)+(0,4985\*-0,7503

=-2,7457

Fungsi aktivasi pada output layer:

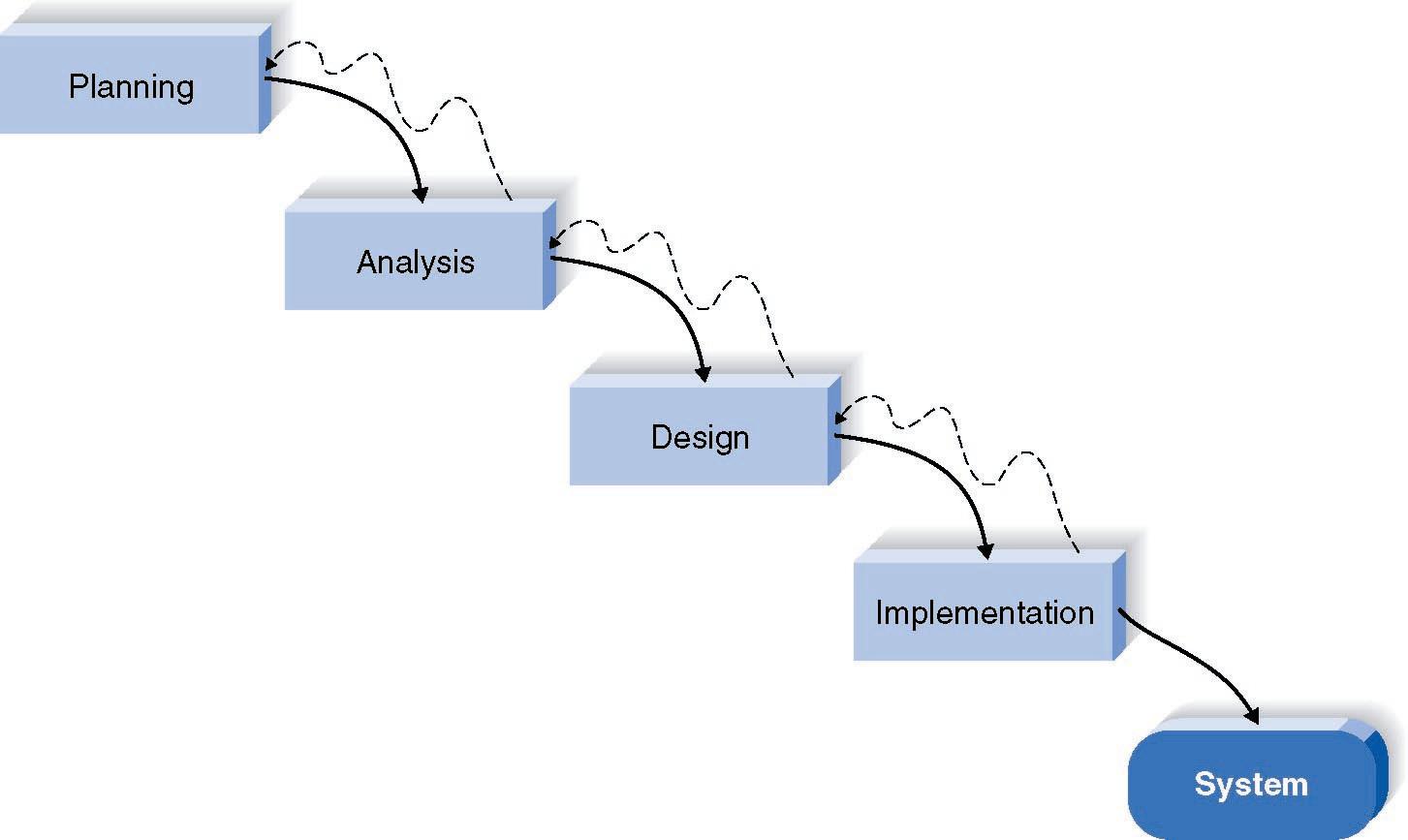
Fungsi aktivasi :T =

Hasil aktivasi : T = 0 (sama dengan target)

Dengan nilai – nialai bobot dan bias tersebut, jaringan dapat mengenali fungsi logika XOR dengan baik.

**2.2.13 Rekayasa Perangkat Lunak**

Proses pengembangan sistem melewati beberapa tahapan dari mulai sistem itu direncanakan sampai dengan sistem tersebut diterapkan, dioperasikan dan dipelihara. Bila operasi sistem yang sudah dikembangkan masih timbul kembali permasalahan-permasalahan yang kritis serta tidak dapat diatasi dalam tahap pemeliharaan sistem, maka perlu dikembangkan kembali suatu sistem untuk mengatasinya dan proses ini kembali ke tahap yang pertama, yaitu tahap perencanaan sistem. Siklus ini disebut dengan siklus hidup suatu sistem (*systems life cycle*). Daur atau siklus hidup dari pengembangan sistem merupakan suatu bentuk yang digunakan untuk menggambarkan tahapan utama dan langkah-langkah didalam tahapan tersebut dalam proses pengembangannya.(Pressman, 2002)



**Gambar 2.11:** Siklus Hidup Pengembangan Sistem

*Software* (perangkat lunak) berhubungan dengan: (1) Perintah (program komputer) yang bila dieksekusi memberikan fungsi dan unjuk kerja seperti yang diinginkan; (2) Struktur data yang memungkinkan program memanipulasi informasi secara proporsional; dan (3) Dokumen yang menggambarkan operasi dan kegunaan program. “*Software engineering* adalah suatu disiplinrekayasa (rancang-bangun) yang terkaitdengan semua aspek produksi perangkat lunak (Pressman, 2002)”. Sedangkan menurut IEEE (Institute of Electrical and Electronic Engineers) “Software engineering adalah: (1) Aplikasi dari sebuah pendekatan yang bersifat kuantifiabel, disiplin, dan sistematis bagi pengembangan, operasi, dan pemeliharaan perangkat lunak; (2) Studi tentang pendekatan-pendekatan seperti pada (1) (Pressman, 2002)”. Dengan demikian bidang penelitian software *engineering* mengacu pada kedua hal tersebut.

Perbedaan utama paradigma berorientasi objek dibandingkan konvensional adalah pada penyatuan proses/fungsi dan data ke dalam bentuk yang terenkapsulasi, sedangkan paradigma konvensional memisahkan data dengan proses. Data merupakan sekumpulan atribut yang di-enkapsulasi (dikemas/dibungkus) bersama algoritma (operasi/metode/servis) untuk melakukan suatu proses berdasarkan pesan (stimulan/message/event) yang masuk ke objek/class. Pesan ini merupakan sarana interface antar-objek.(Pressman, 2002)

Objek-objek merupakan realitas yang unik sehingga tidak efisien bila dideskripsikan satu persatu sementara mereka memiliki beberapa kesamaan. Misal kerbau dengan sapi adalah dua objek yang berbeda, tetapi keduanya adalah pemakan rumput. Oleh sebab itu diciptakan kelas yang memuat karakteristik yang sama dari kerbau dan sapi. Dengan demikian kelas ini dapat menginformasikan berbagai binatang pemakan rumput (tetapi tidak bisa menyebutkan binatang-binatang di kebun binatang). Kelas ini dengan mudah dikembangkan untuk merekrut objek/binatang-binatang lain pemakan rumput secara efisien tanpa mengulang-ulang deskripsi yang sama. Bahkan dapat dibuat hirarkhinya sebagai superkelas atau subkelas yang masing-masing memiliki warisan (inheritance) dari kelas di atasnya. Sedangkan *Polymorphism* diperlukan guna memperluas sistem (software engineering) berbasis objek yang ada secara efisien, yakni dengan nama yang sama tetapi operasinya berbeda.(Pressman, 2002)

**2.2.14 Analisis dan Desain Sistem Berorientasi Objek**

Pertanyaan pada object oriented analysis adalah: *who, what, when, where will the system be* (Alan Dennis, 2010). Tahap ini mengidentifikasi dan mengumpulkan kebutuhan sistem dengan cara menjawab: (1) *Who will use the system*? (2) *What will the system do*? dan (3) *When will it be used*? Menginvestigasi sistem yang sedang berjalan, kemudian mengidentifikasi atau mengusulkan kemungkinan–kemungkinan perbaikannya, dan akhirnya membangun konsep untuk sistem yang baru. Beberapa hal yang dilakukan pada tahap ini adalah sebagai berikut(Alan Dennis, 2010) :

1. *Requirements determination*

* Laporan berupa daftar persyaratan fungsional dan non-fungsional.
* Semua persyaratan tersebut harus dilacak kembali.

1. *Functional modeling*

* *Business process modeling with Actifity Diagram*.
* Dengan *Use Case Diagram*:
* Bagaimana bisnis berinteraksi dengan lingkungannya;
* Kegiatan yang dilakukan oleh pengguna;
* Aktifitas sistem atau apa yang akan dilakukan oleh sistem;
* Digunakan untuk dokumentasi dan pembangunan sistem;
* Dapat digunakan dengan baik sebagai komunikasi dengan pengguna.

1. *Structural modeling*

* Menggambarkan struktur data;
* Struktur data yang direpresentasikan melalui *Class Diagram*.
* Mengurangi *"semantic gap"* antara dunia nyata dan dunia *software*.
* Digunakan sebagai kosakata untuk analis dan pengguna.

1. *Behavioral modeling*

* Untuk proses bisnis digambarkan dengan *Sequence Diagram* dan *Collaboration Diagram*.
* Untuk perubahan data digambarkan dengan *Statechart Diagram*.
* Tujuan *behavioral modeling* adalah:
* Bagaimana objek berkolaborasi
* Menggambarkan pandangan internal proses bisnis
* Menunjukkan efek dari variasi-variasi proses pada sistem.

Sedangkan pertanyaan pada *object oriented design* adalah: *How will the system work*(Alan Dennis, 2010)*.* Dibandingkan dengan pendekatan konvensional, arsitektur *object oriented design* tidak memperlihatkan hirarkhi kontrol tetapi kolaborasi antar objek dengan aliran kontrol.

Beberapa hal yang dilakukan pada tahap ini adalah sebagai berikut(Alan Dennis, 2010):

1. *Moving on to Design*

Tujuan dari tahap object oriented analysis adalah untuk mencari tahu apa kebutuhan bisnis. Sedangkan tujuan dari tahap object oriented desing adalah cara membuat cetak biru sistem.

1. *Physical Architecture Layer Design (Architectur Design)*

* *Server based, client based,* atau *client server based*
* *Network model*
* *Hardware and software specification*
* *Global support and security plan*

1. *Human Computer Interaction Layer Design (Interface Design)*

* *User interface*
* *System interface*
* *Navigation mechanism*
* *Input mechanisim (Form)*
* *Output mechanism (Report)*
* *Graphical User Interface (GUI)*

1. *Data Management Layer Design (Data Design)*

Pada dasarnya format yang digunakan untuk data, yaitu:

* *File*
* *Relational database Pthon*
* *Object relational database (Extended SQL: ad hoc atau SQL3)*
* *Object oriented database* (Sistem manajemen *database* yang terpisah)

1. *Class and Method Design (Program Design)*

Kelasyang sudah adaperlu dipahamidan terorganisir dengan baik, pada umumnya beberapa kode, masih memerlukan instantiatekelasbaru. Komponen-komponen pada paradigma berbasis objek antara lain: (1) *Classes;* (2) *Objects;*(3) *Attributes*; (4) *States*; (5) *Methods*; (6) *Messages/Event.*

Seperti yang telah dipaparkan sebelumnya, alat bantu yang biasanya digunakan pada tahap *object oriented analysis and design* adalah *The Unified Modelling Language* (UML).

*“UML can be used for modeling all processes in the development life cycle and across different implementation technologies. UML is the standard language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system*(Alan Dennis, 2010)*.*

UML merupakan *tools* pemodelan yang bermanfaat dalam rekayasa perangkat lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML dapat dibuat model untuk semua jenis aplikasi perangkat lunak, yang dapat beroperasi pada perangkat keras, sistem operasi, dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk pemodelan perangkat lunak dalam bahasa berorientasi objek seperti python, matlab

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram perangkat lunak. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering). Simbol-simbol yang digunakan pada UML dapat ditunjukkan pada berbagai tabel berikut(Alan Dennis, 2010).

**Tabel 2.5:** Simbol-Simbol Actifity Diagram

|  |  |
| --- | --- |
| Simbol | Keterangan |
|  | *Action/actifity* |
|  | *Object/class* |
|  | *Control/object flow* |
|  | *Initial node* |
|  | *Actifity final node* |
|  | *Decision node* |
|  | *Merge node* |
|  | *Fork node* |
|  | *Join node* |

Sumber: (Alan Dennis, 2010)

**Tabel 2.6:** Simbol-Simbol Use Case Diagram

|  |  |
| --- | --- |
| Simbol | Keterangan |
|  | *Actor* |
|  | *UseCase* |
|  | *Association relationship* |
|  | *Include relationship* |
|  | *Extend relationship* |
|  | *Generalization relationship* |

Sumber: (Alan Dennis, 2010)

**Tabel 2.7:** Simbol-Simbol Class Diagram

|  |  |
| --- | --- |
| Simbol | Keterangan |
|  | *Class/Object* |
|  | *Interface* |
|  | *Enumeration* |
|  | *Package* |
|  | *Association* |
|  | *Aggregation* |
|  | *Composition* |
|  | *Dependency* |
|  | *Inheritance* |

Sumber: (Alan Dennis, 2010)

**Tabel 2.8:** Simbol-Simbol Sequence Diagram

|  |  |
| --- | --- |
| Simbol | Keterangan |
|  | *Actor* |
|  | *Object* |
|  | *Lifeline* |
|  | *Focus of control* |
|  | *Message* |
|  | *Object destruction* |

Sumber: (Alan Dennis, 2010)

**2.2.15 Konstruksi Sistem**

Tahapan implementasi merupakan tahap dimana dilakukan transformasi /penerjemahan dari bahasa *modeling* ke suatu bahasa pemrograman. hal ini merupakan tugas dari pemprogram, pada pengembangan sistem/perangkat lunak berorientasi objek penerjemahan dari setiap diagram-diagram UML yang telah di rancang pada tahap analisis dan desain harus diterjemahkan ke dalam bahasa pemrograman sama persis dengan diagram-diagram yang ada guna menghindari terjadinya perubahan fungsi/tujuan dari pengembangan sistem/perangkat lunak.

**2.2.16 Pengujian Sistem dengan White Box Testing**

Pengujian *White Box*, adalah metode pengujian yang menggunakan struktur kontrol desain prosedur untuk memperoleh *test case*. Dengan menggunakan metode *white box*, perekayasa sistem dapat melakukan *test case* yang memberikan jaminan bahwa semua jalur independen pada suatu modul telah digunakan paling tidak satu kali, menggunakan semua keputusan logis pada sisi *true* dan *false*, mengeksekusi semua *loop* pada batasan mereka dan pada batas operasional mereka, dan menggunakan stuktur data internal untuk menjamin validitasnya. (Pressman, 2002)



**Gambar 2.12 :** Contoh Bagan Alir

Bagan alir digunakan untuk menggambarkan struktur kontrol program dan untuk menggambarkan grafik alir, harus memperhatikan representasi desain prosedural pada bagan alir. Pada gambar dibawah ini, grafik alir memetakan bagan alir tersebut ke dalam grafik alir yang sesuai (dengan mengasumsikan bahwa tidak ada kondisi senyawa yang diisikan di dalam diamond keputusan dari bagan alir tersebut). Masing-masing lingkaran, yang disebut *simpul* grafik alir, merepresentasikan satu atau lebih statemen prosedural. Urutan kotak proses dan permata keputusan dapat memetakan simpul tunggal. Anak panah tersebut yang disebut *edges* atau *links*, merepresentasikan aliran kontrol dan analog dengan anak panah bagan alir. Edge harus berhenti pada suatu simpul, meskipun bila simpul tersebut tidak merepresentasikan statemen prosedural. (Pressman, 2002)

**2.2.17 Pengujian Sistem dengan Black Box Testing**

*Black box approach* adalah suatu sistem dimana *input dan output-*nya Dapat didefinisikan tetapi prosesnya tidak diketahui atau tidak terdefinisi. Metode ini hanya dapat dimengerti oleh pihak dalam (yang menangani sedangkan pihak luar hanya mengetahui masukan dan hasilnya).Sistem ini terdapat pada subsistem tingkat terendah. Metode ujicoba *black box* memfokuskan pada keperluan fungsional dari *software*. Karena itu ujicoba *black box* memungkinkan pengembang *software* untuk membuat himpunan kondisi *input* yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba *black box* bukan merupakan alternatif dari ujicoba *white box*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *white box*.

Ujicoba *black box* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya:

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses*database eksternal*
4. Kesalahan performa
5. Kesalahan inisialisasi dan terminasi

Tidak seperti metode *white box* yang dilaksanakan diawal proses, ujicoba *black box* diaplikasikan dibeberapa tahapan berikutnya karena ujicoba *black box* dengan sengaja mengabaikan struktur kontrol, sehingga perhatiannya difokuskan pada informasi *domain*. Ujicoba didesain untuk dapat menjawab pertanyaan-pertanyaan berikut:

1. Bagaimana validitas fungsionalnya diuji?
2. Jenis *input* seperti apa yang akan menghasilkan kasus uji yang baik?
3. Apakah sistem secara khusus sensitif terhadap nilai input tertentu?
4. Bagaimana batasan-batasan kelas data diisolasi?
5. Berapa rasio data dan jumlah data yang dapat ditoleransi oleh sistem?
6. Apa akibat yang akan timbul dari kombinasi spesifik data pada operasi sistem?

Dengan mengaplikasikan uji coba *black box*, diharapkan dapat menghasilkan sekumpulan kasus uji yang memenuhi kriteria berikut:

1. Kasus uji yang berkurang, jika jumlahnya lebih dari 1, maka jumlah dari uji kasus tambahan harus didesain untuk mencapai ujicoba yang cukup beralasan.
2. Kasus uji yang memberitahukan sesuatu tentang keberadaan atau tidaknya suatu jenis kesalahan, dari pada kesalahan yang terhubung hanya dengan suatu ujicoba yang spesifik.

**2.2.18 Perangkat Lunak Pendukung**

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python mendukung multi paradigma pemrograman, utamanya; namun tidak dibatasi; pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi

**2.3 Kerangka Pemikiran**



**Gambar 2.13:** Kerangka Pikir