BAB 2
LANDASAN TEORI

2.1. Audio

Secara umum audio adalah representasi dari suara atau bunyi, sedangkan suara ialah gelombang longitudinal yang merambat melalui perantara. Sedangkan untuk *Digital Audio* secara mudah ialah cara alternatif dalam merepresentasikan gelombang audio dalam bentuk digital yaitu dalam bentuk *binary* [4].

Frekuensi suara adalah jumlah gelombang suara yang terjadi dalam suatu unit waktu yang telah ditentukan. Satuan frekuensi suara adalah *Hertz* atau gelombang suara per detik. Suara dihasilkan dari getaran sebuah benda, dan setiap benda memiliki frekuensi yang berbeda-beda, ada yang tinggi dan juga rendah, dan semakin tinggi frekuensi suara dari suatu benda maka akan semakin nyaring suara yang dihasilkan.

Audio ataupun suara juga memiliki kualitas bunyi, disebut juga *Timbre* atau warna bunyi. Kualitas inilah yang membedakan suara antar alat musik meskipun memainkan nada yang sama, misalnya gitar dan biola. Kualitas suara ini pula yang membedakan suara manusia maupun alat musik dan sumber-sumber suara lainnya. Pada digital audio perbedaan antar sumber suara tersebut dapat ditemukan dengan terlebih dahulu menganalisa fitur pada audio untuk mendapatkan kualitas suaranya

2.2. *Mixed Type Audio*

Mixed Type Audio adalah sebuah berkas audio yang memiliki lebih dari satu jenis sumber suara, misalnya ialah *speech* atau suara manusia yang digabung dengan musik, ataupun digabung dengan suara lingkungan sekitar [2].

Seperti yang telah dijelaskan pada bagian audio, membedakan *mixed type audio* akan menggunakan warna suara sebagai atribut pembedanya. Biasanya

sebuah audio berisikan musik, akan memiliki rasio harmonis yang lebih tinggi daripada suara manusia, sedangkan suara bicara manusia akan memiliki bunyi konsonan dan huruf vokal didalamnya [2].

Pada penelitian ini, yang menjadi objek pengklasifikasian adalah berkas audio, yanga akan dikelompokan kedalam kelas-kelas *mixed type* diantaranya musik *mixed environment*, dan *environment mixed speech*. Pengelompokan akan berdasarkan pada ciri-ciri khusus pada setiap berkas audio yang diolah.

2.3. Kecerdasan Buatan

Kecerdasan Buatan (*Artificial Intelligence*) didenfinisikan sebagai kecerdasan yang ditunjukan oleh suatu *entitas* buatan. Sistem seperti ini umumnya dianggap computer. Kecerdasan diciptakan dan dimasukkan kedalam suatu mesin (komputer) agar dapat melakukan perkerjaan seperti yang dapat dilakukan manusia [5].

2.3.1. Definisi Kecerdasan Buatan

Kecerdasan buatan merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia. Pada awal diciptakannya, computer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan zaman, maka peranan computer semakin mendominasi kehidupan umat manusia. komputer tidak lagi hanya digunakan sebagai alat hitung, lebih dari itu komputer diharapkan dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan manusia.

Manusia bisa menjadi padai dalam menyelesaikan segala permasalahan di dunia ini karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki seseorang tentu saja diharapkan akan lebih mampu dalam menyelesaikan permasalahan. Namun bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran, mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang mereka miliki. Tanpa memiliki kemampuan untuk menalar

dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik. Demikian pula, dengan kemampuan menalar yang sangat baik, namun tanpa bekal pengetahuan dan pengalaman yang memadai, manusia juga tidak akan bisa menyelesaikan masalah dengan baik.

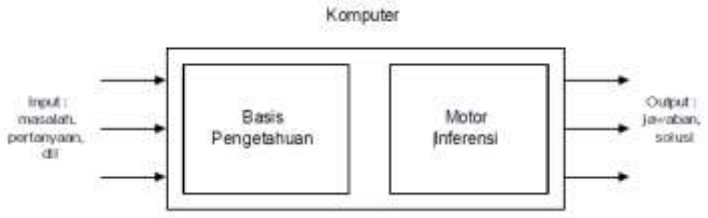
Agar computer bisa bertindak seperti dan sebaik manusia, maka computer juga harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Untuk itu pada *artificial intelegence* (AI) akan mencoba untuk memberikan beberapa metoda untuk membekali computer dengan kedua komponen tersebut agar computer bisa menjadi mesin yang pintar.

Pengertian kecerdasan buatan dapat dipandang dari berbagai sudut pandang antara lain :

a. Sudut pandang kecerdasan
   Kecerdasan buatan akan membuat mesin menjadi cerdas (mampu berbuat seperti apa yang dilakukan oleh manusia).
b. Sudut pandang penelitian
   Kecerdasan buatan adalah suatu studi bagaimana membuat agar computer dapat melakukan sesuatu sebaik yang dikerjakan oleh manusia.
c. Sudut pandang bisnis
   Kecerdasan buatan adalah kumpulan peralatan yang sangat powerful dan metodologis dalam menyelesaikan masalah-masalah bisnis.
d. Sudut pandang pemograman
   Kecerdasan buatan meliputi studi tentang pemograman simbolik, penyelesaian masalah (problem solving) dan pencarian (searching).

Untuk membuat aplikasi kecerdsan buatan ada 2 bagian utama yang sangat dibutuhkan yaitu :

1. Basis pengetahuan (*knowledge base*), berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
2. Motor inferensi (*inference engine),* yaitu kemampuan menarik kesimpulan berdasarkan pengalaman.



Gambar 2.1 Penerapan Konsep Kecerdasan Buatan Pada Komputer

2.3.2. Kecerdasan Buatan dan Kecerdasan Ilmiah

Kecerdasan alamiah adalah kecerdasan yang dimiliki oleh manusia jika dibandingkan dengan kecerdasan buatan ada beberapa keuntungan kecerdasan buatan dibanding kecerdasan alamiah, yaitu [5] :

1. Lebih permanen

   Kecerdasan alamiah akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer dan program tidak diubah

2. Memberikan kemudahan dalam duplikasi dan penyebaran.

   Mentransfer pengetahuan manusia dari satu orang ke orang lain membutuhkan proses yang sangat lama dan keahlian itu juga tidak akan pernah dapat diduplikasi dengan lengkap. Oleh karena itu, jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebut dapat disalin dari komputer tersebut dan dapat dipindahkan dengan mudah ke komputer yang lain.

3. *Relative* lebih murah dari kecerdasan alamiah

   Menyediakan layanan komputer akan lebih mudah dan lebih murah dibandingkan dengan harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka watu yang sangat lama.

4. Konsisten dan teliti

Hal ini disebabkan karena kecerdasan buatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami akan senantiasa berubah-ubah.

5. Dapat didokumentasi

Keputusan yang dibuat oleh komputer dapat didokumentasi dengan mudah dengan cara melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi

6. Dapat mnegerjakan beberapa task dengan lebih cepat dan lebih baik dibanding manusia.

Sedangkan keuntunga kecerdasan alamiah dibanding kecerdasan buatan yaitu :

1. Bersifat lebih kreatif

   Kemampuan untuk menambah ataupun memenuhi pengetahuan itu sangat melekat pada jiwa manusia. Pada kecerdasan buatan, untuk menambah pengetahuan harus dilakukan melalui sistem yang dibangun.

2. Dapat melakukan proses pembelajaran secara langsung, sementara AI harus mendapatkan masukann berupa symbol dan representasi.

3. Focus yang luas sebagai referensi untuk pengambilan keputusan sebaiknya AI menggunakan focus yang sempit.

Komputer dapat digunakan untuk mengumpulkan informasi tentang objek, kegiatan (*events*), proses dan dapat memproses sejumlah besar informasi dengan lebih efisien dari yang dapat dikerjakan manusia. Namun disisi lain, manusia dengan menggunakan insting dapat melakukan hal yang sulit diprogram pada komputer, yaitu kemampuan mengenali (*recognize*) hubungan antara hal-hal tersebut, menilai kualitas dan menemukan pola yang menjelaskan hubungan tersebut.

2.4. Jaringan Syaraf Tiruan

Jaringan saraf tiruan (JST) adalah sistem pemrosesan informasi yang memiliki karakteristik mirip dengan jaringan saraf biologi. Hal ini menyerupai kerja otak dalam dua hal yaitu [6]:

1. Pengetahuan diperoleh oleh jaringan melalui suati proses belajar.
2. Kekuatan hubungan antar sel syaraf yang dikenal dengan bobot sinapsis digunakan untuk menyimpan pengetahuan.

JST merupakan sebuah model komputasi dari otak manusia yang mampu melakukan perhitungan, pengenalan, pengamatan serta pengambilan keputusan. Jaringan syaraf tiruan memanfaatkan struktur pengolahan paralel atas sejumlah pengolah sederhana dan hubungan antar pengolah tersebut. Dalam jaringan syaraf tiruan, terdapat elemen pengolah yang merupakan model dari neuron. Setiap hubungan elemen pengolah menentukan kemampuan JST. Seperti halnya jaringan saraf biologis, JST juga memiliki kemampuan untuk belajar dan beradaptasi terhadap masukan-masukannya. JST tidak perlu diprogram secara eksplisit, karena JST dapat belajar dari beberapa contoh pelatihan. Secara umum proses pembelajaran pada jaringan syaraf tiruan dibagi menjadi dua macam, yaitu:

1. Belajar dengan pengawasan (*Supervised Learning)*

   Pada metode *supervised learning* jaringan diberikan vektor target yang harus dicapai sebagai dasar untuk mengubah hubungan interkoneksi atau bobot pada jaringan. Contoh jaringan yang belajar dengan pengawasan adalah *Backpropagation* dan Perceptron.

2. Belajar Tanpa Pengawasan (*Unsupervised Learning*)

   Berbeda dengan *supervised learning* yang masih membutuhkan pengawasan dan sangat tergantung pada *knowledge* yang diberikan pada pembangunan jaringan, metode *unsupervised learning* ini melatih jaringan terhadap suatu *input*an tanpa adanya pengawasan, dimana vektor target tidak ditentukan. Vektor masukan dimasukkan ke dalam jaringan, kemudian sistem akan

mengatur dirinya sendiri sedemikian rupa sehingga dihasilkan keluaran yang konsisten bilamana pola yang menyerupai vektor masukan tersebut diberikan. Contoh jaringan yang belajar tanpa pengawasan adalah *Adaptive Resonance Theory 2* (ART 2), *self organizing map* (SOM), dan *Learning Vektor Quantization* (LVQ).

2.5. *Backpropagation*

Backpropagation adalah salah satu algoritma *supervised learning* yang digunakan dalam *artificial neural networks. Backpropagation* [7] mencari kombinasi bobot untuk meminimalkan kesalahan output untuk dianggap menjadi solusi yang benar. Secara sederhana *backpropagation* dilakukan dalam dua tahap, yaitu :

1) *Feed-forward*

   Pola yang akan dilatih diset ke setiap unit di *input layer*, lalu *output* yang dihasilkan ditransmisikan ke *layer* selanjutnya terus sampai *output layer.*

2) *Backpropagation*

   Berdasarkan *output* yang diharapkan, setiap bobot disesuaikan agar menghasilkan *error* yang minimal mulai dari bobot yang terhubung ke *ouput neuron*, lalu terus mundur sampai ke *input layer* .

Salah satu *activation function* yang populer untuk jaringan *backpropagation* adalah *binary sigmoid* dengan rentang nilai (0,1) yang didefinisikan sebagai (2.1). untuk tahap *backpropagation* menggunakan fungsi turunannya yaitu (2.2).

$$f(x) = \frac{1}{1+ e^{-x}} \qquad (2.1)$$

$$f'(x) = f(x)(1 - f(x)) \qquad (2.2)$$

Alternatif selain fungsi *binary sigmoid* adalah *bipolar sigmoid* dengan rentang nilai (-1,1) yang didefinisikan sebagai (2.3). untuk tahap *backpropagation* menggunakan fungsi turunannya yaitu (2.4).

$$f(x) = \frac{2}{1+ e^{-x}} - 1 \qquad (2.3)$$

$$f'(x) = \frac{1}{2}(1 + f(x))(1 - f(x)) \qquad (2.4)$$

Berikut tahapan pada proses pealtihan algoritma *backpropagation*.

Langkah 1 : Inisialisasi bobot dan parameter

Langkah 2 : Hitung nilai aktivasi pada *hidden layer*

$$Y_net_j = \sum_{j=1}^{p} Xi\ W1_{i,j} \qquad (2.5)$$

Di mana :

Y_net_j : jumlah total *input* untuk *node* ke-j

P : jumlah *node* pada *hidden layer* sebelumnya, untuk *hidden layer* yang pertama n adalah jumlah *node input*

Xi : nilai node ke-I yang dikeluarkan oleh *hidden layer* sebelumnya, untuk *hidden layer* yang pertama Xi adalah input yang diterima oleh sistem

W1i,j : bobot yang menghubungkan antara *node* ke-i dan *node* ke-j

Hitung *output* setiap *node* pada *hidden layer* dengan fungsi aktivasi

$$Y_j = f(Y_net_j) \qquad (2.6)$$

Di mana :

Y_j : *output* dari *node* ke-j

$$Y_j = \frac{1}{1+ e^{-Y_net_j}} \qquad (2.7)$$

Langkah 3 : Hitung *input* setiap *node* pada *output layer*

$$Z_net_k = \sum_{k=1}^{m} Y_j \, W2_{j,k} \qquad (2.8)$$

Di mana :

Z_net $_k$: jumlah total input untuk *node* ke-k

m : jumlah *node* pada *hidden layer* sebelum *output layer*

W2j,k: bobot yang menghubungkan antara *node* ke-j dan *node* ke-k

Hitung *output* pada setiap *node* pada *output layer*

Zk = f(Z_net $_k$) (2.9)

$$Z_k = \frac{1}{1+ e^{-Z_net\ k}} \qquad (2.10)$$

Langkah 4 : Hitung *error* setiap *node* pada *output layer*

$$\delta_k = (t_k - Z_k)\, f'(Z_net_k) \qquad (2.11)$$

Ket :

k : banyaknya *node* pada *output layer*

t : *output* yang diharapkan

Langkah 5 : Hitung perhitungan perubahan bobot ΔW2 (2.18)

Langkah 6 : Hitung *error* setiap *node* pada *hidden layer*

$$\delta_net_j = \sum_{k=1}^{m} \delta_k w2_{j,k} \qquad (2.12)$$

Ket :

m : jumlah *node* pada *layer* sebelumnya

hitung *error*

$$\delta_j = \delta_net_j f'(Y_net_j) \qquad (2.13)$$

Ket :

f'(Y_net $_j$) : Y_net j * (t – Y_net j) (2.14)

t : *output* yang diharapkan

Langkah 7 : Hitung perhitungan perubahan bobot ΔW1 (2.17)

Langkah 8 : Hitung *update* bobot pada *hidden layer* dan *output layer*

$$W2_{j,k} = W2_{j,k}(lama) + \Delta W2_{j,k} \quad (2.15)$$

$$W1_{i,j} = W1_{i,j}(lama) + \Delta W1_{i,j} \quad (2.16)$$

2.6. *Fixed Size Momentum*

Fixed size momentum pada dasarnya dirancang untuk mengatasi beberapa keterbatasan yang terkait dengan pelatihan *backpropagation.* Dalam rangka untuk mempercepat pelatihan, banyak peneliti menambah bobot update berdasarkan update bobot sebelumnya. Hal ini secara efektif meningkatkan waktu pembelajaran [2]. Banyak algoritma menggunakan informasi dari update bobot sebelumnya untuk menentukan berapa besar sebuah update yang bisa dibuat tanpa divergen.

Fixed size momentum dapat menambah kecepatan melebihi momentum standar. Fixed size momentum di rancang untuk mengatasi panjang history dari update bobot yang sudah ada untuk setiap hubungan yang ada di neural networks. Dengan menggunakan informasi tambahan ini yang disimpan, *fixed size momentum* dapat meningkatkan kecepatan pembelajaran dan juga meningkatkan akurasi [2].

Aturan standar update bobot yang digunakan dalam *backpropagation* adalah :

$$\Delta W_{i,j} = \alpha\, \delta_j\, X_{i,j}$$

Dimana : i = indeks dari sumber node

j = indeks target

α = learning rate

δ_j= backpropagated error term

x= nilai input bobot

cara aturan update bobot ini sangat lambat dan membutuhkan waktu yang lama.

Fixed size momentum menggunakan fixed size window untuk mendapatkan informasi lebih banyak daripada menggunakan standar momentum. Dengan menggunakan lebih banyak memori dapat memungkinkan untuk mengatasi beberapa keterbatasan masalah.

Fixed size momentum dapat mengingat update bobot yang sering muncul dan menggunakan informasi tersebut untuk bobot yang lainnya. Dengan standar momentum , kesalahan dari update sebelumnya sebagian akan diterapkan untuk selanjutnya, untuk kasus terburuk beberapa sampel akan sering memiliki update yang berlawanan. Situasu ini dapat mengganggu momentum yang telah dibangun dan bahkan dapat membuat lebih lama dalam melatih . fixed size momentum dapat melihat history yang lebih luas.

Untuk rumus fixed size momentum adalah sebagai berikut.

$$\Delta W1_{i,j}(t) = \alpha\, \delta_j\, X_i + f(\alpha\, \delta_j X_i, \Delta W1_{ij}(t-1), \Delta W1_{i,j}(t-2), \dots, \Delta W1_{i,j}(t-k)) \quad (2.17)$$

$$\Delta W2_{j,k}(t) = \alpha\, \delta_k\, Y_j + f(\alpha\, \delta_j Y_j, \Delta W2_{j.k}(t-1), \Delta W2_{j.k}(t-2), \dots, \Delta W2_{j,k}(t-k)) \quad (2.18)$$

Dan untuk fungsi f nya adalah sebagai berikut.

$$f(z, \delta_1, \dots, \delta_k) = \begin{cases} -z & if,\ z * \sum \delta_i < 0 \\ 0 & if,\ z * \sum \delta_i \geq 0 \end{cases} \quad (2.19)$$

Terdapat argumen k+1 untuk fungsi f , yang pertama adalah update saat ini dan sisanya adalah update dari k sebelumnya dimana k adalah window size untuk algoritma fixed size momentum. δ_1

Rumus diatas dapat membantu dalam konvergensi lebih cepat dan juga meningkatkan akurasi klasifikasi dan meningkatkan ukuran dari history window yang digunakan untuk melatih jaringan.

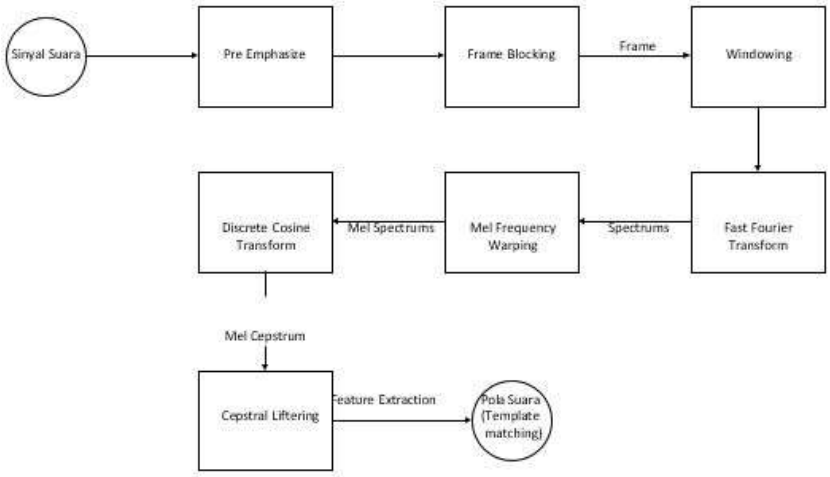
2.7. MFCC (*Mel Frequency Cepstrum Coefficient*)

MFCC (*mel frequency cepstrum coefficient)* merupakan salah satu metode yang banyak digunakan dalam bidang *speech technology*, baik *speaker recognition* maupun *speech recognition.* Metode ini digunakan untuk melakukan *feature extraction* sebuah proses yang mengkonversi sinyal suara menjadi beberapa parameter. Keunggulan dari metode ini adalah :

1. Mampu untuk menangkap karakteristik suara yang sangat penting bagi pengenalan suara atau dengan kata lain mampu menangka informasi-informasi penting yang terkandung dalam sinyal suara.
2. Menghasilkan data seminimal mungkin tanpa menghilangkan informasi-informasi penting yang ada.
3. Mengadaptasi organ pendengaran manusia dalam melakukan persepsi terhadap sinyal suara.

Perhitungan yang dilakukan MFCC menggunakan dasar perhitungan *short-term analysis*. Hal ini dilakukan mengingat sinyal suara yang bersifat quasistationary. Pengujian yang dilakukan untuk periode waktu yang cukup pendek (sekitar 0 sampai 10 milidetik) akan menunjukkan karakteristik sinyal suara yang *stationary*. Tetapi bila dilakukan dalam peridoe waktu yang lebih panjang karakteristik sinyal suara akan terus berubah sesuai dengan kata yang diucapkan.

MFCC *feature extraction* sebenarnya merupakan adaptasi dari sistem pendengaran manusia dimana sinyal suara akan di-filter secara linear untuk frekuensi rendah (dibawah 1000Hz) dan secara logiritmik untuk frekuensi tinggi. berikut ini blok diagram utnuk proses yang dilakukan pada metode MFCC [3]:



Gambar 2.2 Blok Diagram MFCC

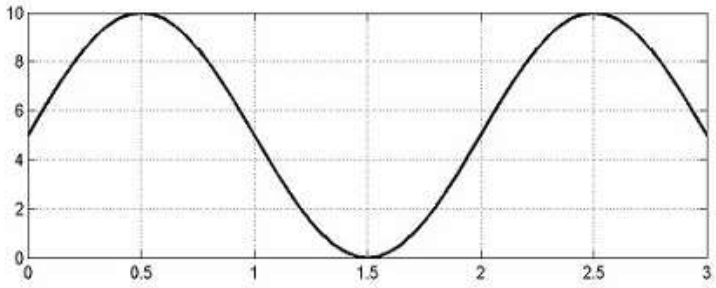
2.7.1. Konversi Analog Menjadi digital

Sinyal–sinyal yang natural pada umumnya, seperti sinyal suara merupakan sinyal *continue* dimana memiliki nilai yang tidak terbatas. Sedangkan pada komputer, semua sinyal yang dapat diproses oleh komputer hanyalah sinyal *discrete* atau sering dikenal dengan istilah digital *signal*. Agar sinyal natural dapat diproses oleh komputer kita harus dapat mengubah data sinyal *continue* menjadi *discrete*. Hal itu dapat melalui tiga proses, diantaranya adalah proses sampling data, proses kuantisasi, dan proses pengkodean.

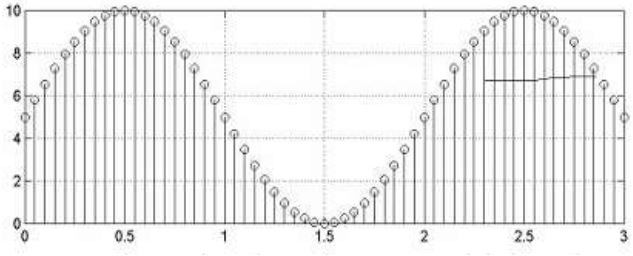
Proses sampling adalah suatu proses untuk mengambil data sinyal *continue* untuk setiap periode tertentu. Dalam melakukan proses sampling data, berlaku aturan Nquist, yaitu bahwa frekuensi sampling (sampling rate) minimal harus dua kali lebih tinggi dari frekuensi maksimum yang akan disamplingkan. Jika sinyal sampling kurang dari dua kali frekuensi maksimum sinyal yang akan disampling, maka akan timbul efek aliasing. Aliasing adalah suatu efek dimana sinyal yang dihasilkan memiliki frekuensi yang berbeda dengan sinyal aslinya,

Proses kuantisasi adalah proses untuk membulatkan nilai data kedalam bilangan-bilangan tertentu yang telah ditentukan terlebih dahulu. Semakin banyak level yang dipakai maka semakin akurat pula data sinyal yang disimpan tetapi akan menghasilkan ukuran data yang besar dan proses lama.

Proses pengkodean adalah proses pemberian kode untuk tiap tiap data sinyal yang telah terkuantisasi berdasarkan level yang ditempati.



Gambar 2.3 Sinyal Sinus



Gambar 2.4 Sinyal Sinus Setelah Tersampling

2.7.2. Pre-Emphasis

Pre-emphasis dilakukan untuk memperbaiki signal dari gangguan *noise*, sehingga dapat meningkatkan tingkat akurasi dari proses feature extraction. Default dari nilai alpha yang digunakan dalam proses pre-emphasis filtering adalah 0,9.

Bentuk yang paling umum digunakan dalam *pre-emphasis filter ά* adalah sebagai berikut:

$$H(z) = 1 - az^{-1}$$

Dimana $0.9 \leq ά \leq 1.0$ dan ά € R, formula diatas dapat diimplementasikan sebagai filter order differentiator, sebagai berikut

$$y[n] = s[n] \text{ a ss } [n - 1]$$

keterangan :

y[n] = sinyal hasil *pre –emphasis filter*

s[n] = sinyal sebelum *pre-emphasis filter*

2.7.3. *Frame Blocking*

Hasil perekaman suara merupakan sinyal analog yang berada dalam domain waktu yang bersifat variant time, yaitu suatu fungsi yang bergantung waktu. Oleh karena itu sinyal tersebut harus dipotong-potong dalam slot-slot waktu tertentu agar dapat dianggap invariant. Setiap potongan sinyal suara tersebut disebut frame.

Untuk menghitung jumlah frame digunakan rumus:

$$Jumlah\ Frame = ((1 - N)/M + 1)$$

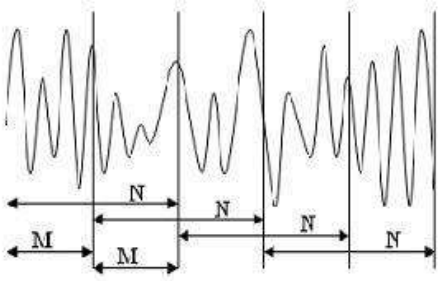
Keterangan :

I = sample rate

N= sample point (sample rate*waktu framing(s))

M= N/2

Potongan *frame* digambarkan seperti gambar dibawah ini :



Gambar 2.5 Framing

2.7.4. *Windowing*

Fungsi *window* yang paling sering digunakan dalam aplikasi *speaker recognation* adalah Hamming Window. Fungsi ini menghasilkan sidelobe level yang tidak terlalu tinggi (kurang lebih -43dB) selain itu *noise* yang dihasilkan pun tidak terlalu besar (kurang lebih 1.36 BINS).

Window Hamming :

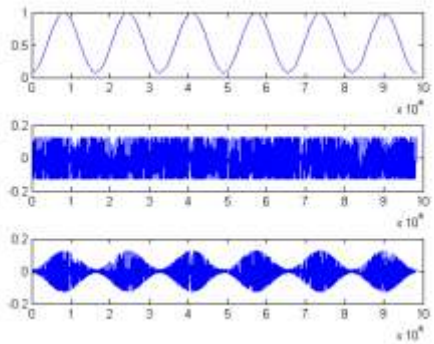
$$W_{ham}(n) = 0.52 - 0.46cos\left[\frac{2\pi n}{N-1}\right] 0 \leq n \leq N-1$$

Keterangan :

$W_{ham}(n)$ = *windowing hamming*

n = sinyal ke [1….n]

N = Sample point



Gambar 2.6 Window Hamming

2.7.5. *Fast Fourier Transform*

FFT (*Fast Fourier Transform*) adalah teknik perhitungan cepat dari DFT. FFT adalah DFT dengan teknik perhitungan yang cepat dengan memanfaatkan sifat periodikal dari *transformasi fourier*. Perhatikan definisi dari FFT :

$$F(k) = \sum_{n=1}^{N} f(n)e^{-j2\pi knT/N}$$

2.7.6. *Mel Frequency Warping* (Filter Bank)

Mel Frequency Warping umumnya dilakukan dengan menggunakan filterbank. Filterbank menggunakan representasi konvolusi dalam melakukan filter terhadap sinyal. Konvolusi dapat dilakukan dengan melakukan multiplikasi antara spektrum sinyal dengan koeffisien *filterbank*.

Berikut ini adalah rumus yang digunakan dalam perhitungan *filterbank.*

$$Y[i] = \sum_{j=1}^{N} S[j]H_i[j]$$

Keterangan :

N = jumlah magnitude spectrum

S[j] = magnitude spectrum pada frekuensi j

$H_i[j]$ = koefisien *filterbank* pada frekuensi j ($1 \leq I \leq M$)

M = jumlah channel dalam *filterbank*

Dimana $H_i = \frac{mel\ f}{xi/2}$

2.7.7. *Discrete Cosine Transform*

Proses ini merupakan langkah akhir dari *feature extraction*. Hasil dari DCT ini adalah fitur-fitur yang dibutuhkan untuk melakukan proses analisa terhadap pengenalan suara tersebut. Menggunakan rumus :

$$T_n = \sum_{k=1}^{K} (\log S_k)\ cos\left[n(k - \frac{1}{2})\frac{\pi}{k}\right]$$

Keterangan :

S_k = keluaran dari proses filterbank pada indeks k

K = jumlah koefisien yang diharapkan

Koefisien ke-nol dari DCT pada umumnya akan dihilangkan, walaupun sebenarnya mengindikasikan energi dari *frame* sinyal tersebut. Hal ini dilakukan karena, berdasarkan penelitian-penelitian yang pernah dilakukan, koefisien ke nol ini tidak *reliable* terhadap *speaker recognation.* Tetapi koefisien ke nol dari DCT sangatlah berguna dalam menentukan *end point* dari suatu suku kata maupun kata. Hal ini disebabkan karena pada *end point* dari suatu suku kata maupun kata mempunyai energi yang lebih rendah daripada *point-point.*

2.7.8. *Cepstral Liftering*

Cepstral, hasil dari proses utama MFCC *feature extraction*, memiliki beberapa kelemahan. *Low-order* dari *cepstral coefficients* sangat sensitif terhadap *spectral slope*, sedangkan bagian *high order*-nya sangat sensitif terhadap *noise*. Oleh karena itu, maka *cepstral liftering* menjadi salah satu standar teknik yang diterapkan untuk meminimalisasi sensitifitas tersebut.

Cepstral liftering dapat dilakukan dengan mengimplementasikan fungsi window terhadap *cepstral features*.

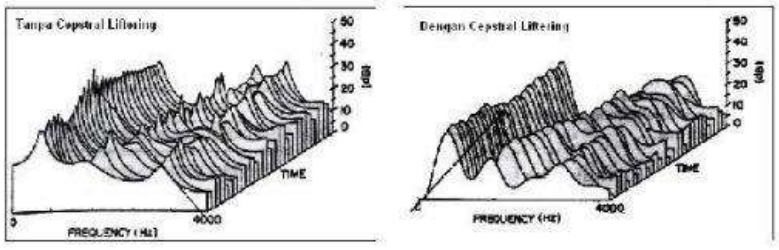
$$W[n] = \left\{1 + \frac{L}{2}\, sin\left(\frac{n\pi}{L}\right)\right\}$$

Keterangan :

L = jumlah cepstral coefficients

n = index dari cepstral coefficients

cepstral liftering menghaluskan spectrum hasil dari main processor sehingga dapat digunakan lebih baik untuk *pattern matching.*



Gambar 2.7 Perbandingan spectrum dengan dan tanpa Cepstral Liftering

Berdasarkan gambar diatas dapat dilihat bahwa pola spektrum setelah dilakukannya *cepstral liftering* lebih halus daripada spektrum yang tidak melalui tahap *cepstral liftering*. Dalam beberapa jurnal dikatakan bahwa *cepstral liftering* dapat meningkatkan akurasi dari aplikasi pengenalan suara, baik *speaker recognation*, maupun *speech recognition.*

Berikut adalah contoh bagaimana mendapatkan 13 nilai koefisien dari MFCC [8]:

1. Membuat sebuah frame menjadi frame kecil
2. Setiap frame akan dihitung perkiraan periodogramnya dari daya spektrum.
3. Gunakan mel filterbank untuk daya spektrum, dan jumlahkan energy dari masing masing filter
4. Ambil perhitungan algoritma dari semua energy filterbanks.
5. Ambil DCT energy log filterbank.
6. Simpan koefisien DCT 2-13, dan sisanya diabaikan.

2.8. jAudio *Feature Extraction*

jAudio adalah digital signal processing yang dibangun untuk menyediakan program yang mudah digunakan untuk melakukan ekstraksi fitur pada audio [9].

Fitur audio yang di ekstrak seperti MFCC, LPC, Beat Histogram dan lain-lain. jAudio Feature Extractor bisa digunakan secara graphical atau command line. Sistem yang dibuat diintegrasikan dengan cara mengeksekusi jAudio dengan command line, yang selanjutnya hasilnya disimpan di file xml dan diambil oleh sistem sebagai input.

2.9. *Java Programming Language*

Java programming language [10] adalah bahasa pemogramman berorientasi objek dan berbasis *class* untuk keperluan umum dan dirancang cukup sederhana untuk *programmer*. *Java programming language* tergolong bahasa pemogramman tingkat tinggi dimana representasi mesin tidak tersedia melalui bahasa. *Java programming language* memiliki manajemen penyimpanan otomatis untuk menghindari maslaah keamanan dealokasi secara eksplisit. *Java programming language* mendukung pembatasan akses eksternal ke member dari

packages, classes, dan *interface.* Hal ini membantu dalam menulis program dengan skala besar.

Java programming language disusun dengan intruksi *bytecode* dan *binary format* yang didefinisikan dalam format *java virtual machine* (JVM). Program yang dibuat dengan *java programming language* berjalan di *java runtime environment* (JRE), sehingga kita tidak perlu membuat program ulang atau melakukan kompilasi ulang untuk sistem operasi yang berbeda. Sistem operasi yang didukung antara lain *linux, BSD, Mac OS X, Solaris* dan *windows.*

2.10. *Unified Model Language*

UML [11] adalah bahasa yang dimaksudkan untuk memodelkan aplikasi komputer. Salah satu tujuan dari UML adalah untuk menyediakan lingkungan *development* dengan bahasa desain yang stabil dan umum yang dapat digunakan untuk mengmbangkan dan membangun aplikasi komputer. Dengan menggunakan UML, *developer* bisa membaca dan mendistribusikan struktur system dan rencana desain aplikasi dengan lebih mudah. UML menyediakan beberapa jenis diagram yang meningkatkan kemudahan memahami aplikasi dalam proses *development.* Beberapa diagram yang digunakan pada penelitian ini antara lain :

1) *Use Case Diagram*

   Use case diagram [11] menggamnbarkan unot fungsionalitas yang disediakan oleh sistem. Tujuan utama dait *use-case diagram* adalah untuk membantu memvisualisasikan kebutuhan fungsional dari sistem, termasuk hubungan "aktor" dengan *use-case,* serta hubungan antara *use-case* yang berbeda.

2) *Activity Diagram*

   Activity diagram [11] menunjukkan aliran kontrol antara dua atau lebih objek saat memproses suatu kegiatan. *Activity diagram* dapat digunakan untuk

memodelkan tindakan objek pada *higher-level processes*, karena *activity diagram* yang "kurang teknis" jika dibandingkan dengan *sequence diagram.*

3) *Class Diagram*

Class diagram [11] menunjukkan bagaimana entitas yang berbeda berhubungan satu sama lain. *Class diagram* menampilkan struktur statis dari sistem. *Class diagram* juga menunjukkan implementasi dari *class* yang ada di sistem.

4) *Sequence Diagram*

Sequence diagram [11] menunujukkan aliran rinci untuk *use-case* tertentu atau bahkan hanya bagian dari *use-case* tertentu. *Sequence diagram* mennunjukkan hubungan antara objek yang berbeda dalam urutan, dan dapat menunujukkan secara rinci urutan pemanggilan kepada objek yang berbeda.