BAB 2
TINJAUAN PUSTAKA

2.1 Kecerdasan Buatan

Kecerdasan buatan (*Artificial intelligence*) adalah salah satu cabang ilmu pengetahuan yang berhubungan dengan pemanfaatan mesin untuk memecahkan persoalan yang rumit dengan cara yang lebih manusiawi. Hal ini biasanya dilakukan dengan mengikuti / mencontoh karakteristik dan analogi berpikir dari kecerdasan manusia, dan menerapkannya sebagai algoritma yang dikenal oleh komputer.

Semakin pesatnya perkembangan teknologi menyebabkan adanya perkembangan dan perluasan lingkup yang membutuhkan kehadiran kecerdasan buatan. Karakteristik cerdas sudah mulai dibutuhkan di berbagai disiplin ilmu dan teknologi. Kecerdasan Buatan tidak hanya merambah di berbagai disiplin ilmu yang lain. Irisan antara psikologi dan kecerdasan buatan melahirkan sebuah area yang dikenal dengan nama *cognition* & *psycolinguistics*. Irisan antara teknik elektro dengan kecerdasan buatan melahirkan berbagai ilmu seperti pengolahan citra, teori kendali, pengenalan pola dan robotika.

Kecerdasan buatan digunakan untuk menganalisis pemandangan dalam citra dengan perhitungan simbol-simbol yang mewakili isi pemandangan tersebut setelah citra diolah untuk memperoleh ciri khas. Kecerdasan buatan bisa dilihat sebagai tiga kesatuan yang terpadu yaitu persepsi, pengertian dan aksi. Persepsi menerjemahkan sinyal dari dunia nyata dalam citra menjadi simbol-simbol yang lebih sederhana, pengertian memanipulasi simbol-simbol tadi untuk memudahkan penggalian suatu informasi tertentu, dan aksi menerjemahkan simbol-simbol yang telah dimanipulasi menjadi sinyal lain yang dapat merupakan hasil akhir atau hasil antara sesuai dengan keperluan[4].

2.2 Pola

Pola adalah suatu entitas yang terdefinisi dan dapat diidentifikasi serta diberi nama melalui ciri-cirinya (*feature*). Ciri-ciri tersebut digunakan untuk membedakan suatu pola dengan pola yang lainnya. Ciri yang baik adalah ciri yang memiliki daya pembeda yang tinggi, sehingga pengelompokan pola berdasarkan ciri yang dimiliki dapat dilakukan dengan keakuratan yang tinggi. Pola adalah komposit/gabungan dari ciri yang merupakan sifat dari sebuah objek.

Beberapa contoh pola :

1. Huruf, memiliki ciri-ciri seperti tinggi, tebal, titik sudut, dan lengkungan garis.
2. Suara, memiliki ciri-ciri seperti amplitudo, frekuensi, nada, dan intonasi.
3. Tanda tangan, memiliki ciri-ciri seperti panjang, kerumitan, dan tekanan.
4. Sidik jari, memiliki ciri-ciri seperti lengkungan, dan jumlah garis.

Ciri pada suatu pola diperoleh dari hasil pengukuran terhadap objek uji. Khusus pada pola yang terdapat di dalam citra, ciri-ciri yang dapat diperoleh berasal dari informasi.

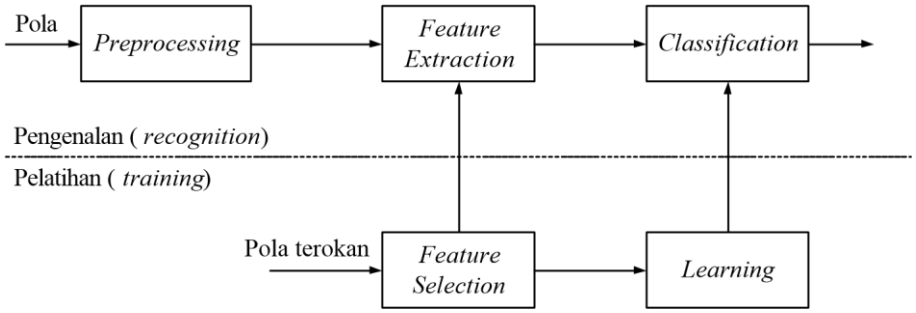
1. Spasial: intensitas *pixel*, *histogram*.
2. Tepi: arah, kekuatan.
3. Kontur: garis, elips, lingkaran.
4. Wilayah/bentuk: keliling, luas, pusat massa.
5. Hasil transformasi *Fourier*: frekuensi.

2.2.1 Pengenalan Pola

Pengenalan Pola adalah mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (dalam hal ini komputer). Pengenalan pola bertujuan menentukan kelompok atau kategori pola berdasarkan ciri-ciri yang dimiliki oleh pola tersebut. Dengan kata lain, pengenalan pola membedakan suatu objek dengan objek lain. Terdapat dua pendekatan yang dilakukan dalam pengenalan pola: pendekatan secara statistik dan pendekatan secara sintaktik atau struktural.

2.2.1.1 Pengenalan Pola secara Statistik

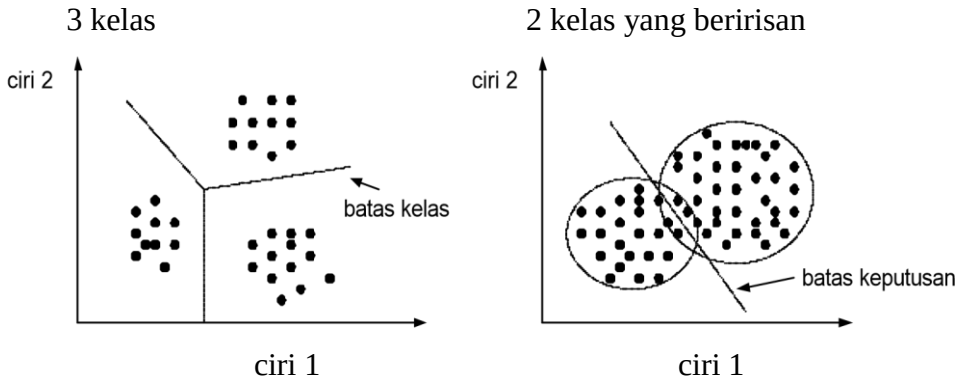
Pendekatan ini menggunakan teori-teori ilmu peluang dan statistik. Ciri-ciri yang dimiliki oleh suatu pola ditentukan distribusi statistiknya. Pola yang berbeda memiliki distribusi yang berbeda pula. Dengan menggunakan teori keputusan di dalam statistik, kita menggunakan distribusi ciri untuk mengklasifikasikan pola. Sistem pengenalan pola dengan pendekatan statistik ditunjukan oleh *Diagram* pada Gambar 2.1:



Gambar 2.1 Sistem pengenalan pola dengan pendekatan statistic

Ada dua fase dalam sistem pengenalan pola: (i) fase pelatihan dan (ii) fase pengenalan. Pada fase pelatihan, beberapa contoh citra dipelajari untuk menentukan ciri yang akan digunakan dalam proses pengenalan serta prosedur klasifikasinya. Pada fase pengenalan, citra diambil cirinya kemudian ditentukan kelas kelompoknya.

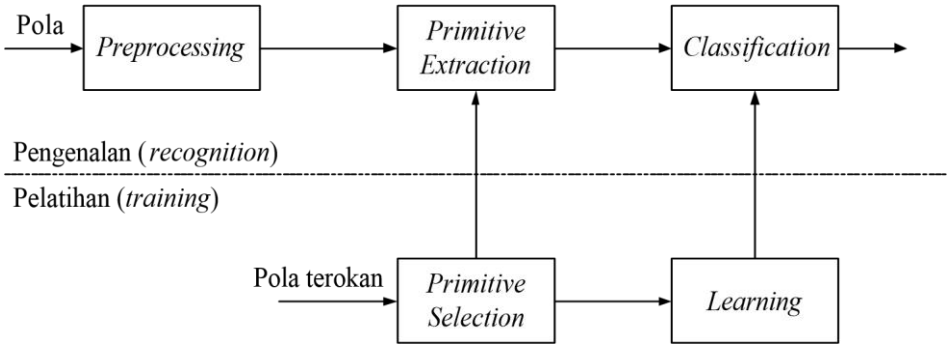
Kumpulan ciri dari suatu pola dinyatakan sebagai vektor ciri dalam ruang bahumatra (multi dimensi). Jadi, setiap pola dinyatakan sebagai sebuah titik dalam ruang bahumatra. Ruang bahumatra dibagi menjadi sejumlah uparuang (sub-ruang). Tiap uparuang dibentuk berdasarkan pola-pola yang sudah dikenali kategori dan ciri-cirinya (melalui fase pelatihan). Lihat Gambar 2.2:



Gambar 2.2 Contoh pembagian kelas pola

2.2.1.2 Pengenalan Pola secara Sintaktik

Pendekatan ini menggunakan teori bahasa formal. Ciri-ciri yang terdapat pada suatu pola ditentukan primitif dan hubungan struktural antara primitif kemudian menyusun tata bahasanya. Dari aturan produksi pada tata bahasa tersebut kita dapat menentukan kelompok pola. Gambar (2.3) memperlihatkan sistem pengenalan pola dengan pendekatan sintaktik. Pengenalan pola secara sintaktik lebih dekat ke strategi pengenalan pola yang dilakukan manusia, namun secara praktek penerapannya relatif sulit dibandingkan pengenalan pola secara statistic.



Gambar 2.3 Sistem pengenalan pola dengan pendekatan sintaktik

Pendekatan yang digunakan dalam membentuk tata bahasa untuk mengenali pola adalah mengikuti kontur (tepi batas) objek dengan sejumlah segmen garis terhubung satu sama lain, lalu mengkodekan setiap garis tersebut (misalnya dengan kode rantai). Setiap segmen garis merepresentasikan primitif pembentuk objek.

2.3 Pengolahan Citra

Pengolahan citra (*image processing*) adalah pemrosesan citra, yang bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (dalam hal ini komputer). Teknik – teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas lebih baik dari pada citra masukan [5]. Umumnya, operasi – operasi pada pengolahan citra diterapkan pada citra bila :

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra.
2. Elemen di dalam citra perlu dikelompokkan, dicocokkan, atau diukur.
3. Sebagian citra perlu digabung dengan bagian citra yang lain.

2.3.1 Operasi Pengolahan Citra

Operasi – operasi yang dilakukan di dalam pengolahan citra banyak ragamnya. Namun, secara umum operasi pengolahan citra dapat diklasifikasikan dalam beberapa jenis sebagai berikut :

1. Perbaikan kualitas citra (*image enhancement*)

   Jenis operasi ini bertujuan memperbaiki kualitas citra dengan cara memanipulasi parameter – parameter citra. Dengan operasi ini, ciri – ciri khusus yang terdapat di dalam citra lebih ditonjolkan.

   Contoh – contoh operasi perbaikan citra :

   a. Perbaikan kontras gelap/terang
   b. Perbaikan tepian objek (*edge enhancement*)

c. Penajaman (*sharpening*)
d. Pemberian warna semu (*pseudocoloring*)
e. Penapisan derau (*noise filtering*)

2. Pemugaran citra (*image restoration*)

Operasi ini bertujuan menghilangkan / meminimumkan cacat pada citra. Tujuan pemugaran citra hampir sama dengan operasi perbaikan citra. Bedanya, pada pemugaran citra penyebab degradasi gambar diketahui.

a. Penghilangan kesamaran (*deblurring*)
b. Penghilangan derau (*noise*)

3. Pemampatan citra (*image compression*)

Jenis operasi ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam pemampatan adalah citra yang telah dimampatkan harus tetap mempunyai kualitas gambar yang bagus.

4. Segmentasi citra (*image segmentation*)

Jenis operasi ini bertujuan untuk memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi ini berkaitan erat dengan pengenalan pola.

5. Pengorakan citra (*image analysis*)

Jenis operasi ini bertujuan menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Teknik pengorakan citra mengekstraksi ciri-ciri tertentu yang membantu dalam identifikasi objek. Proses segmentasi kadangkala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya.

Contoh-contoh operasi pengorakan citra:

a. Pendeteksian tepi objek (*edge detection*)

b. Ekstraksi batas (*boundary*)
c. Representasi daerah (*region*)

6. Rekonstruksi citra (*image reconstruction*)

   Jenis operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekonstruksi citra banyak digunakan dalam bidang medis. Misalnya beberapa foto *rontgen* dengan sinar *X* digunakan untuk membentuk ulang gambar organ tubuh.

2.3.2 Jenis – Jenis Citra

2.3.2.1 Citra Biner

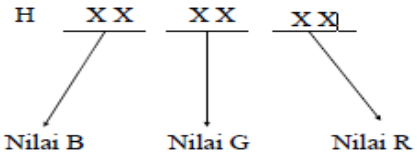
Citra *biner* adalah citra *Digital* yang hanya memiliki dua kemungkinan nilai piksel yaitu hitam dan putih. Citra *biner* disebut juga citra B & W (*Black and White*) atau citra monokrom. Hanya dibutuhkan 1 bit untuk mewakili nilai setiap piksel dari citra *biner*[6].



Gambar 2.4 Contoh Citra Biner

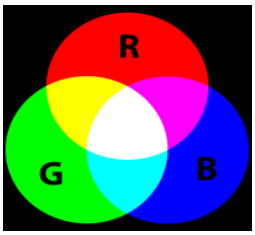
2.3.2.2 Citra Warna RGB (*Red, Green, Blue*)

Dalam pengolahan citra, warna dipresentasikan dalam *range* nilai H000000 sampai 0x00ffffff. Warna hitam adalah 0x00000000 dan warna putih adalah Hffffff. Definisi nilai warna di atas sebagai berikut. variabel H menyatakan angka dibelakangnya adalah *hexadecimal* [5].



Gambar 2.5 Definisi nilai warna pada citra

Dengan demikian *range* warna yang digunakan adalah (2 8) (2 8) (2 8) = 2 24 (atau yang dikenal dengan istilah *True Colour* pada *Windows*). Nilai warna yang digunakan di atas merupakan gabungan warna cahaya merah, hijau dan biru seperti yang terlihat pada Gambar 2.6:



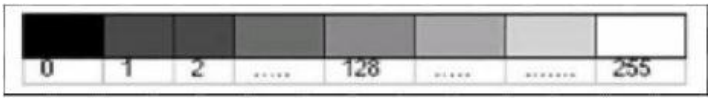
Gambar 2.6 komposisi warna RGB

Dengan demikian, untuk menentukan suatu warna yang bukan warna dasar digunakan gabungan skala kecerahan dari setiap warnanya

2.3.2.3 Citra *Grayscale*

Grayscale adalah warna-warna piksel yang berada dalam rentang gradasi warna hitam dan putih. Intensitas *Grayscale* biasanya disimpan sebagai data citra 8 bit atau 256 intensitas warna *gray* dari nilai 0 (hitam) dan 255 (putih). Citra *Grayscale* merupakan citra *Digital* yang hanya memiliki satu nilai kanal pada setiap pikselnya, dengan kata lain nilai bagian *RED* = *GREEN* = *BLUE*. Nilai tersebut digunakan untuk menunjukan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan keabuan disini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih [7]. Model warna ini

menyimpan informasi warna sebesar 8 bit. Jadi warna yang bisa ditampilkan pada mode warna *Greyscale* berjumlah sampai 256. Skala perhitungan numeriknya bisa dengan pendekatan nilai *brightness* yang berskala 0 sampai 255 seperti pada Gambar 2.7:



Gambar 2.7 Komposisi Warna *Grayscale*

Untuk mengkonversikan citra yang memiliki warna *RGB* ke derajat keabuan bisa menggunakan rumus 2.1 :

$$X = R + G + B \,/\, 3 \qquad [2.1]$$

Dimana :

X = Citra *Greyscale*

R = *Red*

G = *Green*

B = *Blue*

R = nilai *Red* * 15 / 255

G = nilai *Green* * 15 / 255

B = nilai *Blue* * 15 / 255

Contoh gambar dari Hasil dari proses *grayscaling* ditunjukkan pada Gambar 2.16 dibawah ini:

(a) (b)



Gambar 2.8 Citra *RGB* (a) dan citra *Grayscale (b)*

2.3.3 Teknik Pengambilan Citra *Digital*

Ada beberapa teknik pengambilan citra *Digital* yang bisa dilakukan, antara lain dengan menggunakan kamera *Digital* atau menggunakan *scanner*, atau *barcode scanner*. Hasil dari citra dengan menggunakan teknik kamera atau *scanner* berupa citra *raster* atau citra dengan model *matrix*. Teknik pengambilan citra selain membutuhkan peralatan *input*, membutuhkan juga suatu *card* yang disebut *frame grabber* yang berupa rangkaian untuk mengolah citra secara *hardware*.

Selain itu pengambilan citra juga bisa dilakukan dengan menggunakan teknik grafika komputer, yaitu dengan membentuk citra komputer sesuai dengan teknik pengambilan citra. Hasil dari citra dengan menggunakan teknik grafika komputer ini bisa berupa citra *raster* atau bisa berupa *vector* [8].

2.3.4 Preprocessing

Preprocessing adalah proses pengolahan data asli sebelum data tersebut diolah dengan jaringan syaraf tiruan. Tahap ini merupakan tahap awal penerjemahan. Dalam *preprocessing*, dilakukan penghilangan *noise* agar pola-pola yang tidak dibutuhkan dihilangkan.

Tujuan *preprocessing*, diantaranya*:*

1. Menghilangkan *noise*
2. Memperjelas *features* (fitur) data
3. Memperkecil / memperbesar ukuran data
4. Mengkonversi data asli agar diperoleh data yg sesuai kebutuhan

2.4 Tekstur

Tekstur secara umum mengacu pada repetisi elemen-elemen tekstur dasar yang sering disebut primitif atau *texel* (*texture element*). Untuk membentuk suatu tekstur setidaknya ada dua persyaratan yang harus dipenuhi antara lain :

1. Terdiri dari satu atau lebih piksel yang membentuk pola-pola primitif (bagian-bagian terkecil). Bentuk-bentuk pola primitif ini dapat berupa titik, garis lurus,

garis lengkung, luasan dan lain-lain yang merupakan elemen dasar dari sebuah bentuk.

2. Munculnya pola-pola primitif yang berulang-ulang dengan interval jarak dan arah tertentu sehingga dapat diprediksi atau ditemukan karakteristik perulangannya.

   Suatu citra memberikan interpretasi tekstur yang berbeda apabila dilihat dengan jarak dan sudut yang berbeda, manusia memandang tekstur berdasarkan deskripsi yang bersifat acak, seperti halus, kasar, teratur, tidak teratur, dan lain sebagainya. Hal ini merupakan deskripsi yang tidak tepat dan non-kuantitatif, sehingga diperlukan adanya suatu deskripsi yang kuantitatif (matematis) untuk memudahkan analisis.

2.4.1 Analisis Tekstur

Analisis tekstur merupakan dasar dari berbagai macam aplikasi, aplikasi dari analisis tekstur antara lain adalah penginderaan jarak jauh, pencitraan medis, identifikasi kualitas suatu bahan (kayu, kulit, tekstil dan lain-lain). Pada analisis citra, dikategorikan menjadi lima kategori utama yaitu statistis, struktural, geometri, model dasar dan pengolahan sinyal. Pendekatan statistis mempertimbangakan bahwa internsitas dibangkitkan oleh medan acak dua dimensi, metode ini berdasar pada frekuensi-frekuensi ruang. Contoh metode statistis adalah fungsi *autokorelasi*, *run-length*, *matriks kookurensi*, tranformasi *Fourier*, frekuensi tepi. Teknik struktural berkaitan dengan penyusupan bagian-bagian terkecil suatu citra. contoh metode struktural adalah model *fraktal*. Metode geometri berdasar atas perangkat geometri yang ada pada elemen tekstur. Contoh model dasar adalah medan acak. Sedangkan metode pengolahan sinyal adalah metode yang berdasarkan analisis frekuensi seperti transformasi *gabor* dan transformasi *wavelet*.

2.4.2 Analisis Tekstur Dengan Pendekatan Statistik

Analisis tekstur Metode statistik menggunakan perhitungan statistik distribusi derajat keabuan (*histogram*) dengan mengukur tingkat kekontrasan, granularitas, dan

kekasaran suatu daerah dari hubungan ketetanggaan antar piksel di dalam citra. Metode Statistik ini penggunaannya tidak terbatas untuk tekstur – tekstur alami yang tidak terstruktur dari sub pola dan himpunan aturan (mikrostruktur). Berdasarkan orde statistiknya, analisis tekstur dapat dikategorikan menjadi 3, yaitu analisis tekstur orde satu, orde dua, dan orde tiga.

2.5 *Principal Component Analysis (PCA)*

PCA merupakan metode ekstrasi fitur untuk pengenalan pola. *PCA* melakukan kompresi berdasarkan informasi orde kedua dari *Mean Square Error* (MSE) [11].

Secara matematis *PCA* didefenisikan sebagai berikut [11]. Misalkan terdapat vektor kolom citra Xi = [X1, X2….XM] dimana M merupakan jumlah citra dan merupakan jumlah kolom dalam matrik Xi. Maka μ, yaitu *mean* dari matrik Xi dapat dihitung sebagai berikut:

$$\mu = \frac{1}{M}\sum_{i}^{M} Xi \qquad [2.2]$$

Pengurangan data *matriks* Xi dengan rata – rata citra μ akan menghasilkan data dengan rata-rata nilai nol. Proses ini disebut juga dengan *normalisasi* data.

$$\varphi = Xi - \mu \; ; i = 1,2 \ldots \ldots M \qquad [2.3]$$

Hasil dari matrik ini kemudian digunakan untuk membangun matrik kovarian sebagai berikut.

$$\text{Cov} = (1/n)*[\text{Transpose}(X-mx)*(X-mx)] \qquad [2.4]$$

Setelah matrik kovarian didapat, dilanjutkan mencari *eigenvalue* dari matrik kovarian tersebut dengan persamaan sebagai berikut :

$$\textit{Determinant}\ (\text{Cov} - \lambda\, I) = 0 \qquad [2.5]$$

Setelah didapatkan *eigenvalue*, dipilihlah *eigenvalue* terbesar untuk mencari nilai *eigenvector*. Dengan rumus sebagai berikut :

$$(\lambda I - \text{Cov}) = 0 \qquad [2.6]$$

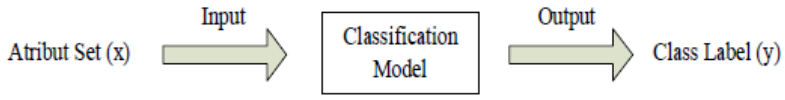
2.6 *Two Dimensional Principal Component Analysis (2DPCA)*

Metode *2DPCA* merupakan pengembangan dari metode *PCA* yang berfungsi sebagai ekstraksi fitur untuk kompresi data. Metode *2DPCA* memiliki kelebihan dari metode *PCA* dari segi akurasi data dan kompleksitas waktu namun memiliki kekurangan yaitu membutuhkan banyak koefisien dalam menyimpan data[12]. Perbedaan dari *2DPCA* dan *PCA* adalah pada langkah awal pengambilan matrik suatu citra. Jika pada metode Jika pada metode *PCA* suatu matrik citra dua dimensi (m x n) maka matrik tersebut ditransformasi menjadi matrik baris atau matrik satu dimensi. Metode *2DPCA* jika suatu matrik citra (m x n) maka matrik citra tersebut tidak perlu ditranformasi ke matrik satu dimensi. Suatu citra A dengan matrik berukuran (m x n) dan X menyatakan vektor kolom kesatuan-n dimensi. Untuk memproyeksikan gambar A, (m x n) *matriks* ke X dengan transformasi *linear* persamaan berikut.

$$Y = AX \qquad [2.7]$$

2.7 Metode Klasifikasi

Klasifikasi adalah proses untuk menemukan model atau fungsi yang menjelasakan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu objek yang labelnya tidak diketahui. Dalam mencapai tujuan tersebut, proses klasifikasi membentuk satu model yang mampu membedakan data kedalam kelas – kelas yang berbeda berdasarkan aturan atau fungsi tertentu. Model itu tersendiri biasa berupa aturan "jika-maka", berupa pohon keputusan, atau *formula* matematis.

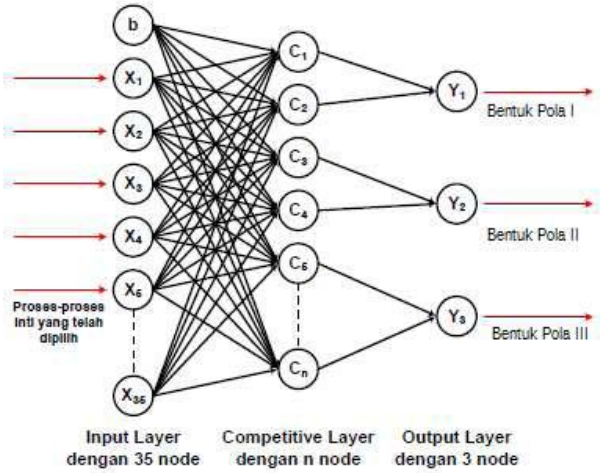


Gambar 2.9 *Blok Diagram* Model Klasifikasi

2.7.1 Metode Learning Vektor Quantization (*LVQ*)

Metode *Learning Vektor Quantization* (*LVQ*) yaitu merupakan metode untuk melakukan pelatihan terhadap lapisan – lapisan kompetitif *supervised*. Lapisan kompetitif akan belajar secara otomatis untuk melakukan klasifikasi terhadap vektor *input* yang diberikan. Apabila beberapa vektor *input* memiliki jarak yang sangat berdekatan, maka *vector – vector input* tersebut akan dikelompokkan dalam kelas yang sama.

Dari hal ini *LVQ* merupakan metode klasifikasi data adaptif berdasarkan pada data pelatihan dengan informasi kelas yang diinginkan yang bisa pula di ambil dari sebuah JST dari *levenberg Marquardt* yang di ambil yaitu nilai jumlah pelatihan *epoh*-nya dan nilai *error (MSE)*. Walaupun merupakan suatu metoda pelatihan *supervised* tetapi *LVQ* menggunakan teknik data *clustering unsupervised* untuk pra proses set data dan penentuan *cluster center*-nya. Arsitektur jaringan *LVQ* hampir menyerupai suatu jaringan pelatihan kompetitif kecuali pada masing-masing unit *output*-nya yang dihubungkan dengan suatu kelas tertentu[14]. Proses pelatihan pada jaringan dilakukan untuk mengeset nilai bobot jaringan (wi) yang akan mengalami perubahan pada setiap pelatihan yang dilakukan.



Gambar 2.10 Arsitektur *Learning Vector Quantization (LVQ)*

2.7.2 Cara Kerja *Learning Vector Quantization (LVQ)*

Pada Metoda pembelajaran yang dipilih pada penelitian ini adalah *Learning Vector Quantization (LVQ)*. Algoritma pembelajaran *LVQ* terdiri dari dua tahap, yaitu:

1. Metoda pelajaran data *unsupervised* digunakan untuk menempatkan beberapa *cluster center* tanpa menggunakan informasi kelas.
2. Informasi kelas digunakan untuk *fine-tune cluster center* untuk memperkecil banyaknya kasus kesalahan klasifikasi.

2.7.3 Proses Pelatihan *Learning Vector Quantization (LVQ)*

Dari tujuan diadakannya proses pelatihan pada JST adalah agar JST dapat diandalkan dalam mengenali pola yang di*input*kan sehingga dapat diperoleh hasil yang diinginkan. Karena itu sangatlah penting untuk melatih JST dengan *training set* yang sesuai dengan tugas JST itu sendiri dalam mengenali suatu pola bentuk pada permukaan bahan kulit[15].

Kinerja yang diukur pada proses pelatihan jaringan syaraf tiruan pada metode *Learning Vector Quantization* adalah :

1. Lamanya proses pelatihan jumlah *(epoch)* Semakin cepat proses pelatihan maka semakin baik kinerja proses pelatihan tersebut. Hal ini berarti jumlah *epoch* untuk mencapai nilai *error* yang diinginkan adalah semakin mendekati nol.
2. Perhitungan *error* (MSE).

   Perhitungan *error* merupakan pengukuran bagaimana jaringan dapat belajar dengan baik sehingga jika dibandingkan dengan pola yang baru akan dengan mudah dikenali. Apabila nilai *error* ini masih cukup besar, hal ini mengidikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi hingga nilai *error* mendekati nol. Apabila *output* jaringan tepat sama dengan target akan bernilai nol.

Koefisien *regresi* (R) terhadap respon *output* jaringan dan target yang diinginkan. Apabila *output* jaringan tepat sama dengan target maka koefisien *regresi* akan bernilai satu.

2.8 Bahan Kulit

Bahan Kulit adalah bahan yang fleksibel dan tahan lama yang dibuat dengan proses penyamakan kulit hewan, umumnya kulit sapi, domba atau kambing. Bahan kulit banyak digunakan sebagai bahan baku pembuatan pakaian, *interior* kendaraan, *furniture*, sepatu, *accecoris* dan yang lainnya. Kulit menjadi pilihan yang tepat karena menawarkan kenyamanan dan ketahanan. Bahan kulit asli memiliki poros yang tinggi sehingga tepat digunakan dimusim hujan maupun musim panas dan bahkan musim salju. Bahan jenis kulit yang sudah disiap pakai untuk proses pembuatan barang mempunyai jenis – jenis. Jenis itu memiliki tekstur yang berbeda beda sesuai jenis dan kebutuhannya. Sehingga konsumen atau pemakai bahan kulit sulit membedakan.

Jenisnya membedakan penggunaan bahan jenisnya sesuai kebutuhan yang diperlukan. Setiap jenis bahan kulit yang akan digunakan untuk membuat suatu barang menggunakan bahan jenis kulit yang berbeda beda sesuai kebutuhan barang tersebut. seperti contohnya apabila ingin membuat sepatu formal atau *pantofel*, tetapi karena tidak tahu bahan jenis kulitnya dan membeli bahan jenis kulit *suede* atau *milling*. Tentu salah dan merugikan, karena tidak sesuai kebutuhan sebab sepatu formal biasa memakai bahan jenis *brush-off*. Maka dari itu perlu pengetahuan mengenai jenis bahan kulit supaya tidak salah guna atau tidak terjadi yang bukan pada penggunaannya karena masih ada para penjual yang tidak mengerti. Jadi membuat konsumen bahan kulit atau pemakai cukup sulit dalam memilih dan membeli bahan kulit.

2.8.1 Jenis – jenis bahan kulit

Berbagai macam kulit hewan baik sapi, kerbau, kambing dan domba pada dasarnya dapat dibuat menjadi kulit-kulit di bawah ini.

1. *Full grain/full top grain leather*

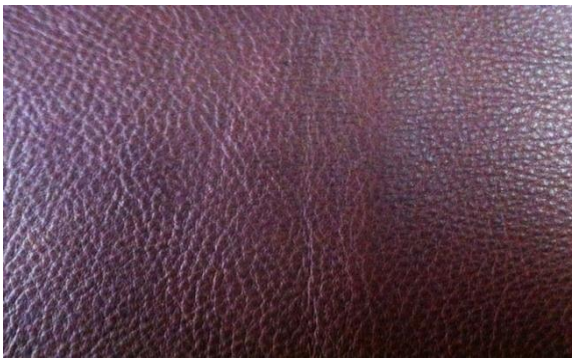
   Jenis bahan kulit yang di proses secara utuh tanpa mengubah atau memodifikasi struktur permukaan kulit bawaanya. Sehingga motif permukaan pori-pori kulit (*grain*) masih terlihat alami atau natural, sehingga tekstur kulit binatang masih terlihat cukup jelas. Bahan kulit ini sering di buat untuk mabel, tas, jaket dan juga sepatu. Jenis kulit ini merupakan yang umum beredar di pasaran, dikarenakan kualitasnya yang baik. Keunggulan jenis ini adalah sulit untuk bertransformasi alias awet. Bagian luar dari kulit ini secara utuh masih alami sehingga *texture* dari kulit binatang terlihat cukup jelas.



Gambar 2.11 *Full grain*

2. *Milling/Corrected grain leather*

   Kulit yang memiliki permukaan tambahan/buatan yang diemboss ke dalamnya setelah dihaluskan lebih bagian luar kulit yang kurang bagus. Bahan kulit yang di proses melalui tehknik tertentu dan menghasilkan corak menyerupai kulit jeruk, terlihat kasar tetapi jenis kulit ini mempunyai daya tahan yang baik, lembut dan elastis. *Milled Leather* diambil dari bagian atas kulit sapi muda ataupun dewasa.



Gambar 2.12 *Milling*

3. Nappa leather

Mulanya hanya kulit domba yang dinamakan *Nappa*. Tetapi belakangan ini kata '*Nappa*' menjadi istilah kulit lain yang berarti 'lembut' seperti kulit sapi *Nappa*. Jenis kulit yang diproses khusus dengan permukaan halus yang tertutup cat khusus kulit baik yang *Nappa* doff ataupun yang mengkiilap. Umumnya sepatu model Dress Shoes (sepatu kantor), tas, dompet, sabuk, juga jaket yang berasal dari kulit sapi maupun kulit domba terbuat dari kulit jenis *Nappa*.



Gambar 2.13 *Nappa leather*

4. Patent leather (Brush-off)

Ketika kulit sapi dikerjakan dengan bahan akhir yang protektif seperti cat acrylic atau bahan tahan air untuk memproduksi hasil akhir yang sangat mengkilap. Bahan jenis ini sering dipakai untuk sepatu. sepatu kulit ini memilki kualitas permukaan halus dan mengkilat. Kelebihan lain dari bahan sepatu ini

mudah dibersihkan. Jenis kulit Patent *Leather* adalah jenis kulit yang diproses dengan dilapisi bahan protektif seperti *acrylic*. Sehingga menghasilkan jenis kulit yang sangat mengkilap, berwarnana terang, dan licin. Sepatu yang biasa menggunakan kulit jenis ini adalah sepatu formal.



Gambar 2.14 *Patent/Brush-off*

5. *Buk / Nubuck leather*

Jenis bahan kulit *Buk / Nubuck* terlihat sama dengan jenis kulit *Suede*, tetapi tak bisa dikategorikan sebagai *suede*. Perbedaanya adalah pada proses pembuatanya. Tidak seperti kulit *Suede,* jenis kulit Nubuck tidak dibuat dari bagian dalam kulit binatang. Melainkan dibuat dari bagian luar kulit binatang. Namun permukaannya dihaluskan menggunakan permukaan yang kasar, sehingga menghasilkan permukaan kulit yang halus seperti pada jenis kulit *Suede*. Permukaan kulit *Nubuck* disikat untuk menciptakan tekstur seperti beludru, sehingga seringkali dikira *suede*.



Gambar 2.15 *Buk / Nubuck leather*

6. *Suede leather*

Jenis kulit *Suede* adalah jenis bahan kulit yang diproses secara terbalik, yaitu disamak dari bagian dalam kulit. Kulit *Suede* memiliki tekstur yang lebih lembut. Sepatu kulit yang menggunakan bahan kulit *suede* terkesan lebih santai dan tidak formal / casual.



Gambar 2.16 *suede leather*

7. *Pull-up leather*

Kulit yang memperlihatkan efek warna meretak bila kulit ditarik ketat. Kulit ini menggunakan bahan celup *full aniline*, dan sebagai tambahan memiliki sejenis minyak dan/atau *wax* aplikasi, yang menyebabkan warna menjadi terlihat lebih muda ketika kulit ditarik.



Gambar 2.17 *Pull-up leather*

2.9 OOP (Object Oriented Programing)

Pemrograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktifitas analisis beorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientais objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentranformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan saat ini sangat beragam (aplikasi bisnis, *real-time, utility* dan sebagainya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut :

a. Meningkatkan Produktivitas

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

b. Kecepatan Pengembangan

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

c. Kemudahan Pemeliharaan

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

d. Adanya Konsistensi

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

e. Meningkatkan Kualitas Perangkat Lunak

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

Berikut salah satu contoh bahasa pemograman yang mendukung pemrograman berorinentasi objek adalah *C#*.

C# adalah bahasa pemrograman yang diciptakan oleh *Microsoft* (dikembangkan dibawah kepemimpinan *Anders Hejlsberg* yang notabene juga telah menciptakan berbagai macam bahasa pemrograman termasuk *Borland Turbo C++* dan *Borland Delphi*). Bahasa *C#* juga telah di standarisasi secara internasional oleh *ECMA*.

Seperti halnya bahasa pemrograman yang lain, *C#* bisa digunakan untuk membangun berbagai macam jenis aplikasi, seperti aplikasi berbasis *windows* (*desktop*) dan aplikasi berbasis *web* serta aplikasi berbasis *web services*.

Kelebihan menggunakan *C#* diantaranya :

1. Sederhana (*simple*)

C# menghilangkan beberapa hal yang bersifat kompleks yang terdapat dalam beberapa macam bahasa pemrograman seperti *Java* dan *C++*, termasuk diantaranya mengilangkan macro, templates, multiple inheritance dan virtual base classes. Hal hal tersebut yang dapat menyebabkan kebingungan pada saat menggunakannya, dan juga berpotensial dapat menjadi masalah bagi para programmer *C++*. Jika anda pertama kali belajar bahasa *C#* sebagai bahasa pemrograman, maka hal-hal tersebut diatas tidak akan membuat waktu anda terbuang terlalu banyak untuk mempelajarinya. *C#* bersifat sederhana, karena bahasa ini didasarkan kepada bahasa C dan *C++*. Jika anda familiar dengan C dan *C++* atau bahkan *Java*, anda akan menemukan aspek-aspek yang begitu familiar, seperti *statements*, *expression*, *operators*, dan beberapa fungsi yang diadopsi

langsung dari *Java, C* dan *C++*, tetapi dengan berbagai perbaikan yang membuat bahasanya menjadi lebih sederhana.

2. *Modern*

   Beberapa fitur seperti *exception handling, garbage collection, extensible data types,* dan *code security* (keamanan kode/bahasa pemrograman). Dengan adanya fitur-fitur tersebut, menjadikan bahasa *C#* sebagai bahasa pemrograman yang *modern*.

3. *Object-Oriented Language*

   Kunci dari bahasa pemrograman yang bersifat *Object Oriented* adalah *encapsulation, inheritance,* dan *polymorphism*. Secara sederhana, istilah-istilah tersebut bisa didefinisikan sebagai berikut (definisi dan penjelasan lebih lanjut akan diuraikan pada bab - bab selanjutnya). *encapsulation*, dimana semua fungsi ditempatkan dalam satu paket (*single package*). *inheritance*, adalah suatu cara yang terstruktur dari suatu kode - kode pemrograman dan fungsi untuk menjadi sebuat program baru dan berbentuk suatu paket. *Polymorphism*, adalah kemampuan untuk mengadaptasi apa yang diperlukan untuk dikerjakan. Sifat-sifat tersebut di atas, telah di miliki oleh *C#* sehingga bahasa *C#* merupakan bahasa yang bersifat *Object Oriented*.

4. *Powerfull dan fleksibel*

   C# bisa digunakan untuk membuat berbagai macam aplikasi, seperti aplikasi pengolah kata, grafik, *spreadsheets,* atau bahkan membuat *kompiler* untuk sebuah bahasa permrograman.

5. Efisien

   C# adalah bahasa pemrograman yang menggunakan jumlah kata-kata yang tidak terlalu banyak. *C#* hanya berisi kata-kata yang biasa disebut dengan *keywords. Keywords* ini digunakan untuk menjelaskan berbagai macam informasi. Jika anda berpikiran bahwa bahasa pemrograman yang menggunakan sangat banyak kata-kata (*keywords*) akan lebih *powerfull*, maka jawabannya adalah

"pemikiran itu tidak selalu benar", karena hal itu justru bisa menambah kerumitan para *developer* pada saat membuat suatu aplikasi.

6. Modular

   Kode *C#* ditulis dengan pembagian masing *Class-Class (classes)* yang terdiri dari beberapa routines yang disebut sebagai member *methods. Class-Class* dan metode metode ini dapat digunakan kembali oleh program atau aplikasi lain. Hanya dengan memberikan informasi yang dibutuhkan oleh *Class* dan metode yang dimaksud, maka kita akan dapat membuat suata kode yang dapat digunakan oleh satu atau beberapa aplikasi dan program.

2.9.1 Konsep Dasar Berorientasi Objek

Berikut adalah konsep dasar pemrograman berorientasi objek :

a. Objek (*Object*)

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

b. Kelas (*Class*)

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

c. Pembungkusan (*Encapsulation*)

   Pembungkusan atribut data dan layanan yang mempunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

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

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

e. Metode

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

f. Polimorfisme

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

2.9.2 UML (Unified Modelling Language)

Unified Modelling Language (UML) adalah sekumpulan spesifikasi yang dikeluarkan oleh *OMG. UML* terbaru adalah *UML* 2.3 yang terdiri dari 4 macam spesifikasi, yaitu : *Diagram Interchange Spesification, UML Infrastrukture, UML Superstrukture, dan Object Constraint Language (OCL)*. Pada *UML* 2.3 terdiri 13 macam *diagram* yang dikelompokan pada 3 kategori, yaitu:

A. Struktur *Diagram,* yaitu kumpulan *Diagram* yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. Pada Struktur *Diagram* dibagi menjadi 6 bagian.

1. *Diagram* Kelas

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

2. *Diagram* Objek

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

3. *Diagram* Komponen

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

4. Composite Structure Diagram

   Composite Structure Diagram bru muali ada pada UML versi 2.0. *Diagram* ini dapat digunakan untuk menggambarkan struktur dari bagian-

bagian yang saling terhubung maupun mendeskripsikan struktur pada saat berjalan (*runtime*) .

5. *Package Diagram*

   Package Diagram menyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam *Diagram* UML. Hampir semua *Diagram* dalam UML dapat dikelompokan menggunakan *Package Diagram*.

6. *Deployment Diagram*

   Deployment Diagram menunjukan konfigurasi komponen dalam proses eksekusi aplikasi.

B. *Behavior Diagram*, yaitu kumpulan *Diagram* yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Pada *Behavior Diagram* dibagi menjadi 3 bagian :

1. *Use Case Diagram*

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

2. *Activity Diagram*

   Activity Diagram menggambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

3. *State Machine Diagram*

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

C. *Interactions Diagram*

Interactions Diagram yaitu kumpulan *Diagram* yang digunakan untuk menggambarkan interaksi antar subsistem pada suatu sistem. Pada *Interactions Diagram* dibagi menjadi 4 bagian :

1. *Sequence Diagram*

Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.

2. *Communication Diagram*

   Communication Diagram menggambarkan interaksi antar bojek/bagian dalam bentuk urutan pengiriman pesan. *Diagram* komunikasi merepresentasikan informasi yang diperoleh dari *Diagram* kelas, *Diagram sequence*, dan *Diagram use case* untuk mendeskripsikan gabungan antara struktur statis dan tingkah laku dinamis dari suatu sistem.

3. *Timing Diagram*

   Timing Diagram merupakan *Diagram* yang fokus pada penggambaran terkait batas waktu.

4. *Interaction Overview Diagram*

   Interaction overview Diagram mirip dengan *Diagram* aktivitas yang berfungsi untuk menggarbarkan sekumpulan urutan aktivitas, *Diagram* ini adalah bentuk aktivias *Diagram* yang setiap titik merepresentasikan *Diagram* interaksi.