

# **PENERAPAN METODE MEL FREQUENCY CEPTRAL COEFFICIENT DAN LEARNING VECTOR QUANTIZATION UNTUK TEXT- DEPENDENT SPEAKER VERIFICATION**

## **TUGAS AKHIR**

Diajukan sebagai syarat untuk menyelesaikan  
Program Studi Strata-1 Departemen Teknik Informatika

Oleh :

**Sukoreno Mukti Widodo**

**1112051**



**INSTITUT  
TEKNOLOGI  
HARAPAN  
BANGSA**

*School of Telematics*

**DEPARTEMEN TEKNIK INFORMATIKA  
INSTITUT TEKNOLOGI HARAPAN BANGSA  
BANDUNG**

**2016**

# **BAB I**

## **PENDAHULUAN**

Bab ini menjelaskan mengenai latar belakang, lingkup permasalahan, tujuan dan manfaat penelitian, batasan masalah, dan sistematika penulisan.

### **1.1 Latar Belakang**

Layanan keamanan pada umumnya menggunakan kata sandi untuk membatasi dan mengontrol akses layanan tersebut. Kata sandi yang biasa digunakan sering kali berbentuk teks, sehingga setiap kali pengguna ingin melakukan akses terhadap layanan tersebut, pengguna harus memasukkan kata sandi dengan cara mengetik kata sandi tersebut ke dalam *password field* yang tersedia. Penggunaan kata sandi dengan bentuk teks dianggap masih kurang aman karena sering kali terjadi kebocoran. Maka dari itu dibutuhkanlah bentuk lain dari kata sandi, untuk meningkatkan keamanan dalam mengakses layanan atau data tertentu. Salah satunya adalah dalam bentuk suara.

Manusia dapat mengenali seseorang hanya dengan mendengarkan orang tersebut berbicara. Maka, beberapa detik ucapan sudah cukup untuk mengidentifikasi pembicara. *Speaker Recognition* adalah proses untuk mengidentifikasi pembicara secara otomatis berdasarkan karakteristik suara [KSH12]. Melalui konsep ini, *Speaker Recognition* memungkinkan untuk menggunakan karakteristik suara pembicara sebagai kata sandi untuk memverifikasi identitas pembicara dan mengakses layanan tersebut.

Ada dua jenis *Speaker Recognition*, yaitu [ZHI13]: *Text Independent Speaker Verification* (TI-SV) dan *Text Dependent Speaker Verification* (TD-SV). TI-SV adalah jenis proses verifikasi suara pembicara tanpa membatasi pengenalan dengan kata-kata tertentu. TI-SV membutuhkan pelatihan data yang banyak untuk mendapatkan akurasi yang baik. Sedangkan TD-SV adalah jenis proses verifikasi suara pembicara dengan menggunakan kata-kata yang sama, tipe ini lebih cocok

untuk digunakan untuk teknik identifikasi dan verifikasi suara pembicara karena untuk proses verifikasi pembicara bisa didapatkan hanya dengan mengekstraksi ciri suara melalui beberapa kata sebagai *sample*.

Metode-metode yang dapat digunakan untuk mengekstraksi fitur suara diantaranya adalah LPC, LPCC, dan MFCC. Berdasarkan penelitian yang dilakukan oleh Utpal Bhattacharjee pada tahun 2013, metode MFCC yang digunakan untuk pengenalan suara dengan kondisi *noise* sebesar 20dB memberikan akurasi sebesar 97.03% sedangkan metode LPCC hanya memberikan akurasi sebesar 73.76% [BHA13]. Untuk metode LPC, tidak direkomendasikan untuk pengenalan pembicara karena lebih cocok untuk komputasi linear, sedangkan suara manusia pada dasarnya adalah non-linear [DAV13].

Sedangkan metode yang digunakan untuk pengenalan pola suara adalah LVQ, selain metode LVQ, penelitian *Speaker Recognition* dan *Speech Recognition* biasanya juga menggunakan metode HMM atau GMM. Metode LVQ lebih cocok digunakan untuk TD-SV karena pengenalan pembicara jenis ini tidak memerlukan banyak pelatihan data seperti TI-SV sehingga lebih efisien dalam hal data latih dan waktu pelatihan data untuk penelitian ini [PEN15]. Sedangkan metode GMM dan HMM lebih banyak digunakan pada model stokastik, dimana lebih cocok digunakan untuk kasus TI-SV [GEE14].

Pada penelitian ini, metode yang digunakan untuk ekstraksi fitur suara adalah metode MFCC. Metode ini merupakan metode yang paling terkenal dan paling sering digunakan dalam melakukan ekstraksi fitur suara dalam penelitian *Speaker Recognition* dan *Speech Recognition*, karena mampu untuk mengekstraksi karakteristik sinyal suara secara jelas, yang relatif berbeda dari setiap sifat saluran suara pembicara dan lebih efektif digunakan untuk pengenalan suara yang mengandung *noise*. Metode LVQ digunakan sebagai pengenalan pola untuk mengidentifikasi pembicara sesuai dengan fitur suara yang telah diekstraksi menggunakan metode MFCC. Menurut penelitian yang dilakukan oleh Geeta Nijhawan pada tahun 2014, kombinasi metode MFCC dan LVQ ini dianggap sangat baik untuk kasus TI-SV dengan menghasilkan akurasi sebesar 95%, sehingga

penulis mendapat ide dari saran penelitian tersebut untuk melakukan penelitian TD-SV dengan menggunakan kombinasi metode MFCC dan LVQ.

## **1.2 Rumusan Masalah**

Berdasarkan latar belakang di atas, rumusan masalah dari latar belakang tersebut adalah sebagai berikut:

1. Apakah metode MFCC mampu mengekstraksi ciri suara hingga mendapatkan warna suara yang unik dari masing-masing pembicara ?
2. Bagaimana hasil dari identifikasi dan verifikasi suara pembicara dengan menggunakan kombinasi metode MFCC dan LVQ ?

## **1.3 Tujuan**

Tujuan dari penelitian ini adalah selain mampu untuk mengidentifikasi pembicara melalui pengenalan suara, juga untuk mengetahui performa dari kombinasi metode MFCC dan LVQ untuk identifikasi dan verifikasi suara pembicara.

## **1.4 Batasan Masalah**

Untuk mempersempit masalah yang diteliti maka disusunlah batasan masalah seperti berikut :

1. Kata yang akan digunakan untuk melakukan verifikasi dibatasi hanya berupa buka, kunci, unlock dan lock.
2. Aplikasi yang dibuat hanya menerima masukan berupa suara dan keluarannya merupakan hasil identifikasi pembicara dan kata-kata terdaftar.

## **1.5 Kontribusi Penelitian**

Membuat pengembangan aplikasi yang mampu mengidentifikasi dan memverifikasi pembicara dengan pengenalan suara serta mendapatkan

perbaikan akurasi yang lebih baik yang pada penelitian sebelumnya yang dilakukan oleh Daniel Christian T. dengan Judul “Penerapan Metode Mel-Frequency Cepstral Coefficients Dan K-Means Clustering Untuk Pengenalan Pembicara” yang sebelumnya menggunakan metode MFCC dan K-Mean Clustering dan akan dikembangkan dengan menggunakan kombinasi metode MFCC dan LVQ untuk identifikasi dan verifikasi pembicara.

## 1.6 Metodologi Penelitian

Tahap-tahap yang penulis lakukan untuk pembuatan aplikasi adalah sebagai berikut:

1. Studi Literatur

Pada tahap ini penulis mencari data dan informasi mengenai pengenalan suara pembicara, *decoding* sinyal suara, perekaman data suara, serta metode Mel Frequency Ceptral Coefficient (MFCC) dan Learning Vector Quantization (LVQ).

2. Data Sampling

Data sampling merupakan perekaman suara pembicara dengan mengucapkan kata buka, kunci, unlock dan lock.

3. Analisis Permasalahan

Penulis melakukan analisis masalah, batasan dan kebutuhan untuk melakukan penelitian.

4. Perancangan

Penulis melakukan perancangan aplikasi seperti melakukan desain *database*, dan *user interface*.

5. Implementasi

Penulis melakukan implementasi dari perancangan yang dilakukan di tahap sebelumnya dan metode untuk mengidentifikasi suara pembicara dengan menggunakan metode Mel Frequency Ceptral Coefficient (MFCC) dan Learning Vector Quantization (LVQ).

## 6. Pengujian

Penulis melakukan pengujian terhadap aplikasi yang sudah diimplementasikan oleh penulis dengan melakukan *input* data yang beragam.

## 1.7 Sistematika Penulisan

Laporan tugas akhir ini dibagi menjadi 5 bab, yaitu :

1. Bab I Pendahuluan yang berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian.
2. Bab II Landasan Teori yang berisi landasan teori dalam pembuatan tugas akhir ini.
3. Bab III Analisis dan Perancangan yang berisi analisis masalah dan pemodelan dari tugas akhir ini.
4. Bab IV Implementasi dan Pengujian yang berisi implementasi sistem dan pengujian terhadap sistem yang telah dibangun tersebut.
5. Bab V Penutup yang berisi kesimpulan dari tugas akhir ini dan saran untuk pengembangan lebih lanjut dari tugas akhir ini.

## BAB II

### LANDASAN TEORI

Pada bab ini akan di bahas mengenai review studi literatur yang akan digunakan untuk penelitian, Metode yang akan digunakan dan juga Kerangka pemikiran awal untuk metode.

#### 2.1 Tinjauan Studi

Dalam penelitian ini peneliti menggunakan referensi dari 3 jurnal yaitu

Penghua Li, Shunxing Zhang, Huizong Feng, dan Yuanyuan Li (2015), “Speaker Identification using Spectrogram and Learning Vector Quantization”, JCIS Journal of Computational Information Systems, Vol. 11, Issue 9. Dalam jurnal ini peneliti melakukan pengenalan pembicara melalui suara menggunakan metode *Spectrogram* sebagai ekstraksi fitur dan Learning Vector Quantization (LVQ).

Dalam penelitian tersebut, terdapat 3 langkah yang dilakukan oleh peneliti, yaitu *preprocessing*, *Feature extraction*, dan *Classifier*. Pada tahap *preprocessing*, dilakukan *emphasizing* dan *windowing* pada *input* suara. Setelah itu, *input* suara yang telah melalui *preprocessing* ditransformasikan menggunakan *short-time Fast Fourier Transform* untuk mendapatkan *spectrogram* dari suara tersebut. Kemudian dilanjutkan dengan *Gabor Filter* untuk meningkatkan tekstur suara dan menggunakan algoritma *Local Binary Pattern* untuk vektor fitur suaranya. Lalu pada tahap klasifikasi, vektor fitur suara tersebut di *input* ke jaringan LVQ untuk data latih dan percobaan identifikasi pembicara.

Data suara dikumpulkan dari 5 responden (3 pria dan 2 wanita) masing-masing responden mengulangi ucapan sebanyak 10 kali. Spesifikasi suara yang didapat yaitu ukuran *frame* sebesar 30ms dan panjang *overlapping* nya adalah 15ms, sedangkan frekuensi maksimumnya adalah 8 kHz.

Dari hasil penelitian yang dilakukan, dengan menggunakan metode *Spectrogram* dan Learning Vector Quantization menunjukkan akurasi yang cukup baik yaitu sebesar 90%.

Kemudian dalam penelitian yang dilakukan oleh Kshamamayee Dash, Debananda Padhi, Bhoomika Panda dan Sanghamitra Mohanty (2012). “Speaker Identification using Mel Frequency Cepstral Coefficient and BPNN”, IJARCSSE International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 4. Metode yang digunakan adalah Mel Frequency Cepstral Coefficient (MFCC) dan Back Propagation Neural Network (BPNN).

Pada penelitian tersebut langkah pertama yang dilakukan adalah *Front-end processing* yang dimana bagian ini akan mengubah sinyal suara menjadi kumpulan vektor-vektor dari fitur suara yang memiliki karakteristik masing-masing dari suara yang di *input*. Lalu selanjutnya dilakukan *Speaker Modeling*, bagian ini membagi-bagi data karakteristik suara menjadi beberapa model sebagai data pembandingan untuk identifikasi pembicara. Bagian terakhir adalah *decision logic* yaitu menentukan identitas pembicara dengan cara membandingkan vektor fitur suara yang baru dengan yang ada di *database* dan memilih model terbaik.

Hasil yang didapatkan dari penelitian adalah dari 100 suara *sampling* didapatkan akurasi sebesar 85% dengan *error rate* diantara -1 sampai 2.5. Jenis pengenalan pembicara yang digunakan dalam penelitiannya adalah *Text Dependent Speaker Verification* dengan kata-kata yang digunakan adalah “*welcome to the world of speaker identification*”.

Dalam penelitian yang dilakukan oleh Rupali S Chavan dan Ganesh S. Sable (2013). “Text Dependent Speaker Independent Isolated Word Speech Recognition Using HMM”, IJECCCE International Journal of Electronics Communication and Computer Engineering, Volume 4, Issue 4, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209. Metode yang digunakan pada penelitian tersebut adalah MFCC untuk mengekstraksi fitur suara dan HMM/GMM untuk pengenalan.

Pada penelitian tersebut, langkah pertama yang dilakukan adalah *preprocessing*, suara diambil menggunakan *microphone* spesifikasi suara yang diambil adalah suara selama 2 detik dengan kedalaman 16 bit, menggunakan *sampling rate* sebesar 11025Hz. Selanjutnya dilakukan *feature extraction* langkah-langkah yang dilakukan pada *feature extraction* adalah *pre-emphasis*, *framing*,



*windowing*, *FFT*, *Mel Filterbank*, dan *DCT*. Pelatihan dan pengenalan pada penelitian tersebut sama-sama menggunakan HMM.

Dalam pembuatan data latih, penelitian tersebut mengumpulkan 5 responden pembicara dan diminta untuk mengucapkan 4 kali untuk kata yang sama pada masing-masing responden, sehingga didapatkan 20 total sampel suara untuk 1 kata. Setiap *input* suara diambil 2 detik dengan kedalaman 16 bit, dengan *sampling rate* sebesar 11025Hz.

Hasil dari penelitian ini menunjukkan bahwa metode MFCC dan HMM yang digunakan untuk mengidentifikasi pembicara dengan kata-kata yang digunakan adalah bahasa Inggris menghasilkan performa yang baik dalam melakukan pengenalan pembicara, performa yang baik ini dihasilkan karena terdapat kemiripan antara arsitektur dari HMM data latih dan berbagai data bicara.

### 2.1.1 *State of the Art* dari rujukan penelitian

Dalam penelitian ini, terdapat pula penelitian – penelitian sebelumnya yang membahas hal yang sama. Penelitian sebelumnya adalah sebagai berikut :

Tabel 2.1 State of the Art

Peneliti	Tahun	Topik	Metode
Penghua Li, Shunxing Zhang, Huizong Feng, Yuanyuan Li	2015	Speaker Identification using Spectrogram and Learning Vector Quantization	1. Spectrogram 2. Learning Vector Quantization
Kshamamayee Dash, Debananda Padhi, Bhoomika Panda, Sanghamitra Mohanty	2012	Speaker Identification using Mel Frequency Cepstral Coefficient and BPNN	1. Mel Frequency Cepstral Coefficients

Peneliti	Tahun	Topik	Metode
			2. Back Propagated Neural Network
Rupali S Chavan, Ganesh S. Sable	2013	Text Dependent Speaker Independent Isolated Word Speech Recognition Using HMM	1. Mel Frequency Ceptral Coefficients 2. Hidden Markov Model

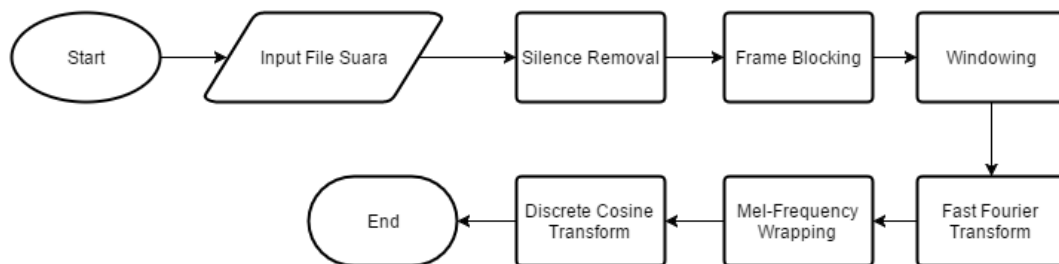
## 2.2 Tinjauan Pustaka

### 2.2.1 Mel Frequency Ceptral Coefficients (MFCC)

Ekstraksi fitur merupakan proses perhitungan dari sinyal suara untuk merepresentasikan karakteristik dari sinyal tersebut dalam bentuk data diskrit. Salah satu metode yang paling populer untuk mengekstraksi fitur adalah MFCC. MFCC merupakan metode untuk mengekstraksi fitur yang menghitung koefisien *cepstral* berdasarkan variasi frekuensi kritis pada sistem pendengaran manusia.

Beberapa keunggulan metode MFCC adalah [BHA15]:

1. Mampu menghasilkan data dengan akurasi tinggi dalam kondisi suara banyak *noise*.
2. Mampu mengekstraksi fitur suara selengkap mungkin dengan data seminimal mungkin.
3. Dapat mereplikasi sistem pendengaran suara manusia, sehingga akurasi tinggi.

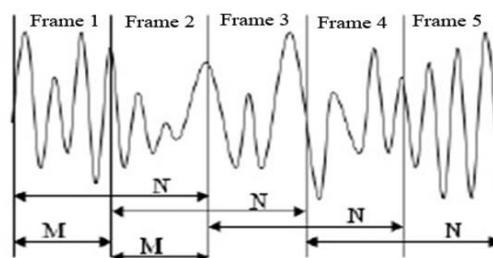


Gambar 2.1 Tahap-tahap MFCC

### 2.2.2 *Frame Blocking* [ANG11]

Pada tahap *frame blocking*, sinyal suara dibagi menjadi beberapa *frame* dengan panjang pada umumnya sebesar 20-30ms yang berisi  $N$  sampel masing-masing *frame* dipisahkan oleh  $M$  ( $M < N$ ) dimana  $M$  adalah banyaknya pergeseran antar *frame*.

*Frame* pertama berisi sampel  $N$  pertama. *Frame* kedua dimulai  $M$  sampel setelah permulaan *frame* pertama, sehingga *frame* kedua ini *overlap* terhadap *frame* pertama sebanyak  $N-M$  sampel. *Frame blocking* diperlukan karena sinyal suara mengalami perubahan dalam jangka waktu tertentu.



Gambar 2.2 Proses Frame Blocking

### 2.2.3 *Windowing*

Proses *Windowing* dilakukan untuk meminimalisir diskontinuitas yang terjadi pada sinyal, yang disebabkan oleh kebocoran spektral pada saat proses *frame blocking* dilakukan dimana sinyal yang baru, memiliki frekuensi yang berbeda dengan sinyal aslinya.

Konsep dari *windowing* adalah meruncingkan ujung sinyal menjadi nol pada bagian awal dan akhir setiap *frame*. Proses *windowing* dilakukan dengan cara mengalikan tiap *frame* dari dengan jenis *window* yang digunakan, yang dapat dituliskan dalam persamaan berikut [ANG11]:

$$y(n) = x(n)w(n), \quad 0 \leq n \leq N - 1 \quad (2.1)$$

Dimana,

$y(n)$  = sinyal hasil *windowing* sampel ke-  $n$

$x(n)$  = nilai sampel ke-  $n$  sebelum di *windowing*

$w(n)$  = nilai *window* ke-  $n$

$N$  = jumlah sampel setiap *frame*

$n$  = indeks sampel dalam suatu *frame*

Karena dalam penelitian suara banyak menggunakan *hamming window* dengan kesederhanaan rumus serta hasil dari *windowing* yang baik maka jenis *window* yang digunakan adalah *hamming window* yang dituliskan ke dalam persamaan berikut [LYO11]:

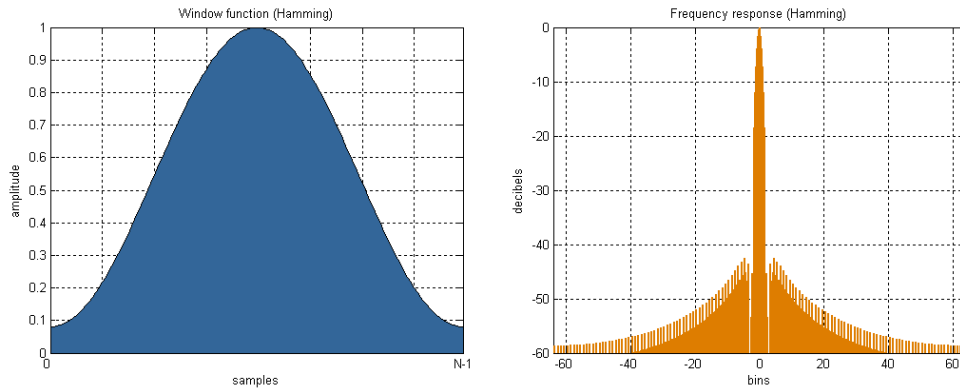
$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N - 1}\right), \quad 0 \leq n \leq N - 1 \quad (2.2)$$

Dimana,

$w(n)$  = nilai *window* ke-  $n$

$N$  = jumlah sampel setiap *frame*

$n$  = indeks sampel dalam suatu *frame*



Gambar 2.3 *Hamming Window*

Sumber: <http://kc.flexradio.com/Uploads%5CImages/>

#### 2.2.4 *Discrete Fourier Transform*

Untuk mendapatkan analisis frekuensi dari sinyal waktu diskrit  $x(n)$  maka perlu didapatkan representasi frekuensi dari sinyal yang biasanya dinyatakan dalam domain waktu. *Discrete Fourier Transform* (DFT) adalah prosedur yang digunakan dalam pemrosesan sinyal digital dan filterisasi digital. DFT memungkinkan kita untuk menganalisa, memanipulasi dan mensintesis sinyal yang tidak mungkin dilakukan pada sinyal analog [LYO11]. DFT berasal dari *Continue Fourier Transform*  $X(f)$  yang didefinisikan sebagai berikut:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft} dt \quad (2.3)$$

Dimana  $x(t)$  adalah sinyal kontinu dalam domain waktu. Persamaan tersebut digunakan untuk mentransformasikan fungsi sinyal dalam domain waktu  $x(t)$  ke dalam domain frekuensi  $X(f)$ . DFT sendiri dapat dinyatakan ke dalam persamaan matematis berikut bentuk eksponensial:

$$X(m) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nm/N} \quad (2.4)$$

Dimana  $x(n)$  adalah urutan nilai diskrit sampel dalam domain waktu dari suatu sinyal dan  $j$  adalah  $\sqrt{-1}$ . Persamaan tersebut dapat dinyatakan dalam bentuk *rectangular* dengan persamaan sebagai berikut:

$$X(m) = \sum_{n=0}^{N-1} x(n) \left[ \cos\left(\frac{2\pi nm}{N}\right) - j \sin\left(\frac{2\pi nm}{N}\right) \right] \quad (2.5)$$

Dimana,

$X(m)$  = komponen output DFT ke- $m$

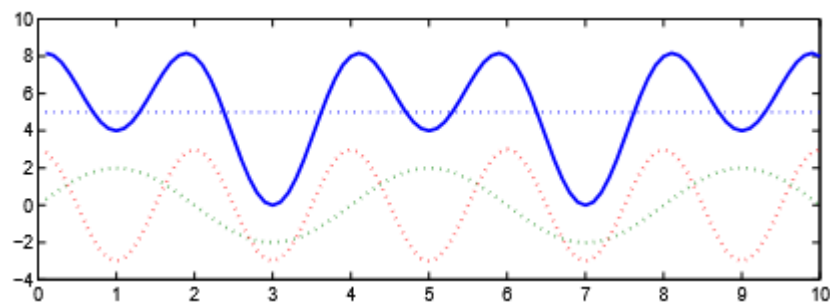
$m$  = indeks *output* DFT dalam domain frekuensi,  $m = 0, 1, 2, 3, \dots, N-1$

$x(n)$  = urutan *input* sampel,  $x(0), x(1), x(2), \dots$

$n$  = indeks input sampel dalam domain waktu,  $n = 0, 1, 2, 3 \dots, N-1$

$j = \sqrt{-1}$

$N$  = jumlah urutan input sampel dan jumlah titik frekuensi dalam output DFT



Gambar 2.4 Contoh Sinyal DFT

Nilai frekuensi dari setiap sinusoid yang berbeda tergantung kepada sampel sinyal asli ( $f_s$ ) dan jumlah sampel  $N$ . Nilai frekuensi sinusoid dapat dihitung

dengan  $f_s/N$ . Sedangkan analisis frekuensi  $X(m)$  adalah perkalian dengan frekuensi sinusoid, yang bisa dihitung dengan rumus sebagai berikut:

$$f_{analysis}(m) = \frac{mf_s}{N} \quad (2.6)$$

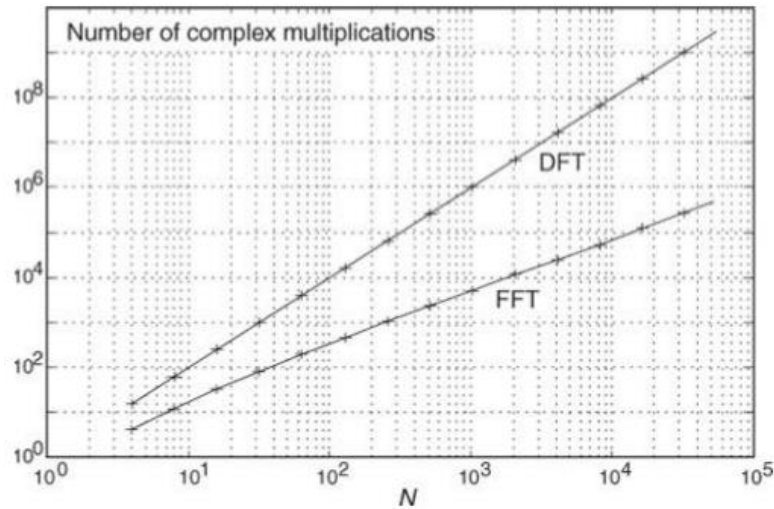
Dari persamaan tersebut kita mendapati bahwa  $X(m)$  menghasilkan nilai *magnitude* dari komponen sinyal *input* yang didapat dari hasil akar dari penjumlahan bilangan real dan bilangan imajiner yang dikuadratkan dan dapat dituliskan dalam persamaan berikut:

$$Xmag(m) = |X(m)| = \sqrt{X_{real}(m)^2 + X_{imag}(m)^2} \quad (2.7)$$

### 2.2.5 Fast Fourier Transform [LYO11]

*Fast Fourier Transform* (FFT) adalah tahap untuk mengubah setiap *frame* yang terdiri dari  $N$  sampel dari domain waktu ke dalam domain frekuensi, sama dengan DFT, yang membedakan adalah operasi DFT membutuhkan operasi  $O(N^2)$  sedangkan FFT mengurangi operasi komputasinya menjadi  $O(N \log_2(N))$ . FFT dilakukan untuk mendapatkan besaran frekuensi pada setiap *frame*. Keluaran dari proses FFT ini adalah berupa spektrum atau periodogram.

Salah satu algoritma dari FFT yang paling populer adalah radix-2. Sebagai pembandingan dengan DFT, untuk jumlah sampel  $N$  yang besar seperti contoh  $N = 512$ , dengan menggunakan perhitungan DFT membutuhkan perhitungan sebesar 114 kali lebih banyak daripada yang dibutuhkan oleh perhitungan FFT. Semakin besar jumlah sampel  $N$ , maka perhitungan semakin kompleks jika menggunakan DFT.



Gambar 2.5 Grafik perbandingan kompleksitas perhitungan DFT dan FFT

*Sumber: Understanding Digital Signal Processing*

Langkah pertama untuk menginterpretasikan FFT adalah dengan menghitung nilai frekuensi dari setiap sampel tengah dari FFT. Jika sampel waktu yang diterima FFT dalam bentuk real, maka hanya keluaran  $X(m)$  dari  $m = 2$  sampai  $m = N/2$  yang independen. Dalam kasus ini, kita hanya perlu menghitung nilai frekuensi FFT untuk  $m$  selama  $0 \leq m \leq N/2$ . Jika sampel waktu yang diterima berbentuk kompleks, kita harus menghitung semua nilai frekuensi FFT untuk  $m$  selama  $0 \leq m \leq N - 1$ . Untuk menghitung amplitudo dari FFT digunakan rumus sebagai berikut:

$$X(m) = X_{real}(m) + jX_{imag}(m) \quad (2.8)$$

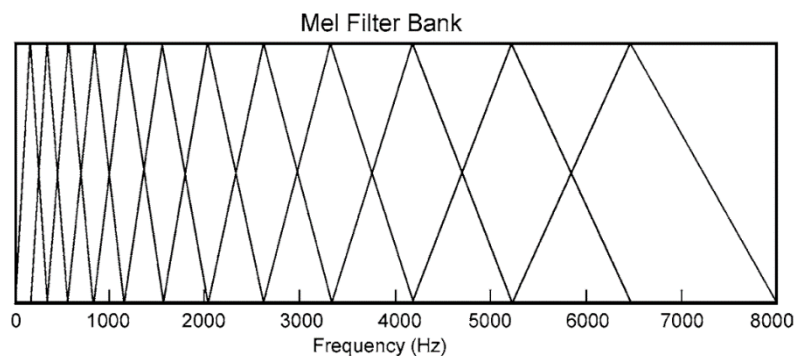
Dan untuk menghitung nilai *magnitude* dari FFT, digunakan persamaan:

$$X_{mag}(m) = |X(m)| = \sqrt{X_{real}(m)^2 + X_{imag}(m)^2} \quad (2.9)$$



### 2.2.6 Mel-Frequency Wrapping

*Mel-Frequency Wrapping* menggunakan *Filterbank* untuk menyaring sinyal suara yang telah diubah menjadi ke dalam bentuk domain frekuensi. *Filterbank* adalah sistem yang membagi *input* sinyal ke dalam kumpulan analisis sinyal, yang masing-masing sesuai dengan wilayah yang berbeda sesuai spektrum [RYA08]. Biasanya, daerah di spektrum yang diberikan oleh analisis sinyal kolektif menjangkau seluruh suara yang terdengar oleh pendengaran manusia, yaitu dari sekitar 20 Hz sampai 20 kHz. *Filterbank* yang digunakan dalam metode MFCC khususnya untuk proses *Mel-Frequency Wrapping* adalah *mel-filterbank*. *mel-filterbank* yang terdiri dari rangkaian *Triangular Window* yang saling *overlap* akan menyaring sinyal sebanyak N sampel.



Gambar 2.6 Mel Filterbank

Nilai *mel* dapat dihitung dengan persamaan berikut:

$$mel(f) = 1125 \ln\left(1 + \frac{f}{700}\right) \quad (2.10)$$

Sedangkan untuk mendapatkan nilai frekuensi dari nilai *mel* dapat dihitung dengan persamaan berikut:

$$mel^{-1}(f) = 700 \left( \exp\left(\frac{mel(f)}{1125}\right) - 1 \right) \quad (2.11)$$

Dimana,

$mel(f)$  = Nilai  $mel$ , konversi dari nilai frekuensi

$mel^{-1}(f)$  = Nilai frekuensi, konversi dari nilai  $mel$

$f$  = Nilai frekuensi

Batas atas frekuensi maksimal adalah setengah dari frekuensi sampling dari file suara. Setelah itu menentukan nilai  $mel$ , nilai  $mel$  didefinisikan sebagai  $mel(f)$ .

Nilai  $mel$  didapat dirubah ke dalam Hertz (Hz) dengan menggunakan persamaan 2.14, tetapi untuk dapat membuat *filterbank*, nilai frekuensi  $f$  harus dikonversikan ke dalam nilai sampel FFT terdekat dengan menggunakan persamaan berikut [HUA01]:

$$f[m] = \left(\frac{N}{F_s}\right) mel^{-1} \left( mel(f_i) + m \frac{mel(f_h) - mel(f_i)}{M + 1} \right) \quad (2.12)$$

Dimana,

$f[m]$  = Nilai sample FFT

$N$  = Jumlah FFT tiap frame

$F_s$  = Frekuensi sampling

$f_h$  = Frekuensi batas atas

$f_i$  = Frekuensi batas bawah

$M$  = Jumlah filter

Untuk menghitung *filterbank* dapat digunakan persamaan berikut [HUA01]:

$$H_m[k] = \begin{cases} 0 & k < f[m - 1] \\ \frac{k - f[m - 1]}{f[m] - f[m - 1]} & f[m - 1] \leq k \leq f[m] \\ \frac{f[m + 1] - k}{f[m + 1] - f[m]} & f[m] \leq k \leq f[m + 1] \\ 0 & k > f[m + 1] \end{cases} \quad (2.13)$$

Dimana,

$H_m[k]$  = Nilai filterbank

$m$  = Indeks filter

$k$  = Indeks input sampel FFT ( $k = 0, 1, \dots, N - 1$ )

$f[m]$  = Nilai sampel FFT ke- $m$

Proses pemfilteran sinyal dilakukan untuk mendapatkan *log energy* pada tiap filter dengan menggunakan persamaan [HUA01]:

$$S[m] = \ln \left[ \sum_{k=0}^{N-1} |X_a[k]|^2 H_m[k] \right], \quad 1 \leq m \leq M \quad (2.14)$$

Dimana,

$S[m]$  = Nilai log energy

$X_a[k]$  = Nilai magnitude (Hz)

$H_m[k]$  = Nilai filterbank

$N$  = Jumlah nilai FFT tiap frame

$m$  = Indeks filter

$k$  = Indeks input sampel FFT ( $k = 0, 1, \dots, N - 1$ )

### 2.2.7 Ceptrum [HUA01]

Pada tahap terakhir ini, nilai mel akan dikonversikan kembali ke dalam domain waktu, yang hasilnya disebut Mel Frequency Ceptral Coefficient. Konversi ini dilakukan dengan menggunakan Discrete Cosine Transform (DCT).  $C_0$  adalah nilai rata-rata dalam dB yang dapat digunakan untuk estimasi energi yang berasal dari filterbank. Koefisien DCT adalah nilai amplitudo dari spektrum yang dihasilkan. Perhitungan DCT dapat dilihat pada persamaan berikut:

$$c_i = \sum_{m=0}^{M-1} S[m] \cos\left(\frac{\pi n (m - 0.5)}{M}\right) \quad 0 \leq n < M \quad (2.15)$$

Dimana,

$c_i$  = Nilai koefisien C (*cepstrum*) ke-i

$S[m]$  = Nilai log energy

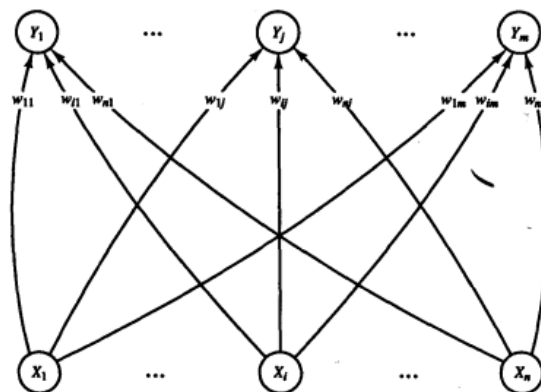
$M$  = Jumlah filter

$m$  = Indeks filter ,  $n$  = Indeks koefisien DCT

### 2.2.8 Learning Vector Quantization [FAU94]

Learning Vector Quantization (LVQ) adalah satu metode untuk melakukan klasifikasi pola yang dimana setiap keluarannya merepresentasikan sebuah kelas atau kategori tertentu. LVQ merupakan salah satu algoritma dari *supervised neural network*.

LVQ terdiri dari lapisan *input*, lapisan kompetitif, dan lapisan *output*. Lapisan input adalah masukan data, lapisan kompetitif adalah pada saat terjadinya kompetisi pada *input* untuk masuk dalam suatu kelas berdasarkan kedekatan jaraknya. Dalam lapisan kompetitif, proses pembelajaran dilakukan secara terawasi. *Input* akan bersaing untuk dapat masuk ke dalam suatu kelas.



Gambar 2.7 Arsitektur LVQ

Dalam LVQ terdapat dua jenis proses, yaitu proses pembelajaran dan proses pengujian. Proses pembelajaran dilakukan melalui beberapa iterasi sampai batas iterasi maksimal tercapai. Algoritma dalam melakukan LVQ adalah sebagai berikut:

0. Inisialisasi vektor referensi; inisialisasi *learning rate* ( $\alpha$ ), dimana  $0 < \alpha < 1$ .
1. Selama kondisi berhenti adalah *false*, lakukan langkah 2-4.
2. untuk setiap *input* pelatihan vektor  $x$ , lakukan langkah 3 – 4.
3. Cari  $J$  hingga  $\|x - w_j\|$  adalah nilai minimal.
4. Perbarui  $w_j$  dimana kondisinya adalah:

Jika  $T = C_j$  maka

$$w_j(\text{baru}) = w_j(\text{lama}) + \alpha[x - w_j(\text{lama})] \quad (2.19)$$

Jika  $T \neq C_j$  maka

$$w_j(\text{baru}) = w_j(\text{lama}) - \alpha[x - w_j(\text{lama})] \quad (2.20)$$

5. Kurangi *learning rate*.
6. Uji kondisi berhenti, seperti kondisi yang mungkin menetapkan sebuah jumlah tetap dari iterasi.

Untuk Euclidean Distance dapat dihitung dengan persamaan berikut:

$$\|x - w_j\| = \sqrt{\sum_{i=1}^n (x - w_j)^2} \quad (2.21)$$

Keterangan:

$x$  = vektor pelatihan ( $x_1, x_2, x_3, \dots x_n$ )

$T$  = Kategori atau kelas yang tepat untuk vektor pelatihan.

$w_j$  = bobot vektor untuk *output* ke  $-j$

$C_j$  = kategori atau kelas yang direpresentasikan dari unit *output* ke  $-j$

$\|x - w_j\|$  = jarak Euclidean antara vektor *input* dan bobot vektor untuk *output* ke  $-j$

$\alpha$  = *Learning Rate*,  $0 < \alpha < 1$ .

Pengurangan *Learning Rate* =  $\alpha - (LR \text{ Decrement} * \alpha)$

### 2.2.9 Sound eXchange (SoX)

SoX atau sering juga disebut Sound eXchange, *the Swiss Army knife of audio manipulation* adalah sebuah *library* lintas platform yang dapat mengkonversi berbagai format *file* audio komputer ke dalam format lain dan dapat menerapkan berbagai efek untuk *file-file* audio. SoX dapat membaca dan menulis *file* audio dalam format yang paling populer dan dapat menerapkan efek kepada keluaran *file* audio secara opsional.

Secara mendetail, SoX dapat melakukan berbagai macam manipulasi terhadap audio seperti, menggabungkan *input*, mensintesis audio, dan pada kebanyakan sistem, SoX dapat bertindak sebagai pemutar atau perekam audio [BAG13].

Pada penelitian ini, *library* ini digunakan untuk melakukan *silence removal*, yaitu menghilangkan diam pada rekaman suara yang dianggap tidak penting untuk ekstraksi fitur. Fungsi yang digunakan adalah [BAG13]:

**silence** [-l] *above-periods* [duration threshold[d|%]] [*below-periods* duration threshold[d|%]]

Keterangan:

*above-periods* : batas atas periode minimum diam.

*below-periods* : batas bawah periode minimum diam.

*duration* : batas minimum durasi suara diam (dalam detik).

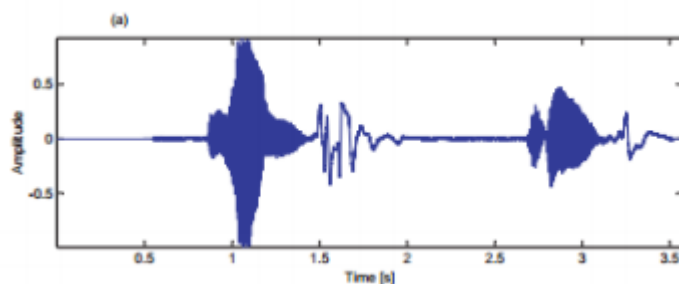
*threshold* : nilai *threshold* yang akan dihilangkan sebagai suara diam.

## 2.3 Tinjauan Objek Penelitian

Berdasarkan penelitian yang dilakukan penulis, berikut merupakan objek yang dibahas dalam penelitian ini.

### 2.3.1 Suara

Suara adalah bunyi yang dikeluarkan dari mulut manusia, seperti pada waktu berbicara, menyanyi, tertawa, menangis, dan lain sebagainya [SET16]. Pada umumnya suara merambat melalui udara. Suara tidak dapat merambat melalui ruang hampa. Manusia pada umumnya dapat memproduksi suara dengan frekuensi 70Hz hingga 10.000 Hz sedangkan suara yang dapat didengar oleh manusia adalah suara yang memiliki frekuensi dari rentang 20 sampai 20.000 Hz [MAN11].



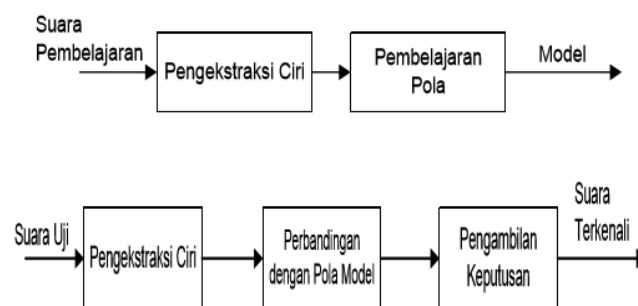
Gambar 2.8 Representasi Sinyal Suara

*Sumber: Speech Recognition using Hidden Markov Model: performance evaluation in noisy environment*

### 2.3.2 Pengenalan Suara

Pengenalan suara adalah sebuah teknik yang memungkinkan sebuah sistem komputer untuk mengenali huruf, kata atau kalimat yang diucapkan. Dalam dunia komputer, pengenalan suara biasa dikenal sebagai *Automatic Speech Recognition* atau *Computer Speech Recognition*. Teknologi ini memungkinkan suatu perangkat untuk mengenali dan memahami kata-kata yang diucapkan dengan cara digitalisasi kata dan mencocokkan sinyal digital tersebut dengan suatu pola tertentu yang tersimpan dalam suatu perangkat. Secara umum pengenalan suara terdiri dari empat tahapan, yaitu [KSH12]:

1. **Ekstraksi Fitur.** Proses mengekstrak karakter suara dari data sinyal suara yang telah diubah menjadi data digital.
2. **Pembentukan Model Suara.** Proses pembelajaran yang membentuk model referensi agar sistem dapat mengenali pembicara. proses ini membutuhkan data berupa vektor-vektor fitur yang telah diekstraksi.
3. **Pencocokan Pola.** Proses pencocokan pola dengan menerima data ekstraksi fitur sebagai *input* dan model referensi yang sebelumnya didapatkan dan memberikan skor-skor kesesuaian dengan pola.
4. **Pembuatan Keputusan.** Proses pembuatan keputusan untuk hasil akhir, pada pengenalan pembicara tahapan ini menghasilkan identifikasi dan verifikasi. Proses ini akan menerima skor-skor hasil kesesuaian dari pencocokan pola, dan akan menghasilkan identitas pembicara serta verifikasi pembicara.



Gambar 2.9 Tahapan Pengenalan Suara



### 2.3.3 Pengenalan Pembicara

Pengenalan pembicara secara umum terbagi menjadi dua bagian, yaitu identifikasi pembicara dan verifikasi pembicara. Identifikasi pembicara adalah proses untuk menentukan identitas pembicara sedangkan verifikasi secara otomatis berdasarkan karakteristik suara [KSH12]. Pengenalan pembicara memungkinkan sebuah sistem komputer untuk menerima *input* berupa suara dari seseorang lalu dapat mengenali atau mengidentifikasi orang yang menghasilkan suara tersebut.

Ada dua jenis pengenalan suara, yaitu [ZHI13]: *Text Independent Speaker Verification* (TI-SV) dan *Text Dependent Speaker Verification* (TD-SV). TD-SV merupakan pengenalan pembicara yang mengharuskan sumber pembicara untuk mengatakan kata-kata yang spesifik, seperti *password*. Sedangkan TI-SV tidak mengharuskan sumber pembicara untuk mengatakan kata-kata yang spesifik, melainkan pembicara dapat bebas mengatakan kata-kata apapun untuk melakukan pengenalan pembicara.

Pengenalan pembicara dapat dimanfaatkan untuk diaplikasikan dalam berbagai macam bidang, seperti [KSH12]:

#### 1. Untuk autentikasi

Pengenalan pembicara untuk autentikasi memungkinkan pengguna menggunakan suara mereka untuk melakukan autentikasi identitas pengguna tersebut. Cara ini dianggap jauh lebih mudah dibanding dengan cara tradisional seperti menggunakan PIN atau *password*, karena pengguna harus mengingat PIN atau *password* tersebut.

#### 2. Untuk pengawasan

Pengenalan pembicara memungkinkan pihak-pihak berwajib untuk melakukan pengawasan terhadap orang-orang tertentu yang dicurigai melalui suara mereka. Cara ini juga dapat digunakan untuk mendapatkan informasi data-data tentang orang tersebut.

#### 3. Untuk Identifikasi Forensik

Rekaman suara dapat menjadi bukti-bukti untuk menghukum penjahat dengan adanya pengenalan suara.

## **BAB III**

### **ANALISIS DAN PERANCANGAN**

Pada bab ini akan dijelaskan analisis dan perancangan sistem yang dapat mengidentifikasi pembicara. Adapun dalam membangun sistem ini, diperlukan suatu metode untuk melakukan ekstraksi fitur yang dimana fitur-fitur tersebut membawa karakter yang khusus dari suara pembicara tersebut dan untuk dapat mengenali pembicara, diperlukan *machine learning* yang akan melakukan identifikasi terhadap fitur-fitur yang telah diekstrak.

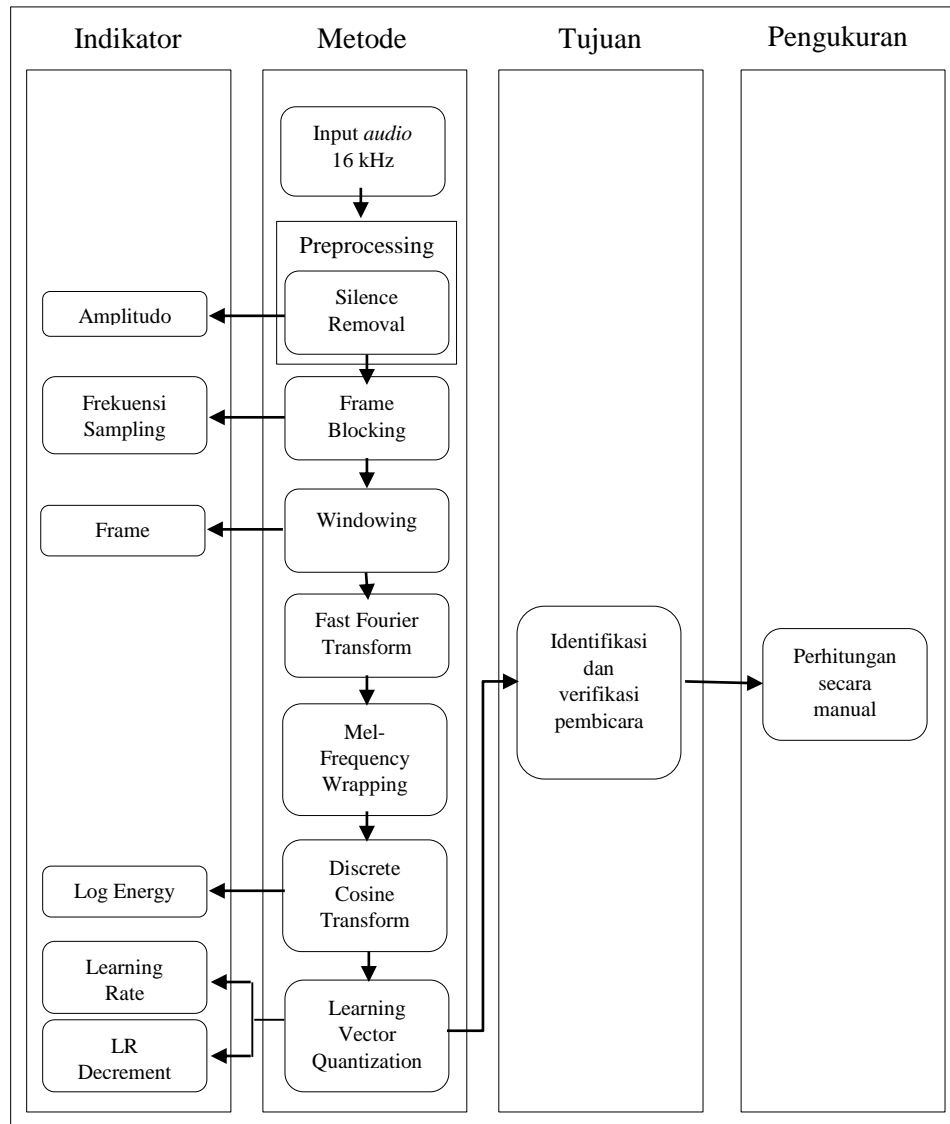
#### **3.1 Analisis Masalah**

Masalah yang akan dibahas pada penelitian ini adalah bagaimana melakukan identifikasi dan verifikasi pembicara melalui sebuah *input* suara dengan ucapan terbatas. Ucapan yang akan diterima pada tahap verifikasi hanya *lock*, *unlock*, buka dan kunci. Selain itu maka verifikasi dianggap gagal meskipun identifikasi pembicara dianggap benar. Untuk dapat melakukan pengenalan pembicara, sebuah metode yang dapat memodelkan pendengaran manusia perlu dibuat untuk diterapkan pada sistem sebuah *speaker recognition*. Sistem akan melakukan ekstraksi fitur untuk mendapatkan karakteristik dari *input* suara tersebut untuk dijadikan parameter-parameter yang akan digunakan untuk mengidentifikasi pembicara.

Dalam penelitian ini metode Mel Frequency Ceptral Coefficients berperan sebagai pengestraksi fitur dari *input* suara, metode ini digunakan karena dianggap paling baik untuk melakukan ekstraksi fitur dari *input* suara dengan berbagai kondisi, salah satunya kondisi suara dengan banyak *noise*.

Untuk mendapatkan klasifikasi hasil identifikasi dan verifikasi pembicara, maka digunakan metode Learning Vector Quantization (LVQ). LVQ digunakan untuk melakukan pengambilan keputusan untuk identifikasi pembicara karena LVQ merupakan salah satu algoritma dari *supervised neural network*. Sehingga hasil yang diperoleh dapat lebih akurat menggunakan metode ini.

### 3.2 Kerangka Pemikiran



Gambar 3.1 Kerangka Pemikiran

Kerangka pemikiran dalam pembuatan tugas akhir ini adapun sebagai berikut:

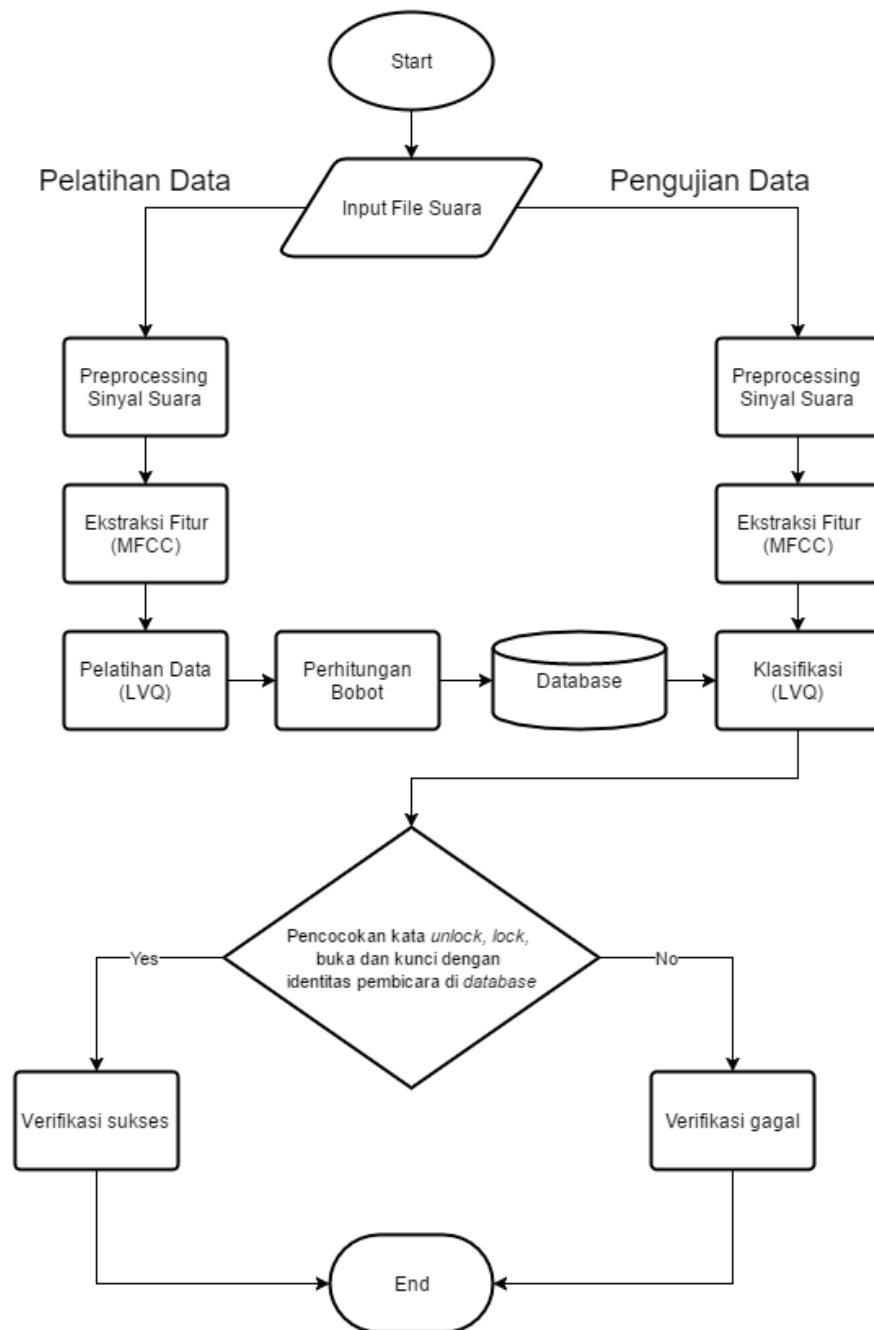
1. *File* suara hasil rekaman dengan frekuensi sampling sebesar 16 kHz dimasukan sebagai *input*.
2. Selanjutnya dilakukan *silence removal* yang amplituda nya kecil akan dihilangkan.

3. *Frame Blocking* dilakukan untuk membagi sinyal kedalam beberapa bagian.
4. *Windowing* akan dilakukan pada setiap *frame* dari hasil *frame blocking* pada taha sebelumnya, *windowing* yang akan digunakan adalah Hamming Window.
5. Selanjutnya dilakukan perhitungan Fast Fourier Transform untuk mengubah sinyal dari domain waktu ke domain frekuensi yang nantinya frekuensi tersebut akan digunakan untuk mendapatkan koefisien-koefisien *acoustic vector*.
6. Untuk mendapatkan *acoustic vector* atau nilai koefisien *cepstral mel-frequency* maka dilakukan perhitungan Discrete Cousine Transform dengan menghitung nilai *log-energy* yang didapatkan dari proses Mel-Frequency Wrapping.
7. Selanjutnya *acoustic vector* akan diproses menggunakan Learning Vector Quantization untuk mendapatkan hasil identifikasi dan verifikasi pembicara pada proses pengujian dan akan disimpan sebagai basis data pembicara pada proses pelatihan.

### 3.3 Perancangan

*Flowchart* yang dapat dilihat pada gambar 3.2 dibawah merupakan gambaran umum dari alur kerja sistem yang akan dibuat. Pertama *user* akan menginput *file* suara yang akan diproses. Lalu akan dilakukan *preprocessing* pada sinyal suara untuk menghasilkan suara yang jernih. Selanjutnya dilakukan ekstraksi fitur untuk mendapatkan karakteristik dari suara tersebut yang nantinya data tersebut akan digunakan untul pencocokan identifikasi dan verifikasi pembicara.

Pada gambar 3.2 dapat dilihat proses pelatihan data dari menginput *file* suara sampai dilakukan perhitungan MFCC untuk mendapatkan koefisien *mel-frequency ceptral* yang akan disimpan didalam database yang nantinya akan digunakan untuk pencocokan data untuk identifikasi dan verifikasi pembicara.



Gambar 3.2 Flowchart text-dependent speaker verification

### 3.3.1 Input File Suara

*Input File* Suara yang digunakan pada penelitian ini adalah data suara berupa *file* suara yang didapat dari rekaman secara langsung. Setiap *file* suara berdurasi kurang lebih 1-3 detik tergantung ucapan yang diberikan sebagai *input*. *File* suara yang digunakan pada penelitian ini memiliki frekuensi *sampling* sebesar 16kHz karena pada umumnya suara percakapan manusia adalah 300 Hz – 8000 Hz. Untuk memenuhi *Nyquist Shannon Criterion*, maka frekuensi *sampling* yang dipakai adalah 16 kHz, yang dimana minimum frekuensi *sampling* adalah 2 kali frekuensi sinyal [GEE14].

Pada penelitian ini, penulis menggunakan 5 sampel pembicara yang terdiri dari 3 pria dan 2 wanita. Suara masing-masing pembicara akan direkam saat mengucapkan kata buka, kunci, *lock*, dan *unlock*. Setiap kata akan direkam sebanyak 5 kali, sehingga total untuk masing-masing pembicara adalah 20 kali perekaman suara. Data tersebut akan digunakan sebagai data belajar untuk melakukan identifikasi dan verifikasi pembicara.

Sebagai contoh, *file* suara yang digunakan adalah rekaman suara ucapan “buka” sepanjang 2 detik dengan frekuensi *sampling* sebesar 16 kHz. Jumlah sampel yang didapat dari rekaman tersebut adalah sebanyak 15975 sampel.

Table 3.1 Nilai Sampel *Input* Suara

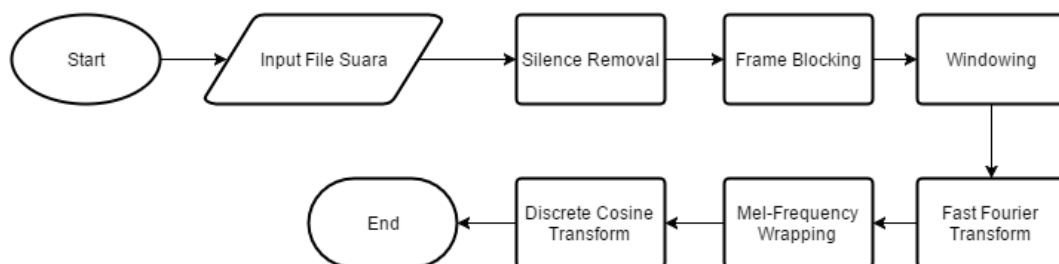
Sampel ke-	Nilai Sampel
1	0.00006
2	-0.00006
3	0.00012
4	-0.00021
5	0.00012
...	...
15970	-0.00006
15971	-0.00009
15972	-0.00006

Sampel ke-	Nilai Sampel
15973	-0.00012
15974	-0.00006
15975	-0.00006

### 3.3.2 Preprocessing dan Ekstraksi Fitur

Dalam melakukan ekstraksi fitur, pertama-tama dilakukan *preprocessing*. Tahap yang dilakukan adalah melakukan *silence removal*, yaitu menghilangkan sinyal suara yang amplitudonya sangat kecil sehingga suara tersebut tidak terdengar. Selanjutnya akan dilakukan *frame blocking* untuk membagi sinyal kedalam beberapa bagian untuk dilakukan *windowing*. Jenis *window* yang digunakan dalam penelitian ini adalah Hamming Windows.

Tahap selanjutnya adalah mengubah sinyal suara dari domain waktu ke dalam domain frekuensi dengan menggunakan Fast Fourier Transform. Untuk memodelkan pendengaran manusia, maka perlu dilakukan Mel-Frequency Wrapping dengan menerapkan *mel filterbank* untuk melakukan *filtering* pada sinyal *input* yang selanjutnya akan menghasilkan Mel-Frequency Ceptral Coefficients.



Gambar 3.3 Flowchart Preprocessing dan Ekstraksi Fitur

#### 3.3.2.1 Silence Removal

Pada tahap ini, suara diam dalam rekaman suara akan dihilangkan menggunakan *library* Sound eXchange. Nilai parameter yang digunakan pada penelitian ini didapat dari forum *library* SoX. Adapun nilai parameter yang digunakan adalah sebagai berikut [NAV09]:

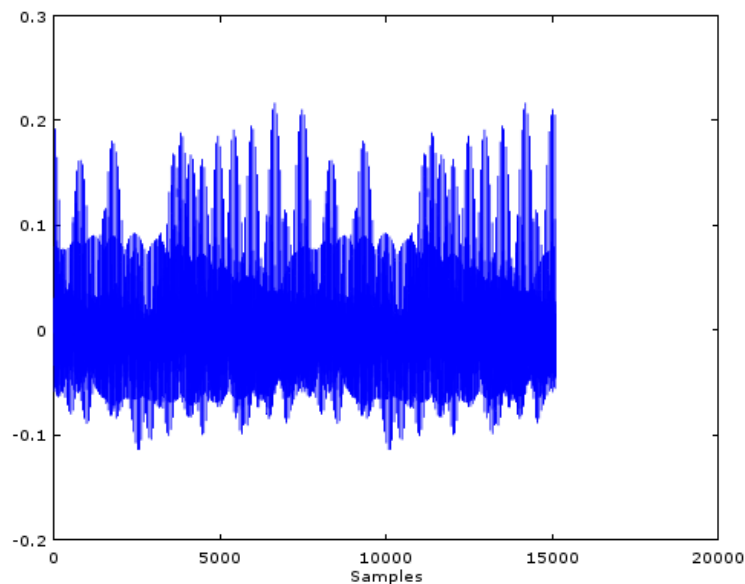
Table 3. 2 Nilai Parameter SoX Silence

<i>above-period</i>	<i>below-period</i>	<i>duration</i>	<i>threshold</i>
1	-1	0.05s	0.3%

Setelah *silence removal* dilakukan, maka jumlah sampel *file* suara tersebut akan berkurang dari 15975 menjadi 5340 sampel.

### 3.3.2.2 *Frame Blocking*

Pada tahanan *frame blocking* sinyal suara akan dibagi menjadi beberapa bagian dalam rentang waktu tertentu. Pada penelitian yang dilakukan oleh Kshamamayee Dash, penelitian tersebut menggunakan ukuran sampel per *frame* sebesar 256 sehingga penulis menggunakan ukuran sampel per *frame* sebesar 256 dan *overlapping* sebesar 1/2 dari sampel per *frame* pada penelitian ini [KSH12]. Biasanya ukuran *frame* merupakan bilangan *power of two* agar dapat digunakan pada proses FFT.



Gambar 3.4 Frame blocking



Pada penelitian ini, jumlah sampel setelah dilakukan *silence removal* adalah 7774 sampel dan *overlapping* sebesar 1/2 dari sampel per *frame*, maka untuk mengetahui jumlah *frame* yang ada, dilakukan perhitungan sebagai berikut:

$$\begin{aligned}\text{numFrame} &= \text{floor}((\text{sample-frameSize}) / \text{overlap}) + 1 \\ &= \text{floor}(7774 - 256) / 128 + 1 \\ &= 59\end{aligned}$$

Dari perhitungan yang dilakukan di atas maka dihasilkan jumlah *frame* dari *file input* suara dengan ukuran sampel per *frame* sebesar 256 dan *overlapping* sebesar 128 adalah 59 buah *frame*.

### 3.3.2.3 Windowing

Setelah dilakukan proses *frame blocking*, selanjutnya dilakukan proses *windowing*. Jenis *windowing* yang digunakan dalam penelitian ini adalah Hamming Window. Untuk mendapatkan nilai dari Hamming Window maka dilakukan perhitungan dengan menggunakan persamaan 2.2.

Table 3.3 Nilai sampel pada *frame*

Frame ke-	Sampel ke-	Nilai Sampel
1	1	-0.00021362
1	2	-0.00024414
1	3	-0.00027466
1	4	-0.00021362
1	5	0.00021362
1	6	0.00021362
1	7	0.00027466
1	8	0.00033569
1	9	-0.00021362
1	10	-0.00021362

Frame ke-	Sampel ke-	Nilai Sampel
...	...	...
1	254	-0.0024109
1	255	-0.0025024
1	256	-0.0024719

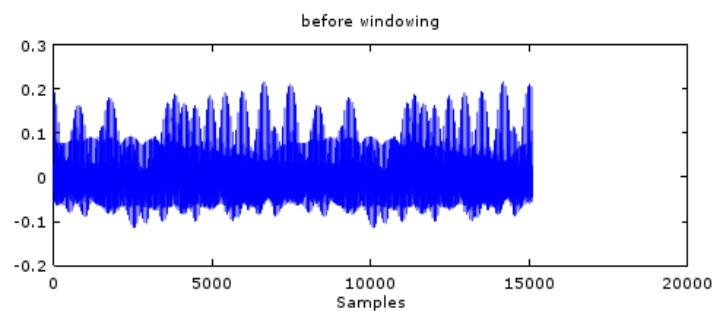
Untuk mendapatkan nilai Hamming Window, maka perhitungan akan dilakukan dengan menggunakan persamaan 2.2 seperti berikut ini:

$$\begin{aligned}
 w_0 &= 0,54 - 0,46 \cos\left(\frac{2*0*\pi}{256-1}\right) \\
 &= 0,08 \\
 y_0 &= -0.00021362 * 0,08 \\
 &= -0,000017 \\
 w_1 &= 0,54 - 0,46 \cos\left(\frac{2*1*\pi}{256-1}\right) \\
 &= 0.08014 \\
 y_1 &= -0.00024414 * 0.08014 \\
 &= -0,000019 \\
 &\dots \\
 w_{255} &= 0,54 - 0,46 \cos\left(\frac{2*255*\pi}{256-1}\right) \\
 &= 0.08 \\
 y_{256} &= -0.0024719 * 0.08 \\
 &= -0,00019775
 \end{aligned}$$

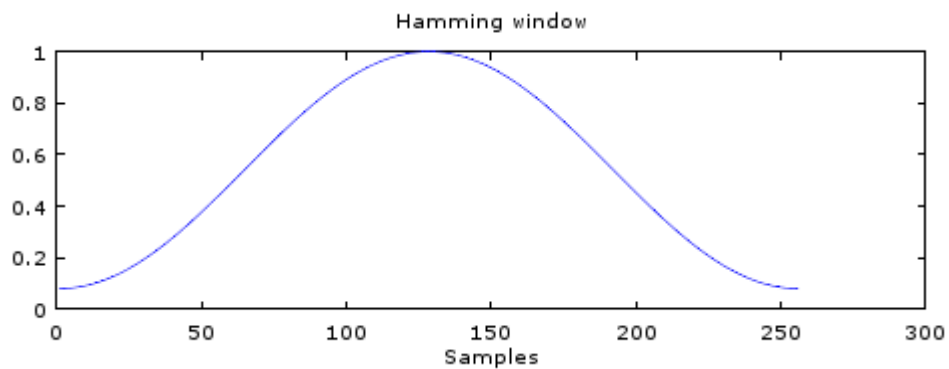
Table 3.4 Nilai setelah *windowing*

Frame ke-	Sampel ke-	Nilai setelah <i>windowing</i>
1	1	-0.000017
1	2	-0.000019
1	3	-0.000022
1	4	-0.000017

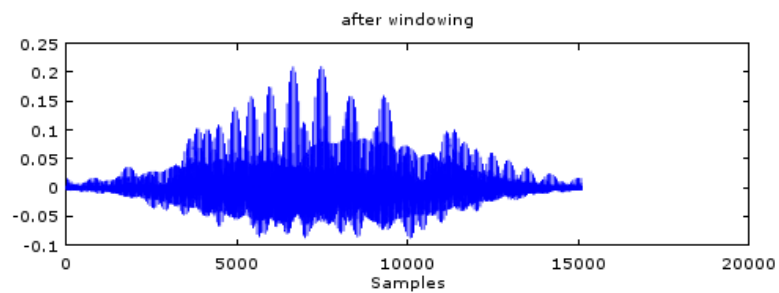
Frame ke-	Sampel ke-	Nilai setelah <i>windowing</i>
1	5	0.000017
...	...	...
1	254	-0.00019422
1	255	-0.00020054
1	256	-0.00019775



Gambar 3.5 sinyal sebelum *windowing*



Gambar 3.6 Hamming Window



Gambar 3. 7 Sinyal setelah windowing

### 3.3.2.4 Zero Padding

*Zero Padding* adalah penambahan nilai 0 pada sinyal data agar memenuhi jumlah sampel per *frame* yang telah ditentukan pada *frame* terakhir. Sebagai contoh, jika pada frame terakhir jumlah sampelnya masih kurang dari 256, maka akan dilakukan proses *zero padding* dengan menambahkan nilai 0 sampai jumlah sampel pada frame terakhir tersebut jumlah sampelnya adalah 256.

### 3.3.2.5 Fast Fourier Transform

Untuk mengubah sinyal dari domain waktu ke domain frekuensi maka dilakukan proses Fast Fourier Transform. Nilai-nilai frekuensi yang didapat dari proses ini nantinya akan digunakan dalam tahap *filtering* untuk mendapatkan koefisien-koefisien vektor.

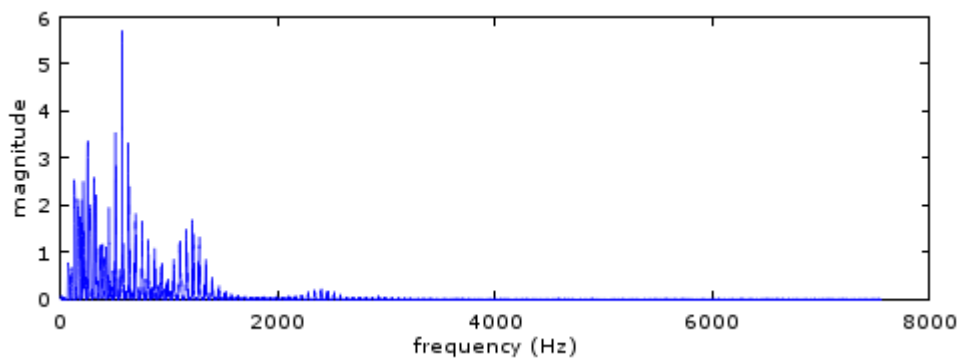
Fast Fourier Transform ini dilakukan pada setiap *frame* yang telah melalui proses *windowing*. Fast Fourier Transform dilakukan dengan membagi N buah titik pada transformasi diskrit menjadi 2, masing masing (N/2) titik transformasi. Proses memecah menjadi (N/4) dan seterusnya hingga diperoleh titik minimum.

Nilai-nilai pada tabel 3.3 adalah nilai sampel setelah melalui proses *windowing* yang dilakukan dengan Hamming Window. Nilai-nilai tersebut akan diproses dengan Fast Fourier Transform menggunakan algoritma Radix-2. Hasil setelah dilakukan Fast Fourier Transform dapat dilihat pada tabel berikut.

Table 3.5 Hasil FFT

Frame ke-	Sampel ke-	Nilai Magnitude
1	1	0.0015689
1	2	0.0036518
1	3	0.010927
1	4	0.025087
1	5	0.023391
...	...	...

Frame ke-	Sampel ke-	Nilai Magnitude
1	124	0.010754
1	125	0.023391
1	126	0.025087
1	127	0.010927
1	128	0.0036518



Gambar 3.8 Hasil FFT

### 3.3.2.6 Mel-Frequency Wrapping

Performa dari MFCC juga dipengaruhi oleh salah-satunya adalah jumlah filter, pada penelitian yang dilakukan oleh Vibha Tiwari pada tahun 2010, peneliti tersebut menggunakan jumlah *filter* sebesar 32 filter, dan menghasilkan akurasi yang lebih baik. Jumlah *filter* yang terlalu banyak atau terlalu sedikit akan menghasilkan akurasi yang tidak baik [TIW10]. Maka dari itu penulis juga akan menggunakan jumlah *filter* sebanyak 32 *filter*. Sedangkan jumlah *cepstrum* yang akan diambil sebagai nilai akhir adalah sebanyak 13 buah.

Pada tahap ini akan dilakukan *filter* pada sinyal untuk setiap *frame*. *Filter* yang digunakan pada tahap ini menggunakan *mel filterbank*. Untuk membuat *mel filterbank*, pertama-tama tentukan batas atas dan batas bawah frekuensi. Untuk batas bawah yang ditentukan adalah 300 Hz dan batas atas adalah Frekuensi *Sampling* / 2 yaitu 8000 Hz. Untuk menghitung nilai mel, digunakan persamaan 2.10.

$$\begin{aligned} mel(300) &= 1125 \ln(1 + (300/700)) \\ &= 401.26 \end{aligned}$$

$$\begin{aligned} mel(8000) &= 1125 \ln(1 + (8000/700)) \\ &= 2835 \end{aligned}$$

Jumlah *filter* yang digunakan adalah 32 buah, yang artinya *mel filterbank* memiliki *triangular window* sebanyak 32 buah, dan memiliki 34 titik. Setelah mengetahui nilai mel untuk frekuensi batas bawah dan frekuensi batas atas, nilai mel pada 34 titik antara titik awal dan titik akhir berjarak secara linear antara 401.26 dan 2835 sehingga akan didapatkan hasil seperti dibawah ini.

Table 3.6 Nilai Mel

Mel ke-	Nilai Mel
1	401.26
2	475.01
3	475.01
4	622.51
5	696.26
...	...
30	2540
31	2613.7
32	2687.5
33	2761.2
34	2835

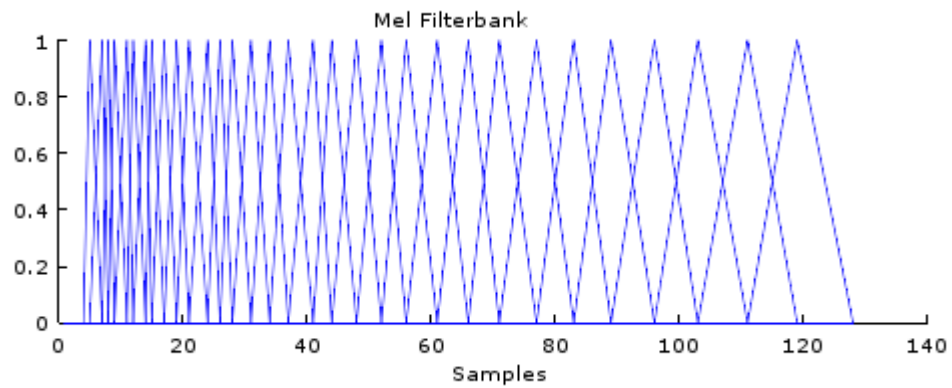
Setelah itu nilai mel pada tabel 3.5 dikonversikan ke dalam frekuensi (Hz) dengan persamaan 2.14 sehingga akan didapatkan hasil sebagai berikut:

$$\begin{aligned} mel^{-1}(1) &= 700 (\exp(401.26/1125)) - 1) \\ &= 300 \text{ Hz} \end{aligned}$$

...

$$mel^{-1}(34) = 700 (\exp(2835/1125)) - 1$$

$$= 8000 \text{ Hz}$$



Gambar 3.9 Mel Filterbank

Perhitungan di atas dilakukan kepada seluruh nilai mel yang berjumlah 34, sehingga akan didapatkan hasil konversi mel ke frekuensi sebagai berikut:

Table 3.7 Nilai Konversi mel ke frekuensi

No	Frekuensi (Hz)
1	300
2	367.75
3	440.09
4	517.34
5	599.81
...	...
31	6446.7
32	6931
33	7448
34	8000

Selanjutnya, nilai-nilai frekuensi pada tabel 3.6 di atas akan dikonversikan menjadi nilai FFT bin yang terdekat dengan menggunakan persamaan 2.15 sehingga didapatkan hasil seperti berikut:

$$\begin{aligned}
 f[0] &= (256/16000) * mel^{-1}(401.26 + 0 * (2835 - 401.26)/32 + 1) \\
 &= 0.016 * mel^{-1}(401.261) \\
 &= 0.016 * 299.37 \\
 &= 4.7899 = 4
 \end{aligned}$$

...

$$\begin{aligned}
 f[33] &= (256/16000) * mel^{-1}(401.26 + 33 * (2835 - 401.26)/32 + 1) \\
 &= 0.016 * mel^{-1}(2835) \\
 &= 0.016 * 8000 \\
 &= 128
 \end{aligned}$$

Table 3.8 Nilai FFT bin

Nilai sampel FFT bin											
m	0	1	2	3	4	...	29	30	31	32	33
FFT bin	4	5	7	8	9	...	96	103	111	119	128

Setelah didapatkan nilai FFT bin seperti pada tabel 3.7, selanjutnya akan dihitung nilai *log energy* dari setiap *filter* menggunakan persamaan 2.17, adapun hasil yang didapat adalah sebagai berikut :

Table 3.9 Nilai *log energy*

frame ke-	filter ke-	Nilai <i>log energy</i>
1	1	0.028768
1	2	0.016435
1	3	0.0077781



<i>frame ke-</i>	<i>filter ke-</i>	Nilai <i>log energy</i>
1	4	0.0056781
1	5	0.0050628
...	...	...
1	28	0.0043019
1	29	0.0082302
1	30	0.0066933
1	31	0.0090088
1	32	0.0075837

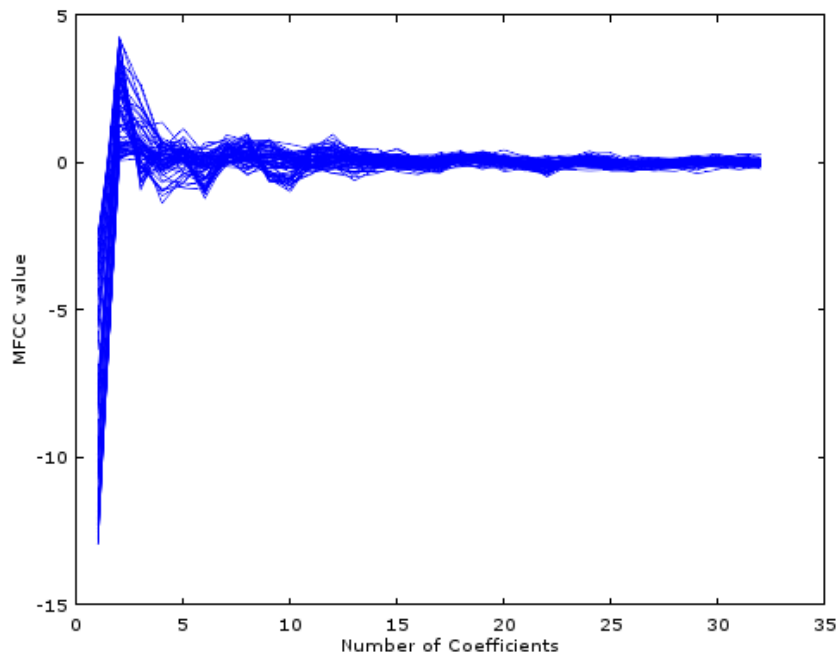
### 3.3.2.7 Discrete Cosine Transform

Proses ini adalah proses terakhir dari ekstraksi fitur. Untuk mendapatkan Mel-Frequency Cepstral Coefficients, maka dilakukan perhitungan dengan menggunakan persamaan 2.18. Pada tahap ini jumlah *ceptrum* yang diambil adalah sebanyak 13 buah pada satu *frame* [TIW10]. Adapun hasil perhitungan tersebut adalah sebagai berikut:

Table 3.10 Nilai Mel-Frequency Cepstral Coefficients pada *frame* ke-1

<i>frame ke-</i>	DCT <i>ke-</i>	Nilai DCT
1	1	-12.554095
1	2	0.257731
1	3	0.347545
1	4	0.693372
1	5	0.69337
1	6	0.398867
1	7	0.330066
1	8	0.109447
1	9	-0.022126
1	10	-0.092325

<i>frame ke-</i>	<i>DCT ke-</i>	<i>Nilai DCT</i>
1	11	0.216490
1	12	0.262256
1	13	0.059163

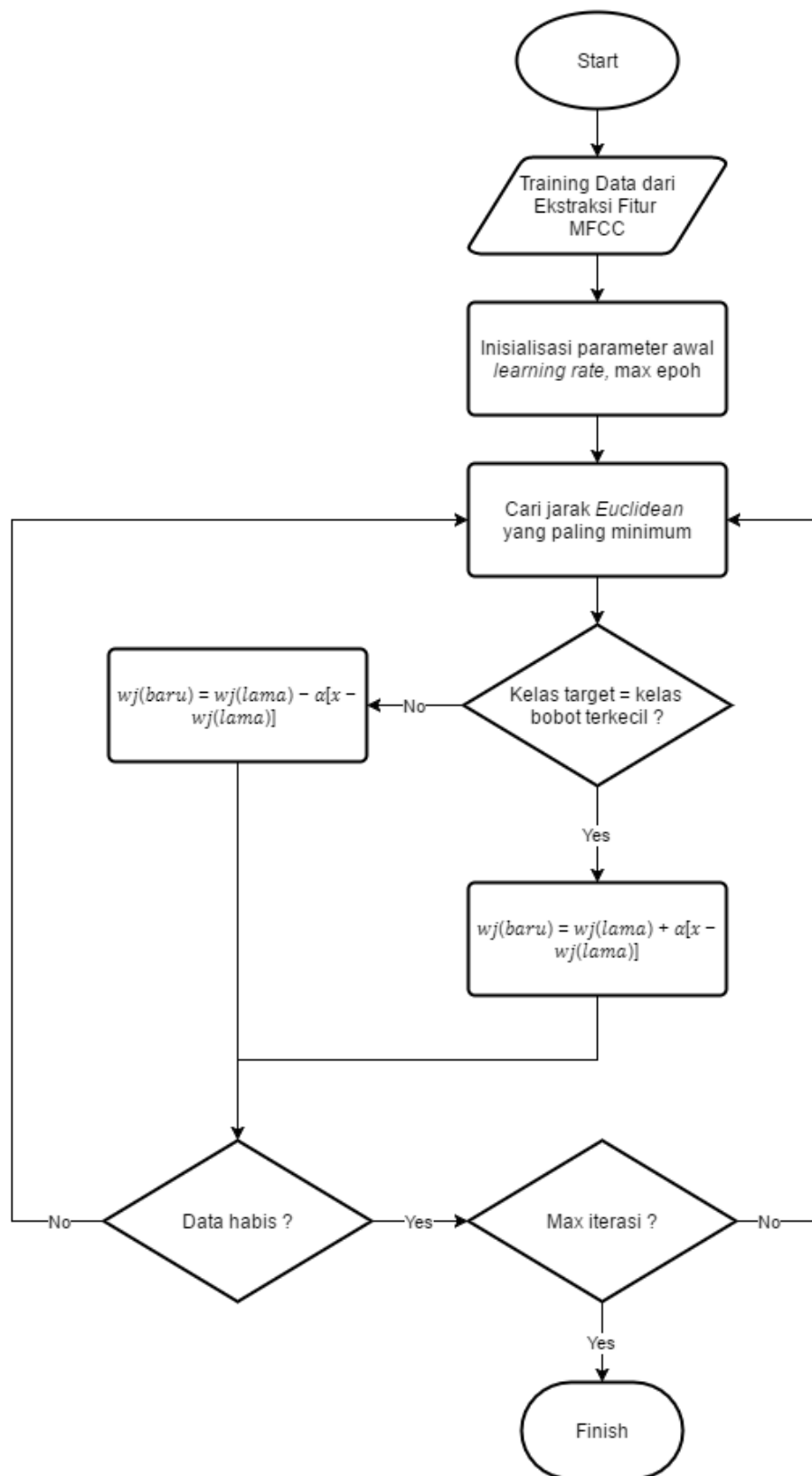


Gambar 3.10 Grafik MFCC

### 3.3.3 Learning Vector Quantization

Pada tahapan ini akan dijelaskan cara kerja sistem untuk klasifikasi menggunakan Learning Vector Quantization yang terdiri dari dua tahap yaitu pelatihan data, dan proses pengujian data. Data yang digunakan sebagai input layer didapatkan dari neuron yang dihasilkan dari proses ekstraksi sebelumnya, dan output layer yang merupakan representasi dari hasil yang akan di capai dari penelitian ini yaitu identitas pembicara dan verifikasi kata.

Pada tahap pelatihan data sistem akan melakukan proses pelatihan terhadap *machine learning* dari sistem agar dapat melakukan perbandingan terhadap input.



Gambar 3.11 Flowchart LVQ

Sedangkan pada tahap pengujian, sistem akan dilakukan perbandingan antara data input terhadap *database* untuk dicari data yang paling cocok dengan data input.

Jumlah *neuron* masukan LVQ pada penelitian adalah 13 buah yang didapat dari hasil ekstraksi Mel-Frequency Ceptral Coefficients yaitu 13 koefisien dan *neuron* pada keluaran memiliki 4 *neuron* yang merepresentasikan jumlah kelas yaitu kelas pembicara-Buka, pembicara-Kunci, pembicara-Unlock, pembicara-Lock. Berdasarkan rujukan penelitian sebelumnya, ditetapkan learning rate ( $\alpha$ ) sebesar 0.05 dengan pengurangan setiap kali iterasi  $0.977 \times \alpha$ , maksimum iterasi adalah 1000 [ANG11]. Pada tabel 3.10 diketahui vektor masukan yang didapat dari hasil ekstraksi fitur *input* suara dari 4 kata (Buka, Kunci, Lock, Unlock) yang dari setiap kata di ambil perekaman sebanyak 5 kali. Masing-masing hasil suara, memiliki jumlah *input* yang berbeda-beda, tergantung dari jumlah fitur yang diperoleh dari hasil ekstraksi fitur.

Table 3.11 *Input Vector*

Input	X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	...	X <sub>13</sub>	Output
1	-12.554	0.25773	0.34754	...	0.05916	Reno-Buka
2	-10.45	0.83613	0.94603	...	0.09659	Reno-Buka
...	...	...	...	...	...	...
59	-12.007	-0.28962	-0.11881	...	0.23359	Reno-Buka
60	-12.06	-0.00362	0.27599	...	0.49975	Reno-Kunci
61	-12.71	0.21683	0.58889	...	0.32464	Reno-Kunci
...	...	...		....	...	...
150	-12.432	0.37681	0.70664	...	0.015075	Reno-Kunci
151	-12.235	0.41048	0.73952	...	0.07273	Reno-Lock
152	-12.128	1.5358	1.4412	...	-0.0224	Reno-Lock
...	...	...		...	...	...
219	-12.161	0.27789	0.20759	...	-0.11699	Reno-Lock
220	-12.8	0.24217	0.60348	...	0.10297	Reno-Unlock

Input	X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	...	X <sub>13</sub>	Output
221	-12.393	0.3369	0.35531	...	0.075556	Reno-Unlock
...	...	...		...	...	...
297	-12.273	0.19314	0.40462	...	0.34727	Reno-Unlock

Dari 297 vektor masukan di atas, dipilih satu vektor masukan dari setiap kelas yang akan diinisialisasikan sebagai inisialisasi vektor referensi untuk masing-masing kelas. Pada Tabel 3.11 terdapat hasil dari inisialisasi vektor referensi untuk setiap kelas. Sedangkan sisanya sebanyak 281 vektor masukan dijadikan sebagai data latih.

Table 3.12 Inisialisasi vektor referensi

No	Vektor	Kelas
1	[-12.554, 0.25773, 0.34754, ..., 0.05916]	Reno-Buka
2	[-12.060, -0.00362, 0.27599, ..., 0.49975]	Reno-Kunci
3	[-12.235, 0.41048, 0.73952, ..., 0.07273]	Reno-Lock
4	[-12.8, 0.24217, 0.60348, ..., 0.10297]	Reno-Unlock

Proses LVQ dimulai dengan mencari jarak *Euclidean* antara vektor masukan (X) dan semua bobot neuron (W<sub>m</sub>) pada lapisan kompetitif. Neuron dengan jarak yang paling kecil akan memenangkan kompetisi. Jarak *Euclidean* bisa didapatkan dengan menggunakan perhitungan dari persamaan 2.21.

Data latih pertama iterasi pertama:

$$\begin{aligned}
 \text{a. } ||X - W_1|| &= \sqrt{((-10.45) - (-12.554))^2 + \dots + (0.09659 - 0.05916)^2} \\
 &= 2.4886
 \end{aligned}$$

$$\begin{aligned}
 \text{b. } ||X - W_2|| &= \sqrt{((-10.45) - (-12.06))^2 + \dots + (0.09659 - 0.49975)^2} \\
 &= 2.4228
 \end{aligned}$$

$$\begin{aligned} \text{c. } ||X - W_2|| &= \sqrt{((-10.45) - (-12.235))^2 + \dots + (0.09659 - 0.07273)^2} \\ &= 2.1585 \end{aligned}$$

$$\begin{aligned} \text{d. } ||X - W_2|| &= \sqrt{((-10.45) - (-12.8))^2 + \dots + (0.09659 - 0.10297)^2} \\ &= 2.6255 \end{aligned}$$

Hasil dari penghitungan jarak *Euclidean* antara data latih dengan vektor referensi didapatkan jarak terkecil yaitu bobot ketiga. Dikarenakan kelas terkecil yang didapat berbeda dengan kelas target data latih, maka operator yang digunakan untuk memperbaharui bobot pemenang menggunakan operator (-). Bobot keempat yang baru, diantaranya:

$$\begin{aligned} W_{31} &= W_{31} - \alpha * (X_{11} - W_{31}) \\ &= -12.235 - 0.05 (-10.45 - (-12.235)) \\ &= -12.324 \end{aligned}$$

...

$$\begin{aligned} W_{3...} &= W_{3...} - \alpha * (X_{1...} - W_{3...}) \\ &= 0.49975 - 0.05 (0.09659 - 0.49975) \\ &= 0.51991 \end{aligned}$$

Sehingga diperoleh bobot ketiga yang baru yaitu:

$$W_3 = (-12.485, 0.38919, 0.72919, \dots, 0.071537)$$

Selanjutnya dilakukan perhitungan yang sama untuk data latih kedua hingga data latih ke 297 dengan menggunakan bobot yang telah diperbaharui sebelumnya. Setelah semua data telah dilatih seluruhnya untuk iterasi pertama, maka dilakukan pengurangan *learning rate* ( $\alpha$ ) untuk setiap iterasi dengan cara :

$$\alpha = \alpha - (0.977 * \alpha)$$

Proses penghentian pembelajaran dilakukan jika telah mencapai iterasi maksimum yang telah ditentukan sebelumnya atau data latih sudah selesai diproses semua. Jika proses pembelajaran telah selesai, maka nilai akhir bobot akan disimpan di dalam *database* dan akan digunakan nanti pada proses pengujian. Sehingga didapatkan bobot akhir adalah sebagai berikut:

Table 3.13 Bobot akhir setelah proses pelatihan data

No	vektor	Kelas
1	[-20.087, 0.41237, 0.55607, ..., 0.09466]	Reno-Buka
2	[-19.296, -0.0058071, 0.44158, ..., 0.7996]	Reno-Kunci
3	[-19.576, 0.65676, 1.1832, ..., 0.11637]	Reno-Lock
4	[-20.48, 0.38747, 0.96556, ..., 0.16475]	Reno-Unlock

Jika disimulasikan *input* data baru dan didapatkan vektor ekstraksi fiturnya adalah sebagai berikut:

Table 3.14 Input vektor data uji

Input	X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	...	X <sub>13</sub>
1	-12.061	0.71374	0.59465	...	0.07847
2	-11.872	0.55795	0.58062	...	0.02105
...	...	...		...	...
87	-12.029	0.4011	0.54767	...	0.14775
88	-12.231	0.38182	0.52007	...	-0.00219
89	-12.429	0.20994	0.36829	...	0.17308
90	-12.432	0.37681	0.70664	...	0.01507

Maka untuk mendapatkan hasil kelas dari *input* yang dimaksud adalah dengan menghitung kembali bobot dengan jarak terpendek dari bobot akhir pada tabel 3.12.

$$\begin{aligned} \text{a. } ||X - W_1|| &= \sqrt{((-12.061) - (-20.087))^2 + \dots + (0.07847 - 0.09466)^2} \\ &= 8.2155 \end{aligned}$$

$$\begin{aligned} \text{b. } ||X - W_2|| &= \sqrt{((-12.061) - (-19.296))^2 + \dots + (0.07847 - 0.7996)^2} \\ &= 7.4205 \end{aligned}$$

$$\begin{aligned} \text{c. } ||X - W_3|| &= \sqrt{((-12.061) - (-19.576))^2 + \dots + (0.07847 - 0.11637)^2} \\ &= 7.6278 \end{aligned}$$

$$\begin{aligned} \text{d. } ||X - W_4|| &= \sqrt{((-12.061) - (-20.48))^2 + \dots + (0.07847 - 0.16475)^2} \\ &= 8.5672 \end{aligned}$$

Dari hasil perhitungan jarak bobot di atas, didapatkan bahwa bobot kedua merupakan jarak terkecil diantara bobot-bobot lainnya, sehingga vektor *input* pertama tersebut dapat di masukan ke dalam kelas bobot pertama, yaitu Reno-Kunci. Langkah tersebut diulangi sampai seluruh data *input* selesai diproses, sehingga akan didapatkan banyak kelas sejumlah data *input*. Untuk mendapatkan hasil akhir data *input* tersebut masuk ke dalam kelas yang mana, maka dilakukan proses *voting*, yaitu memilih hasil kelas yang terbanyak sebagai kelas akhir dari data *input* tersebut.

Table 3.15 Jumlah hasil kelas *output*

	Reno-Buka	Reno-Kunci	Reno-Lock	Reno-Unlock
<b>Jumlah hasil</b>	0	76	11	0

Jika dilihat dari hasil pada tabel 3.14, maka data *input* tersebut paling banyak masuk ke kelas Reno-Kunci, sehingga dapat disimpulkan kelas akhir dari data *input* adalah Reno-Kunci.



## **BAB IV**

### **IMPLEMENTASI DAN PENGUJIAN**

Pada bab ini dibahas mengenai analisis kebutuhan dari sistem yang akan dibangun dan bagaimana perancangan dari sistem yang akan dibangun.

#### **4.1 Lingkungan Pengembangan**

Pada bagian ini akan dijelaskan mengenai spesifikasi perangkat keras yang digunakan dan perangkat lunak yang dipakai dalam membangun dan mengembangkan sistem untuk pengenalan pembicara menggunakan MFCC dan LVQ.

##### **4.1.1 Lingkungan Perangkat Keras**

Spesifikasi perangkat keras yang digunakan dalam pengembangan aplikasi ini adalah sebagai berikut:

1. *Processor Intel® Core™ i5-4200M @ 2.50 GHz (4 CPUs)*
2. *Memory 8 GB*
3. *Hard Disk 500 GB*

##### **4.1.2 Lingkungan Perangkat Lunak**

Pengembangan aplikasi ini membutuhkan beberapa perangkat lunak, diantaranya adalah sebagai berikut:

1. Sistem Operasi Linux Ubuntu 14.04.04 LTS
2. PyCharm Community Edition
3. Python 2.7.10
4. PyQt 4.8
5. Audacity 2.10
6. SQLite 3.0

## 4.2 Implementasi Aplikasi

Pada bagian ini akan dijelaskan mengenai implementasi aplikasi ekstraksi fitur pada suara, metode pemrosesan, alur pemrosesan dan cara penggunaan aplikasi.

Tabel 4.1 Class FileReader

<b>Attribute:</b> signal fs codec				
Method	Parameter		Return	Deskripsi
	Type	Attribute		
read_audio	File	audio	list	<i>Function</i> untuk membaca sinyal dalam file audio.
add_temp	File	file	String	<i>Function</i> untuk membuat <i>temporary</i> file hasil dari silence removal.

Tabel 4.2 Class AudioPlayer

<b>Attribute:</b> -				
Method	Parameter		Return	Deskripsi
	Type	Attribute		
__init__	ui_element	volume_slider seek_slider lcd_number	None	<i>Function</i> Constructor kelas AudioPlayer.
init_audio_player	-	None	None	<i>Function</i> untuk inisialisasi awal parameter-parameter pemutar audio.

Method	Parameter		Return	Deskripsi
	Type	Attribute		
set_audio_source	File	audio_file	None	<i>Function</i> untuk menetapkan sumber <i>file</i> audio.
tick	int	time	None	<i>Function</i> untuk sinkronisasi waktu pada tampilan LCD Number.
state_change	PyQt4.Phonon.state	newState oldState	None	<i>Function</i> untuk menangkap <i>event state change</i> pada pemutar <i>audio</i> .

Tabel 4.3 class MFCC

<b>Attribute:</b> above_period = 1 below_period = -1 duration = 0.05 threshold = 0.3% frame_size = 256 overlap = 128 num_filter = 32 freq_low = 300				
Method	Parameter		Return	Deskripsi
	Type	Attribute		
freq_high	float	fs	float	<i>Function</i> untuk menghitung batas atas frekuensi dari <i>frekuensi sampling</i> .
remove_silence	File	audio	list	<i>Function</i> untuk meng-hilangkan <i>silence</i> pada <i>audio</i> .
frame_blocking	numpy.ndarray	audio_signal	list	<i>Function</i> untuk me-lakukan proses <i>frame blocking</i> .

Method	Parameter		Return	Deskripsi
	Type	Attribute		
hamm_window	numpy.ndarray	framed_signal	numpy.ndarray	<i>Function</i> untuk melakukan proses <i>windowing</i> dengan <i>hamming window</i> .
calc_fft	numpy.ndarray	signal	numpy.ndarray	<i>Function</i> untuk melakukan proses FFT.
DFT	numpy.ndarray	signal	numpy.ndarray	<i>Function</i> untuk menghitung DFT menggunakan persamaan 2.4
FFT	numpy.ndarray	signal	numpy.ndarray	<i>Function</i> untuk menghitung FFT.
mel	int	fs	numpy.ndarray	<i>Function</i> untuk menghitung dan menghasilkan nilai mel menggunakan persamaan 2.10
melinhz	numpy.ndarray	mel	numpy.ndarray	<i>Function</i> untuk menghitung dan menghasilkan nilai mel dalam frekuensi menggunakan persamaan 2.11
calc_fft_bin	numpy.ndarray int	melinhz fs	numpy.ndarray	<i>Function</i> untuk menghitung nilai fft bin menggunakan persamaan 2.12

Method	Parameter		Return	Deskripsi
	Type	Attribute		
fbank	numpy.ndarray	fft	tuple	<i>Function</i> untuk membuat <i>filterbank</i> dan meng-hitung nilai <i>log energy</i> menggunakan persamaan 2.13 dan 2.14
features	numpy.ndarray	energy	numpy.ndarray	<i>Function</i> untuk menghitung nilai koefisien MFCC.
dct	numpy.ndarray	energy	numpy.ndrray	<i>Function</i> untuk menghitung DCT menggunakan persamaan 2.15

Tabel 4.4 Class LVQ

<b>Attribute:</b>				
-				
Method	Parameter		Return	Deskripsi
	Type	Attribute		
init_ref_vector	-	-	numpy.object	<i>Function</i> untuk mengambil vektor referensi awal dari data input.
init_data_set	numpy.object	ref_vectors	numpy.object	<i>Function</i> untuk mengambil vektor data <i>input</i> selain dari vektor referensi awal.
eucl	numpy.ndarray numpy.ndarray	x y	float	<i>Function</i> untuk menghitung jarak Euclidean.

Method	Parameter		Return	Deskripsi
	Type	Attribute		
start_training	numpy.object numpy.object int float float	ref_vectors data_set max_epoh alpha alpha_decay	numpy.ndarray	<i>Function</i> untuk melakukan pelatihan data LVQ.
test_data	numpy.ndarray	features	String	<i>Function</i> untuk melakukan pengujian data <i>feature</i> untuk mendapatkan kelas akhir klasifikasinya.

Tabel 4.5 class DatabaseConnector

<b>Attribute:</b> conn				
Method	Parameter		Return	Deskripsi
	Type	Attribute		
__init__	String	database_name	-	<i>Function</i> Constructor kelas DatabaseConnector
insert_features	list	arg	int	<i>Function</i> untuk meng- <i>input</i> <i>features</i> dan kelas <i>output</i> ke dalam <i>database</i> .
insert_weight	list	arg	int	<i>Function</i> untuk menginput bobot akhir setelah pelatihan data ke dalam <i>database</i> .
whereClause	dict	params	String	<i>Function</i> untuk membuat <i>query</i> WHERE CLAUSE

Method	Parameter		Return	Deskripsi
	Type	Attribute		
adapt_array	numpy.ndarray	arr	sql.Binary	<i>Function</i> untuk mengkonversi numpy array ke dalam sql Binary
convert_array	String	text	numpy.ndarray	<i>Function</i> untuk meng-konversi sql Binary String ke dalam numpy array

#### 4.2.1 Implementasi *Silence Removal*

Pada proses *silence removal* ini digunakan *library* SoX untuk menghilangkan suara diam pada *audio*, ada beberapa parameter yang digunakan untuk menghilangkan suara diam pada *audio* yaitu : `above_period = 1`, `below_period = -1`, `duration = 0.05s`, dan `threshold = 0.3%`. tahap – tahap dalam melakukan *silence removal* adalah sebagai berikut:

1. Panggil *command* sox *silence* dengan parameter-parameter masukan sesuai diatas. Hasil keluaran nya adalah *file audio* sementara yang sudah di *silence*.
2. Ambil nilai sinyal dan frekuensi sampling dari *file audio* yang telah di *silence removal* dan simpan nilai nya kedalam *dictionary of array* yang berisi sinyal dan frekuensi sampling.

```
call(['sox', audio, FileReader.add_temp(audio), 'silence',
      self.above_period,
      self.duration,
      self.threshold,
      self.below_period,
      self.duration,
      self.threshold])
```

#### 4.2.1 Implementasi *frame blocking*

Pada proses ini, sinyal *audio* akan dibagi ke dalam beberapa *frame* tergantung pada jumlah sinyal pada *frame*, *overlap*, dan jumlah sampel sinyal. Tahap-tahap dalam proses *frame blocking* adalah sebagai berikut:

1. Menghitung jumlah *frame* dengan membagi jumlah sampel sinyal *audio* dikurangkan dengan ukuran *frame* dibagi dengan setengah dari jumlah sampel sinyal pada satu *frame*.
2. Inisialisasi `numpy.ndarray` dengan dimensi jumlah *frame* x jumlah sampel dalam satu *frame*.
3. Masukkan nilai-nilai sampel ke dalam *frame* sejumlah ukuran *frame* dengan melakukan iterasi. Pada setiap iterasi nilai sampel yang dimasukan dimulai dari setengah dari jumlah sinyal pada satu *frame*.
4. Hasil yang dikembalikan berupa tuple yang berisi jumlah *frame* dan sinyal setelah dilakukan *frame blocking*.

#### 4.2.3 Implementasi *windowing*

Pada proses *windowing*, nilai sampel tiap *frame* akan dikalikan dengan nilai-nilai window Hamming. Tahap-tahap implementasi proses *windowing* adalah sebagai berikut:

1. Hitung nilai Hamming Windows dengan menggunakan persamaan 2.2
2. Inisialisasi `numpy.ndarray` untuk menyimpan nilai *windowing* sejumlah banyak *frame* dikali dengan jumlah sampel tiap *frame* lalu isi dengan nilai 0
3. Lakukan *looping* untuk setiap *frame*, lalu kalikan nilai sampel dalam *frame* tersebut dengan nilai Hamming Windows sesuai indeksnya dan simpan dalam variable dengan tipe data `numpy.ndarray`.
4. Hasil yang dikembalikan berupa `numpy.ndarray` berisi sinyal setelah dilakukan proses *windowing*.



#### 4.2.4 Implementasi Fast Fourier Transform

Untuk mengubah sinyal ke dalam domain waktu, maka dilakukan perhitungan Fast Fourier Transform. Tahap-tahap implementasi proses perhitungan Fast Fourier Transform adalah sebagai berikut:

1. Inisialisasi `numpy.ndarray` untuk menyimpan sinyal `fft` sejumlah sinyal *windowed* lalu isi dengan nilai 0.
2. Lakukan *looping* untuk setiap *frame* pada sinyal *windowed* lalu masukan nilai perhitungan FFT setiap sinyal pada *frame*, dan masukan kedalam `numpy.ndarray` penyimpanan sinyal `fft`.
3. Untuk perhitungan FFT, dilakukan dengan cara rekursif dengan membagi sinyal yang ganjil dan yang genap, untuk dilakukan perhitungan DFT dengan menggunakan persamaan 2.4
4. Untuk setiap rekursif sinyal ganjil dan genap, akan digabungkan lalu di kalikan dengan persamaan DFT.
5. Nilai-nilai magnitude yang dihasilkan akan disimpan didalam `numpy.ndarray` sinyal `fft` untuk setiap *frame*.

#### 4.2.5 Implementasi Mel Frequency Wrapping

Pada proses Mel Frequency Wrapping, akan dibuat sebuah *filterbank* dengan jumlah filter adalah 32 filter, tujuan dari proses ini adalah untuk menghitung nilai *log energy* dari setiap *frame* sinyal. Tahap-tahap implementasi proses perhitungan Mel Frequency Wrapping adalah sebagai berikut:

1. Hitung nilai mel dengan menggunakan persamaan 2.10
2. Ubah nilai mel tersebut ke dalam nilai frekuensi dengan menggunakan persamaan 2.11
3. Hitung nilai-nilai `fft` bin dengan menggunakan persamaan 2.12
4. Inisialisasi sebuah `numpy.ndarray` untuk menyimpan nilai-nilai *filterbank*.

5. Lakukan *looping* sejumlah *filter* dan jumlah sampel dalam *frame* dibagi 2, untuk membentuk *filterbank* dengan menggunakan kondisi pada persamaan 2.13
6. Hitung nilai *log energy* dengan mengalikan nilai-nilai magnitude yang dihasilkan dari proses FFT dengan nilai-nilai *filterbank*.
7. Hasil yang dikembalikan melalui proses ini adalah sebuah tuple yang berisi `numpy.ndarray log energy` dan `numpy.ndarray filterbank`.

#### 4.2.6 Implementasi Discrete Cosine Transform

Pada proses ini, fitur dari suara akan didapatkan dengan cara menghitung nilai DCT dari nilai *log energy* yang didapatkan dari proses Mel Frequency Wrapping. Tahap-tahap implementasi proses Discrete Cosine Transform adalah sebagai berikut :

1. Hitung nilai DCT dari setiap nilai *log energy* menggunakan persamaan 2.15
2. Simpan nilai DCT dari indeks ke-1 sampai indeks ke-13 yang dimana merupakan fitur dari suara yang telah diekstraksi.
3. Hasil yang dikembalikan melalui proses ini adalah sebuah `numpy.ndarray` yang berisi nilai DCT.

#### 4.2.7 Implementasi Insert Features

Setelah dilakukan ekstraksi fitur dan didapatkan sebuah fitur dari *file audio* tersebut, maka proses selanjutnya adalah memasukan fitur tersebut ke dalam *database*. Agar proses *Insert Features* dapat dilakukan dengan cepat, maka penulis mengoptimasi proses *Insert Features* ini menggunakan *Thread*. Tahap-tahap implementasi proses *Insert Features* adalah sebagai berikut:

1. Inisialisasi sebuah kelas *Thread* dengan menggunakan `QThread`.
2. Lakukan *looping* untuk semua *file audio* yang akan diekstraksi.
3. Untuk setiap *file*, setelah dilakukan ekstraksi, jalankan sebuah *Thread* untuk memasukan fitur hasil ekstraksi, sehingga proses ekstraksi *file*

selanjutnya dapat dilakukan tanpa menunggu proses menulis data ke dalam *database*.

```
insert_feature = ExtractThread(self,
                                self.audio_files,
                                str(self.databaseNameVal.text()))
insert_feature.start()

QtCore.QObject.connect(insert_feature,
                        QtCore.SIGNAL("finish()"),
                        self.finish_thread)
QtCore.QObject.connect(insert_feature,
                        QtCore.SIGNAL("update()"),
                        self.update_progress)
```

#### 4.2.8 Implementasi Learning Vector Quantization

Implementasi LVQ dibagi menjadi dua bagian, yaitu pelatihan data untuk menentukan bobot akhir tiap kelas *output* dan pengujian data untuk menentukan kelas akhir dari data suara baru. Tahap – tahap implementasi Learning Vector Quantization adalah sebagai berikut:

1. Proses Pelatihan Data:

- a. Jumlah *neuron* masukan adalah sebanyak 13 buah, untuk 1 masukan *audio file* dapat memiliki beberapa *input*, tergantung kepada jumlah *frame* yang dihasilkan dari proses ekstraksi.
- b. Tentukan vektor bobot awal, diambil satu data acak dari data *input* setiap kelas. Pengambilan data secara acak menggunakan SQL Query.

```
SELECT * FROM (SELECT * FROM feature ORDER BY
random()) GROUP BY class
```

- c. Ambil vektor dataset yang bukan merupakan vektor bobot awal yang ditentukan pada tahap sebelumnya.
- d. Lakukan *looping* selama jumlah maximum iterasi belum terpenuhi, pada setiap iterasi, lakukan looping sejumlah dataset *input*.
- e. Hitung nilai jarak Euclidean dari bobot vektor awal dan data *input* dengan menggunakan persamaan 2.21

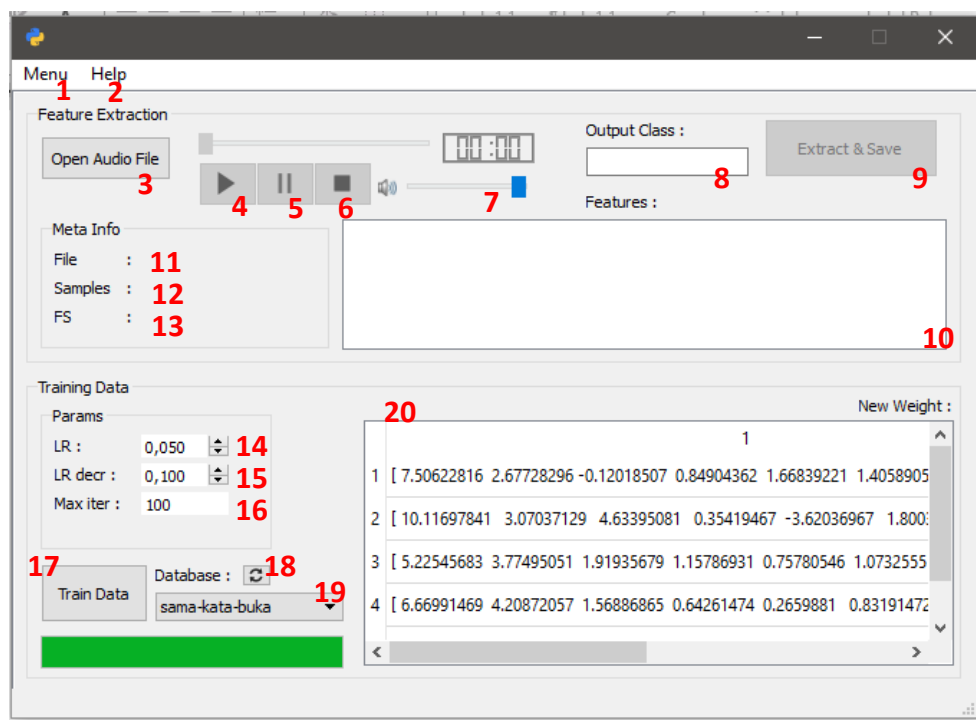
- f. Setelah didapatkan jarak Euclidean setiap vektor, cari nilai yang paling minimal, jika kelas target sesuai dengan kelas vektor tersebut, maka perbaharui bobot menggunakan persamaan 2.19, jika tidak perbaharui bobot menggunakan persamaan 2.20
- g. Pada setiap iterasi, kurangi nilai *learning rate* sejumlah nilai parameter yang ditentukan.
- h. Setelah seluruh iterasi terpenuhi, simpan bobot akhir ke dalam database untuk selanjutnya dibandingkan dengan data uji.

## 2. Proses Pengujian Data:

- a. Lakukan ekstraksi fitur pada data suara yang baru dimasukkan.
- b. Lakukan *looping* sejumlah dataset fitur yang didapat pada tahap sebelumnya.
- c. Hitung nilai jarak Euclidean dari bobot vektor akhir dan data input dengan menggunakan persamaan 2.21
- d. Cari nilai paling minimal dari jarak Euclidean, lalu tampung hasil kedalam list.
- e. Lakukan *voting* untuk hasil terbanyak yang terdapat pada list yang berisi hasil dari perhitungan nilai minimal dari jarak Euclidean.
- f. Hasil akhir kelas adalah vektor dengan jumlah *voting* terbanyak.

### 4.3 Antarmuka Aplikasi

Bagian ini menjelaskan tampilan akhir aplikasi yang dibuat beserta penjelasan kegunaan setiap halaman.

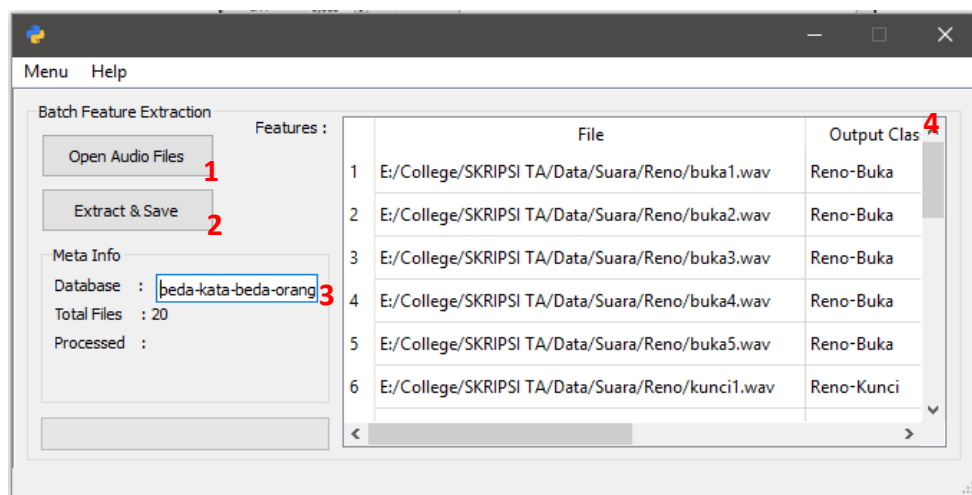


Gambar 4.1 Antarmuka aplikasi untuk pelatihan data

Keterangan:

1. Menu untuk mengganti ke tampilan training, testing atau batch features extraction.
2. Menu untuk menampilkan *option help*.
3. Tombol untuk memilih satu *audio file*.
4. Tombol untuk *play audio*.
5. Tombol untuk *pause audio*.
6. Tombol untuk *stop audio*.
7. Kontrol *volume audio*.
8. *Input field* kelas keluaran.
9. Tombol untuk mengekstraksi fitur dan menyimpannya ke dalam *database*.

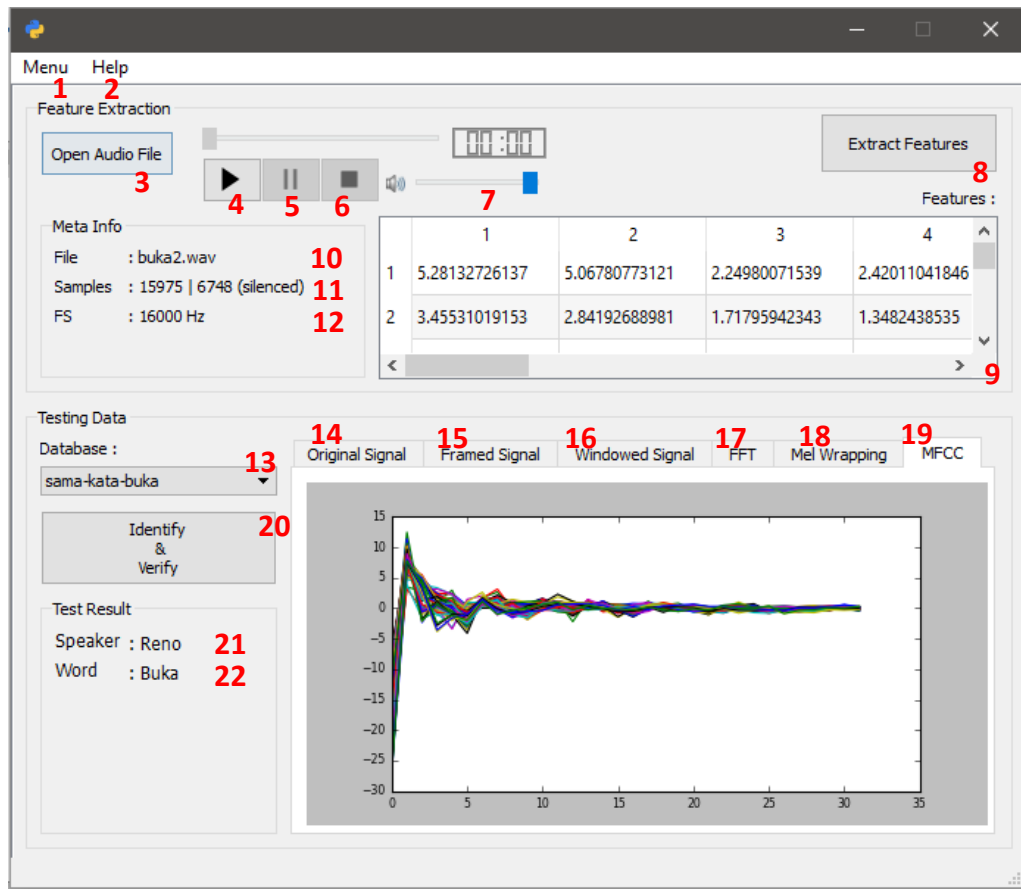
10. Table hasil ekstraksi fitur.
11. Informasi nama *audio file*.
12. Informasi jumlah sampel pada *audio file*.
13. Informasi *Frequency Sampling* pada *audio file*.
14. *Double spinbox* untuk mengganti nilai parameter *Learning Rate*.
15. *Double spinbox* untuk mengganti nilai parameter pengurangan *Learning Rate*.
16. *Input field* untuk mengganti nilai parameter jumlah iterasi.
17. Tombol untuk mengeksekusi pelatihan data.
18. Tombol untuk *reload* daftar *database* yang baru ditambahkan.
19. *Combo box* untuk memilih *database* yang ingin digunakan.
20. Tabel hasil bobot akhir pelatihan data.



Gambar 4.2 Antarmuka aplikasi untuk batch features extraction

Keterangan:

1. Tombol untuk memilih file-file *audio*.
2. Tombol untuk mengekstraksi fitur dari semua *audio* file yang dipilih dan menyimpannya ke dalam *database*.
3. Tombol untuk memilih satu *audio file*.
4. Tabel untuk menampilkan file-file *audio* yang dipilih.



Gambar 4.3 Antarmuka aplikasi untuk pengujian data

Keterangan:

1. Menu untuk mengganti ke tampilan *training* atau *testing*.
2. Menu untuk menampilkan *option help*.
3. Tombol untuk memilih satu *audio file*.
4. Tombol untuk *play audio*.
5. Tombol untuk *pause audio*.
6. Tombol untuk *stop audio*.
7. Kontrol *volume audio*.
8. *Input field* kelas keluaran.
9. Tombol untuk mengekstraksi fitur.
10. Informasi nama *audio file*.
11. Informasi jumlah sampel pada *audio file*.
12. Informasi *Frequency Sampling* pada *audio file*.

13. Combo box untuk memilih database yang ingin digunakan.
- 14-19. *Tab* yang menampilkan *plotting* dari sinyal *audio*.
20. Tombol untuk mengidentifikasi pembicara.
21. Informasi nama pembicara yang teridentifikasi.
22. Informasi kata yang teridentifikasi.

#### **4.4 Pengujian**

Pada bagian ini akan dijelaskan bagaimana skenario pengujian yang dilakukan dan bagaimana hasilnya serta analisis untuk setiap hasil pengujian yang didapatkan. Pengujian dilakukan dengan menggunakan jumlah data latih yang beragam, sedangkan data uji yang digunakan ada sebanyak 4 buah *file audio* untuk setiap 1 pembicara.

Data uji diambil dari data yang bukan merupakan data latih. Setiap orang akan diambil 1 rekaman suara untuk setiap kata, jumlah data uji setiap pembicara adalah 4 suara, sehingga total data uji adalah 20 buah suara. Pada percobaan ini penulis melakukan 3 kali pelatihan data dengan parameter yang sama untuk mendapatkan 3 bobot akhir yang berbeda, yang dimana akurasi akhir yang didapat adalah hasil akurasi rata-rata dari 3 kali pengujian pada parameter yang sama.

##### **4.4.1 Pengujian dengan Kata yang Berbeda**

Pada pengujian ini, terdapat 4 kata berbeda yang diucapkan untuk data latih dan uji. Data latih yang digunakan untuk setiap kata adalah 10 buah, untuk pembicaranya sendiri, terdapat 5 orang pembicara, sehingga didapatkan 40 buah suara untuk satu orang pembicara. Sehingga total data latih untuk pengujian ini adalah sebanyak 200 buah.

Pada pengujian ke 1 parameter *Learning Rate* yang digunakan adalah 0.2, *Learning Rate Decay* adalah 0.1 dan *Maximum Epoch* adalah 100. Akurasi pembicara-kata rata-rata yang didapatkan dari pengujian dengan adalah 26.6%,



sedangkan untuk akurasi pembicara saja didapatkan 43.3%. Hasil identifikasi pembicara dan kata pengujian ke-1 dapat dilihat di lampiran A.

Pada pengujian ke 2 parameter *Learning Rate* yang digunakan adalah 0.1, *Learning Rate Decay* adalah 0.1 dan *Maximum Epoch* adalah 100. Akurasi pembicara-kata rata-rata yang didapatkan dari pengujian dengan adalah 41.6%, sedangkan untuk akurasi pembicara saja didapatkan 56.6%. Hasil identifikasi pembicara dan kata pengujian ke-1 dapat dilihat di lampiran B.

Pada pengujian ke 3 parameter *Learning Rate* yang digunakan adalah 0.05, *Learning Rate Decay* adalah 0.097 dan *Maximum Epoch* adalah 100. Akurasi pembicara-kata rata-rata yang didapatkan dari pengujian dengan adalah 48.3%, sedangkan untuk akurasi pembicara saja didapatkan 63.3%. Hasil identifikasi pembicara dan kata pengujian ke-1 dapat dilihat di lampiran C.

Pada pengujian ke 4 parameter *Learning Rate* yang digunakan adalah 0.05, *Learning Rate Decay* adalah 0.097 dan *Maximum Epoch* adalah 200. Akurasi pembicara-kata rata-rata yang didapatkan dari pengujian dengan adalah 48.3%, sedangkan untuk akurasi pembicara saja didapatkan 71.6%. Hasil identifikasi pembicara dan kata pengujian ke-1 dapat dilihat di lampiran D.

Pada pengujian ke 5 parameter *Learning Rate* yang digunakan adalah 0.03, *Learning Rate Decay* adalah 0.095 dan *Maximum Epoch* adalah 100. Akurasi pembicara-kata rata-rata yang didapatkan dari pengujian dengan adalah 48.3%, sedangkan untuk akurasi pembicara saja didapatkan 65%. Hasil identifikasi pembicara dan kata pengujian ke-1 dapat dilihat di lampiran D.

Tabel 4.6 Hasil pengujian beda kata

No	parameter			Pengujian Pembicara-Kata 1	Pengujian Pembicara-Kata 2	Pengujian Pembicara-Kata 3	AVG	Pengujian Pembicara 1	Pengujian Pembicara 2	Pengujian Pembicara 3	AVG
	alpha	decay	epoch								
1	0.2	0.1	100	25%	25%	30%	<b>26.6%</b>	40%	45%	45%	<b>43.3%</b>
2	0.1	0.1	100	50%	35%	40%	<b>41.6%</b>	55%	55%	60%	<b>56.6%</b>
3	0.05	0.097	100	40%	50%	55%	<b>48.3%</b>	60%	70%	60%	<b>63.3%</b>
4	0.05	0.097	200	55%	50%	40%	<b>48.3%</b>	70%	75%	70%	<b>71.6%</b>
5	0.03	0.095	100	50%	30%	65%	<b>48.3%</b>	60%	45%	90%	<b>65%</b>

#### 4.4.2 Pengujian dengan Kata yang Sama

Pada pengujian ini, pembicara hanya mengucapkan 1 kata yaitu ‘Buka’ untuk data latih dan uji. Data latih yang digunakan untuk setiap orang adalah 10 buah, untuk pembicaranya sendiri, terdapat 5 orang pembicara. Sehingga total data latih untuk pengujian ini adalah sebanyak 50 buah. Sedangkan untuk data uji, masing-masing pembicara hanya mengucapkan 1 kali, sehingga data uji yang didapat adalah sebanyak 5 buah.

Pada pengujian ke 1 parameter *Learning Rate* yang digunakan adalah 0.1, *Learning Rate Decay* adalah 0.1 dan *Maximum Epoch* adalah 100. Akurasi pembicara didapatkan sebesar 33.3%. Hasil identifikasi pembicara pengujian ke-1 dapat dilihat di lampiran E.

Pada pengujian ke 2 parameter *Learning Rate* yang digunakan adalah 0.05, *Learning Rate Decay* adalah 0.1 dan *Maximum Epoch* adalah 100. Akurasi pembicara didapatkan sebesar 53.3%. Hasil identifikasi pembicara pengujian ke-2 dapat dilihat di lampiran E.

Pada pengujian ke 3 parameter *Learning Rate* yang digunakan adalah 0.05, *Learning Rate Decay* adalah 0.097 dan *Maximum Epoch* adalah 200. Akurasi

pembicara didapatkan sebesar 20%. Hasil identifikasi pembicara pengujian ke-3 dapat dilihat di lampiran E.

Pada pengujian ke 4 parameter *Learning Rate* yang digunakan adalah 0.05, *Learning Rate Decay* adalah 0.097 dan *Maximum Epoch* adalah 100. Akurasi pembicara didapatkan sebesar 66.6%. Hasil identifikasi pembicara pengujian ke-4 dapat dilihat di lampiran E.

Pada pengujian ke 5 parameter *Learning Rate* yang digunakan adalah 0.03, *Learning Rate Decay* adalah 0.097 dan *Maximum Epoch* adalah 100. Akurasi pembicara didapatkan sebesar 80%. Hasil identifikasi pembicara pengujian ke-5 dapat dilihat di lampiran E.

Tabel 4.7 Hasil pengujian sama kata

No	parameter			Pengujian Pembicara 1	Pengujian Pembicara 2	Pengujian Pembicara 3	AVG
	alpha	decay	epoch				
1	0.1	0.1	100	0%	40%	60%	<b>33.3%</b>
2	0.05	0.1	100	80%	40%	40%	<b>53.3%</b>
3	0.05	0.097	200	20%	20%	20%	<b>20%</b>
4	0.05	0.097	100	60%	60%	80%	<b>66.6%</b>
5	0.03	0.097	100	100%	60%	80%	<b>80%</b>

## **BAB V**

### **PENUTUP**

Dalam bab ini terdapat kesimpulan serta saran yang didapatkan oleh penulis berdasarkan penelitiannya.

#### **5.1 Kesimpulan**

Untuk menjawab tujuan dari penelitian ini, maka kesimpulan yang dapat diambil adalah:

1. Nilai *alpha* dan *alpha decay* cukup mempengaruhi akurasi identifikasi dan verifikasi pembicara. Pada penelitian ini nilai *alpha* dan *alpha decay* yang dianggap paling baik adalah  $\alpha = 0.03$  dan  $\alpha \text{ decay} = 0.097$ .
2. Akurasi identifikasi pembicara dengan data latih hanya satu kata yang didapatkan lebih baik daripada akurasi identifikasi pembicara pada data latih banyak kata.
3. Metode MFCC lebih menghasilkan akurasi yang cukup tinggi untuk identifikasi pembicara saja, akurasi yang didapat paling tinggi untuk pembicara adalah sebesar 80%. Sehingga metode ini lebih cocok untuk pengenalan pembicara daripada pengenalan kata.

#### **5.2 Saran**

Setelah melihat kesimpulan di atas, maka saran yang dapat diberikan oleh penulis adalah sebagai berikut :

1. Pada penelitian berikutnya, pada proses ekstraksi fitur bisa menggunakan metode lain seperti Gammatone Frequency Cepstral Coefficients (GFCC) untuk melakukan pengenalan pembicara. sedangkan untuk klasifikasi hasil pengenalan dapat dicoba menggunakan Hidden Markov Model.

2. Metode MFCC sebaiknya hanya digunakan untuk pengenalan pembicara saja, dapat dilihat hasil akurasi dalam mengenali pembicara berikut kata yang diucapkan masih kurang.
3. Optimasi-optimasi parameter LVQ dan MFCC dapat dikembangkan lebih jauh lagi untuk mendapatkan akurasi pengenalan yang terbaik.

## DAFTAR PUSTAKA

- [ANG11] Angga Setiawan, Achmad Hidayanto, R. Rizal Isnanto. 2011. “Aplikasi Pengenalan Ucapan dengan Ekstraksi Mel-Frequency Cepstrum Coefficients (MFCC) Melalui Jaringan Syaraf Tiruan (JST) Learning Vector Quantization (LVQ) untuk Mengoperasikan Kursor Komputer”, TRANSMISI, 13 (3), 2011, 82-86
- [BAG13] Bagwell, Chris. 2013. Sound eXchange, the Swiss Army knife of audio manipulation, <http://sox.sourceforge.net/sox.html>. Diakses April 2016
- [BHA13] Bhattacharjee, Utpal. Januari 2013. “A COMPARATIVE STUDY OF LPCC AND MFCC FEATURES FOR THE RECOGNITION OF ASSAMESE PHONEMES”, International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 1
- [DAV13] Dave, Namrata. Juli 2013. “FEATURE EXTRACTION METHODS LPC, PLP AND MFCC IN SPEECH RECOGNITION”. International Journal For Advance Research In Engineering And Technology, Vol 1, Issue 4
- [FAU94] Fausett, Laurene. 1994. “Fundamental of Neural Networks: Architectures, Algorithms, and Applications”.
- [GEE14] Geeta Nijhawan, M.K Soni. Juli 2014. “SPEAKER RECOGNITION USING MFCC AND VECTOR QUANTISATION”. International Journal on Recent Trends in Engineering and Technology, Vol 11 No. 1

- [HUA01] Huang, Xuedong, Alex Acero and Hsiao-Wuen Hon. 2001. Spoken Language Processing: A Guide To Theory, Algorithm And System Development. Prentice Hall, New Jersey.
- [KSH12] Kshamamayee Dash, Debananda Padhi, Bhoomika Panda and Prof. Sanghamitra Mohanty. April 2012. "SPEAKER IDENTIFICATION USING MEL FREQUENCY CEPSTRAL COEFFICIENT AND BPNN". International Journal of Advanced Research in Computer Science and Software Engineering, Vol 2, Issue 4
- [LYO11] Lyons, Richard G. 2011. Understanding Digital Signal Processing 3rd Edition. Prentice Hall, Boston
- [NAV09] Navarrete, Jason. Agustus 2009. The Sox of Silence. <http://digitalcardboard.com/blog/2009/08/25/the-sox-of-silence/>. Diakses April 2016
- [MAN11] Mandalia, Darshan and Gareta, Pravin. Mei 2011. "Speaker Recognition Using MFCC and Vector Quantization Model". Department of Electrical Engineering Electronics & Communication Engineering Program Institute of Technology. NIRMA University.
- [PEN15] Penghua LI, Shunxing ZHANG, Huizong FENG, Yuanyuan LI. 2015. "SPEAKER IDENTIFICATION USING SPECTROGRAM AND LEARNING VECTOR QUANTIZATION". Journal of Computational Information Systems, Vol 11, Issue 9
- [SET16] Setiawan, Ebta. 2016. Suara, Kamus Besar Bahasa Indonesia Online. <http://kbbi.web.id/suara>. Diakses Maret 2016

- [TIW10] Tiwari, Vibha. 2010. "MFCC And Its Application In Speaker Recognition". International Journal on Emerging Technologies.
- [ZHI13] Zhizheng Wu, Anthony Larcher, Kong Aik Lee, Eng Siong Chn, Tomi Kinnunen and Haizhou Li. August 2013. "Vulnerability evaluation of speaker verification under voice conversion spoofing: the effect of text constraints". ISCA



## LAMPIRAN A

Hasil Pengujian dengan parameter sebagai berikut :

1. Alpha = 0.2
2. Alpha Decay = 0.1
3. Max epoh = 100

Table 1 Pengujian ke-1 pelatihan data ke-1

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Hulda	Buka
	Kunci	Jeremy	Kunci
	Lock	Reno	Unlock
	Unlock	Jeremy	Kunci
Rifky	Buka	Rifky	Kunci
	Kunci	Rifky	Kunci
	Lock	Reno	Unlock
	Unlock	Riana	Buka
Riana	Buka	Reno	Buka
	Kunci	Jeremy	Kunci
	Lock	Reno	Buka
	Unlock	Reno	Buka
Hulda	Buka	Riana	Buka
	Kunci	Hulda	Kunci
	Lock	Hulda	Lock
	Unlock	Riana	Lock
Jeremy	Buka	Reno	Kunci
	Kunci	Jeremy	Kunci
	Lock	Jeremy	Lock
	Unlock	Jeremy	Lock

Table 2 Pengujian ke-1 pelatihan data ke-2

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Jeremy	Unlock
	Kunci	Reno	Kunci
	Lock	Reno	Buka
	Unlock	Jeremy	Unlock
Rifky	Buka	Hulda	Kunci
	Kunci	Rifky	Kunci
	Lock	Hulda	Kunci
	Unlock	Hulda	Kunci
Riana	Buka	Riana	Unlock
	Kunci	Hulda	Kunci
	Lock	Riana	Unlock
	Unlock	Riana	Lock
Hulda	Buka	Riana	Unlock
	Kunci	Hulda	Kunci
	Lock	Jeremy	Buka
	Unlock	Riana	Lock
Jeremy	Buka	Reno	Buka
	Kunci	Reno	Kunci
	Lock	Jeremy	Lock
	Unlock	Jeremy	Unlock

Table 3 Pengujian ke-1 pelatihan data ke-3

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Reno	Buka
	Kunci	Riana	Kunci
	Lock	Jeremy	Unlock
	Unlock	Reno	Lock
Rifky	Buka	Reno	Kunci

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
	Kunci	Reno	Kunci
	Lock	Rifky	Lock
	Unlock	Reno	Kunci
Riana	Buka	Reno	Buka
	Kunci	Riana	Kunci
	Lock	Riana	Lock
	Unlock	Hulda	Lock
Hulda	Buka	Rifky	Buka
	Kunci	Reno	Kunci
	Lock	Riana	Lock
	Unlock	Hulda	Lock
Jeremy	Buka	Jeremy	Buka
	Kunci	Jeremy	Kunci
	Lock	Riana	Lock
	Unlock	Jeremy	Kunci

## LAMPIRAN B

Hasil Pengujian dengan parameter sebagai berikut :

1. Alpha = 0.1
2. Alpha Decay = 0.1
3. Max epoch = 100

Table 4 Hasil pengujian ke-2 pelatihan data ke-1

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Hulda	Lock
	Kunci	Hulda	Kunci
	Lock	Reno	Lock
	Unlock	Reno	Lock
Rifky	Buka	Hulda	Kunci
	Kunci	Hulda	Kunci
	Lock	Rifky	Lock
	Unlock	Rifky	Unlock
Riana	Buka	Riana	Buka
	Kunci	Riana	Kunci
	Lock	Riana	Lock
	Unlock	Hulda	Lock
Hulda	Buka	Hulda	Buka
	Kunci	Hulda	Kunci
	Lock	Hulda	Lock
	Unlock	Hulda	Unlock
Jeremy	Buka	Rifky	Unlock
	Kunci	Reno	Unlock
	Lock	Hulda	Lock
	Unlock	Reno	Lock

Table 5 Hasil pengujian ke-2 pelatihan data ke-2

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Rifky	Unlock
	Kunci	Jeremy	Kunci
	Lock	Jeremy	Lock
	Unlock	Reno	Lock
Rifky	Buka	Rifky	Kunci
	Kunci	Rifky	Kunci
	Lock	Jeremy	Unlock
	Unlock	Rifky	Lock
Riana	Buka	Riana	Buka
	Kunci	Jeremy	Kunci
	Lock	Riana	Lock
	Unlock	Riana	Lock
Hulda	Buka	Hulda	Buka
	Kunci	Hulda	Kunci
	Lock	Riana	Unlock
	Unlock	Riana	Unlock
Jeremy	Buka	Reno	Buka
	Kunci	Jeremy	Kunci
	Lock	Jeremy	Lock
	Unlock	Riana	Buka

Table 6 Hasil pengujian ke-2 pelatihan data ke-3

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Reno	Buka
	Kunci	Reno	Kunci
	Lock	Jeremy	Lock
	Unlock	Hulda	Buka

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Rifky	Buka	Rifky	Buka
	Kunci	Rifky	Buka
	Lock	Rifky	Lock
	Unlock	Rifky	Kunci
Riana	Buka	Riana	Buka
	Kunci	Riana	Kunci
	Lock	Hulda	Buka
	Unlock	Jeremy	Lock
Hulda	Buka	Hulda	Buka
	Kunci	Rifky	Kunci
	Lock	Jeremy	Lock
	Unlock	Hulda	Lock
Jeremy	Buka	Rifky	Lock
	Kunci	Jeremy	Unlock
	Lock	Rifky	Lock
	Unlock	Jeremy	Unlock

## LAMPIRAN C

Hasil Pengujian dengan parameter sebagai berikut :

1. Alpha = 0.05
2. Alpha Decay = 0.097
3. Max epoh = 100

Table 7 Hasil pengujian ke-3 pelatihan data ke-1

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Jeremy	Unlock
	Kunci	Reno	Kunci
	Lock	Riana	Lock
	Unlock	Reno	Unlock
Rifky	Buka	Rifky	Buka
	Kunci	Rifky	Kunci
	Lock	Riana	Kunci
	Unlock	Rifky	Kunci
Riana	Buka	Jeremy	Unlock
	Kunci	Reno	Kunci
	Lock	Riana	Unlock
	Unlock	Hulda	Lock
Hulda	Buka	Hulda	Lock
	Kunci	Rifky	Kunci
	Lock	Hulda	Lock
	Unlock	Hulda	Lock
Jeremy	Buka	Jeremy	Buka
	Kunci	Jeremy	Kunci
	Lock	Riana	Lock
	Unlock	Jeremy	Unlock

Table 8 Hasil pengujian ke-3 pelatihan data ke-2

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Reno	Buka
	Kunci	Rifky	Unlock
	Lock	Rifky	Unlock
	Unlock	Reno	Unlock
Rifky	Buka	Rifky	Kunci
	Kunci	Rifky	Kunci
	Lock	Riana	Unlock
	Unlock	Rifky	Buka
Riana	Buka	Riana	Buka
	Kunci	Riana	Kunci
	Lock	Riana	Lock
	Unlock	Riana	Kunci
Hulda	Buka	Reno	Buka
	Kunci	Rifky	Kunci
	Lock	Reno	Buka
	Unlock	Hulda	Lock
Jeremy	Buka	Jeremy	Buka
	Kunci	Jeremy	Kunci
	Lock	Jeremy	Lock
	Unlock	Jeremy	Unlock

Table 9 Hasil pengujian ke-3 pelatihan data ke-3

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Reno	Buka
	Kunci	Jeremy	Kunci
	Lock	Reno	Lock



Pembicara	Kata	Identifikasi	
		Pembicara	Kata
	Unlock	Hulda	Lock
Rifky	Buka	Rifky	Buka
	Kunci	Rifky	Kunci
	Lock	Jeremy	Lock
	Unlock	Rifky	Unlock
Riana	Buka	Jeremy	Buka
	Kunci	Riana	Kunci
	Lock	Riana	Lock
	Unlock	Riana	Buka
Hulda	Buka	Hulda	Buka
	Kunci	Rifky	Kunci
	Lock	Jeremy	Unlock
	Unlock	Hulda	Unlock
Jeremy	Buka	Riana	Buka
	Kunci	Jeremy	Kunci
	Lock	Riana	Buka
	Unlock	Jeremy	Unlock

## Lampiran D

Hasil Pengujian dengan parameter sebagai berikut :

1. Alpha = 0.05
2. Alpha Decay = 0.097
3. Max epoch = 200

Table 10 Hasil pengujian ke-4 pelatihan data ke-1

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Reno	Buka
	Kunci	Riana	Kunci
	Lock	Jeremy	Lock
	Unlock	Reno	Unlock
Rifky	Buka	Rifky	Kunci
	Kunci	Rifky	Kunci
	Lock	Reno	Lock
	Unlock	Rifky	Kunci
Riana	Buka	Jeremy	Buka
	Kunci	Riana	Kunci
	Lock	Riana	Lock
	Unlock	Riana	Lock
Hulda	Buka	Reno	Buka
	Kunci	Hulda	Kunci
	Lock	Hulda	Lock
	Unlock	Reno	Buka
Jeremy	Buka	Jeremy	Buka
	Kunci	Jeremy	Kunci
	Lock	Jeremy	Lock
	Unlock	Jeremy	Unlock

Table 111 Hasil pengujian ke-4 pelatihan data ke-2

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Reno	Unlock
	Kunci	Reno	Kunci
	Lock	Riana	Unlock
	Unlock	Reno	Buka
Rifky	Buka	Rifky	Unlock
	Kunci	Rifky	Unlock
	Lock	Rifky	Kunci
	Unlock	Hulda	Buka
Riana	Buka	Riana	Buka
	Kunci	Riana	Kunci
	Lock	Riana	Buka
	Unlock	Riana	Unlock
Hulda	Buka	Jeremy	Lock
	Kunci	Hulda	Kunci
	Lock	Riana	Unlock
	Unlock	Jeremy	Lock
Jeremy	Buka	Jeremy	Buka
	Kunci	Jeremy	Kunci
	Lock	Jeremy	Lock
	Unlock	Jeremy	Unlock

Table 12 Hasil pengujian ke-4 pelatihan data ke-3

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Reno	Lock
	Kunci	Rifky	Lock
	Lock	Reno	Unlock
	Unlock	Reno	Unlock
Rifky	Buka	Rifky	Buka

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
	Kunci	Rifky	Buka
	Lock	Reno	Unlock
	Unlock	Rifky	Lock
Riana	Buka	Riana	Lock
	Kunci	Riana	Kunci
	Lock	Riana	Lock
	Unlock	Riana	Unlock
Hulda	Buka	Hulda	Buka
	Kunci	Rifky	Lock
	Lock	Hulda	Lock
	Unlock	Hulda	Unlock
Jeremy	Buka	Rifky	Lock
	Kunci	Jeremy	Kunci
	Lock	Reno	Unlock
	Unlock	Reno	Unlock

## LAMPIRAN E

Hasil Pengujian dengan parameter sebagai berikut :

1. Alpha = 0.03
2. Alpha Decay = 0.095
3. Max epoch = 100

Table 13 Hasil pengujian ke-5 pelatihan data ke-1

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Reno	Unlock
	Kunci	Hulda	Kunci
	Lock	Reno	Lock
	Unlock	Reno	Unlock
Rifky	Buka	Hulda	Buka
	Kunci	Hulda	Buka
	Lock	Rifky	Lock
	Unlock	Hulda	Buka
Riana	Buka	Riana	Buka
	Kunci	Rifky	Unlock
	Lock	Jeremy	Unlock
	Unlock	Riana	Unlock
Hulda	Buka	Hulda	Lock
	Kunci	Hulda	Kunci
	Lock	Riana	Unlock
	Unlock	Hulda	Unlock
Jeremy	Buka	Jeremy	Buka
	Kunci	Hulda	Kunci
	Lock	Jeremy	Lock
	Unlock	Jeremy	Unlock

Table 14 Hasil pengujian ke-5 pelatihan data ke-2

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Riana	Lock
	Kunci	Reno	Kunci
	Lock	Hulda	Kunci
	Unlock	Reno	Buka
Rifky	Buka	Rifky	Buka
	Kunci	Rifky	Buka
	Lock	Rifky	Lock
	Unlock	Jeremy	Buka
Riana	Buka	Riana	Kunci
	Kunci	Jeremy	Lock
	Lock	Rifky	Lock
	Unlock	Riana	Unlock
Hulda	Buka	Hulda	Buka
	Kunci	Hulda	Kunci
	Lock	Riana	Lock
	Unlock	Riana	Lock
Jeremy	Buka	Riana	Kunci
	Kunci	Reno	Kunci
	Lock	Rifky	Lock
	Unlock	Riana	Lock

Table 15 Hasil pengujian ke-5 pelatihan data ke-3

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
Reno	Buka	Reno	Buka
	Kunci	Reno	Kunci
	Lock	Reno	Lock
	Unlock	Reno	Buka
Rifky	Buka	Rifky	Buka

Pembicara	Kata	Identifikasi	
		Pembicara	Kata
	Kunci	Rifky	Kunci
	Lock	Rifky	Lock
	Unlock	Rifky	Unlock
Riana	Buka	Riana	Buka
	Kunci	Rifky	Buka
	Lock	Riana	Lock
	Unlock	Riana	Lock
Hulda	Buka	Hulda	Lock
	Kunci	Hulda	Kunci
	Lock	Hulda	Unlock
	Unlock	Hulda	Unlock
Jeremy	Buka	Rifky	Lock
	Kunci	Jeremy	Kunci
	Lock	Jeremy	Lock
	Unlock	Jeremy	Kunci

## LAMPIRAN F

Pengujian	Pelatihan	Pembicara	Identifikasi
1	1	Reno	Jeremy
		Rifky	Reno
		Jeremy	Reno
		Hulda	Reno
		Riana	Reno
1	2	Reno	Reno
		Rifky	Rifky
		Jeremy	Rifky
		Hulda	Reno
		Riana	Rifky
1	3	Reno	Reno
		Rifky	Rifky
		Jeremy	Rifky
		Hulda	Hulda
		Riana	Hulda
2	1	Reno	Reno
		Rifky	Rifky
		Jeremy	Jeremy
		Hulda	Hulda
		Riana	Hulda
2	2	Reno	Reno
		Rifky	Jeremy
		Jeremy	Hulda



Pengujian	Pelatihan	Pembicara	Identifikasi
		Hulda	Reno
		Riana	Riana
2	3	Reno	Jeremy
		Rifky	Rifky
		Jeremy	Rifky
		Hulda	Jeremy
		Riana	Riana
3	1	Reno	Hulda
		Rifky	Rifky
		Jeremy	Riana
		Hulda	Jeremy
		Riana	Jeremy
3	2	Reno	Rifky
		Rifky	Rifky
		Jeremy	Rifky
		Hulda	Rifky
		Riana	Rifky
3	3	Reno	Hulda
		Rifky	Reno
		Jeremy	Reno
		Hulda	Hulda
		Riana	Hulda
4	1	Reno	Reno
		Rifky	Rifky
		Jeremy	Jeremy
		Hulda	Riana
		Riana	Jeremy
4	2	Reno	Reno
		Rifky	Rifky

Pengujian	Pelatihan	Pembicara	Identifikasi
		Jeremy	Rifky
		Hulda	Riana
		Riana	Riana
4	3	Reno	Reno
		Rifky	Rifky
		Jeremy	Jeremy
		Hulda	Reno
		Riana	Riana
5	1	Reno	Reno
		Rifky	Rifky
		Jeremy	Jeremy
		Hulda	Hulda
		Riana	Riana
5	2	Reno	Reno
		Rifky	Rifky
		Jeremy	Riana
		Hulda	Hulda
		Riana	Riana
5	3	Reno	Hulda
		Rifky	Rifky
		Jeremy	Jeremy
		Hulda	Hulda
		Riana	Jeremy