BAB 2
LANDASAN TEORI

2.1 Bahasa Isyarat

Menurut Reynolds dan Mann bahasa isyarat adalah istilah umum yang mengacu pada setiap gestural/bahasa visual yang menggunakan bentuk dan gerakan jari-jari, tangan, dan lengan yang spesifik, serta gerakan mata, wajah, kepala, dan tubuh. Tidak ada system internasional yang dipahami semua orang tunarungu. Terdapat bahasa isyarat Inggris, bahasa isyarat Spanyol, dan mungkin bahasa isyarat di setiap negara di mana orang tunarungu telah diperlukan untuk berkomunikasi di antara mereka sendiri dengan cepat, efisien, dan secara visual tanpa menggunakan kertas dan pensil.

Senada dengan Reynold and Mann, A. Van Uden mengatakan bahasa isyarat adalah bahasa dengan menggunakan tangan, walaupun dalam kenyataan, ekspresi muka dan lengan juga digunakan untuk berperan. Terdapat bahasa isyarat Inggris, bahasa isyarat Spanyol, dan mungkin bahasa isyarat di setiap negara di mana orang tunarungu telah diperlukan untuk berkomunikasi di antara mereka sendiri dengan cepat, efisien, dan secara visual tanpa menggunakan kertas dan pensil.

Menurut Permanarian Somad dan Tati Herawai sistim isyarat ini terdapat dua jenis komponen. Yang berfungsi sebagai penentu atau pembeda makna, sedangkan yang lain sebagai penunjang. Demikian juga menurut L. Evans dan Lenneberg mengatakan bahwa kontak anak tunarungu melalui bahasa akan menjadi sangat miskin dibandingkan dengan anak dengar bila hanya pada baca ujaran. Dengan menggunakan bahasa isyarat selain membaca ujaran anak tunarungu juga dapat membaca isyarat yang diberikan kepadanya, dengan begitu ada pilihan bagi anak tunarungu untuk memahami lawan bicaranya[8].

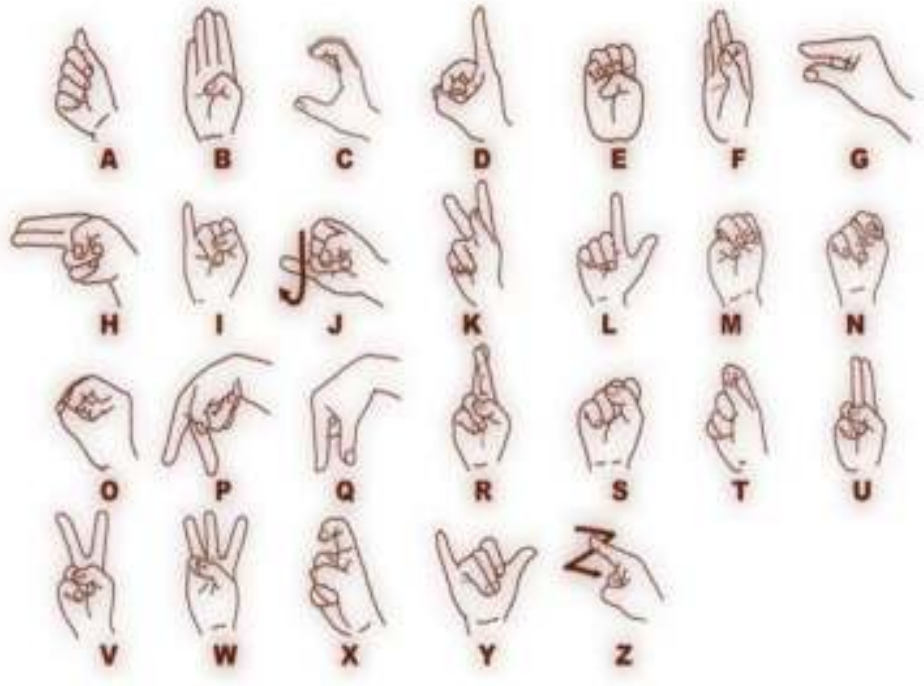
2.1.1 FingerSpelling

Fingerspelling adalah pengodean bahasa tertulis secara manual yang digunakan oleh penandatangan untuk mengisi celah leksikal dalam bahasa isyarat.

Fingerspelling didasarkan pada bentuk kata tertulis, yang tersedia untuk orang tuli melalui membaca. Sama seperti orang yang mendengar dapat mengeja nama yang tidak dikenal dengan mengucapkan setiap huruf, demikian juga seorang *fingerspeller* dapat menggunakan alfabet manual untuk mengucapkan kata-kata yang tidak dikenal. *Fingerspelling* mengkodekan sistem penulisan suatu bahasa (Kode Sekunder); oleh karena itu, ini dapat di anggap sebagai kode tersier[9].

2.1.2 Sistem Isyarat Bahasa Indonesia (SIBI)

Sistem Isyarat Bahasa Indonesia (SIBI) yang dibakukan itu merupakan salah satu media yang membantu komunikasi sesama kaum tunarungu di dalam masyarakat yang lebih luas. Wujudnya adalah tatanan yang sistematis tentang seperangkat isyarat jari, tangan, dan berbagai gerak yang melambangkan kosa kata bahasa Indonesia[10].



Gambar 2.1 Abjad Sistem Isyarat Bahasa Indonesia

2.2 Kecerdasan Buatan

Kecerdasan Buatan (*Artificial Intelligence*)merupakan satu bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat

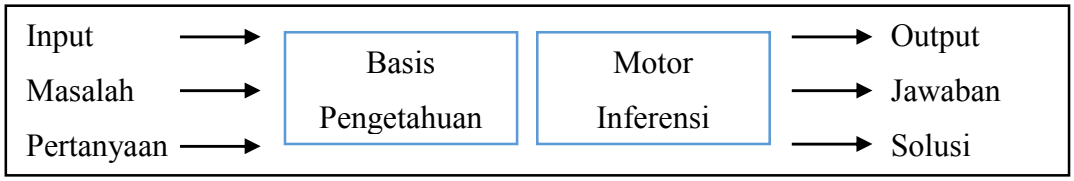
melakukan pekerjaan seperti sebaik yang dilakukan oleh manusia. Untuk mengetahui dan memodelkan proses-proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia. Cerdas, berarti memiliki pengetahuan ditambah pengalaman, penalaran (bagaimana membuat keputusan dan mengambil tindakan), moral yang baik.

Manusia pandai/cerdas dalam menyelesaikan permasalahan karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki tentu akan lebih mampu dalam menyelesaikan permasalahan. Tapi bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran, mengambil kesimpulan berdasarkan pengalaman dan pengetahuan yang dimiliki. Tanpa memiliki kemempuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik. Demikian juga dengan kemampuan menalar yang sangat baik, namun tanpa bekal pengetahuan dan pengalaman yang memadai, manusia juga tidak akan bisa menyelesaikan masalah dengan baik[9].

Demikian juga agar mesin bisa cerdas (bertindak seperti dan sebaik manusia) maka harus diberi bekal pengetahuan, sehingga mempunyai kemampuan untuk menalar. Untuk membuat aplikasi kecerdasan buatan ada 2 bagian utama yang sangat dibutuhkan:

1. Basis pengetahuan (*Knowledge Base*), bersifat fakta-fakta, teori, pemikiran dan hubungan antar satu dengan yang lainnya.
2. Motor Inferensi (*Inference Engine*), kemampuan menarik kesimpulan berdasarkan pengetahuan dan pengelaman.

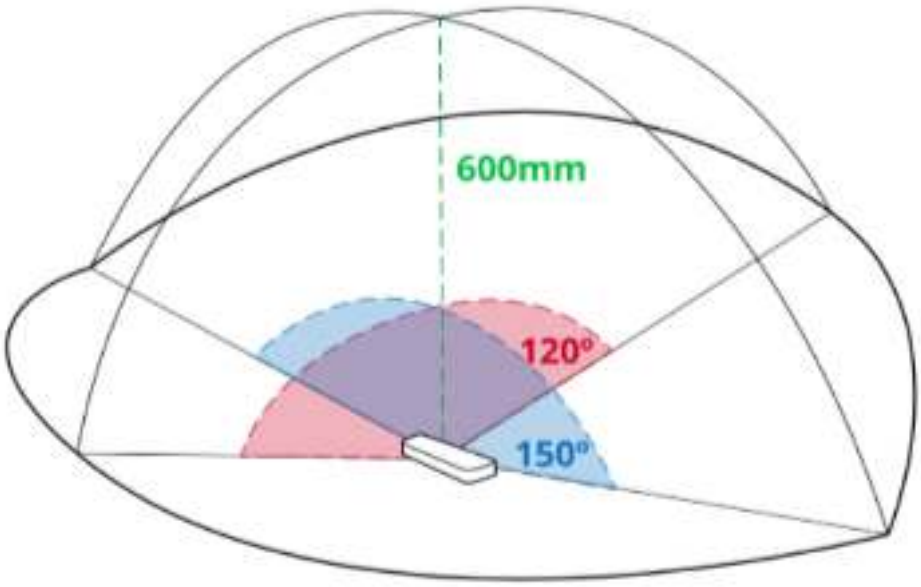
Penerapan konsep kecerdasan buatan pada komputer adalah sebagai berikut:



Gambar 2.2 Penerapan Konsep Kecerdasan Buatan Di Komputer

2.3 *Leap Motion Controller*

Leap Motion Controller adalah perangkat yang berguna untuk mendeteksi gerak isyarat tangan yang tidak bisa diwakili oleh tetikus dan papan tombol. Gerak isyarat tangan ditangkap oleh perangkat ini kemudian dipetakan dalam berbagai bentuk. Salah satu contohnya adalah untuk memutar, menggerakkan secara bebas sebuah objek, atau menambahkan efek tertentu di dalam sebuah aplikasi. Leap Motion Controller mampu mendeteksi jari tangan dari jarak 25 milimeter hingga 600 milimeter. Alat ini memiliki dua kamera monokromatik dan tiga LED inframerah dengan akurasi hingga 1/100 milimeter. Area jangkauan penjejakan *Leap Motion Controller* diilustraikan pada gambar 2.3[11].



Gambar 2.3 Area Jangkauan *Leap Motion Controller*

A. *Hand Gesture Detection*

LMC mempunyai *library* yang menyediakan nilai berupa vektor dari setiap *gesture* tangan yang di tangkap menggunakan LMC. Vektor yang di ambil dari LMC terdiri dari 6 vektor, yaitu :

1. *Palm Position* : Pusat telapak tangan yang terbagi menjadi tiga vektor yaitu *palm1, palm2,* dan *palm3*
2. *Hand Pitch* : Sudut sumbu x
3. *Yaw* : Sudut sumbu y
4. *Roll* : Sudut sumbu z

2.4 Jaringan Syaraf Tiruan

JST merupakan suatu sistem pemrosesan informasi yang mempunyai karakteristik jaringan syaraf biologi yang memcoba meniru kinerja otak manusia. Baik tidaknya suatu model JST ditentukan oleh pola antar *neuron* (arsitektur jaringan), metode untuk menentukan dan mengubah bobot (metode pembelajaran), dan fungsi aktivasi. Kelebihan dari JST mampu mengakuisisi pengetahuan walau tidak ada kepastian dan mampu melakukan generalisasi dan ekstraksi dari suatu pola tertentu. Sedangkan kekurangannya yaitu kurang mampu untuk melakukan operasi-operasi numerik dengan presisi tinggi dan lamanya proses training yang dapat terjadi dalam waktu yang sangat lama untuk jumlah data yang besar. Contoh dari metode JST yaitu Backpropagation merupakan salah satu metode pelatihan yang terawasi (klasifikasi) untuk operasi pada JST *multilayer*. Selanjutnya metode LVQ merupakan suatu metode untuk melakukan pelatihan terhadap lapisan-lapisan kompetitif untuk mempelajari dan mencari pola-pola menarik pada masukan yang diberikan. Hasil dari penelitian bahwa dengan metode LVQ lebih baik dari segi waktu dengan tetap mengedepankan akurasi [12].

i. Klasifikasi dengan Jaringan Syaraf Tiruan

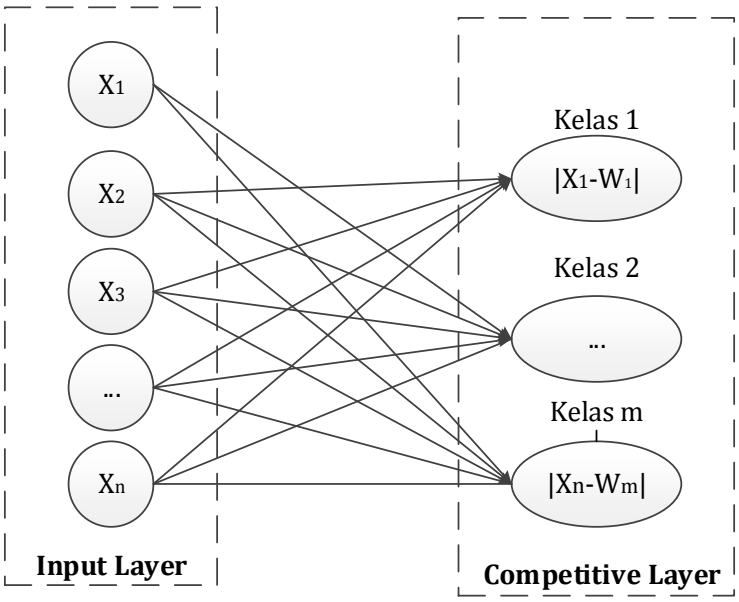
Klasifikasi adalah suatu proses menemukan sekumpulan model atau fungsi yang menjelaskan dan membedakan data kedalam kelas-kelas tertentu. Secara umum terdapat dua proses klasifikasi pada JST yaitu proses pelatihan dan pengujian. Pada proses pelatihan, JST diberikan pengetahuan yang berupa pola-pola data sebagai masukan untuk dilatih dan menghasilkan sebuah model JST. Pada tahap pengujian, JST mengenali pola masukan yang diujikan yang kemudian dicocokkan dengan model hasil dari proses pelatihan. Terdapat beberapa penelitian terdahulu pengenalan pola wajah dengan metode wavelet. Proses identifikasi wajah dimulai dari masukan berupa citra dari *webcame*, menentukan lokasi titik ciri, sampai proses pengidentifikasian dengan menghitung kesamaan global antara wajah yang diuji dengan citra wajah yang ada pada referensi. Gabor Wavelet digunakan untuk memunculkan ciri-ciri khusus dari citra wajah, setelah dilakukan

pengujian diperoleh hasil bahwa program aplikasi dapat mengidentifikasi wajah dengan baik [13].

ii. Klasifikasi dengan Learning Vector Quantization (LVQ)

LVQ merupakan suatu metode klasifikasi pola yang masing-masing unit keluaran mewakili kategori atau kelas tertentu. Metode LVQ digunakan untuk pengelompokan dengan jumlah target atau kelas sudah ditentukan sebelumnya. Pada penelitian sebelumnya proses pembelajaran menggunakan metode LVQ setiap vektor gambar dipelajari dan menghasilkan satu vektor nilai bobot dengan banyaknya pengujian 25 kali untuk setiap parameter [12].

Setiap kelas diwakili atas satu set data yang digunakan dalam pelatihan, yang kemudian disebut bobot. Suatu lapisan kompetitif secara otomatis belajar untuk mengklasifikasikan vektor-vektor masukan. Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitif ini hanya tergantung pada jarak antara vektor-vektor masukan dengan vektor bobot, yang kemudian bobot dari kelas pemenang direvisi. Jika dua vektor masukan mendekati sama, maka lapisan kompetitif meletakkan kedua vektor masukan tersebut ke dalam kelas yang sama. Prinsip kerja dari algoritma LVQ adalah pengurangan *node-node* yang pada akhirnya hanya ada satu *node* output yang terpilih. Gambar 2.8 menunjukkan arsitektur LVQ.



Gambar 2.4 Arsitektur LVQ

keterangan:

X_1-X_n adalah vektor masukan

W_1-W_m adalah vektor bobot pada lapisan kompetitif.

Tahap awal pembelajaran LVQ inisialisasi bobot untuk tiap-tiap *node* yang didapatkan dari *node-node* input yang mewakili masing-masing kelas. Setelah menentukan bobot, maka jaringan diberi masukan sejumlah dimensi *neuron* masukan. Setelah masukan diterima jaringan, maka jaringan mulai melakukan perhitungan jarak vektor yang didapatkan dengan jarak antara vektor masukan dengan vektor bobot.

Akhir pelatihan diperoleh bobot akhir (W). Bobot-bobot ini nantinya digunakan untuk identifikasi data yang lain. Pada saat pelatihan hanya bobot pemenang dengan nilai minimum yang dilakukan revisi. Tahapan algoritma LVQ, yaitu:

1. Tetapkan:
   a. Bobot (w),
   b. Maksimum iterasi (*MaxEpoch*)
   c. Laju pembelajaran (α)
   d. Laju pembelajaran minimum (eps)
   e. *Epoch* = 0
2. Kerjakan jika iterasi belum mencapai maksimum (*Epoch* < *MaxEpoch)* atau laju pembelajaran lebih besar dari laju pembelajaran minimum (*α* > *eps)*
   a. *Epoch* = *epoch* + 1
   b. Kerjakan untuk i = 1 sampai maksimum iterasi (*MaxEpoch*)
   c. Tentukan jarak (J) untuk mencari w minimum dengan Persamaan (2.4).

$$J = \sqrt{(x_{11} - w_{11})^2 + \ldots + (x_{1n} - w_{1n})^2} \quad (2.4)$$

   d. Perbaiki w dengan ketentuan:

      Jika kelas target sama dengan kelas pemenang maka:

$$W_j = W_j + \alpha\,(x - W_j) \quad (2.5)$$

      Jika kelas target berbeda dengan kelas pemenang maka:

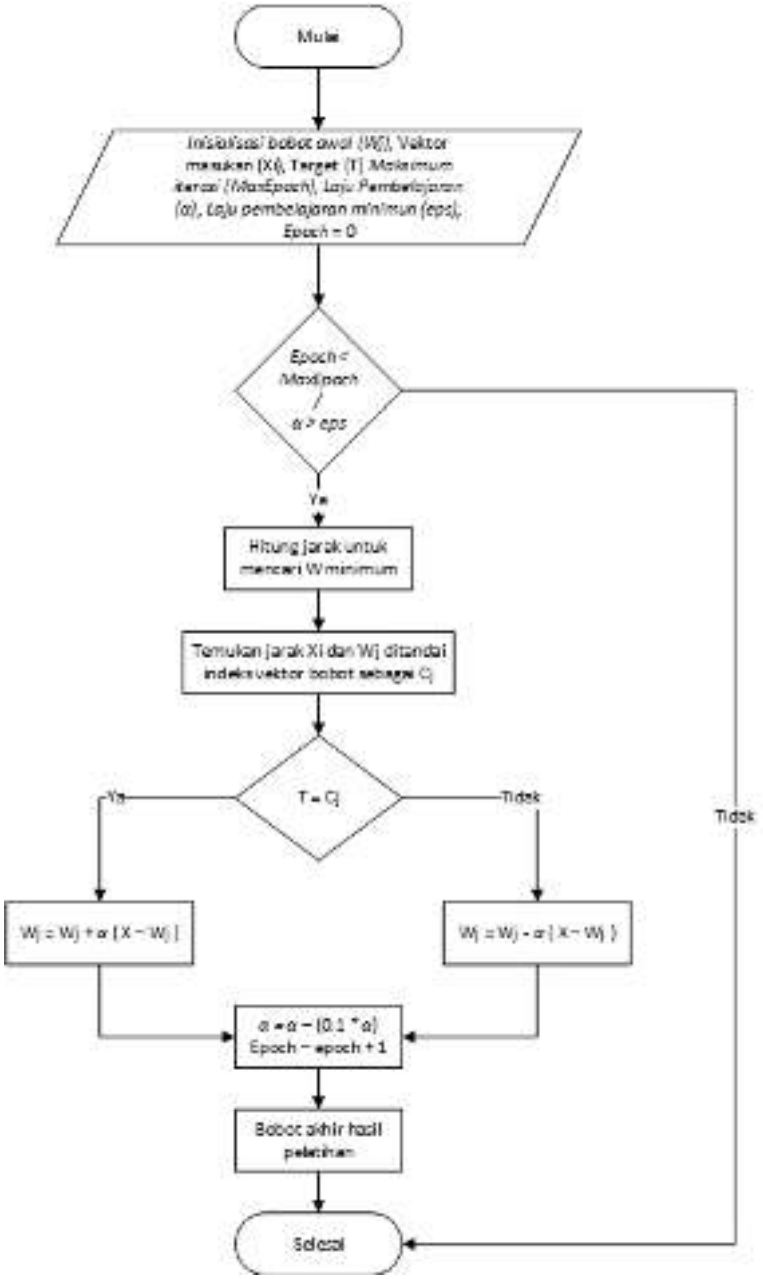
$$W_j = W_j - \alpha\,(x - W_j) \tag{2.6}$$

kurangi nilai α.

$$\alpha = \alpha - (0.1 * \alpha) \tag{2.7}$$

Maksimum iterasi yang digunakan yaitu 1.000, laju pembelajaran 0.01. Jika α terlalu cepat atau terlalu lambat hasil pengenalan untuk tingkat akurasi berpotensi rendah, maka laju pembelajaran yang digunakan yaitu 0.01[14].

Bobot hasil akhir pelatihan disimpan pada basis data, selanjutnya dilakukan klasifikasi menggunakan LVQ. Vektor masukan data uji diklasifikasi untuk mencari jarak minimum terhadap vektor bobot hasil pelatihan sesuai dengan Persamaan (2.4) Bobot dengan jarak minimum merupakan target atau kelasnya.



Gambar 2.5 Bagan Alir Pembelajaran (*Training*) LVQ1

Pembelajaran LVQ3 dalam sistem dikembangkan berdasarkan algoritma LVQ1 dan ketentuan dasar teori LVQ3.

1. Tetapkan:
   f. Bobot (w),
   g. Maksimum iterasi (*MaxEpoch*)
   h. Laju pembelajaran (α)
   i. Laju pembelajaran minimum (eps)

j. *Epoch* = 0

2. Kerjakan jika iterasi belum mencapai maksimum (*Epoch* < *MaxEpoch)* atau laju pembelajaran lebih besar dari laju pembelajaran minimum (*α* > *eps)*

e. *Epoch* = *epoch* + 1

f. Kerjakan untuk i = 1 sampai maksimum iterasi (*MaxEpoch*)

g. Tentukan jarak (J) untuk mencari w minimum dengan Persamaan 2.7.

$$J = \sqrt{(x_{11} - w_{11})^2 + \ldots + (x_{1n} - w_{1n})^2} \quad (2.7)$$

h. Nilai window (ε), digunakan sebagai daerah yang harus dipenuhi untuk memperbaharui vektor referensi pemenang (Y1) dan *runner-up* (Y2) jika berada di kelas yang berbeda. Persamaan window (ε):

$$\text{Min} = \left(\frac{\text{dc1}}{\text{dc2}}, \frac{\text{dc2}}{\text{dc1}}\right) > (1 - \varepsilon)(1 + \varepsilon) \quad (2.8)$$

Nilai window (ε) = 0.3 (Fausett:1994)

i. Jika memenuhi kondisi window (ε), maka pembaharuan menggunakan rumus:

$$w1,t+1 = w1,t - \alpha t(x - w1t) \quad (2.9)$$

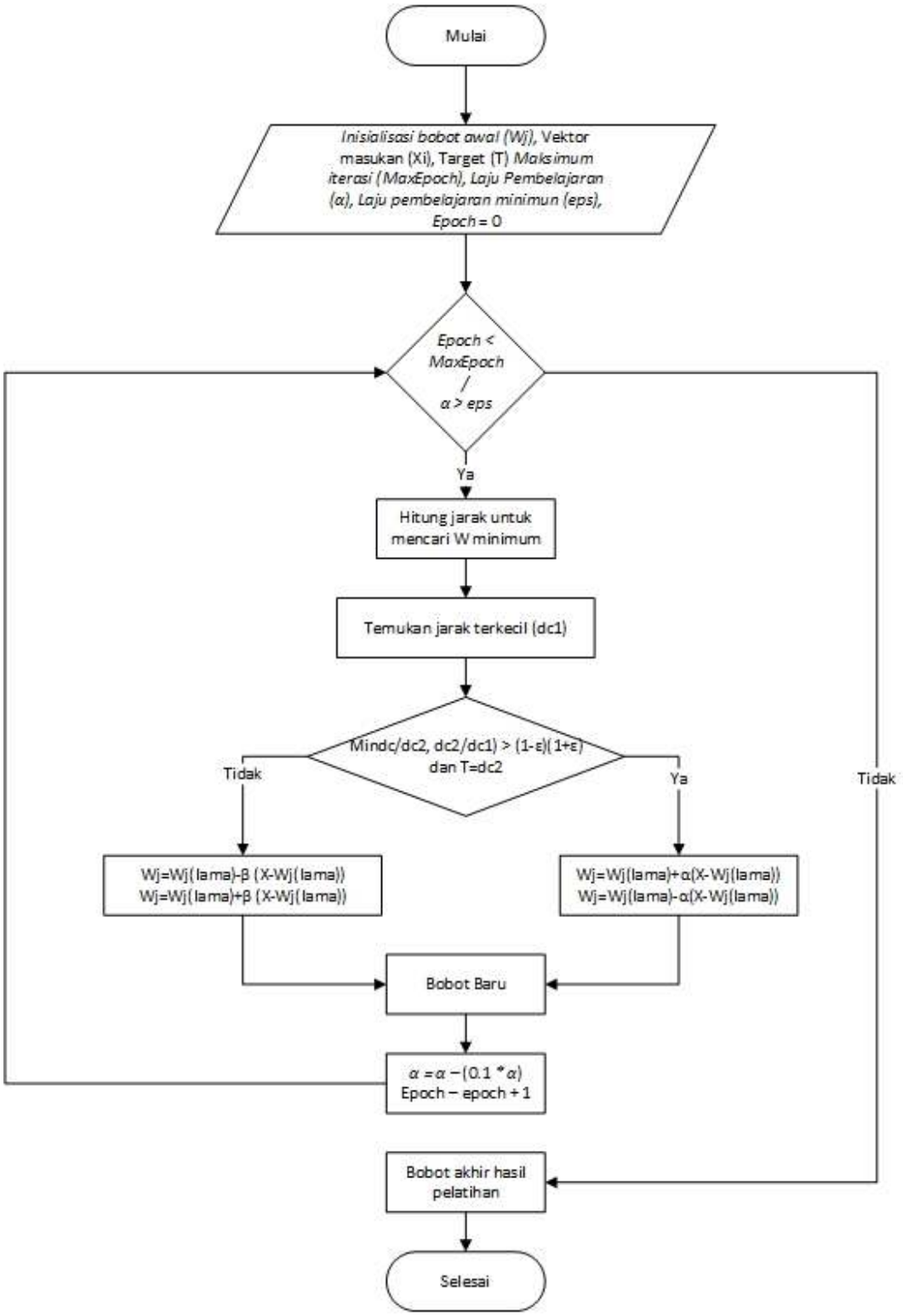
$$w1,t+1 = w1,t + \alpha t(x - w1t)$$

j. Sedangkan jika tidak memenuhi maka pembaharuan menggunakan rumus:

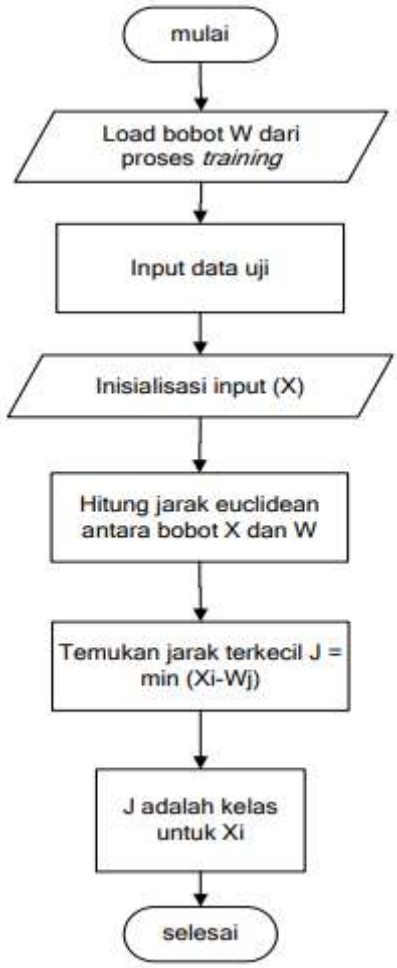
$$w1,(t+1) = w1(t) - \beta(t)\,(x - w1(t)) \quad (2.10)$$

$$w1,(t+1) = w1(t) + \beta(t)\,(x - w1(t))$$

k. $\beta(t) = \varepsilon\alpha(t)$ (2.11)



Gambar 2.6 Bagan Alir Pembelajaran (*Training*) LVQ3 Dalam Sistem



Gambar 2.7 Bagan Alir Pengujian (Testing) LVQ

2.5 Perancangan Berorientasi Objek

Pada metode pengembangan sistem ini akan menggunakan *object oriented Progrmaming* (OOP) yaitu paradigma dalam rekayasa perangkat lunak yang didasarkan pada objek dan kelas. Object-oriented programming merupakan metode terbaik dalam rekayasa perangkat lunak dan mencakup bidang aplikasi yang sangat luas. Karena luasnya cakupan object-oriented programming maka terdapat beberapa hal yang membuat bingung mengenai beberapa istilah dan konsep object-oriented programming. Istilah dan konsep yang berkenaan dengan object-oriented programming ini adalah:

1. *Object-oriented Analysis* adalah metode analisis yang memeriksa requirement (syarat/keperluan yang harus dipenuhi suatu sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemukan dalam ruang lingkup permasalahan.
2. *Object-oriented Design* adalah metode untuk mengarahkan arsitektur perangkat lunak yang didasarkan pada manipulasi objek-objek sistem atau subsistem.
3. *Object Oriented Analysis and Design* (OOAD) merupakan pendekatan yang menekankan pada solusi logis berbasis objek. Beberapa konsep dasar dalam OOAD adalah sebagai berikut:

a. Objek

Objek adalah "benda", secara fisik atau konseptual dapat ditemui dalam kehidupan sehari-hari. Perangkat keras, pernagkat lunak, dokumen, manusia bahkan konsep semuanya adalah konsep objek. Sebuah objek memiliki keadaaan sesaat (*state*) dan perilaku (*behaviour*). State dari sebuah objek adalah kondisi dari objek tersebut atau himpunan dari keadaan yang menggambarkan objek tersebut.

b. Kelas

Kelas adalah definisi umum untuk himpunan objek sejenis. Kelas menetapkan spesifikasi prilaku dan atribut objek-objek tersebut.

Object Oriented Programming merupakan metode yang paling baik dalam rekayasa perangkat lunak diantaranya *procedure-oriented*, *object-oriented*, data *struktur-oriented*, data *flow-oriented*, dan *constraint-oriented*. Sehingga dengan metode *object-oriented programming* dapat diaplikasikan dalam seluruh ruang lingkup rekayasa perangkat lunak.

Untuk memahami keunggulan OOAD (*Object Oriented Analysis and Design*) maka perlu memahami masalah yang dihadapi, antara lain sebagai berikut:

1. Software sulit untuk dimodifikasi bila memerlukan pengembangan.
2. Proses pembuatan software memerlukan waktu yang cukup lama sehingga kadang kala melebihi anggaran dalam pembuatannya.

Para programmer selalu membuat software dari dasar karena tidak adanya kode yang bisa digunakan ulang (*reuse*).

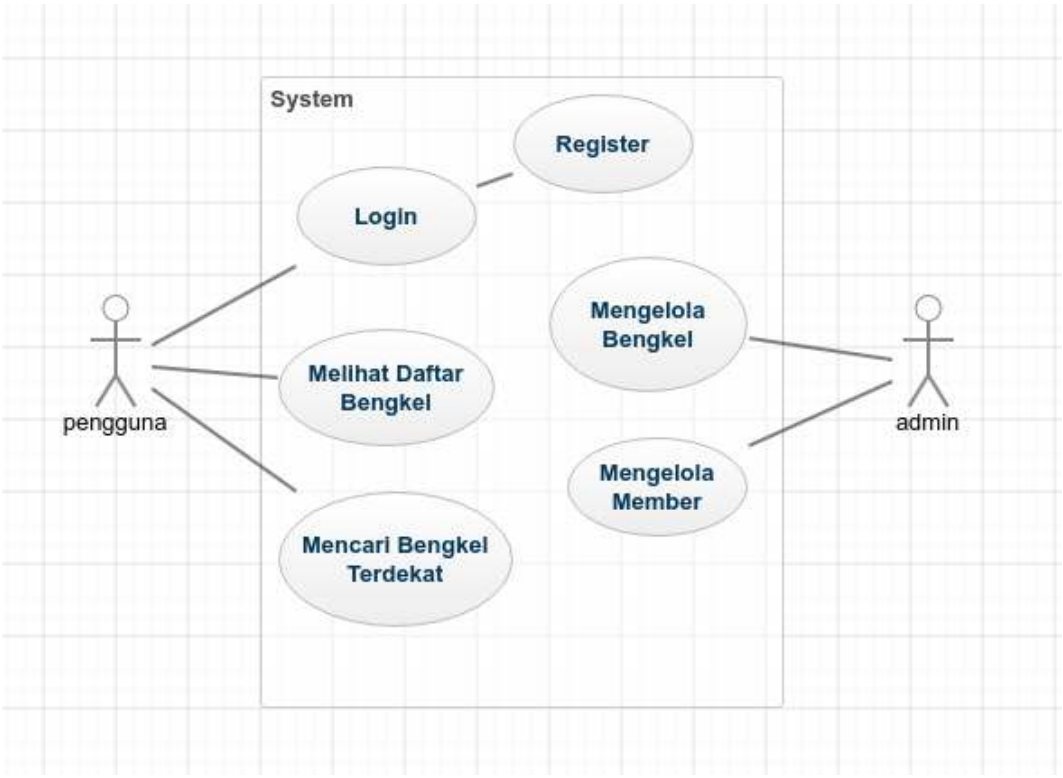
Dari beberapa keunggulan diatas, dengan menggunakan object oriented programming maka sangat menguntungkan bagi programmer karena programmer dapat mendesign program dalam bentuk objek-objek dan hubungan antara objek-objek tersebut untuk kemudian dimodelkan dalam sistem nyata. Keuntungan yang lain adalah proses pembuatan perangkat lunak dapat dilakukan dengan lebih cepat karena perangkat lunak dibangun dalam objek-objek standar, sehingga dapat digunakan secara berulang-ulang[15].

2.6 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah termasuk ke dalam rumpun jenis pemodelan notasi grafis yang didukung oleh meta-model tunggal. Pemodelan ini berguna untuk membantu dalam menjelaskan dan merancang perangkat lunak yang dibangun dengan *object-oriented* (OO). UML merupakan standar terbuka yang dikelola oleh *Open Management Group* (OMG) yang berada dibawah naungan perusahaan-perusahaan konsorsium terbuka [16]. UML merupakan bahasa pemodelan yang terdiri banyak model diantaranya adalah:

1. *Use Case Diagram*

Use case adalah teknik untuk menggambarkan fungsional sebuah sistem. *Use case diagram* memperlihatkan hubungan diantara *actor* dan *use case*. Aktor mempresentasikan seorang user atau subsistem lain yang akan berinteraksi dengan sistem. Sedangkan *use case* merupakan urutan kejadian yang menggambarkan interaksi antara user dengan sistem. Fungsionalitas sistem didefinisikan ke dalam use case dari sudut eksternal sistem yang berguna untuk diuji kelayakan sistem.



Gambar 2.8 Contoh *Use Case Diagram*

2. *Sequence Diagram*

Sequence diagram menggambarkan interaksi diagram yang menunjukan bagaimana kelompok-kelompok objek saling berkolaborasi dalam beberapa behavior, Sebuah *Sequence diagram* secara khusus menjabarkan *behavior* sebuah skenario tunggal. Diagram tersebut menunjukan sejumlah objek contoh dan pesan-pesan yang melewati objek ini dalam *use case*. *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/even untuk menghasilkan output tertentu.

Gambar 2.9 Contoh *Sequential Diagram*

3. *Class Diagram*

Class diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka. Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendifinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas, atribut mendeskripsikan properti dengan sebaris teks di dalam kotak kelas tersebut. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

2.7 Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (suite) 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 [17]. 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 diatas .NET Compact Framework).

2.8 Pengujian *Black Box*

Pengujian yang mengabaikan mekanisme internal sistem atau komponen dan fokus semata-mata pada output yang dihasilkan yang merespon input yang dipilih dan kondisi eksekusi. Pengujian yang dilakukan untuk mengevaluasi pemenuhan sistem atau komponen dengan kebutuhan fungsional tertentu. [19]

Tujuan Black Box adalah menemukan:

1. Fungsi yang tidak benar atau hilang
2. Kesalahan interface
3. Error pada struktur data atau akses database external
4. Error pada kinerja
5. Error pada saat inisialisasi dan terminasi
6. Kesensitifan sistem terhadap nilai input tertentu
7. Batasan dari suatu data

2.9 Confusion Matrix

Confusion Matrix berisi informasi tentang klasifikasi aktual dan yang telah diprediksi yang dilakukan oleh sistem klasifikasi. Kinerja sistem tersebut umumnya dievaluasi dengan menggunakan data dalam matriks [20]. Tabel berikut menunjukkan confusion matrix untuk klasifikasi dua kelas.

Tabel 2.1 Tabel Confusion Matrix

		Predicted Class	
		Positif	**Negatif**
Actual Class	**Positif**	True Positif	False Positif
	Negatif	False Negative	True Negative

True positives adalah jumlah record positif yang diklasifikasikan sebagai positif, false positives adalah jumlah record positif yang diklasifikasikan sebagai negatif, false negatives adalah jumlah record negatif yang diklasifikasikan sebagai positif, true negatives adalah jumlah record negatif yang diklasifikasikan sebagai negatif. Setiap kolom dari confusion matrix merupakan contoh di kelas yang telah diprediksi, sedangkan setiap baris mewakili contoh di kelas yang sebenarnya. Setelah didapat true positives, false positives, true negatives dan false negatives, selanjutnya hitung nilai precision dan akurasinya. Precision adalah ukuran terhadap suatu kelas yang telah diprediksi. Berikut persamaan dari precision dan akurasi.

Dimana :

$$Akurasi = \frac{TP+TN}{TP+FP+TN+FN} \quad (2.12)$$

$$Precision = \frac{TP}{TP+FP} \quad (2.13)$$

$$Recall = \frac{TP}{TP+FN} \quad (2.14)$$

TP : Jumlah *True Positive*

FP : Jumlah *False Positive*

TN : Jumlah *True Negative*

FN : Jumlah *False Negative*