BAB 2
LANDASAN TEORI

2.1 Psikologi

2.1.1 Sejarah Perkembangan Psikologi

Sejak zaman filsuf-filsuf besar seperti Sokrates pada tahun 469-399 SM, Plato tahun 427-347 dan Aristoteles tahun 384-322 SM telah berkembang filsafat mental yang berusaha memberikan pengalaman-pengalaman kejiwaan atau membahas persoalan "jiwa raga" (*body-mind problems*). Seorang filsuf yang secara jelas membahas persoalan ini adalah Rene Descartes tahun 1596-1650. Beliau memandang manusia mempunyai dua unsur yang tidak dapat dipisahkan, yaitu: jiwa dan raga. Kedua unsur ini saling mempengaruhi, terutama karena adanya kelenjar *pinealis* yang terdapat dalam otak[3].

2.1.2 Pengertian BAUM Test

BAUM *test* adalah tes psikologi berupa tes menggambar pohon dengan ketentuan tertentu. Umumnya pohon yang digambar adalah pohon yang berkayu, memiliki akar, batang, dahan, daun, serta terkadang bunga dan buah.

2.1.3 Kelebihan Tes Gambar Pohon

Dalam tes BAUM atau tes menggambar pohon terdapat beberapa kelebihan sehingga psikolog memilih gambar pohon, yaitu sebagai berikut:

1. Sistem hidup pohon merupakan sistem terbuka. Maksudnya, pola tumbuh (kondisi pertumbuhan) pohon dapat secara jelas terlihat dari luar, terbuka (sangat mudah diamati). Inilah salah satu perbedaan utama sistem hidup pohon dengan sistem hidup manusia atau hewan. (Sifat pohon mudah diinterpretasikan).
2. Sistem hidup manusia merupakan sistem tertutup. Semua organ-organnya sudah ada sejak awal. Di dalam tubuh semua organ diberi makan dengan kekuatan yang sama seumur hidupnya, dan dikendalikan oleh organ-organ

pusat. Kondisi yang terjadi di dalamnya tidak terlihat dengan jelas, tertutup. Hal lainnya, manusia mempunyai hati dan akal, yang tentu saja sulit diinterpretasikan dari luar. Hal tersebut adalah cara tegas membedakan sifat manusia dengan pohon. (Sifat manusia sulit diinterpretasikan).

3. Sifat pohon yang cenderung terbuka ini (mudah dibaca), bertentangan dengan sifat manusia yang tertutup, dimanfaatkan oleh psikolog untuk membaca kepribadian seseorang. Gambar pohon yang dibuar oleh seseorang, mencerminkan kondisi kepribadian orang tersebut. Lewat gambar pohon, psikolog akan mudah membaca kepribadiannya, karena sifat pohon yang terbuka [3].

2.1.4 Tujuan Tes Gambar Pohon

Tes gambar adalah tes untuk mengukur atau menginterpretasikan pribadi subjek, dengan melihat sosok gambar yang ditampilkan. Sebuah pohon, digambarkan sebagai sesuatu yang bisa memberi keteduhan bagi makhluk yang ada di sekitarnya. Dalam hubungannya dengan sifat pohon ini, maka tes menggambar pohon bertujuan, anatara lain untuk mengukur ego, emosionalitas, kepekaan, sikap, dan adaptasi.

2.1.5 Motif dan Struktur Soal

Dalam tes ini, peserta tes akan diberi alat yang dibutuhkan berupa kertas dan pensil. Peserta tes dilarang menggunakan penghapus. Biasanya waktu yang disediakan cukup lama. Kemudian penguji akan memberi instruksi kepada peserta tes untuk menuliskan identitas diri di kertas lainnya. Tahap selanjutnya, penguji akan memberi instruksi untuk membuat gambar pohon kepada peserta tes. Ketentuannya adalah peserta menggambar sebuah pohon apa saja sesuai dengan imajinasi pada saat itu, dengan syarat pohon yang digambar adalah:

1. Pohon berkayu, berkambium, dan berdahan banyak, seperti pohon rambutan, jambu, atau mangga.
2. Jangan menggambar pohon kelapa, pisang, semak belukar (padi, tebu), atau cemara karena pohon-pohon itu tidak bercabang.

3. Gambarlah secara detail di tiap-tiap bagian pohon, seperti pada tangkai, bentuk daun, kerapatan daun, buah akar, bahkan alur pohon.
4. Beri nama pohon di bawah gambar.
5. Gambar yang dibuat haruslah proposional dari bawah ke atas [4].

2.2 Citra Digital

2.2.1 Pengertian Citra Digital

Citra digital merupakan sebuah larik (*array*) yang berisi nilai-nilai real maupun komplek yang direpresentasikan dengan deretan bit tertentu. Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar dua dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data dua dimensi.

2.2.2 Kategori Pengolahan Citra

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

1. Kategori rendah, melibatkan operasi-operasi sederhana seperti prapengolahan citra untuk mengurangi derau, pengaturan kontras, dan pengaturan ketajaman citra. Pengolahan kategori rendah ini memiliki input dan output berupa citra.
2. Kategori menengah, melibatkan operasi-operasi seperti segmentasi dan klasifikasi citra. Proses pengolahan citra menengah ini melibatkan input berupa citra dan output berupa atribut (fitur) citra yang dipisahkan dari citra input.
3. Kategori tinggi, melibatkan proses pengenalan dan deskripsi citra.[5]

2.2.3 Pengenalan Pola

Pengenalan pola adalah proses mengenali suatu objek secara independen atau berdasarkan secara kemiripan dengan data yang ada sebelumnya, prinsip kerjanya meniru kemampuan manusia mengenali objek-objek berdasarkan cirri-ciri dan pengtahuan yang sudah diamatinya dari objek-objek tersebut. Pada pengenalan pola terdapat beberapa metode pengenalan pola. Yaitu sebagai berikut:

1. Sintaks, berdasarkan ciri-ciri fisik yang jelas dengan toleransi yang telah ditentukan sebelumnya.
2. Statistic, berdasarkan data yang bersifat statistic
3. Jaringan Syaraf Tiruan (JST), metode yang menggabungkan metode sintaks dan statistic yaitu mengenali objek dari cirri-ciri fisik yang jelas kemudian mengambil kesimpulan berdasarkan data statistic, cara kerja JST seolah-olah meniru cara kerja otak manusia[12].

2.2.4 Citra Grayscale

Citra *grayscale* adalah citra yang citra yang terdiri dari variasi dari warna hitam dan putih, tetapi variasi warnanya sangat banyak. Citra *grayscale* merupakan citra yang memiliki derajat keabuan 256 (8 bit).

Untuk mengubah citra berwarna menjadi citra *grayscale* dilakukan konversi dengan mengambil nilai rata-rata (R+G+B)/3 [9] dimana:

R = Unsur warna merah pada basis warna RGB

G = Unsur warna hijau pada basis warna RGB

B = Unsur warna biru pada basis warna RGB

Berikut akan ditampilkan contoh gambar pohon *grayscale* yang telah dibuat, seperti pada gambar 2.1.



Gambar 2.1 Citra Grayscale

2.2.5 Thresholding

Thresholding digunakan untuk mempertegas citra dari citra keabuan (*grayscale*) menjadi citra biner. Pemilihan nilai *thresholding* akan mempengaruhi hasil citra biner. Nilai *thresholding* (nilai batas/ambang) akan diproses dengan syarat nilai pixel yang berada di bawah nilai *threshold* (nilai ambang) maka akan diubah menjadi putih atau bernilai 0, dan apabila nilai pixel yang berada di atas nilai *threshold* (nilai ambang maka akan diubah menjadi warna hitam atau bernilai 1.

2.2.6 Citra Biner

Citra biner adalah citra yang hanya mempunyai dua nilai derajat keabuan: hitam dan putih. Meskipun saat ini citra berwarna lebih disukai karena memberi kesan yang lebih kaya daripada citra biner, namun tidak membuat citra biner mati. Pada pengidentifikasian karakteristik seseorang dibutuhkan citra biner untuk dapat dikenali polanya dan dilakukan pengajaran pola[8].

2.2.7 Konversi Citra Grayscale ke Citra Biner

Pengkonversian citra hitam-putih (*grayscale*) menjadi citra biner dilakukan untuk alasan-alasan sebagai berikut:

1. Untuk mengidentifikasi keberadaan objek, yang direpresentasikan sebagai daerah (region) di dalam citra. Misalnya kita ingin memisahkan (segmentasi) objek dari gambar latar belakangnya. *Pixel-pixel* objek dinyatakan dengan nilai 1 sedangkan *pixel* lainnya dengan 0. Objek ditampilkan seperti gambar siluet. Untuk memperoleh siluet yang bagus, objek harus dapat dipisahkan dengan mudah dari gambar latar belakangnya.
2. Untuk lebih memfokuskan pada analisis bentuk morfologi, yang dalam hal ini intensitas *pixel* tidak terlalu penting dibandingkan bentuknya. Setelah objek dipisahkan dari latar belakangnya, properti geometri dan morfologi/topologi objek dapat dihitung dari citra biner. Hal ini berguna untuk pengambilan keputusan.

3. Untuk menampilkan citra pada piranti keluaran yang hanya mempunyai resolusi intensitas satu bit, yaitu piranti penampil dua-aras atau biner seperti pencetak (*printer*).
4. Mengkonversi citra yang telah ditingkatkan kualitas tepinya (*edge enhancement*) ke penggambaran garis-garis tepi. Ini perlu untuk membedakan tepi yang kuat yang berkoresponden dengan batas-batas objek dengan tepi lemah yang berkoresponden dengan perubahan *illumination*, bayangan, dll.

Konversi dari citra *grayscale* ke citra biner dilakukan dengan operasi pengambangan (*thresholding*). Operasi pengambangan mengelompokkan nilai derajat keabuan setiap *pixel* ke dalam 2 kelas, hitam dan putih.

Dua pendekatan yang digunakan dalam operasi pengambangan adalah pengambangan secara global dan pengambangan secara lokal.

a. Pengambangan secara global (*global image thresholding*)

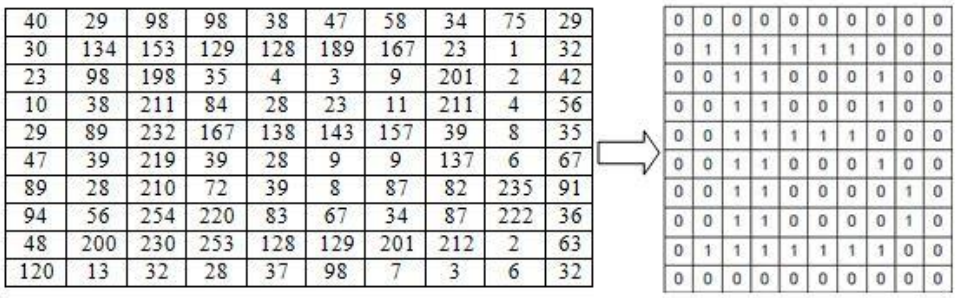
Setiap *pixel* di dalam citra dipetakan ke dua nilai, 1 atau 0 dengan fungsi pengambangan:

$$f_B(i,j) = \begin{cases} 1, & f_g(i,j) \geq T \\ 0, & lainnya \end{cases} \quad (1)$$

yang dalam hal ini, $f_g(i, j)$ adalah citra *grayscale*, $f_B(i, j)$ adalah citra biner, dan T adalah nilai ambang yang dispesifikasikan. Dengan operasi pengambangan tersebut, objek dibuat berwarna gelap (1 atau hitam) sedangkan latar belakang berwarna terang (0 atau putih).

Nilai ambang T dipilih dimisalkan adalah 128, apabila nilai $f_g \geq 128$ maka nilai $f_B = 1$. Sebaliknya, apabila nilai $f_g < 128$ maka nilai $f_B = 0$. Dengan cara seperti ini, kita tidak hanya mengkonversi citra *grayscale* ke citra biner, tetapi sekaligus melakukan segmentasi objek dari latar belakangnya[8].

Contoh citra *grayscale* 10 x 10 yang dikonversikan ke dalam citra biner, terdapat pada gambar 2.2.

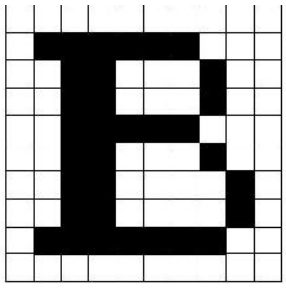


40	29	98	98	38	47	58	34	75	29
30	134	153	129	128	189	167	23	1	32
23	98	198	35	4	3	9	201	2	42
10	38	211	84	28	23	11	211	4	56
29	89	232	167	138	143	157	39	8	35
47	39	219	39	28	9	9	137	6	67
89	28	210	72	39	8	87	82	235	91
94	56	254	220	83	67	34	87	222	36
48	200	230	253	128	129	201	212	2	63
120	13	32	28	37	98	7	3	6	32

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	1	1	1	0	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0
0	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.2 Konversi Citra Grayscale Menjadi Citra Biner

Berikut dihasilkan citra biner yang telah dikonversikan dengan menggunakan nilai ambang (*threshold*) = 128. Citra biner dapat dilihat pada gambar 2.3.



Gambar 2.3 Citra Biner yang Telah dikonversikan

2.2.8 Kompresi *Lossless* dan *Lossy*

Berdasarkan kandungan informasi pada kandungan informasi pada citra hasil maka sifat kompresi data dapat dikelompokkan menjadi dua yaitu kompresi *lossless* (nir-rugi) dan kompresi *lossy* (ber-rugi).

2.2.8.1 Kompresi Lossless

Pada kompresi jenis ini informasi yang terkandung pada citra hasil sama dengan informasi pada citra asli. Citra hasil proses kompresi dapat citra asli. Citra hasil proses kompresi dapat dikembalikan secara sempurna menjadi citra asli, tidak terjadi kehilangan informasi, tidak terjadi kesalahan informasi. Oleh karena itu metode ini disebut juga *error free compression*. Kompresi tipe ini cocok diterapkan pada berkas basis data (*data base*), *spread sheet*, berkas *word processing*, citra biomedis dan lain sebagainya.

2.2.8.2 Kompresi *Lossy*

Kompresi data yang bersifat *lossy* mengijinkan terjadinya kehilangan sebagian data tertentu dari pesan tersebut, sehingga dapat menghasilkan kompresi yang tinggi. Apabila citra tersebut terkompresi di rekonstruksi kembali maka hasilnya tidak sama dengan citra aslinya, tetapi informasi yang terkandung tidak sampai berubah atau hilang. Sebagian besar kompresi data *lossy* memiliki pengaturan tingkat kompresi yang berbeda-beda. Hal ini dilakukan agar kompresinya lebih efektif dan informasi yang terkandung pada citra tidak sampai berubah dan hilang. Kompresi data *lossy* ini efektif jika diterapkan pada penyimpanan data analog yang didigitasi seperti gambar, video, dan suara.

2.3 Artificial Intelegence (AI)

Artificial Intelligence atau kecerdasan buatan merupakan salah satu bagian ilmu komputer yang membuat agar mesin dapat melakukan pekerjaan seperti dan sebaik-baik yang dilakukan oleh manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kehidupan umat manusia[10].

2.3.1 Sejarah Artificial Intelegence

Artificial Intelegence atau kecerdasan buatan termasuk bidang ilmu yang relatif muda. Pada tahun 1950-an para ilmuan dan peneliti mulai memikirkan bagaimana caranya agar mesin dapat melakukan pekerjaannya seperti yang bisa dikerjakan oleh manusia. Alan Turing, seorang matematikawan dari Inggris pertama kali mengusulkan adanya tes untuk melihat bisa tidaknya sebuah mesin dikatakan cerdas. Kecerdasan buatan atau *Artificial Intelegence* itu sendiri dimunculkan oleh seorang professor dari Massachusetts Institute of Technology yang bernama Jhon McCarthy pada tahun 1956 pada Dartmouth Conference yang dihadiri oleh pada peneliti AI. Pada koferensi tersebut juga didefinisikan tujuan utama dari kecerdasan buata, yaitu mengetahui dan memodelkan proses-proses berfikir manusia dan mendesain mesin agar dapat menirukan kelakuan manusia tersebut.

2.3.2 Soft Computing

Soft computing adalah koleksi dari beberapa metodologi yang bertujuan untuk mengeksploitasi adanya toleransi terhadap ketidaktepatan, ketidakpastian, dan kebenaran parsial untuk dapat diselesaikan dengan mudah, *robustness*, dan biaya penyelesaiannya murah. Definisi ini pertama kali diungkapkan oleh Prof. Lofti A. zadeh pada tahun 1992.

Unsur-unsur pokok dalam *soft computing*, adalah:

1. Sistem Fuzzy (mengakomodasi ketidaktepatan);
2. Jaringan Syaraf (menggunakan pembelajaran);
3. *Probabilistic Reasoning* (mengakomodasi ketidakpastian);
4. *Evolutionary Computing* (optimasi).

2.3.3 Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan (JST), yaitu metode *learning* yang biasa digunakan untuk permasalahan diskrit, real, maupun vektor. JST juga merupakan pemodelan sistem syaraf manusia dalam melaksanakan tugas tertentu. Menurut Haykin, proses belajar terdapat kejadian-kejadian sebagai berikut:

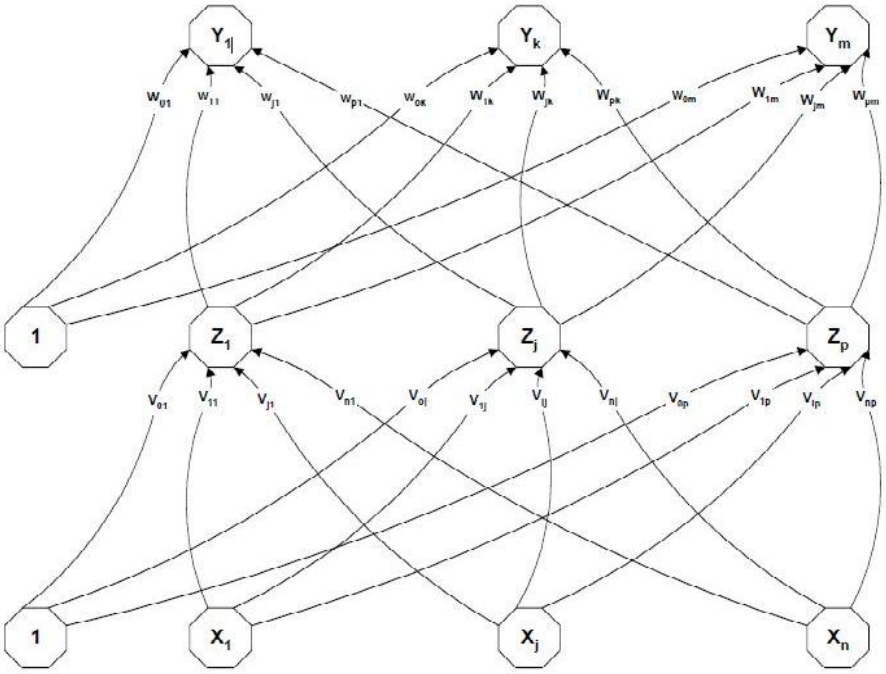
1. JST dirangsang oleh lingkungan
2. JST mengubah dirinya sebagai hasil rangsangan ini
3. JST memberikan respon dengan cara yang baru kepada lingkungan, disebabkan perubahan yang terjadi dalam struktur internalnya sendiri.

Proses belajar pada jaringan syaraf tiruan juga terdiri dari belajar dengan pengawasan (*Supervised Learning*) dan belajar tanpa pengawasan (*Unsupervised Learning*).

Supervised Learning adalah proses belajar yang membutuhkan pengetahuan lingkungan.yang direpresentasikan sebagai sekumpulan sampel *input-output.* Model JST yang tergolong *supervised learning* adalah JST dengan arsitektur *Multi Layer Percepton* (MLP) yang banyak digunakan untuk pendidikan.

2.3.3.1 Multi Layer Percepton (MLP)

MLP merupakan model JST yang paling banyak digunakan dalam bidang pendidikan dan aplikasi. Arsitektur dan proses belajar yang sederhana sangat memudahkan untuk dipelajari. Arsitektur MLP diilustrasikan oleh gambar 2.4:



Gambar 2.4 Model Multi Layer Percepton

Keterangan:

Neuron-neuron:

Xi = input (i = 1, 2, .., n)

Zj = neural hidden (j = 1, 2, …, p)

Yk = output (k = 1, 2, …, m)

Bobot:

V0j = bias unit hidden

Vij = bobot unit hidden

W0k = bias unit output

Wjk = bobot unit output

Secara umum proses pelatihan yang dilakukan adalah sebagai berikut:

a. Pengambilan input
b. Penelusuran error
c. Penyesuaian bobot

Salah satu algoritma pelatihan yang diusulkan untuk melatih MLP adalah algoritma pelatihan *backpropagation* atau propagasi balik. Algoritma ini

melakukan dua tahap perhitungan, yaitu: perhitungan maju untuk menghitung galat antara keluaran aktual dan target, dan perhitungan mundur yang mempropagasi balik galat tersebut untuk memperbaiki bobot-bobot sinaptik pada semua *neuron* yang ada.

Pada kasus pengidentifikasian ini, algoritma *backpropagation* yang digunakan karena algoritma ini dapat mewujudkan sistem yang tahan akan kerusakan (*robustness*), yang artinya sistem tidak akan terpengaruh oleh gangguan yang akan mengacaukan sistem. *Backpropagation* juga konsisten bekerja dengan baik, dikarenakan hasil keluaran bobot paling akhir akan lebih adaptif ketimbang sistem matematis. Contoh sistem yang matematis, misal x = 4*2, dengan rentang 5 < x < 10. Maka x terpenuhi. Contoh sistem yang adaptif, misal x = 15, dengan rentang 5 < x < 10. Artinya x memiliki lebih dari rentang ketentuan yang dianggap hanya gangguan kecil, maka sistem tidak akan berubah.

2.3.3.2 Algoritma Nguyen Widrow dan Backpropagation

Backpropagation merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh percepton dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyinya. Algoritma *backpropagation* menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error* ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan menggunakan fungsi aktivasi sigmoid [10]. Fungsi aktivasi ini digunakan karena secara khusus menguntungkan dalam penggunaan jaringan syaraf yang dilatih dengan *backpropagation*, karena hubungan yang sederhana diantara nilai fungsi pada suatu titik dan nilai *derivative* (turunan) pada titik itu mengurangi beban perhitungan selama pelatihan, fungsi sigmoid dan fungsi turunannya seperti pada persamaan 2 dan 3:

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

$$= (1+e^{-x})^{-1}, \text{misal } y = 1+e^{-x}$$

$$f(x) = (y)^{-1}$$

$$f'(x) = \frac{df}{dx} = \frac{df}{dy}\frac{dy}{dx}$$

$$\frac{df}{dx} = (-y^{-2})(-e^{-x})$$

$$\frac{df}{dx} = -\frac{1}{(1+e^{-x})^2} - e^{-x}$$

$$= \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{e^{-x}}{1+e^{-x}}\right)$$

$$= f(x)\left(\frac{1-1+e^{-x}}{1+e^{-x}}\right)$$

$$= f(x)\left[\left(\frac{1+e^{-x}}{1+e^{-x}}\right) - \left(\frac{1}{1+e^{-x}}\right)\right]$$

$$f'(x) = f(x)(1 - f(x)) \quad (3)$$

Dengan:

$f(x) =$ fungsi aktivasi sigmoid

$e =$ epsilon

$x =$ nilai yang akan dihitung nilai aktivasinya

Pada kasus pengidentifikasian gambar ini terdapat algoritma *backpropagation* (pelatihan propagasi balik). Algoritma ini akan menghitung *error-error* pada data yang dilatih dan yang akan di testing. Pada data yang dilatih akan dihasilkan bobot-bobot baru yang bobot-bobotnya akan digunakan pada inisialisasi data testing yang kemudian dihitung hanya *feedforward*-nya. Akan tetapi algoritma *backpropagation* pada data yang akan dilatih membutuhkan pemilihan bobot dan bias awal yang tidak terlalu kecil dan tidak terlalu besar. Agar nilai turunan aktivasi pun tidak terlalu kecil juga tidak terlalu besar juga. Oleh karena itu dalam standar *backpropagation*, bobot dan bias diisi dengan bilangan acak kecil.

Pada tahun 1990, Nguyen dan Widrow mengusulkan cara membuat inisialisasi bobot dan bias ke unit tersembunyi sehingga menghasilkan iterasi lebih cepat. Misal

n = jumlah unit masukkan

p = jumlah unit tersembunyi

p = (n/2) + 1

m = T_k = jumlah unit keluaran

β = faktor skala = 0.7 $\sqrt[n]{p}$

Algoritma inisialisasi Nguyen Widrow adalah sebagai berikut:

1. Inisialisasi semua bobot ($V_{ij}(lama)$) (i = 1, 2, …, n; j = 1, 2, …,p) dengan bilangan acak dalam interval [-0.5, 0.5]
2. Hitung $\|V_j\| = \sqrt{{V_{1j}}^2 + {V_{2j}}^2 + \ldots + {V_{nj}}^2}$ (4)
3. Bobot yang dipakai sebagai inisialisasi: $V_{ij} = \frac{\beta V_{ij}(lama)}{\|V_j\|}$ (5)
4. Bias yang dipakai sebagai inisialisasi: V_{0j} = bilangan acak antara –β dan β [11].

Algoritma *backpropagation*, yaitu sebagai berikut:

1. Inisialisasi bobot (ambil bobot awal dengan nilai random yang cukup kecil dengan menggunakan algoritma Nguyen Widrow seperti yang telah dijelaskan sebelumnya).
2. Kerjakan langkah-langkah berikut selama kondisi berhenti bernilai “false” :

Feedforward:

3. Tiap-tiap unit input (x_i, i=1,2,3,…,n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada di atasnya (lapisan tersembunyi).
4. Tiap-tiap unit tersembunyi (z_j, j=1,2,3,…,p) menjumlahkan sinyal-sinyal input terbobot untuk suatu nilai j, pada persamaan 6:

$z_in_j = V_{0j} + \sum_{i=1}^{n} X_i V_{ij}$ (6)

Hitung sinyal output dengan menggunakan fungsi aktivasi pada persamaan 7 dan 8:

$z_j = f\left(z_{in_j}\right)$ (7)

$z_j = \frac{1}{1+e^{-z_in_j}}$ (8)

5. Insialisasi w_{jk} dan w_{0k} menggunakan algoritma nguyen widrow.
6. Tiap-tiap unit output (y_k, k=1,2,3,…,m) menjumlahkan sinyal-sinyal input terbobot untuk suatu nilai k, pada persamaan 9:

$$y_in_k = w_{0k} + \sum_{i=1}^{p} z_j w_{jk} \quad (9)$$

Hitung sinyal output dengan menggunakan fungsi aktivasi, pada persamaan 10 dan 11:

$$y_k = f(y_in_k) \quad (10)$$

$$y_k = \frac{1}{1+ e^{-y_in_k}} \quad (11)$$

Backpropagation:

7. Tentukan *error* dan *error* maksimum (k=1, …, m). *Error* dapat diperhitungkan seperti pada persamaan 12:

$$err = \frac{1}{2}\sum_k (T_k - y_k)^2 \quad (12)$$

8. Tentukan jumlah iterasi *epoch* dan *epoch* maksimum. Seperti pada persamaan 13:

$$maks_epoch = \frac{jumlah\ bobot}{maks_error} \quad (13)$$

9. Tiap-tiap unit output terhadap y_k (k=1,2,3,…,m) menerima target pola berhubungan dengan pola input pembelajaran, hitung informasi *error* (δ_k) antara input dan outputnya, seperti persamaan 14 dan15:

$$\delta_k = (T_k - y_k) f'(y_in_k) \quad (14)$$

$$\delta_k = (T_k - y_k) * y_k * (1 - y_k) \quad (15)$$

Hitung koreksi bobot (Δw_{jk}) untuk memperbaiki nilai w_{jk} dengan memberikan nilai *learning rate* (laju pembelajaran) yang diberikan tidak boleh terlalu tinggi karena apabila nilai laju pembelajaran tinggi, maka pembelajarannya akan cepat menurun. Biasanya *learning rate* yang digunakan pada interval [0.1, 0.9], seperti pada persamaan 16:

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (16)$$

$\square = learning\ rate$

Hitung koreksi bias (Δw_{0k}) untuk memperbaiki nilai w_{0k}, pada persamaan 17

$$\Delta w_{0k} = \alpha \delta_k \quad (17)$$

10. Tiap-tiap *error* dikoreksi hingga semua koreksi *error* (δ_in_j) dari jumlah informasi *error* dengan bobot *hidden* ke *output* (k = 1, …,m) dapat dihitung, seperti persamaan 18:

$$\delta_in_j = \sum_{k=1} \delta_k w_{jk} \quad (18)$$

Tiap-tiap unít tersembunyi (δ_j) terhadap z_j (j=1,2,3,…,p) menjumlahkan delta inputnya (dari unit-unit yang berada pada lapisan di atasnya), seperti persamaan 19 dan 20:

$$\delta_j = \delta_{in_j} f'(z_{in_j}) \quad (19)$$

$$\delta_j = \delta_{in_j} * z_j * (1 - z_j) \quad (20)$$

Hitung koreksi bobot untuk memperbaiki nilai V_{ij}, pada persamaan 21:

$$\Delta V_{jk} = \alpha \delta_j x_i \quad (21)$$

Hitung koreksi bias untuk memperbaiki nilai V_{0j}, pada persamaan 22:

$$\Delta V_{0j} = \alpha \delta_j \quad (22)$$

11. Tiap-tiap unit outputnya (y_k, k=1,2,3,…,m) memperbaiki bias dan bobotnya (j=0,1,2,…,p), seperti persamaan 23 :

    $$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (23)$$

    Tiap-tiap unit tersembunyi (z_j, j=1,2,3,…,p) memperbaiki bias dan bobotnya (i=0,1,2,…,n), seperti persamaan 24 :

    $$V_{ij}(baru) = V_{ij}(lama) + \Delta V_{ij} \quad (24)$$

12. Tes kondisi berhenti:
    a. *Err* < maks_*error* ; atau
    b. *Epoch* > maks_*epoch*
13. Jika kondisi nilai *error* atau *epoch* belum memenuhi nilai tes kondisi berhenti, maka akan dilakukan pengulangan dari langkah 3 sampai langkah 11.

2.4 Alat Bantu Pemodelan

2.4.1 Perancangan Berorientasi Objek

Rekayasa adalah bentuk upaya manusia untuk mencari atau merancang produk yang berkualitas. Produk dapat berupa sistem atau perangkat lunak yang sedang atau akan dibangun dengan satu model pengembangan. Orientasi objek adalah cara pandang algoritma yang diimplementasikan pada bahasa pemrograman berorientasi objek.

OOA adalah teknik yang model *driven* yang mengintegrasikan data dan proses yang disebut objek. Model OOA adalah gambar-gambar yang mengilustrasikan objek-objek sistem dari berbagai perspektif, seperti struktur, perilaku dan interasi antar objek. Contoh yang paling terkenal adalah *UML* (*Unified Modelling Language*).

Cara berpikir orientasi objek adalah segala sesuatu dipandang dengan objek. Objek dapat berupa konsep, abstraksi, atau sesuatu dengan batas-batas tegas dan mempunyai arti untuk persoalan yang ditangani. Objek mempunyai identitas dan dapat dibedakan. Ada dua kegunaan objek yaitu : untuk meningkatkan pemahaman dunia nyata dan menyediakan dasar pengetahuan praktis untuk implementasi komputer.

Perangkat lunak masih seperti model yang digunakan dalam metodologi konvensional, yaitu dapat menggunakan model *waterfall*. Dalam model *waterfall* terdapat lima tahapan proses, yaitu : tahap rekayasa sistem, analisis perancangan, pemrograman, pengujian dan pemeliharaan. Dengan menggunakan *waterfall*, memungkinkan adanya pengembangan aplikasi dikemudian hari, sehingga aplikasi yang dibangun dapat bertahan dalam jangka waktu yang panjang.

2.4.2 UML

UML Singkatan *dari Unified Modeling Language* yang berarti bahasa pemodelan standar. Mengutip buku Chonoles and James A Schardt, *UML* memiliki sintak dan semantik. Ketika kita membuat model menggunakan konsep *UML* ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang dibuat berhubungan satu dengan lainnya harus mengikuti standar yang ada. *UML* bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. *UML* diaplikasikan untuk maksud tertentu, biasanya antara lain untuk :

1. Merancang perangkat lunak
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasikan sistem yang ada, proses-proses dan organisasinya.

UML telah diaplikasikan dalam bidang investasi perbankan, lembaga kesehatan, departemen pertahanan, sistem terdistribusi, sistem pendukung alat kerja, retail, sales dan supplier.

Blok pembangunan utama *UML* adalah diagram. Beberapa diagram ada yang rinci (jenis *timing diagram*) dan lainnya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorientasi objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. *UML* memungkinkan para anggota *team* untuk bekerja sama dengan bahasa model yang sama dalam mengaplikasikan beragam sistem. Intinya, *UML* merupakan alat komunikasi yang konsisten dalam men-*support* para pengembang sistem saat ini. Sebagai perancangan sistem, pasti akan menjumpai *UML*, baik dalam pembuatan sendiri ataupun hanya sekedar membaca diagram *UML* buatan orang lain. Tercantum pula logo UML[7] seperti gambar 2.5.



Gambar 2.5 Logo UML

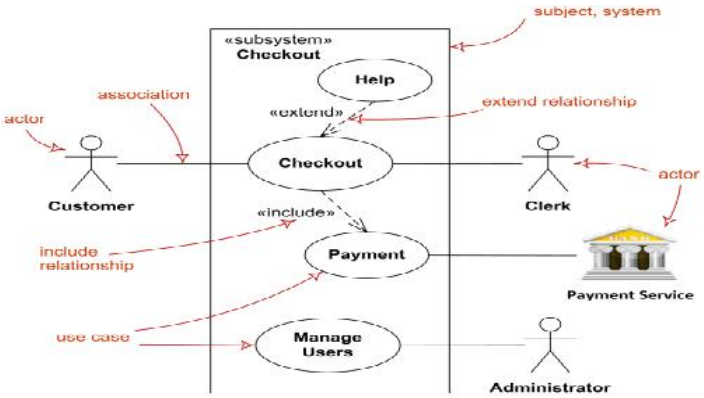
2.4.3 Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya.

Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case* diagram dapat sangat membantu apabila sedang menyusun *requirement*

sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Contoh use case diagram[7] ada pada gambar 2.6.



Gambar 2.6 Use Case Diagram

2.4.4 Class Diagram

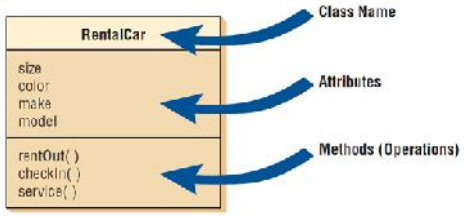
Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Sebuah contoh *class* diagram pada Rental Car [7], pada gambar 2.7

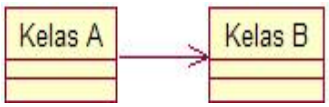


Gambar 2.7 Class Diagram

Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class Diagram* mempunyai 3 macam hubungan, sebagai berikut :

1. Association

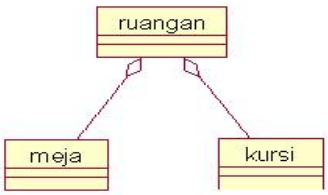
Association adalah hubungan statis antar kelas. Umumnya menggambarkan kelas yang memiliki atribut berupa kelas lain, atau kelas yang harus mengetahui ekstensi kelas lain. Dalam notasi UML kita mengenal asosiasi dua arah (bidirectional) dan satu arah (undirectional)[7]. Seperti contoh gambar 2.8



Gambar 2.8 *Association*

2. Aggregation

Aggregation adalah *association* dimana salah satu kelasnya merupakan bagian dari suatu kumpulan. *Aggregation* memiliki titik pusat yang mencakup keseluruhan bagian[7]. Seperti contoh gambar 2.9

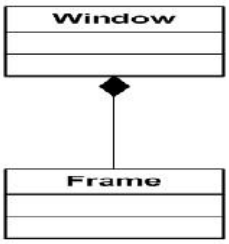


Gambar 2.9 *Aggregation*

3. Composition

Composition adalah *aggregation* dengan ikatan yang lebih kuat. Di dalam *composite aggregation*, siklus hidup *part class* sangat bergantung pada *whole*

class sehingga bila objek *instance* dari *whole class* dihapus maka objek *instance* dari *part class* juga akan terhapus[7]. Seperti pada gambar 2.10.



Gambar 2.10 *Composition*

4. Depedency

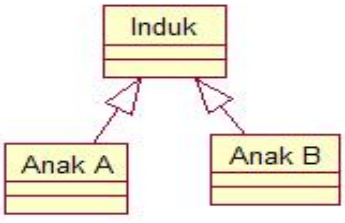
Depedency adalah hubungan antar-*class* di mana sebuah class memiliki ketergantungan pada class lainnya tetapi tidak sebaliknya[7]. Seperti pada gambar 2.11.



Gambar 2.11 *Depedency*

5. Generalization

Generalization diperlukan untuk memperlihatkan hubungan pewarisan (*inheritance*) antar unsur dalam diagram kelas. Pewarisan memungkinkan suatu kelas mewarisi semua atribut, operasi, relasi, dari kelas yang berada dalam hirarki pewarisannya[7]. Dicontohkan pada gambar 2.12.



Gambar 2.12 *Generalization*

2.4.5 Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang

mungkin terjadi dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum[6].

2.4.6 Sequence Diagram

Sequence diagram adalah suatu diagram yang memperlihatkan atau menampilkan interaksi-interaksi antar objek di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. Interaksi antar objek tersebut termasuk pengguna, *display*, dan sebagainya berupa ”pesan/*message*”.

Sequence diagram digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai sebuah respon dari suatu kejadian untuk menghasilkan *output* tertentu. *Sequence* diagram diawali dari apa yang me-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara *internal* dan *output* apa yang dihasilkan[6].

2.5 Media Pembuatan Perangkat Lunak

2.5.1 Java

Java adalah bahasa pemrograman serba guna. Java dapat digunakan untuk membuat suatu program sebagaimana anda membuatnya dengan bahasa seperti pascal atau C++. Yang lebih menarik, java juga mendukung sumber daya Internet yang saat ini popular, yaitu *World Wide Web* atau yang sering disebut Web saja. Java juga mendukung aplikasi klien/server, baik dalam jaringan *local* (LAN) maupun jaringan berskala luas (WAN).

Dalam sejumlah literatur disebutkan bahwa Java merupakan hasil perpaduan sifat dari sejumlah bahasa pemograman, yaitu C, C++, Objek-C, *SmallTalk*, dan Common LISP. Selain itu Java juga dilengkapi dengan unsure

keamanan. Yang tak kalah penting adalah bahwa Java menambahkan paradigma pemograman yang sederhana. Jika Anda telah mengenal C atau C++, yang mengandalkan *pointer* dan Anda dapat merasakan kesulitan, Java justru meninggalkannya sehingga Anda akan memperoleh kemudahan saat menggunakannya.

2.5.2 My SQL

MySQL adalah *multi user* database yang menggunakan bahasa *Strucktured Query Language* (*SQL*). *MySQL* merupakan *database server multi user* dan *multi threaded* yang tangguh. Dengan memiliki banyak *feature MySQL* bisa bersaing dengan *database* komersial sekalipun. Tidak mengejutkan *MySQL* menjadi database pilihan untuk banyak pengguna Delphi 6. *MySQL* termasuk jenis *RDBMS* (*Relational Database Management System*) itulah sebabnya istilah seperti tabel, baris dan kolom digunakan pada *MySQL*. Pada *MySQL* sebuah *database* mengandung satu atau sejumlah tabel. Tabel terdiri dari sejumlah baris dan setiap baris mengandung satu atau beberapa kolom.

MySQL mampu menangani data yang cukup besar. Perusahaan yang mengembangkan *MySQL* yaitu TcX, mengaku menyimpan data lebih dari 40 *database*, 10-000 tabel dan sekitar 7 juta baris, totalnya kurang lebih 100 *Gigabyte* data. SQL adalah bahasa standar yang digunakan untuk mengakses *database server*. Bahasa ini pada awalnya dikembangkan oleh IBM, namun telah diadopsi 25 dan digunakan sebagai standar industri.

Dengan menggunakan SQL, proses akses *database* menjadi lebih *user–friendly* dibandingkan dengan menggunakan DATABASE atau *Clipper* yang masih menggunakan perintah-perintah pemrograman. *MySQL* merupakan *software database* yang paling populer di lingkungan *Linux*, kepopuleran ini karena ditunjang performa *query* dari databasenya yang saat ini bisa dikatakan paling cepat dan jarang bermasalah. *MySQL* ini juga sudah dapat berjalan pada lingkungan *Windows*.

Beberapa karakteristik utama dari *MySQL* :

1. Fully multi-threaded dengan kernel threads. Artinya bisa dengan mudah mempergunakan multiple CPU bila ada.
2. API (Application Programming Interface) dengan C, C++, Delphi, Eiffel, Java, Perl, PHP, VB 6, Python dan Tcl.
3. Beroperasi pada banyak platform berbeda seperti : Linux, Win 95, Win 98, NT, Win 2000, Solaris, SunOS dll.
4. Kita bisa mencampurkan tabel dari database yang berbeda pada query yang sama.
5. Dukungan ODBC (Open-Database-Connectivity), misalkan dengan MS Access bisa melakukan koneksi ke MySQL.
6. Server bisa memberikan pesan kesalahan ke client dalam banyak bahasa.