BAB 2

LANDASAN TEORI

2.1. Citra

Citra adalah gambar pada bidang datar dua dimensi yang terdiri dari pixel – pixel. Sebuah citra digital memiliki tingkat kedalaman warna 24 bit, pada setiap pixelnya terdiri dari 3 unsur warna yaitu *Red* (R), *Green* (G) dan *Blue* (B) yang masing-masing memiliki kedalaman warna 8 bit (0-255) [3]. Nilai pada ketiga unsur warna tersebut (RGB) merupakan kode yang ditentukan oleh suatu badan yang disebut *International Electrotechnical Commision* (IEC) untuk mewakili setiap kombinasi warna pada komputer. Berikut ini merupakan salah satu contoh suatu citra.



Gambar 2.1 Citra Lena

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

2.2. Pengolahan Citra

Pengolahan citra merupakan kegiatan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia/mesin (komputer). Data masukannya adalah citra dan keluarannya juga citra tapi dengan kualitas lebih baik daripada citra masukan, misal citra warnanya kurang tajam, kabur (*blurring*), mengandung *noise* (misal bintik-

bintik putih), sehingga diperlukan pemrosesan untuk memperbaiki citra karena citra tersebut menjadi sulit diinterpretasikan karena informasi yang disampaikan menjadi berkurang [4]. Operasi-operasi pada pengolahan citra dapat diterapkan pada citra bila:

1. Dibutuhkannya peningkatan kualitas citra guna memperbaiki penampakan citra guna memunculkan aspek informasi yang terkandung dalam citra (*image enhancement*). Contoh dari operasi yang dilakukan seperti perbaikan kontras gelap/terang.
2. Terdapat cacat pada citra sehingga perlu dihilangkan atau diperkecil (*image restoration*). Contoh dari operasi yang dilakukan seperti penghilangan *noise*.
3. Adanya analisis citra guna mengetahui informasi yang ada dalam citra melalui elemen dalam citra yang perlu dikelompokkan, dicocokan atau diukur (*image segmentation*). Operasi ini berkaitan erat dengan pengenalan pola.
4. Dibutuhkannya ekstraksi ciri-ciri tertentu yang dimiliki citra untuk membantu dalam pengidentifikasian objek atau mencari informasi dari citra tersebut (*image analysis*). Contoh dari operasi yang dilakukan seperti pendeteksian tepi objek, ektraksi nilai *red-green-blue*.
5. Penggabungan citra dengan citra yang lain (*image reconstruction*). Contoh dari operasi yang dilakukan seperti beberapa foto *rontgen* digunakan untuk membentuk ulang gambar organ tubuh
6. Dibutuhkannya pemampatan citra (*image compression*) Contoh dari operasi yang dilakukan seperti suatu file citra berbentuk BMP berukuran 258 KB dimampatkan dengan metode JPEG menjadi berukuran 49 KB.
7. Digunakan untuk keperluan keamanan data seperti menyembunyikan data rahasia (berupa teks/citra) pada citra sehingga keberadaan data rahasia tersebut tidak diketahui orang (*steganografi*)

Terdapat beberapa operasi pengolahan citra yang dapat digunakan untuk mengolah citra agar dapat diinterpretasi. Beberapa jenis operasi pengolahan citra adalah sebagai berikut:

1. Pengabuan (*Grayscale*)

   Merupakan proses konversi citra menjadi citra keabuan (*grayscale*) dari warna sebenarnya (*true color*).
2. Pengambangan (*Thresholding*)

   Operasi pengambangan digunakan untuk mengubah citra dengan format skala keabuan, yang mempunyai kemungkinan nilai lebih dari 2 ke citra biner yang memiliki 2 buah nilai (yaitu 0 dan 1).
3. Pemotongan (*Cropping*)

   Memotong satu bagian dari citra sesuai kebutuhan. Pada penelitian ini citra menggunkan operasi pemotongan dari citra aslinya agar didapatkan citra tomat yang sesuai untuk digunakan.
4. Peng-skalaan (*Scaling*)

   Mengubah ukuran suatu citra menjadi lebih besar atau lebih kecil.
5. Deteksi Tepi (*Edge Detection*)

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

2.3. Pengenalan Pola

Pengenalan pola adalah suatu cara untuk deskripsi atau klasifikasi suatu data kedalam kelompok tertentu berdasarkan kesamaan karakteristik yang ada dalam citra digital. Pengenalan pola terbagi kedalam tiga cara yaitu metode statistik, metode sinaptik dan metode jaringan syaraf tiruan [5].

1. Metode Statistik

Model statistik didefinisikan sebagai sebuah keluarga dari fungsi kerapatan peluang bersyarat kelas Pr(|) i x c , yakni peluang vektor fitur x jika diberikan kelas ci. Metode ini dapat dilakukan dengan pendekatan *supervised* maupun pendekatan *unsupervised*. Pendekatan *supervised* (dengan pengarahan) menyediakan *training set* untuk mengarahkan atau memberi informasi atau pengetahuan tentang kelas-

kelas yang ada. Contoh pendekatan *supervised* adalah metode k-NN, *Bayes*, *Linear Discriminant Analysis*, dan lain-lain. Pendekatan *unsupervised* (tanpa pengarahan) tidak menyediakan *training set*. Informasi yang disediakan adalah jumlah klaster yang ada. Sehingga pengelompokan dilakukan sepenuhnya berdasarkan karakteristik data. Contoh dari pendekatan ini adalah metode *K-Means* [5].

2. Metode Sinaptik

Dalam metode sintaktik atau metode struktural, pola dipilah berdasarkan keserupaan ukuran struktural, pengetahuan direpresentasikan secara formal grammar atau deskripsi relasional (graf). Metode ini dipakai tidak hanya untuk pemilahan, tetapi juga untuk deskripsi. Biasanya, metode ini memformulasikan deskripsi hirarkis dari pola kompleks yang tersusun dari pola bagian yang lebih sederhana [5].

3. Metode Jaringan Syaraf Tiruan

Metode jaringan syaraf tiruan adalah metode yang kerap digunakan untuk klasifikasi. Dalam metode jaringan syaraf, pemilahan dilakukan berdasarkan tanggapan suatu *neuron* jaringan pengolah sinyal (*neuron*) terhadap stimulus masukan (pola). Pada metode ini informasi diolah berdasarkan *neuron* dan nilai bobot sehingga keakuratannya lebih baik dibandingkan dengan metode pengenalan pola lainnya. Contoh dari metode ini adalah LVQ, *Backprogpagation*, ANFIS, dan lain lain [5].

2.4. Jaringan Syaraf Tiruan

Jaringan syaraf tiruan adalah paradigma pemrosesan suatu informasi yang terinspirasi oleh sistem sel syaraf biologi, sama seperti otak yang memproses suatu informasi. Pembuatan struktur jaringannya meniru jaringan biologis terutama otak manusia. Metode ini mengolah informasi berdasarkan *neuron* dan pengetahuan yang disimpan dalam sambungan antar *neuron* dan kekuatan pembobot sinaptik. *Neuron* pada otak manusia merupakan satuan unit pemroses terkecil. Jaringan syaraf tiruan seperti manusia belajar dari suatu contoh, data – data sebelumnya dan mengenal pola data yang berubah – ubah [6].

2.4.1. Karakteristik Jaringan Syaraf Tiruan

Jaringan syaraf tiruan dibentuk untuk memecahkan suatu masalah tertentu seperti pengenalan pola atau klasifikasi karena proses pembelajaran dari data atau pola yang diberikan. *Neuron* sebagai penampung informasi yang diberikan dan akan diolah akan menstransformasikan informasi yang diterima melalui sambungan keluarannya (bobot) menuju *neuron – neuron* lain pada lapisan yang berbeda. Faktor terpenting yang dapat menentukan perilaku suatu *neuron* dalam suatu lapisan adalah fungsi aktivasi dan pola bobotnya. Karakteristik jaringan syaraf tiruan didasarkan pada ketiga hal [6], yaitu :

1. Arsitektur jaringan syaraf tiruan
2. Metode penentuan bobot
3. Fungsi aktivasi

2.4.1.1. Arsitektur Jaringan Syaraf Tiruan

Arsitektur jaringan syaraf tiruan merupakan gambaran hubungan antar *neuron*. Setiap informasi yang akan diolah ada pada *neuron – neuron* yang terkumpul dalam lapisan – lapisan *neuron* [6]. Adapun lapisan tersebut terbagi menjadi tiga bagian diantaranya :

1. Lapisan Masukan

   Lapisan ini berfungsi untuk menampung data masukan yang akan diteruskan/diolah ke lapisan berikutnya menuju lapisan keluaran.
2. Lapisan Tersembunyi

   Lapisan ini disebut juga dengan lapisan kompetitif yang berfungsi untuk mengolah data masukan untuk menuju lapisan keluaran. Pada lapisan – lapisan ini terdapat perhitungan bobot – bobot yang akan membuat data masukan tepat dalam klasifikasi.
3. Lapisan Keluaran

   Lapisan ini berfungsi untuk menampilkan data hasil pengolahan dari data masukan yang telah diolah.

Lapisan – lapisan tersebut ada pada arsitektur jaringan syaraf tiruan [6]. Adapun arsitektur jaringan syaraf tiruan yang sering digunakan yaitu :

1. Jaringan Syaraf dengan Lapisan Tunggal (*Single Layer Net*)

Jaringan dengan lapisan tunggal hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima masukan kemudian langsung mengolahnya menjadi keluaran tanpa melewati lapisan tersembunyi.

2. Jaringan Syaraf dengan Banyak Lapisan (*Multilayer Net)*

Jaringan syaraf *multilayer* atau disebut juga *mulitayer perception* merupakan pengembangan dari jaringan syaraf tiruan lapis tunggal. Arsitektur ini memiliki layer lebih dari layer pada arsitektur lapis tunggal yang terdiri dari lapisan masukan lapisan tersembunyi dan lapisan keluaran. Beberapa metode yang menggunakan arsitektur ini adalah *Learning Vector Quantization* dan *Adaptive Neuro Fuzzy Inference System*.

2.4.1.2. Metode Penentuan Bobot

Selain dari arsitektur jaringan, penentuan bobot juga merupakan karakteristik yang menentukan beraneka daram jaringan syaraf tiruan. Bobot merupakan suatu hubungan berupa bilangan *real* atau *integer*, tergantung dari jenis permasalahan dan model yang digunakan. Bobot – bobot tersebut dapat ditentukan untuk ada pada interval tertentu. Selama proses pembelajaran, bobot tersebut menyesuaikan dengan pola – pola masukan.

Adanya proses pembelajaran, membuat jaringan dengan sendirinya akan memperbaiki diri secara terus menerus. Setiap ada suatu masalah baru, jaringan dapat belajar dari masalah tersebut yaitu dengan mengatur kembali bobot untuk menyesuaikan karakter nilai. Terdapat tiga jenis proses pembelajaran yang ada pada jaringan syaraf tiruan, diantaranya :

1. Pembelajaran Tak Terawasi (*Unupervised*)

Metode pembelajaran ini tidak membutuhkan target nilai keluaran karena tidak dapat ditentukan hasil yang diharapkan seperti apa selama proses

pembelajaran. Pada proses pembelajaran, nilai bobot disusun dalam suatu range tertentu bergantung pada nilai masukan yang diberikan.

2. Pembelajaran Terawasi (*Supervised*)

Metode pembelajaran terawasi adalah metode yang target nilai keluaran yang diharapkan sudah diketahui sebelumnya. Pada proses pembelajaran, satu pola masukan akan diberikan ke satu *neuron* yang ada dalam lapisan masukan, pola tersebut akan dirambatkan ke *neuron* berikutnya hingga sampai pada *neuron* di lapisan keluaran. Salah satu metode yang termasuk kedalam metode pembelajaran terawasi ini adalah *Learning Vektor Quantivization*.

3. *Hybrid*

Pembelajaran *hybrid* merupakan pembelajaran gabungan antara pembelajaran terawasi dan pembelajaran tak terawasi. Pembelajaran yang dilakukan pada arsitektur ini mengikuti pembelajaran yang ada pada pembelajaran terawasi. Salah satu metode yang menggunakan arsitektur ini adalah *Adaptive Neuro Fuzzy Inference System*.

2.4.1.3. Fungsi Aktivasi

Setiap *neuron* memiliki keadaan internal yang disebut level aktivasi yang merupakan fungsi masukan yang diterima. Suatu *neuron* mengirimkan aktivasi kebeberapa *neuron* lain sebagai sinyal. *Neuron* tersebut hanya dapat mengirimkan satu sinyal sesaat, walaupun sinyal tersebut dapat dipancarkan ke beberapa *neuron* yang lain [6]. Ada beberapa fungsi aktivasi yang sering digunakan seperti fungsi :

1. *Threshold*

Fungsi *threshold* atau fungsi undak biner, fungsi ini menggunakan nilai ambang. Fungsi *threshold* (dengan nilai ambang 0) memiliki memiliki nilai keluaran 0 dan 1. Fungsi *threshold* dirumuskan :

$$y = \begin{cases} 0; & jika\ x < \theta \\ 1; & jika\ x \geq \theta \end{cases} \qquad \text{...(2.1)}$$

Keterangan :

x = Nilai masukan

y = Nilai keluaran

2. Symetric Hard Limit

Fungsi *Symetric Hard Limit* atau fungsi bipolar ini mengonversi masukan dari suatu variabel yang bernilai kontinu ke suatu keluaran berupa 1,0 dan -1. Fungsi *Symetric Hard Limit* dirumuskan :

$$y = \begin{cases} 1\,; & jika\ x > 0 \\ 0; & jika\ x = 0 \\ -1; & jika\ x < 0 \end{cases} \qquad ...(\ 2.2\ )$$

Keterangan :

x = Nilai masukan

y = Nilai keluaran

3. Fungsi *Symetric Saturating Linear*

Fungsi ini akan bernilai -1 jika masukannya kurang dari -1, dan akan bernilai 1 jika masukannya lebih dari 1. Sedangkan jika masukan terletak antara -1 dan 1, maka keluarannya akan bernilai sama dengan nilai masukannya. Fungsi *symetric saturating linear* dirumuskan :

$$y = \begin{cases} 1; jika\ x \geq 1 \\ x; jika -1\ \leq x \leq 1 \ldots \\ -1; jika\ x \leq -1 \end{cases} \qquad ...(\ 2.3\ )$$

Keterangan :

x = Nilai masukan

y = Nilai keluaran

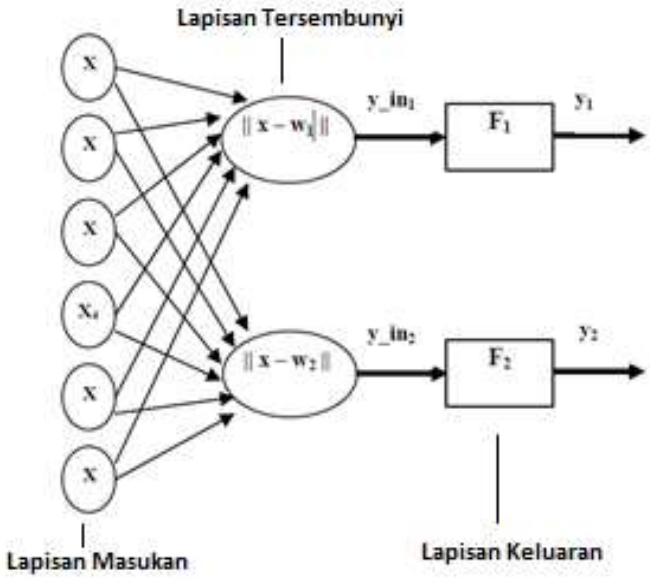
2.5 Learning Vector Quantization (LVQ)

Learning Vector Quantization (LVQ) merupakan salah satu metode *supervised*. Proses pembelajaran ada pada lapisan kompetitif yang terawasi, sehingga pada metode ini pengklasifikasian pola dilakukan dengan setiap unit keluaran mewakili satu kelas tertentu atau satu kategori tertentu.

Metode LVQ menggunakan vektor acuan dari satu unit keluaran yang menjadi acuan bagi kelas/kategori yang diwakili oleh keluaran tersebut. Pendekatan yang dilakukan adalah dengan mengelompokkan vektor masukan berdasarkan jarak terkecil vektor masukan terhadap bobot (metode *Euclidean distance minimum*). Vektor yang memiliki jarak terkecil adalah vektor pewakil pemenang untuk menentukan kategori vektor masukan [6].

1. Langkah –Langkah *Learning Vektor Quantization*

Arsitektur jaringan syaraf LVQ pada dasarnya sama dengan *Kohonen Self Organizing Map*. Jaringan syaraf tiruan LVQ terdiri dari layer masukan, layer kompetitif (layer tersembunyi) dan layer keluaran seperti yang terlihat pada masing-masing keluaran mempunyai kelas yang telah diketahui [6].



Gambar 2.2 Struktur *Learning Vector Quantization*

Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor masukan melalui jarak terdekat antar data pada tiap bobot. Jika dua vektor masukan mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor masukan tersebut ke dalam kelas yang sama. Pemrosesan yang terjadi pada setiap *neuron* adalah mencari jarak antara suatu vektor masukan terhadap bobot yang bersangkutan (w_1 dan w_2). Bobot w_1 adalah vektor yang menghubungkan setiap *neuron* pada lapisan masukan ke *neuron* pertama pada lapisan keluaran, sedangkan w_2 adalah vektor bobot yang menghubungkan setiap *neuron* pada lapisan masukan ke *neuron* kedua pada lapisan keluaran. Kemudian fungsi aktifasi :

F_1 akan memetakan Y_{in1} ke $y_1 = 1$ apabila $||x - w_1|| < ||x - w_2||$, dan $y_1 = 0$ jika sebaliknya $||x - w_1|| > ||x - w_2||$.

F_2 akan memetakan Y_{in2} ke $y_2 = 1$ apabila $||x - w_2|| < ||x - w_1||$, dan $y_2 = 0$ jika sebaliknya $||x - w_2|| > ||x - w_1||$.

Keterangan :

x	=	Vektor pembelajaran
w_j	=	Vektor bobot untuk keluaran ke-J (w_{1j}, ..., w_{ij},.....,w_{nj})
$\|\|x- w_j\|\|$	=	Jarak *Eucladian* antara vektor masukan dan (vektor bobot untuk) keluaran ke-J
F_j	=	Fungsi aktivasi ke-J

Algoritma untuk LVQ adalah dengan mencari keluaran yang terdekat dengan vektor masukan [6] sebagai berikut :

Menetapkan bobot (w), *learning rate* (α), maksimum *epoch* (*MaxEpoch*), *error* minimum yang diharapkan (*Eps*).

1. Masukkan :
   i. *Input* : x(m,n)
   ii. Target : T(1,n)
2. Tetapkan kondisi awal :
   i. *Epoch* = 0;

3. Kerjakan jika : (*epoch*< *MaxEpoch*) atau (α > *eps*)
   i. *Epoch* +1;
   ii. Kerjakan untuk i = 1 sampai n
   iii. Mencari $C_j = \min \|x - w_j\|$.
   iv. *Update* W_J dengan cara :
      a. Jika $T = C_j$ kemudian :

$$w_j(\text{baru}) = w_j(\text{lama}) + \alpha[x - w_j(\text{lama})] \quad ...(2.4)$$

      b. Jika $T \neq C_j$ kemudian :

$$w_j(\text{baru}) = w_j(\text{lama}) - \alpha[x - w_j(\text{lama})] \quad ...(2.5)$$

4. Mengurangi *learning rate* dengan cara

$$\alpha = \alpha - (0.1 * \alpha) \quad ...(2.6)$$

Keterangan :

x	=	Vektor pembelajaran
T	=	Kategori atau kelas tepat untuk vektor pembelajaran
w_j	=	Vektor bobot untuk keluaran ke-J (w_{1j}, ..., w_{ij},.....,w_{nj})
C_j	=	Kategori atau kelas yang ditunjukkan oleh keluaran ke–J
$\|\|x- w_j\|\|$	=	Jarak *Eucladian* antara vektor masukan dan (vektor bobot untuk) keluaran ke-J
F_j	=	Fungsi aktivasi ke-J

Setelah proses pembelajaran LVQ maka langkah selanjutnya adalah proses pengambilan keputusan dari hasil keluaran LVQ. Pengambilan keputusan ini dilakukan hanya pada proses *mapping* (pengenalan). Pada dasarnya, tahapan ini hanya mencari jarak terdekat dengan perhitungan *euclidian* (jarak terdekat).

2.6. Adaptive Neuro Fuzzy Inference System (ANFIS)

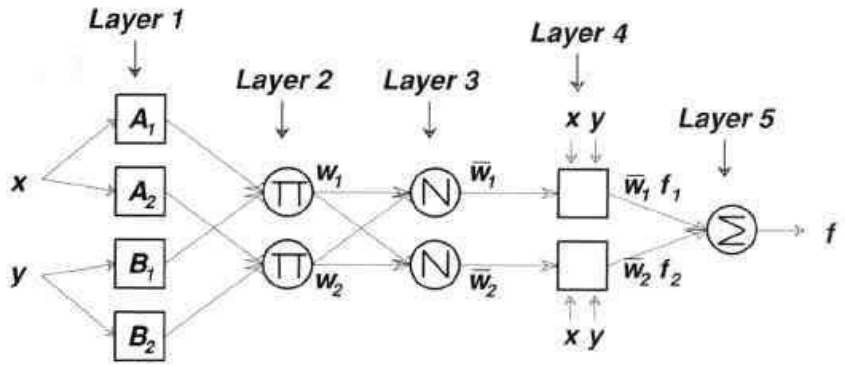
Adaptive Neuro Fuzzy Inference System (ANFIS) merupakan suatu metode gabungan mekanisme dari sistem inferensi fuzzy dengan jaringan syaraf tiruan. Sistem inferensi fuzzy dapat menerjemahkan pengetahuan pakar dalam bentuk aturan – aturan yang diimplementasikan pada bentuk *IF - THEN*, namun karena tidak memiliki kemampuan pembelajaran membuat waktu yang diperlukan untuk

menetapkan fungi keanggotaannya cukup lama. Sedangkan jaringan syaraf tiruan memiliki kemampuan pembelajaran, tetapi tidak bisa menjelaskan proses penalaran yang dilakukan karena pengetahuan yang dimilikinya hanya berupa bobot-bobot sinapsis yang biasanya bernilai real. JST tidak memiliki aturan-aturan *IF – THEN* sama sekali, sehingga metode sistem inferensi fuzzy menggunakan teknik pembelajaran yang dimiliki oleh jaringan syaraf tiruan untuk menetapkan fungsi keanggotaan dengan meminimalis waktu pencarian. Meskipun karakteristik dari sistem inferensi fuzzy dan jaringan syaraf tiruan saling bertolak belakang, namun dengan menggabungkan kedua metode tersebut didapatkan sebuah metode yang sangat baik diterapkan dalam berbagai bidang.

Pada ANFIS, sistem inferensi fuzzy yang digunakan adalah sistem inferensi fuzzy model Tagaki-Sugeno-Kang (TSK) orde satu karena kesederhanaan dan kemudahan komputasi yang ada pada model Tagaki-Sugeno-Kang (TSK) [7].

1. Langkah –Langkah *Adaptive Neuro Fuzzy Inference System*

Sistem inferensi fuzzy model Tagaki-Sugeno-Kang (TSK) direpresentasikan sebagai jaringan syaraf *feed forward* lima lapis khusus pada struktur ANFIS yang digambarkan dalam arsitektur jaringan syaraf tiruan seperti berikut :



Gambar 2.3 Struktur ANFIS

Pada Gambar 2.3 terlihat sistem inferensi fuzzy sistem yang digambarkan pada arsiterktur jaringan syaraf tiruan terdiri atas lima lapisan dengan fungsi yang berbeda- beda. Terdapat juga lambang kotak atau lingkaran pada lapisannya. Lambang kotak menandakan simpul adaptif yang artinya nilai parameter dapat

berubah dengan pembelajaran dan lambang lingkaran menandakan simpul nonadaptif yang nilainya tetap. Berikut merupakan penjelasan dari tiap lapisan yang ada dalam struktur ANFIS :

1. Lapisan 1

Semua simpul pada lapisan ini adalah simpul adaptif (parameter yang berubah) dengan fungsi simpul :

$$O_{1.i} = \mu_A(x), untuk\ i = 1,2\ atau$$
$$O_{1.i} = \mu_{B-2}(y), untuk\ i = 3,4 \quad ...(2.7)$$

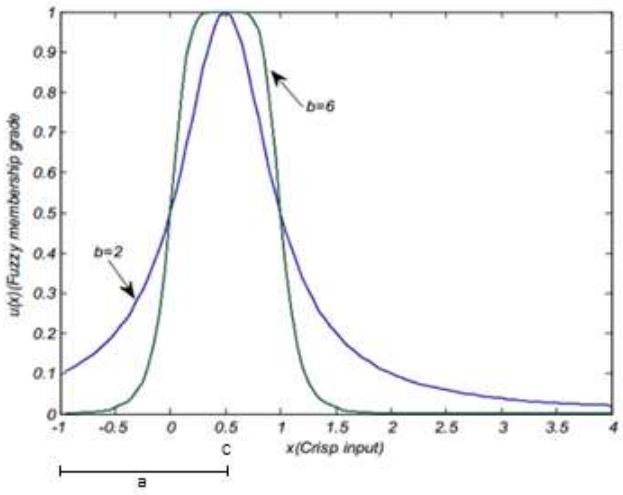
Keterangan :

x	=	Nilai masukan
y	=	Nilai masukan
μ_i	=	Nilai keanggotaan kelas-i
$O_{1,i}$	=	Nilai keluaran lapisan 1 ke-i

Dengan x dan y adalah masukan pada simpul i, A_i (atau B_i-2) adalah fungsi keanggotaan masing – masing simpul. Simpul $O_{1,i}$ berfungsi untuk menyatakan derajat keanggotaan tiap masukan terhadap himpunnan fuzzy A dan B. Fungsi keanggotaan yang dipakai adalah jenis *generalized bell* (*gbell*) yang dirumuskan sebagai berikut :

$$gbell(x, a, b, c) = \frac{1}{1 + \left|\frac{x - c}{a}\right|^{2b}} \quad ..(2.8)$$

Koefisien b: koefisien adalah angka positif, dan biasanya bernilai 1 yang menandakan *slopes* atau nilai terkait dengan bagian atas kurva dan menandakan bahwa kurva berbentuk lonceng dan penyebab naik ke bagian datar. Koefisien a menyatakan lebar dari titik pusat hingga batas keanggotaan dan koefisien c menyatakan titik pusat lonceng. Nilai a,b,c dan jumlah himpunan masukan fuzzy (x) ditentukan pada awal proses dan akan berubah melalui proses pembelajaran dinamakan parameter premis yang adaptif. Fungsi keanggotaan *generalized bell* (*gbell*) tersebut dapat dilihat pada Gambar 2.4 berikut ini :



Gambar 2.4 *Generalized Bell (Gbell)*

2. Lapisan 2

Semua simpul pada lapisan ini adalah nonadaptif (parameter tetap). Fungsi simpul ini adalah mengalikan setiap sinyal masukan yang datang. Fungsi simpul :

$$O_{2.i} = w_i = \mu_{A_i}(x).\mu_{B_i}(y), untuk\ i = 1,2 \quad ...(2.9)$$

Keterangan :

x	=	Nilai masukan
y	=	Nilai masukan
w_i	=	Bobot ke-i
μ_i	=	Nilai keanggotaan kelas-i
$O2_{,i}$	=	Nilai keluaran lapisan 2 ke-i

Tiap keluaran simpul menyatakan derajat pengaktifan (*firing strength*) tiap aturan fuzzy. Fungsi ini dapat diperluas apabila bagian premis memiliki lebih dari dua himpunan fuzzy. Banyaknya simpul pada lapisan ini menunjukkan banyaknya aturan yang dibentuk. Fungsi perkalian yang digunakan adalah interpretasi kata hubung *and* dengan menggunakan operator *t-norm*.

3. Lapisan 3

Lapisan Setiap simpul pada lapisan ini adalah simpul nonadaptif yang menampilkan fungsi derajat pengaktifan ternomalisasi (*normalized firing strength*)

yaitu rasio keluaran simpul ke-i pada lapisan sebelumnya terhadap seluruh keluaran lapisan sebelumnya, dengan bentuk fungsi simpul:

$$O_{3.i} = \bar{w}_i = \frac{w_i}{w_1 + w_2}, untuk\ i = 1,2 \qquad ...(2.10)$$

Keterangan :

w_i	=	Bobot ke-i
$\bar{w}_i$	=	derajat pengaktifan ternomalisasi ke-i
$O_{3,i}$	=	Nilai keluaran lapisan 3 ke-i

Apabila dibentuk lebih dari dua aturan, fungsi dapat diperluas dengan membagi w_i dengan jumlah total w untuk semua aturan.

4. Lapisan 4

Setiap simpul pada lapisan ini adalah simpul adaptif dengan fungsi simpul :

$$O_{4.i} = \bar{w}_i f_i = \bar{w}_i(p_i x + q_i y + r_i) \qquad ...(2.11)$$

Dengan $\bar{w}_i$ adalah derajat perngaktifan ternormalisasi dari lapisan 3 dan parameter p, q, r menyatakan parameter konsekuen yang adaptif.

Keterangan :

$\bar{w}_i$	=	derajat pengaktifan ternomalisasi ke-i
f_i	=	sinyal kontrol ke – i
$O_{4,i}$	=	Nilai keluaran lapisan 4 ke-i

5. Lapisan 5

Pada lapisan ini hanya ada satu simpul tetap yang fungsinya untuk menjumlahkan semua masukan dengan fungsi simpul sebagai berikut :

$$O_{5.i} = \sum_i \bar{w}_i f_i = \frac{\sum \bar{w}_i f_i}{\sum w_i} \qquad ...(2.12)$$

Keterangan :

$\bar{w}_i$ = derajat pengaktifan ternomalisasi ke-i

f_i = sinyal kontrol ke –i

$O_{5,i}$ = Nilai keluaran lapisan 5 ke-i

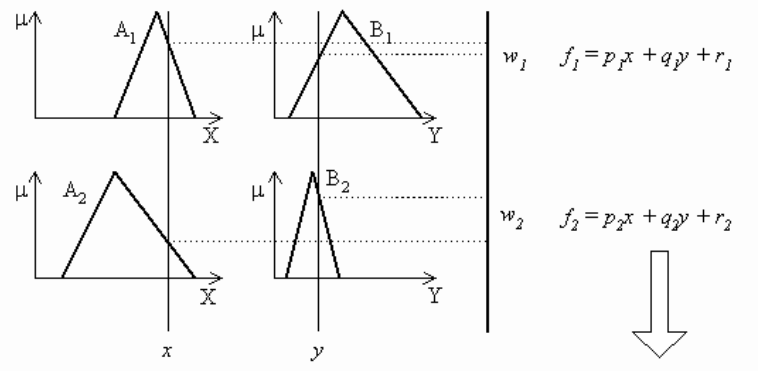
Jaringan adaptif dengan lima lapisan tersebut ekivalen dengan sistem inferensi fuzzy TSK.

Pada ANFIS, contoh mekanisme dari sistem inferensi fuzzy model Tagaki-Sugeno-Kang (TSK) orde satu dengan dua aturan fuzzy *if-then* seperti di bawah ini:

Rule 1 : *if* x is A_1 and y is B_1 *then* $f_1 = p_1x + q_1y + r_1$

Rule 2 : *if* x is A_2 and y is B_2 *then* $f_2 = p_2x + q_2y + r_2$

Nilai x dan y merupakan nilai masukan dan f merupakan konsekuen. Parameter *"x is A_1 and y is B_1"* merupakan parameter premis dan parameter *"$f_1 = p_1x + q_1y + r_1$"* merupakan parameter konsekuen. Dalam struktur fuzzy ANFIS digambarkan pada Gambar 2.5 seperti berikut ini :



$$f=\frac{w_1f_1+w_2f_2}{w_1+w_2} = \overline{w_2}f_2 + \overline{w_2}f_2$$

Gambar 2.5 Sistem Inferensi Fuzzy TSK dua masukan dengan dua aturan

Pada gambar 2.4 terlihat nilai x dan y sebagai suatu masukan *crisp* (tidak fuzzy). Sehingga nilai masukan tersebut perlu dipetakan kedalam fungsi keanggotaannya. Dalam gambar tersebut dimisalkan fungsi keanggotaannya berupa A_1, A_2, B_1 dan B_2.

Nilai masing – masing pasangan dari masukan tersebut lalu diagregasi dengan operasi *T-Norm*. Misalnya dengan operator *AND* menjadi w_1 = (mA_1 *AND* mA_2)

dan w_2 = (mB_1 *AND* mB_2). Sehingga dari basis aturan yang telah dibuat sebelumnya didapatkan :

if w=w_1 *then* $f_1 = p_1x + q_1y + r_1$

if w=w_2 *then* $f_2 = p_2x + q_2y + r_2$

Proses selanjutnya setelah didapat nilai w adalah memproses f_1 dan f_2 yang merupakan sinyal kontrol, terdapat nilai p_1, q_1, r_1, p_2, q_2 dan r_2 yang merupakan parameter konsekuen yang ditentukan oleh nilai awal dan akan berubah dengan pembelajaran. Untuk mendapatkan satu nilai kontrol, nilai f_1 dan f_2 dihitung dengan persamaan :

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} = \overline{w_2} f_2 + \overline{w_2} f_2 \qquad \text{...(2.13)}$$

Keterangan :

x	Nilai masukan
y	Nilai masukan
w_i	Bobot ke-i
$\overline{w}_i$	derajat pengaktifan ternomalisasi ke-i
f_i	sinyal kontrol ke –i
f	sinyal kontrol hasil defuzzifikasi

2. Proses Belajar ANFIS

Proses belajar yang ada digunakan untuk memperbaharui parameter premis dan parameter konsekuen yang ada pada jaringan ANFIS. Pembelajaran *hybrid* yang dilakukan terdiri atas dua bagian yaitu arah maju (*forward pass*) dan arah mundur (*backward pass*) [7].

Pada arah maju, parameter premis dibuat tetap. Dengan menggunakan metode *Recursive Least Square Estimator* (RLSE), parameter konsekuen diperbaiki berdasarkan pasangan data masukan – keluaran. Metode RLSE dapat diterapkan karena parameter konsekuen yang diperbaiki adalah parameter linier. Metode RLSE akan mempercepat proses belajar *hybrid*. Kemudian setelah parameter

konsekuen didapatkan, data masukan dilewatkan jaringan adaptif kembali dan hasil keluaran jaringan adaptif ini dibandingkan dengan keluaran yang sebenarnya.

Pada arah mundur, parameter konsekuen dibuat tetap. Kesalahan yang terjadi antara keluaran jaringan adaptif dan keluaran sebenarnya dipropagasikan balik dengan menggunakan *gradient descent* untuk memperbaiki parameter premis. Pembelajaran ini dikenal sebagai algoritma *Backpropagation-error*. Kedua pembelajaran tersebut dapat dilihat pada tabel 2.1 berikut ini :

Tabel 2.1 Proses Pembelajaran ANFIS

	Arah Maju	Arah Mundur
Parameter Premis	Tetap	*Gradient Descent*
Parameter Konsekuen	RLSE	Tetap
Sinyal	Keluaran Simpul	Laju Kesalahan

3. Proses Belajar dengan RLSE untuk Parameter Konsekuen

Pada proses ANFIS lapisan 4, terlihat terdapat parameter p,q dan r. parameter tersebut merupakan parameter konsekuen yang didapatkan dengan proses pembelajaran RLSE [8]. Persamaan untuk mendapatkan parameter konsekuen adalah :

$$\theta = A^{-1}.Y = (Inv(A^T.A)A^T).Y \qquad ...(2.14)$$

Untuk menghitung nilai pada persamaan 2.14, maka dibutuhkan sebuah matriks yang terdiri atas nilai perkalian antara hasil keluaran lapisan 3 dengan data masukan seperti terlihat pada persamaan 2.15.

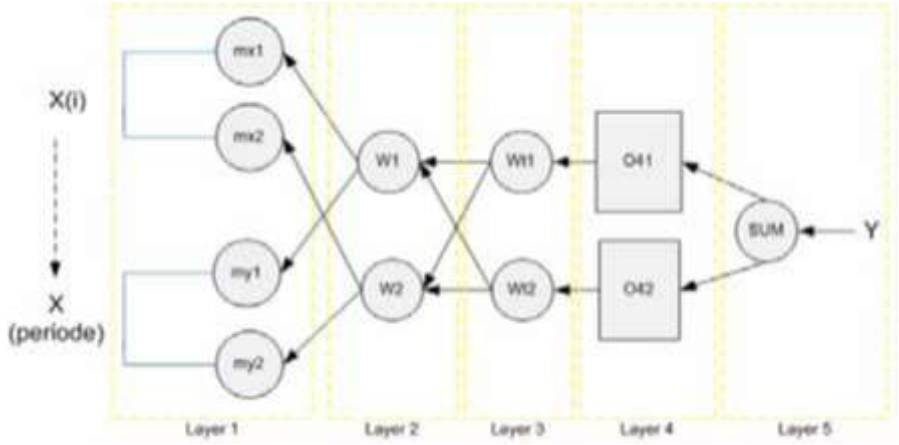
$$A = \begin{bmatrix} X_i(\overline{W_1}) & Y_i(\overline{W_1}) & \overline{W_1} & \cdots & X_i(\overline{W_n}) & Y_i(\overline{W_n}) & \overline{W_n} \\ X_n(\overline{W_1}) & Y_n(\overline{W_1}) & \overline{W_1} & & X_n(\overline{W_n}) & Y_n(\overline{W_n}) & \overline{W_n} \end{bmatrix} \qquad ..(2.15)$$

Setelah dilakukan perhitungan RLSE pada matriks tersebut, nilai keluaran Θ adalah nilai p,q dan r sebagai nilai untuk parameter konsekuen.

$$\Theta = \begin{bmatrix} p_1 \\ q_1 \\ r_1 \\ s_1 \\ \vdots \\ p_{18} \\ q_{18} \\ r_{18} \\ s_{18} \end{bmatrix}^T$$

4. Proses Belajar dengan Backprogpagation-*Error* untuk Parameter Premis

Parameter premis adalah parameter adaptif dengan proses pembelajaran yang menggunakan metode belajar sistem jaringan syaraf *feedforward* dengan *gradient descent*. Untuk memperbaharui parameter premis adalah melalui perhitungan nilai *error* dari tiap simpul/node pada tiap lapisan dengan alur mundur dari lapisan 5 hingga lapisan 1 dengan menggunakan *Error Backprogpagation* [7].



Gambar 2.6 Alur Mundur Proses *Error* Backprogpagation

Pada lapisan 5 dilakukan perhitungan *error* dengan persamaan 2.16. Pada perhitungan *error* lapisan ini melibatkan nilai keluaran lapisan 5 dan Y yang merupakan nilai keluaran yang diharapkan.

$$\varepsilon_{5,1} = -2(Y - O_{5,1}) \qquad \text{...(2.16)}$$

Pada lapisan 4 tidak dilakukan perhitungan *error* karena pada alur mundur tidak akan terjadi pembaharuan nilai parameter konsekuen yang terdapat pada

lapisan 4. Sehingga nilai *error* pada tiap simpul pada lapisan 4 bernilai sama dengan nilai *error* pada lapisan 5.

$$\varepsilon_{4,i} = \varepsilon_5 \left(\frac{\partial f_5}{\partial O_{4,i}}\right) = \varepsilon_5 = 1 \qquad ...(2.17)$$

Pada lapisan 3, perhitungan nilai *error* pada lapisan ini melibatkan nilai keluaran lapisan 4 dan nilai *error* lapisan 4.

$$\varepsilon_{3,i} = O_{4,i} * \varepsilon_{4,i} \qquad ...(2.18)$$

Pada lapisan 2, perhitungan *error* melibatkan nilai keluaran lapisan 2 dan nilai *error* lapisan 3.

$$\varepsilon_{2,i} = \left(\frac{W_i}{(w_1+\cdots+w_{18})^2}\right)\varepsilon_{3,i} \qquad ...(2.19)$$

Pada perhitungan *error* untuk lapisan 1 ini melibatkan nilai keanggotaan pada simpul ada lapisan 1 dan nilai *error* lapisan 2.

$$\varepsilon_{1,i} = \frac{\sum \varepsilon_{2,i} * \mu(i)}{n} \qquad ...(2.20)$$

Langkah selanjutnya setelah mendapatkan nilai *error* untuk setiap lapisan adalah memperbaharui nilai parameter premis dengan cara mencari terlebih dahulu nilai fungsi keanggotaan yang baru.

$$\mu_{\text{i baru}} = \mu_{\text{i lama}} + \varepsilon_{\text{i}} \qquad ...(2.21)$$

Setelah mendapatkan nilai fungsi keanggotaan yang baru, langkah selanjutnya adalah mencari nilai turunan dari *g-bell*. Persamaan 2.22 adalah persamaan turunan untuk parameter a, persamaan 2.23 adalah persamaan turunan untuk parameter b dan 2.24 persamaan turunan untuk parameter c.

$$\frac{d\mu_i}{da} = \frac{2b}{a}\mu_i(1-\mu_i) \qquad ...(2.22)$$

$$\frac{d\mu_i}{db} = \begin{cases} -2\,ln\left|\frac{x-c}{a}\right|\mu_i\,(1-\mu_i) & x \neq c \\ 0 & x = c \end{cases} \qquad ...(2.23)$$

$$\frac{\mathrm{d}\mu_{\mathrm{i}}}{\mathrm{dc}} = \begin{cases} -\frac{2b}{x-c}\mu_i(1-\mu_i) & x \neq c \\ 0 & x = c \end{cases} \qquad ...(2.24)$$

Nilai - nilai hasil perhitungan turunan pada persamaan 2.22 sampai 2.24 akan dijumlahkan dengan nilai parameter premis yang lama untuk mendapatkan nilai parameter premis yang baru.

$$a_{baru}=a_{lama}+\frac{d\mu_i}{da} \quad ...(2.25)$$

$$b_{baru}=b_{lama}+\frac{d\mu_i}{db} \quad ...(2.26)$$

$$c_{baru}=c_{lama}+\frac{d\mu_i}{dc} \quad ...(2.27)$$

Langkah terakhir adalah membagi total jumlah dari parameter premis dengan jumlah data untuk mendapatkan parameter premis yang akan digunakan untuk iterasi selanjutnya.

$$a_{baru}=\frac{\sum a_i}{n} \quad ...(2.28)$$

$$b_{baru}=\frac{\sum b_i}{n} \quad ...(2.29)$$

$$c_{baru}=\frac{\sum c_i}{n} \quad ..(2.30)$$

2.7. Ukuran Ketepatan Pengujian

Untuk mengukur suatu pengujian dapat dilakukan dengan menghitung selisih antara data hasil perhitungan dengan data tujuan aktual. Selisih besaran (ukuran kesalahan pengujian) data pengujian terhadap data aktual yang terjadi merupakan suatu data penting untuk menilai ketepatan suatu metode pengujian. Dengan membandingkan ukuran kesalahan beberapa metode pengujian, akan diperoleh metode mana yang mempunyai ukuran kesalahan terkecil, sehingga nilai pengujian dapat dipakai sebagai acuan dalam menentukan kebutuhan-kebutuhan dimasa yang akan datang. Terdapat beberapa ukuran kesalahan dalam pengujian antara lain RMSE.

a. *Root Mean Squared Error* (RMSE)

RMSE merupakan nilai tengah kesalahan akar kuadrat dengan persamaan sebagai berikut :

$$RMSE=\sqrt{\frac{\sum_{i=1}^{n}(X_t-(F_t))^2}{n}} \quad ...(2.31)$$

RMSE merupakan metode alternatif untuk mengevaluasi teknik peramalan. RMSE adalah rata-rata kuadrat dari perbedaan nilai estimasi dengan nilai observasi suatu variabel. Jika nilai RMSE semakin kecil maka estimasi model atau variabel tersebut semakin valid. Notasi dasar dari persamaan diatas secara ringkas adalah sebagai berikut :

X_t = nilai aktual pada periode *t*

F_t = nilai pengujian pada periode *t*

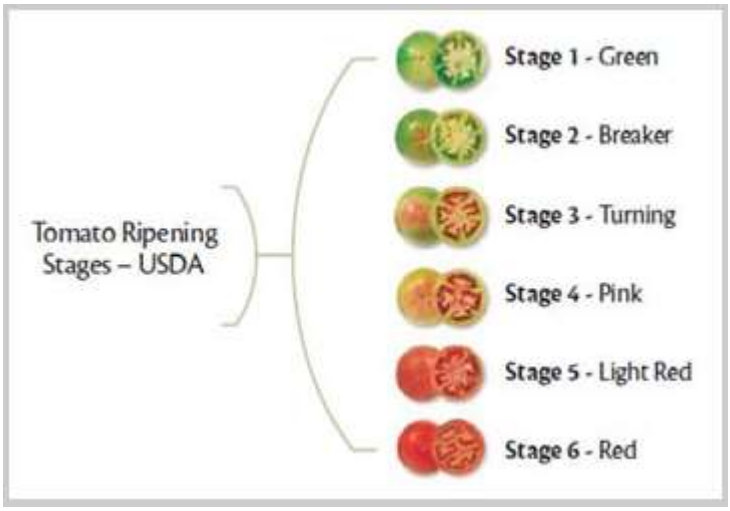
$X_t - F_t$ = nilai kesalahan pengujian

n = jumlah data

2.8. Tomat

Tomat (*Solanum lycopersicum syn. Lycopersicum esculentum*) merupakan tumbuhan topis yang tumbuh didaerah lembab dan dingin. Tomat berbentuk bulat dan berwarna merah ketika matang. Selama proses pematangan, tomat akan mengalami berbagai perubahan baik dari segi fisik maupun kimia. Perubahan segi fisik diantaranya ukuran, warna dan kelunakan buah. Perubahan – perubahan tersebut sangat mempengaruhi mutu tomat.

Pematangan buah tomat dapat diketahui dengan melihat perubahan warna kulit buah tomat. Warna kulit buah tomat akan berubah dari hijau penuh menjadi merah penuh [9]. Klasifikasi perubahan warna kulit tomat dapat dijelaskan pada gambar berikut ini :



Gambar 2.7 Standar Internasional "*Tomato Ripened Stage*"

Setiap *stage* memiliki fase dengan karakteristik fisik yang berbeda – beda yang dapat didekripsikan sebagai berikut :

1. *Stage* 1 (*Green*), pada fase ini permukaan tomat benar-benar warna hijau. Warna hijau dapat bervariasi terang ke gelap.
2. *Stage* 2 (*Breaker*), pada fase ini permukaan tomat berwana hijau namun terdapat warna kuning atau oranye tidak lebih dari 10%.
3. *Stage* 3 (*Turning*), pada fase ini terlihat perubahan yang pasti dari hijau ke kuning atau oranye, namun dominasi warna kuning atau oranye yang ada pada permukaan hanya berkisar antara 10% hingga 30%.
4. *Stage* 4 (*Pink*), pada fase ini warna sudah memasuki warna oranye sudah memasukin warna merah muda yang berkisar antara 30% hingga 60%.
5. *Stage* 5 (*Light Red*), pada fase ini permukaan tomat berwarna merah muda atau oranye gelap yang berkisar antara 60% hingga 90%
6. *Stage* 6 (*Red*), pada fase ini permukaan tomat benar-benar berwarna merah.

2.9. Model Analisis Dan Perancangan

Model analisis yang digunakan untuk membangun sistem informasi ini adalah sebagai berikut :

1. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak. ERD merupakan model jaringan data yang menekankan pada struktur-struktur dan *relationship* data. Diagram hubungan entitas atau yang lebih dikenal dengan sebutan E-R diagram, adalah notasi grafik dari sebuah model data atau sebuah model jaringan yang menjelaskan tentang data yang tersimpan dalam sistem secara abstrak. Diagram hubungan entitas tidak menyatakan bagaimana memanfaatkan data, membuat data, mengubah data dan menghapus data [10].

2. *Unified Modeling Language* (UML)

UML (*Unified Modeling Language*) adalah sebuah bahasa untuk menetukan, visualisasi, kontruksi, dan mendokumentasikan artifact (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. *Artifact* dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan system non perangkat lunak lainnya[[10]]. dan terdapat bagian-bagian diagram sebagai berikut :

1. Use case diagram

Use case diagram merupakan suatu aktivitas sistem dalam menanggapi permintaan pengguna dalam sebuah masalah dimana sistem yang digunakan oleh pengguna (*actor*) [10].

2. Activity diagram

Activity diagram dapat membantu untuk menggambarkan aliran kontrol dari sistem, seperti aturan bisnis menggambarkan jalannya fungsi yang ada dan menggambarkan kasus penggunaan juga proses bisnis [10].

3. Sequence diagram

Sequence diagram merupakan kolaborasi objek berdasarkan urutan waktu. Sequence diagram menunjukkan bagaimana objek berinteraksi dengan objek lain dalam skenario tertentu dari sebuah *use case* [10].

4. Class diagram

Class diagram memberikan gambaran tentang sistem target dengan menggambarkan objek dan kelas dalam sistem dan hubungan antara mereka. Pada *class diagram* juga terdapat atribut dan *method – method* yang ada pada sistem [10].

2.10. Pengujian

Pengujian yang dilakukan pada penelitian ini menggunakan *cross validation* dan *confusion matrix* sebagai berikut :

2.10.1. Cross Validation

Cross validation adalah salah satu cara untuk menemukan parameter terbaik dari satu model dengan cara menguji besarnya *error* pada tes. Dalam *cross validation*, data dibagi ke dalam *k* subset dengan ukuran yang sama secara acak dan diulang sebanyak *k* kali. Untuk menggunakan cross validation ini perlu dilakukan kombinasi fold agar dapat membantu proses pembelajaran dan pengujian dimana salah satu fold akan menjadi data uji dan sisa fold yang lain digunakan untuk pembelajaran. Kemudian error rata – rata di semua cobaan *k* dihitung. Model yang memberikan rata – rata *error* kecil terkecil adalah model yang terbaik [11].

Dari *m* set data, alur k- fold [12] adalah sebagai berikut :

a. Bagi data menjadi subset / *fold* sejumlah *k* yang digunakan secara acak. Ukuran untuk setiap subset adalah *m*/*k.*

b. Ulangi proses *i* = 1 hingga *k*

   i. Lakukan proses pembelajaran untuk semua *fold* kecuali *fold – i*

   ii. Lakukan proses pengujian untuk *fold – i* menggunakan hasil pembelajaran dari fold yang telah digunakan saat proses pembelajaran.

   iii. Hitung *n*, dimana jumlah *error* klasifikasi dari pengujian *fold – i*.

c. Hitung nilai rata – rata *error* klasifikasi (*E*) dengan menggunakan rumus

$$E = \frac{\sum_1^k n_i}{m} \qquad \text{...(2.32)}$$

Untuk mengetahui akurasi metode terhadap klasifikasi yang dilakukan dapat dilakukan dengan melihat hasil dari e (rata – rata pengujian) dan σ (standar deviasi

pengujian) yang didapatkan dari perhitungan nilai *E* dari pengujian yang dilakukan sebanyak *t* kali.

$$e = \frac{\sum_{j=1}^{t} E_j}{t} \quad ...(2.33)$$

Nilai *e* merupakan nilai rata – rata dari seluruh *error* kesalahan klasifikasi oleh metode yang digunakan.

$$V = \frac{\sum_{j=1}^{t} (E_j - e)^2}{t-1} \quad ...(2.34)$$

$$\sigma = \sqrt{V} \quad ...(2.35)$$

Standar deviasi merupakan perkiraan rentang jarak kesalahan klasifikasi dari dari rata - rata seluruh error dalam klasifikasi (*e*), rentang tersebut adalah +σ dan – σ dari *e*.

2.10.2. Confusion Matriks

Confusion matrix digunakan untuk mengevaluasi tingkat kebenaran dari pengujian penelitian yang telah dilakukan [13]. Penjelasan *confusion matrix* pada Tabel 2.2.

Tabel 2.2 *Confusion Matriks*

		Predicted Class	
		1	2
True Class	1	a	b
	2	c	d

Keterangan :

a : Jumlah dari hasil *stage* 1 yang terdeteksi benar sebagai *stage* 1

b : Jumlah dari hasil *stage* 1 yang terdeteksi salah sebagai *stage* 2

c : Jumlah dari hasil *stage* 2 yang terdeteksi salah sebagai *stage* 1

d : Jumlah dari hasil *stage* 2 yang terdeteksi benar sebagai *stage* 2

Beberapa perhitungan yang digunakan untuk mengukur keakuratan pada *confusion matrix* adalah sebagai berikut:

1. Keakuratan, proporsi untuk menghitung jumlah yang benar

$$Ac = \frac{a+d}{a+b+c+d} \quad ...(2.36)$$

2. *True Positif,* untuk menghitung kasus postifif yang diidentifikasi benar.

$$TP = \frac{d}{c+d} \quad ...(2.37)$$

3. *False Positif,* untuk menghitung kasus negative yang diidentifikasi salah sebagai positif.

$$FP = \frac{b}{a+b} \quad ...(2.38)$$

4. *True Negatif,* untuk menghitung kasus postitif yang salah diidentifikasi salah sebagai negatif.

$$TN = \frac{a}{a+b}. \quad ..(2.39)$$

5. *Flase Negative,* untuk menghitung kasus positif yang salah diidentifikasi salah sebagai negatif.

$$FP = \frac{c}{c+d} ...(2.40)$$

2.11. Perangkat Lunak Pendukung

Pembangunan aplikasi simulator perbandingan LVQ dan ANFIS ini dibutuhkan perangkat lunak pendukung diantaranya adalah sebagai berikut :

2.11.1. Microsoft Visual Studio

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

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam native code (dalam bentuk bahasa mesin yang berjalan di atas Windows)

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

Visual Studio kini telah menginjak versi Visual Studio 9.0.21022.08, atau dikenal dengan sebutan Microsoft Visual Studio 2008 yang diluncurkan pada 19 November 2007, yang ditujukan untuk platform Microsoft .NET Framework 3.5. Versi sebelumnya, Visual Studio 2005 ditujukan untuk platform .NET Framework 2.0 dan 3.0. Visual Studio 2003 ditujukan untuk .NET Framework 1.1, dan Visual Studio 2002 ditujukan untuk .NET Framework 1.0. Versi-versi tersebut di atas kini dikenal dengan sebutan Visual Studio .NET, karena memang membutuhkan Microsoft .NET Framework. Sementara itu, sebelum muncul Visual Studio .NET, terdapat Microsoft Visual Studio 6.0 (VS1998) [14].

2.11.2. XAMPP

Xampp adalah perangkat lunak bebas yang mendukung banyak sistem operasi. Xampp merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri (localhost) yang terdiri atas program Apache HTTP Server, MySQL *database* dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama Xampp merupakan singkatan dari X (empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU General Public License dan bebas. Xampp merupakan web server yang mudah digunakan yang dapat melayani tampilan halaman web yang dinamis [15].

Bagian penting XAMPP yang biasa digunakan pada umumnya adalah :

1. XAMPP *Control Panel Aplication* berfungsi mengelola layanan (*service*) XAMPP. Seperti mengaktifkan layanan (*start*) dan menghentikan (*stop*) layanan.
2. *htdoc* adalah folder tempat meletakkan berkas-berkas yang akan dijalankan. Di Windows, folder ini berada di C:/xampp
3. *phpMyAdmin* merupakan bagian untuk mengelola *database*.

2.11.3. MYSQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris : *database management system*) atau DBMS yang *multi-thread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License(GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan di sponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius [16].