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 memasukan 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 mengektraksi 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 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 *Spectogram* 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 *Spectogram* 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 Ceptral 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 selanjutkan dilakukan *Speaker Modeling*, bagian ini membagi-bagi data karakteristik suara menjadi beberapa model sebagai data pembanding 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 penelitan 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 menunjukan 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. Spectogram 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 Ceptral 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

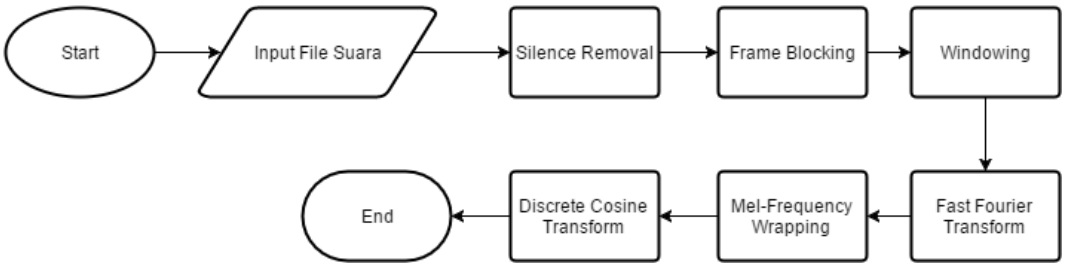
2.2 Tinjauan Pustaka

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 popular 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 akurasinya tinggi.

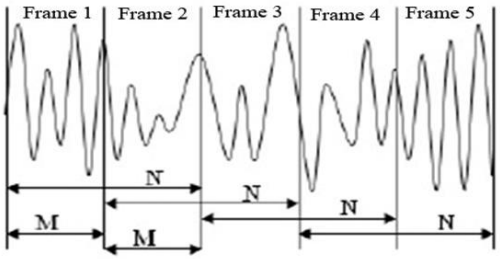


Gambar 2.1 Tahap-tahap MFCC

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]:

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

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]:

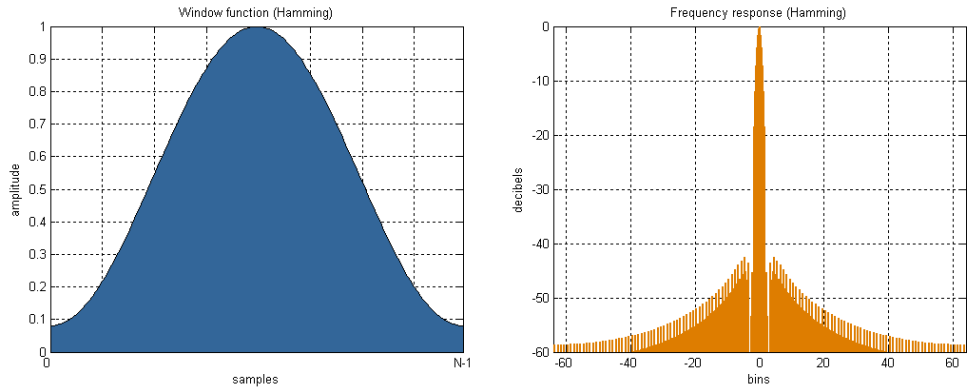
$$w(n) = 0.54 - 0.46\, cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \leq n \leq N-1 \tag{2.2}$$

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:

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

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:

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

Dimana,

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

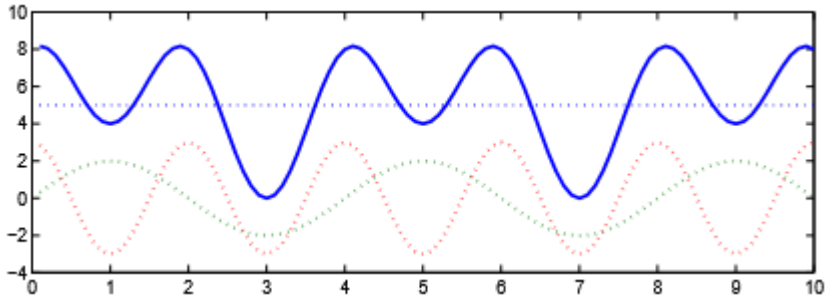
$m =$ indeks *output* DFT dalam domain frekuensi, m = 0, 1, 2, 3, …, N-1

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

n = indeks input sampel dalam domain waktu, n = 0, 1, 2, 3 ... , 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:

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

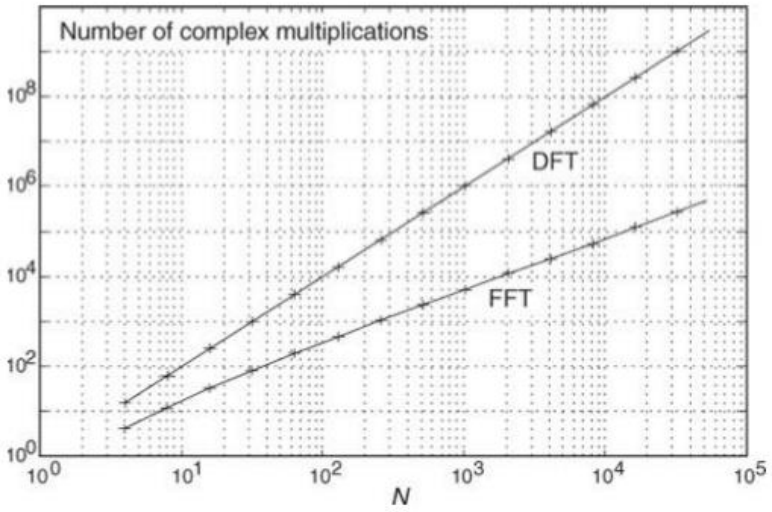
Dari persamaan tersebut kita mendapati bahwa $X(m)$ menghasilkan nilai *magnitude* dari komponen sinyal *input* yang didapat dari hasil akar dari penjumlahan bilangan real dan bilangan imajiner yang dikuadratkan dan dapat dituliskan dalam persamaan berikut:

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

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 popular adalah radix-2. Sebagai pembanding 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 komplek 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:

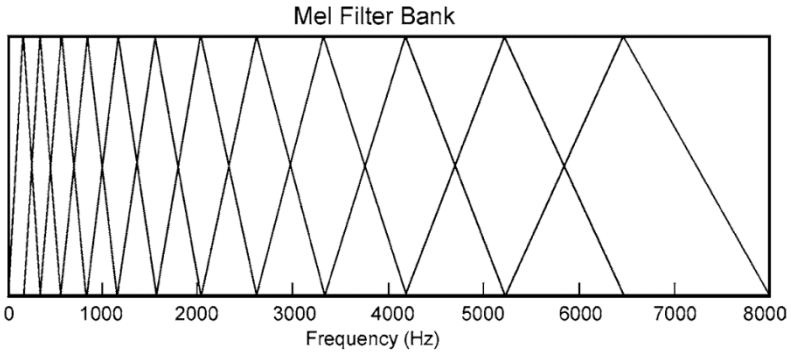
$$X(m) = X_{real}(m) + jX_{imag}(m) \tag{2.8}$$

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

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

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:

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

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

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

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]:

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

Dimana,

$H_m[k]$ = Nilai filterbank

m = Indeks filter

k = Indeks input sampel FFT (k = 0, 1, ... , N - 1)

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

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

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

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, ... , 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:

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

Dimana,

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

$S[m]$ = Nilai log energy

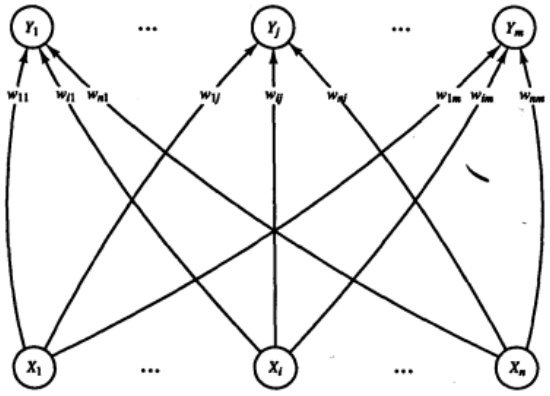
M = Jumlah filter

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 lapiran *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 sampa batas iterasi maksimal tercapai. Algoritma dalam melakukan LVQ adalah sebagai berikut:

0. Inisialisasi vektor referensi; inisialisasi *learning rate* (α), 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

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

   Jika $T \neq C_j$ maka

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

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 $(x_1, x_2, x_3, \ldots x_n)$

$T =$ 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* = $\alpha - (LR\ Decrement * \alpha)$

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).

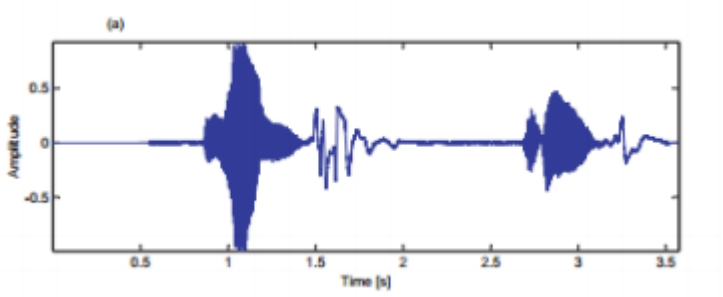
threshold : nilai *threshold* yang akan dihilangkan sebagai suara diam.

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].



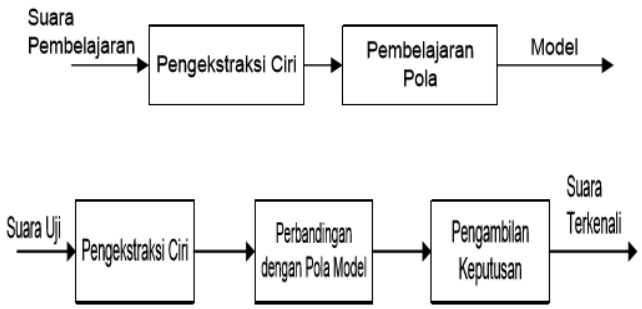
Gambar 2.8 Representasi Sinyal Suara

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

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 keseuaian 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.

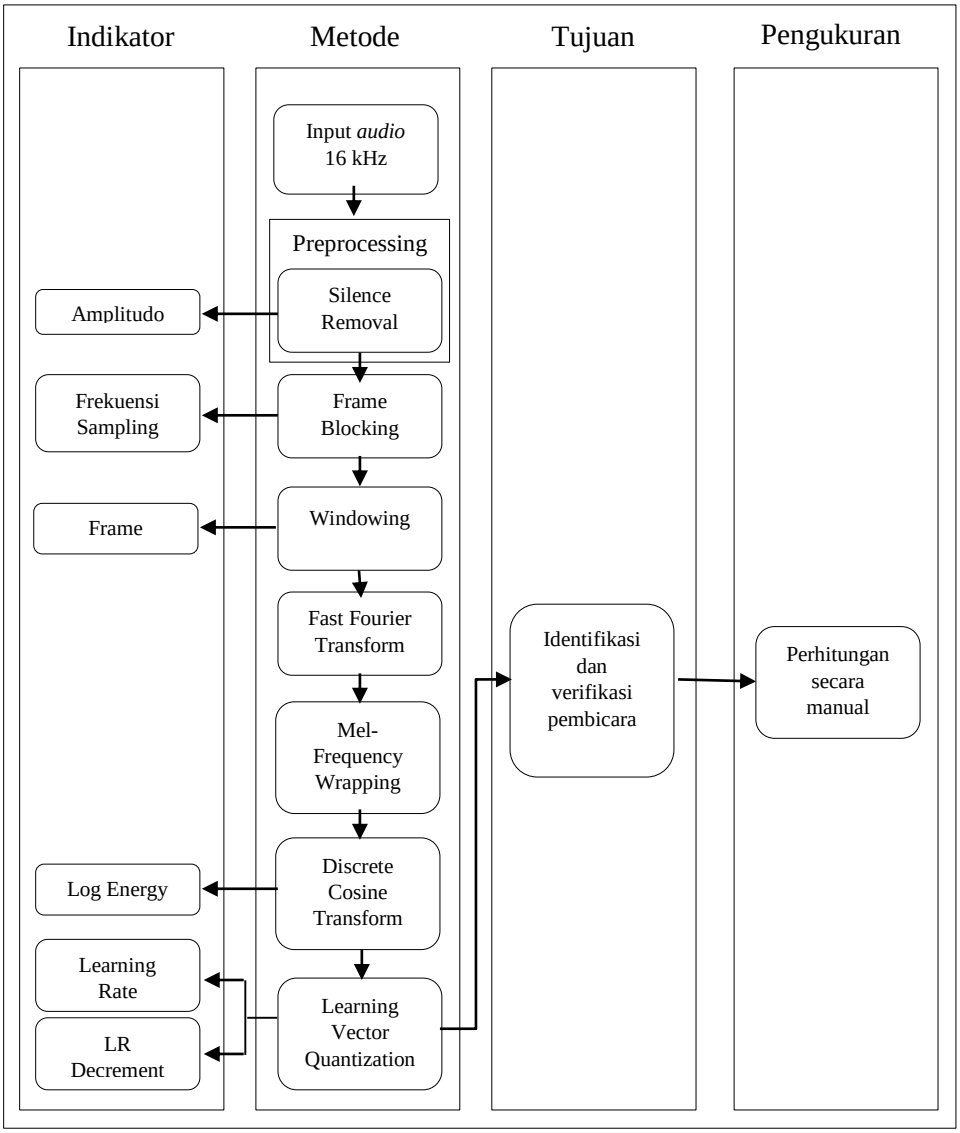
3.1 Analisis Masalah

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 pengekstraksi 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:

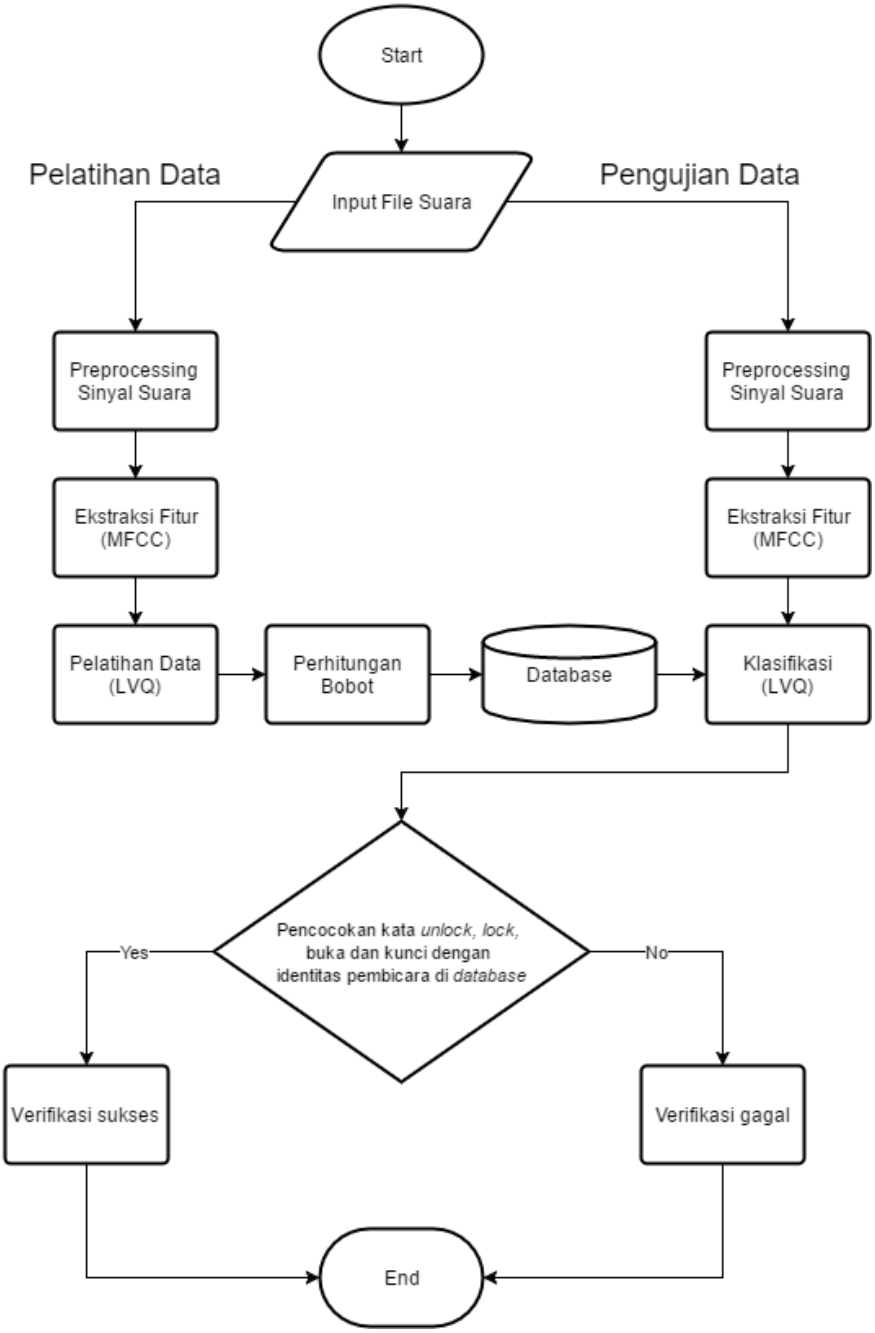
1. *File* suara hasil rekaman dengan frekuensi sampling sebesar 16 kHz dimasukan sebagai *input*.
2. Selanjutnya dilakukan *silence removal* yang amplituda nya kecil akan dihilangkan.

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 *ceptral 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

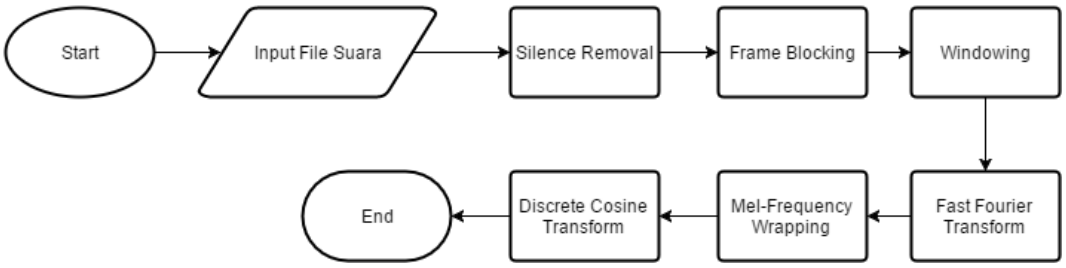
Sampel ke-	Nilai Sampel
1	0.00006
2	-0.00006
3	0.00012
4	-0.00021
5	0.00012
…	…
15970	-0.00006
15971	-0.00009
15972	-0.00006

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]:

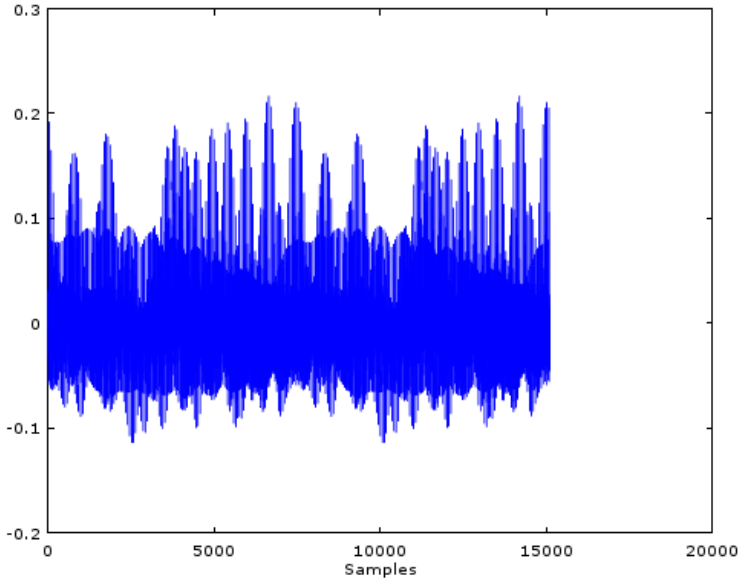
Table 3. 2 Nilai Parameter SoX Silence

above-period	***below-period***	***duration***	***threshold***
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 tahan *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:

numFrame = floor((sample-frameSize) / overlap) + 1
= floor(7774 – 256) / 128) + 1
= 59

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:

$$w_0 = 0{,}54 - 0{,}46\ cos\left(\frac{2*0*\pi}{256-1}\right)$$
$$= 0{,}08$$
$$y_0 = -0.00021362 * 0{,}08$$
$$= -0{,}000017$$
$$w_1 = 0{,}54 - 0{,}46\ cos\left(\frac{2*1*\pi}{256-1}\right)$$
$$= 0.08014$$
$$y_1 = -0.00024414 * 0.08014$$
$$= -0{,}000019$$

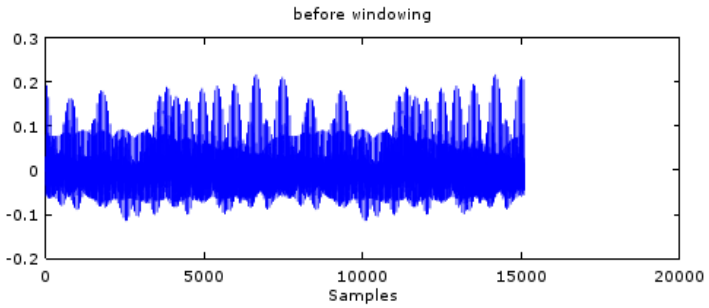
…

$$w_{255} = 0{,}54 - 0{,}46\ cos\left(\frac{2*255*\pi}{256-1}\right)$$
$$= 0.08$$
$$y_{256} = -0.0024719 * 0.08$$
$$= -0{,}00019775$$

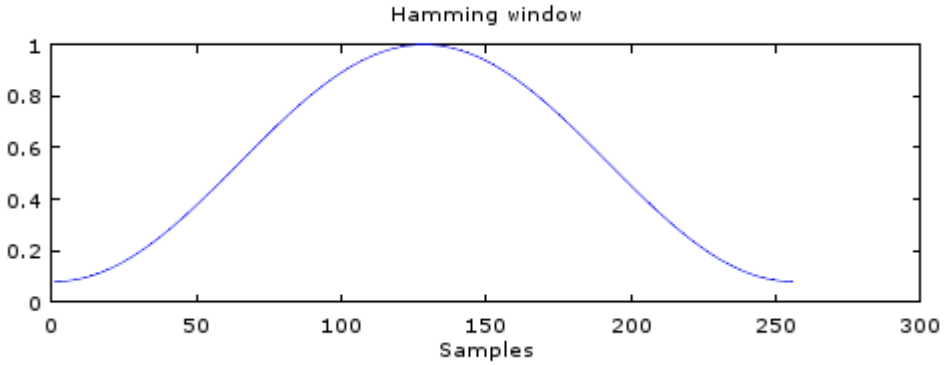
Table 3.4 Nilai setelah *windowing*

Frame ke-	Sampel ke-	Nilai setelah *windowing*
1	1	-0.000017
1	2	-0.000019
1	3	-0.000022
1	4	-0.000017

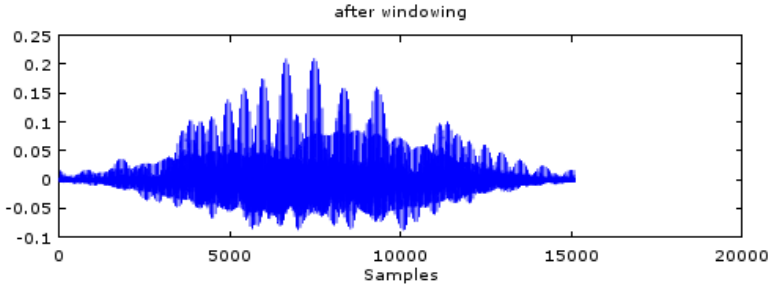
Frame ke-	Sampel ke-	Nilai setelah *windowing*
1	5	0.000017
…	…	…
1	254	-0.00019422
1	255	-0.00020054
1	256	-0.00019775



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.

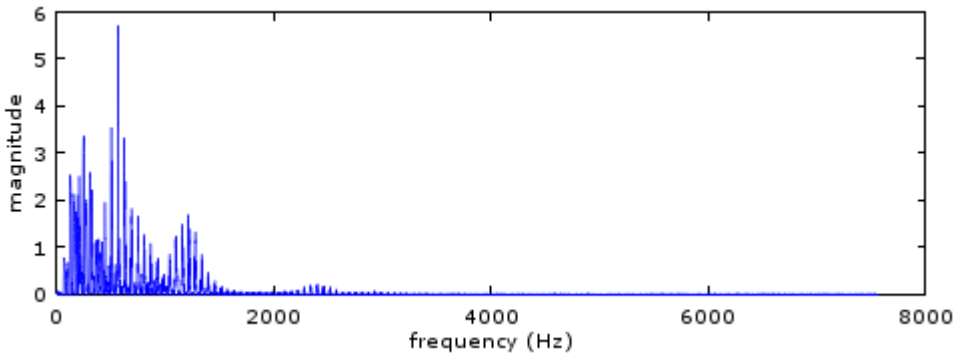
Fast Fourier Transform ini dilakukan pada setiap *frame* yang telah melalui proses *windowing*. Fast Fourier Transform dilakukan dengan membagi N buah titik pada transformasi diskrit menjadi 2, masing masing (N/2) titik transformasi. Proses memecah menjadi (N/4) dan seterusnya hingga diperoleh titik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

$$mel(300) = 1125\ \ln\left(1 + (300/700)\right)$$
$$= 401.26$$

$$mel(8000) = 1125\ \ln\left(1 + (8000/700)\right)$$
$$= 2835$$

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

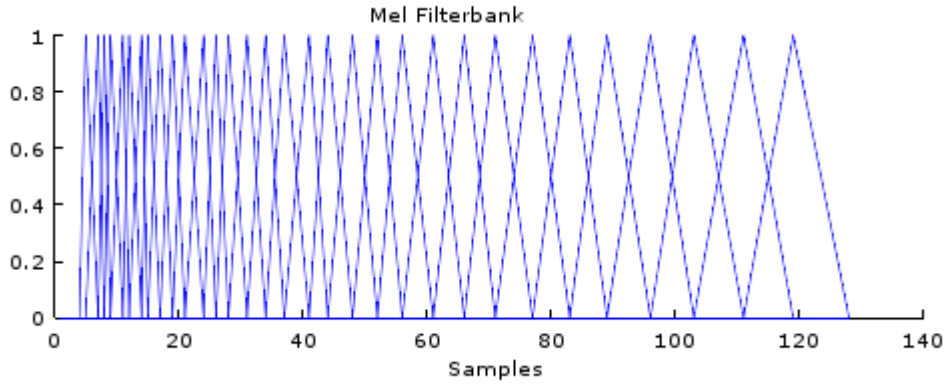
Mel ke-	Nilai Mel
1	401.26
2	475.01
3	475.01
4	622.51
5	696.26
…	…
30	2540
31	2613.7
32	2687.5
33	2761.2
34	2835

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

$$mel^{-1}(1) = 700\ (\exp(401.26/1125)) - 1)$$
$$= 300\ Hz$$

…

$$mel^{-1}(34) = 700\ (\exp(2835/1125)) - 1)$$
$$= 8000\ \text{Hz}$$



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:

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

…

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

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 *log energy***
1	1	0.028768
1	2	0.016435
1	3	0.0077781

frame ke-	*filter ke-*	Nilai *log energy*
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

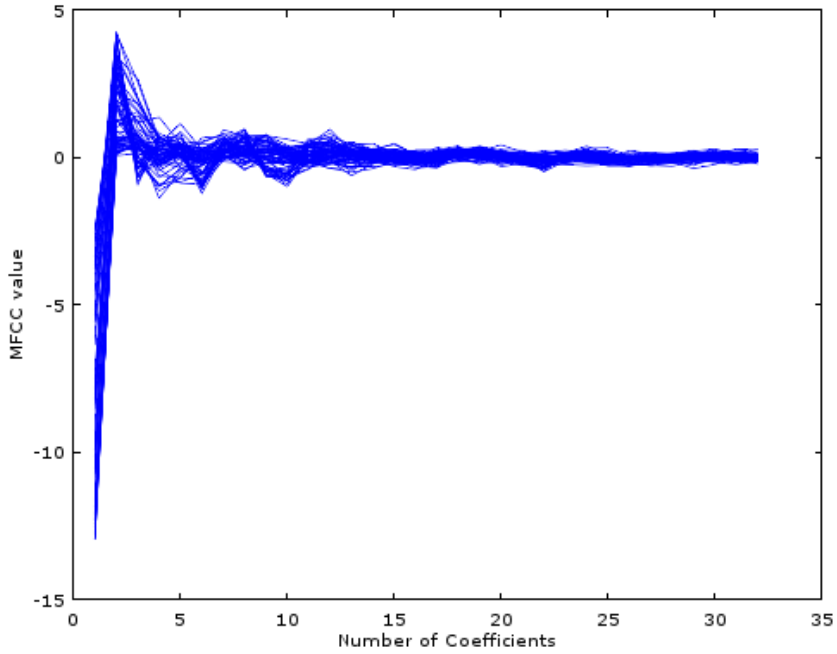
3.3.2.7 Discrete Cosine Transform

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

frame ke-	DCT *ke-*	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

frame ke-	DCT *ke-*	Nilai DCT
1	11	0.216490
1	12	0.262256
1	13	0.059163

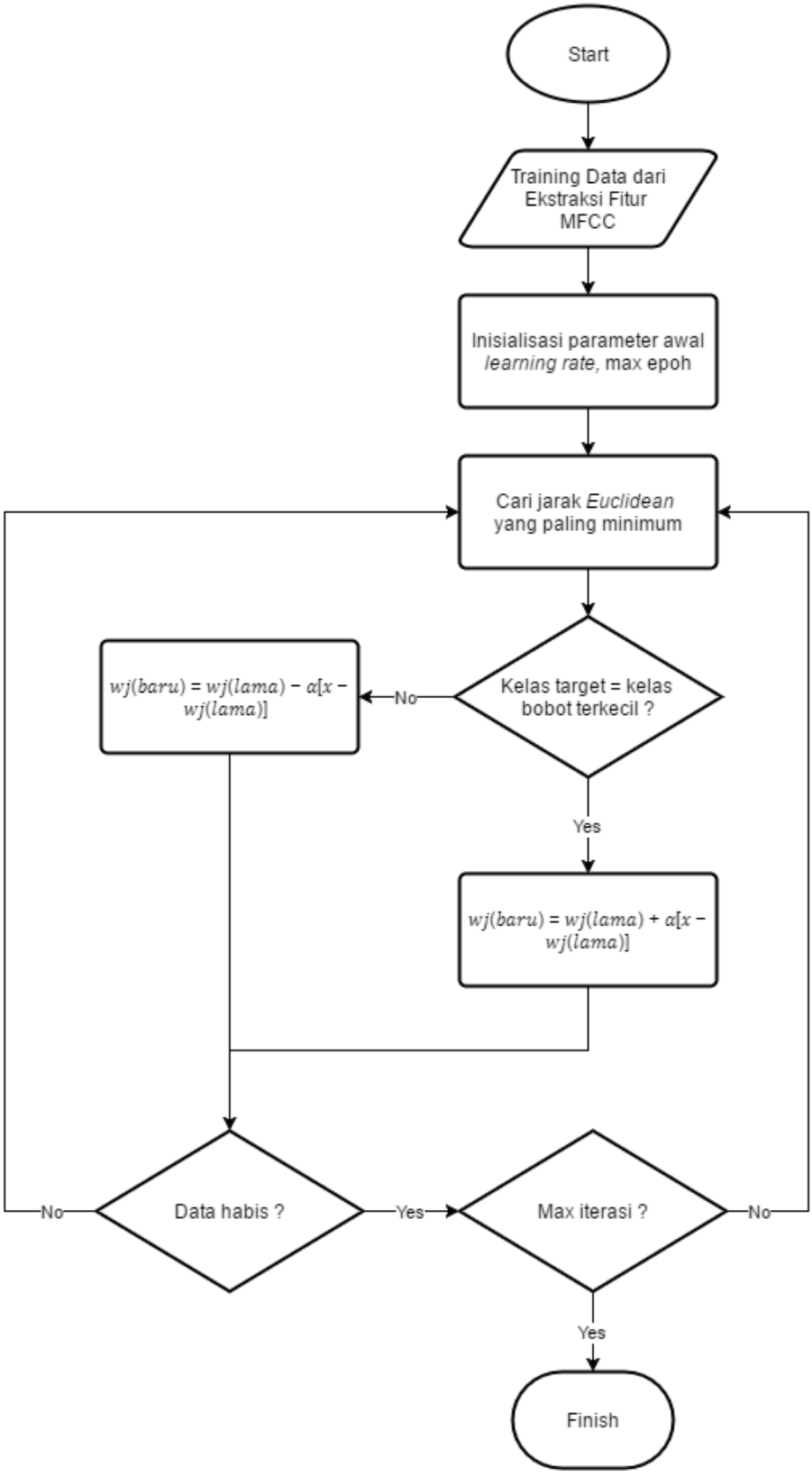


Gambar 3.10 Grafik MFCC

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 (α) sebesar 0.05 dengan pengurangan setiap kali iterasi 0.977 x α, 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_1**	**X_2**	**X_3**	**…**	**X_{13}**	**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_1	X_2	X_3	…	X_{13}	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_m) 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:

a. $||X - W_1|| = \sqrt{((-10.45) - (-12.554))^2 + \cdots + (0.09659 - 0.05916)^2}$

$= 2.4886$

b. $||X - W_2|| = \sqrt{((-10.45) - (-12.06))^2 + \cdots + (0.09659 - 0.49975)^2}$

$= 2.4228$

c. $||X - W_2|| = \sqrt{((-10.45) - (-12.235))^2 + \cdots + (0.09659 - 0.07273)^2}$

$= 2.1585$

d. $||X - W_2|| = \sqrt{((-10.45) - (-12.8))^2 + \cdots + (0.09659 - 0.10297)^2}$

$= 2.6255$

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:

$$\boldsymbol{W_{31} = W_{31} - \alpha * (X_{11} - W_{31})}$$
$$\boldsymbol{= -12.235 - 0.05\ (-10.45 - (-12.235))}$$
$$\boldsymbol{= -12.324}$$

...

$$\boldsymbol{W_{3\ldots} = W_{3\ldots} - \alpha * (X_{1\ldots} - W_{3\ldots})}$$
$$\boldsymbol{= 0.49975 - 0.05\ (0.09659 - 0.49975)}$$
$$\boldsymbol{= 0.51991}$$

Sehingga diperoleh bobot ketiga yang baru yaitu:

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

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* (α) 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_1	X_2	X_3	…	X_{13}
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.

a. $||X - W_1|| = \sqrt{((-12.061) - (-20.087))^2 + \cdots + (0.07847 - 0.09466)^2}$

$= 8.2155$

b. $||X - W_2|| = \sqrt{((-12.061) - (-19.296))^2 + \cdots + (0.07847 - 0.7996)^2}$

$= 7.4205$

c. $||X - W_3|| = \sqrt{((-12.061) - (-19.576))^2 + \cdots + (0.07847 - 0.11637)^2}$

$= 7.6278$

d. $||X - W_4|| = \sqrt{((-12.061) - (-20.48))^2 + \cdots + (0.07847 - 0.16475)^2}$

$= 8.5672$

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**
Jumlah hasil	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

Attribute: signal fs codec			
Method	**Parameter**	**Return**	**Deskripsi**
read_audio	audio	[signal, fs]	*Function* untuk membaca file audio.
add_temp	file	file	*Function* untuk membuat *temporary* file hasil dari silence removal.

Tabel 4.2 Class AudioPlayer

Attribute: volume_slider seek_slider lcd_number audio_play_button audio_pause_button audio_stop_button			
Method	**Parameter**	**Return**	**Deskripsi**
__init__	volume_slider seek_slider lcd_number audio_play_button audio_pause_button audio_stop_button	None	*Function* Constructor kelas AudioPlayer.

Method	Parameter	Return	Deskripsi
init_audio_player	Self	None	*Function* untuk inisialisasi awal parameter-parameter pemutar audio.
set_audio_source	audio_file	None	*Function* untuk menetapkan sumber *file* audio.
tick	time	None	*Function* untuk sinkronisasi waktu pada tampilan LCD Number.
state_change	newState, oldState	None	*Function* untuk menangkap *event state change* pada pemutar *audio*.

Tabel 4.3 class MFCC

Attribute: 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**
freq_high	fs	[fs / 2]	*Function* untuk menghitung batas atas frekuensi dari *frekuensi sampling*.

Method	Parameter	Return	Deskripsi
remove_silence	audio	[silenced_signal, silenced_fs]	*Function* untuk menghilangkan *silence* pada *audio*.
frame_blocking	audio_signal	[num_frames, framed]	*Function* untuk melakukan proses *frame blocking*.
hamm_window	framed_signal	windowed	*Function* untuk melakukan proses *windowing* dengan *hamming window*.
calc_fft	signal	fft_signal	*Function* untuk melakukan proses FFT.
DFT	signal	numpy.dot(M, signal)	*Function* untuk menghitung DFT menggunakan persamaan 2.4
FFT	signal	np.concat(signal)	*Function* untuk menghitung FFT.
mel	fs	mel	*Function* untuk menghitung dan menghasilkan nilai mel menggunakan persamaan 2.10
melinhz	mel	melinhz	*Function* untuk menghitung dan menghasilkan nilai mel dalam frekuensi menggunakan persamaan 2.11

Method	Parameter	Return	Deskripsi
calc_fft_bin	melinhz, fs	fft_bin	*Function* untuk meng-hitung nilai fft bin menggunakan persamaan 2.12
fbank	fft, fs	log_energy, fbank	*Function* untuk membuat *filterbank* dan meng-hitung nilai *log energy* menggunakan persamaan 2.13 dan 2.14
features	energy	dct(energy)	*Function* untuk meng-hitung nilai koefisien MFCC.
dct	energy	ceptstrum	*Function* untuk meng-hitung DCT menggunakan persamaan 2.15

Tabel 4.4 Class LVQ

Attribute: -			
Method	**Parameter**	**Return**	**Deskripsi**
init_ref_vector	None	Object	*Function* untuk mengambil vektor referensi awal dari data input.
init_data_set	ref_vectors	Object	*Function* untuk mengambil vektor data *input* selain dari vektor referensi awal.

Method	Parameter	Return	Deskripsi
eucl	x,y	float	*Function* untuk menghitung jarak Euclidean dari 2 buah vektor.
start_training	ref_vectors, data_set, max_epoh, alpha, alpha_decay	new_weight	*Function* untuk melakukan pelatihan data LVQ.
test_data	features	output_class	*Function* untuk melakukan pengujian data *feature* untuk mendapatkan kelas akhir klasifikasinya.

Tabel 4.5 class DatabaseConnector

Attribute: conn			
Method	**Parameter**	**Return**	**Deskripsi**
__init__	None	-	*Function* Constructor kelas DatabaseConnector
insert_features	args	id	*Function* untuk meng-*input features* dan kelas *output* ke dalam *database*.
insert_weight	args	id	*Function* untuk meng-*input* bobot akhir setelah pelatihan data ke dalam *database*.
adapt_array	arr	sql.Binary	*Function* untuk meng-konversi numpy array ke dalam sql.Binary

Method	Parameter	Return	Deskripsi
convert_array	bin	numpy.array	*Function* untuk meng-konversi sql.Binary 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.

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* dikurangin dengan ukuran *frame* dibagi dengan setengah dari jumlah sampel sinyal pada satu *frame*.
2. Inisialisasi array dengan dimensi jumlah *frame* x ukuran 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*.

4.2.3 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 list 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 list *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 tempat 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 array 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 array sinyal fft untuk setiap frame-nya.

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. Lakukan looping untuk setiap jumlah frame dan jumlah sampel dalam frame dibagi 2, untuk membentuk *filterbank* dengan menggunakan kondisi pada persamaan 2.13
5. Hitung nilai *log energy* dengan mengalikan nilai-nilai magnitude yang dihasilkan dari proses FFT dengan nilai-nilai *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.

4.2.7 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

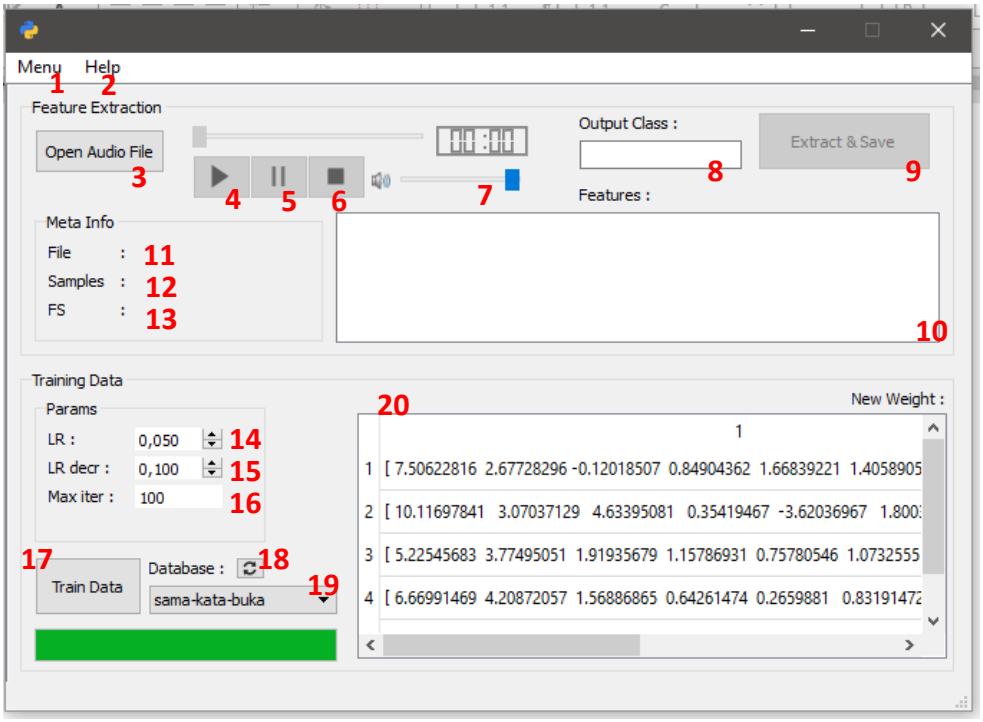
kelas akhir dari data suara baru. Tahap – tahap implementasi Learning Vector Quantization adalah sebagai berikut:

1. Proses Pelatihan Data:
    a. Tentukan vektor bobot awal, diambil satu data random dari data *input* setiap kelas.
    b. Ambil vektor dataset yang bukan merupakan vektor bobot awal yang ditentukan pada tahap sebelumnya.
    c. Lakukan looping selama jumlah maximum iterasi belum terpenuhi, pada setiap iterasi, lakukan looping sejumlah dataset *input*.
    d. Hitung nilai jarak Euclidean dari bobot vektor awal dan data *input* dengan menggunakan persamaan 2.21
    e. 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
    f. Pada setiap iterasi, kurangi nilai *learning rate* sejumlah nilai parameter yang ditentukan.
    g. 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 perhitugan 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.

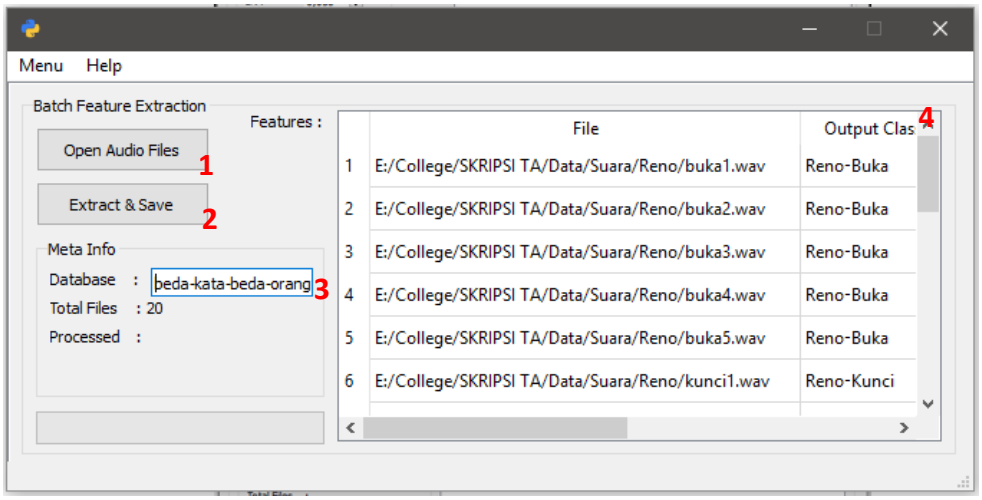


Gambar 4.1 Antarmuka aplikasi untuk pelatihan data

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*.

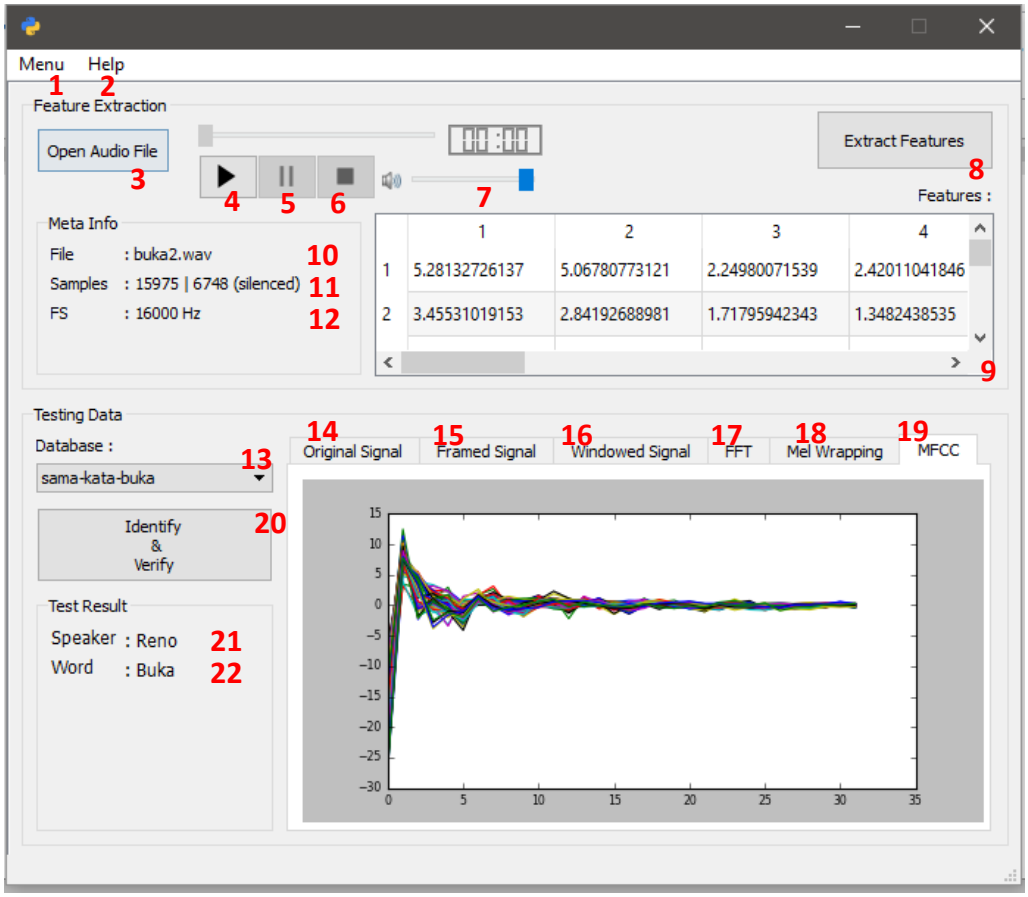
10. Table hasil ekstraksi fitur.
11. Informasi nama *audio file*.
12. Informasi jumlah sampel pada *audio file*.
13. Informasi *Frequency Sampling* pada *audio file*.
14. *Double spinbox* untuk mengganti nilai parameter *Learning Rate*.
15. *Double spinbox* untuk mengganti nilai parameter pengurangan *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.

4.4.1 Pengujian Dengan Kata Berbeda

Pada pengujian ini, terdapat 4 kata berbeda yang diucapkan untuk data latih dan uji. Data latih yang digunakan untuk setiap kata adalah 5 buah, untuk pembicaranya sendiri, terdapat 5 orang pembicara, sehingga didapatkan 20 buah suara untuk satu orang pembicara.

Sedangkan data ujinya, di ambil dari data terpisah yang sama sekali bukan data latih. Pada percobaan ini penulis melakukan 3 kali pelatihan data untuk mendapatkan 3 bobot akhir yang berbeda.

Pada pelatihan ke-1 parameter *Learning Rate* yang digunakan adalah 0.2, *Learning Rate Decay* adalah 0.1 dan *Maximum Epoh* adalah 100. Akurasi yang didapatkan dari pengujian dengan pelatihan ke-1 adalah 40% (dari total 20 data uji, terdapat 8 data uji yang teridentifikasi dengan benar).

Tabel 4.6 Bobot akhir pelatihan ke-1

Bobot Akhir	Kelas
[8.63028289 4.841472 … 0.39847302 0.55291497]	Hulda-Buka

Bobot Akhir	Kelas
[2.02227196 3.17382925 … -1.65402711 1.09977879]	Hulda-Kunci
[8.89854922 2.40641288 … -0.15401663 -0.04253715]	Hulda-Lock
[9.25424757 1.68088612 … -0.24583328 -0.15486346]	Hulda-Unlock
[9.04696148 3.2537203 … 0.90179179 0.13144657]	Jeremy-Buka
[-1.47829739 0.95501046 … -0.59574961 -0.18520316]	Jeremy-Kunci
[8.39233811 3.4030389 … -0.57296106 1.58681872]	Jeremy-Lock
[9.47668497 -0.77107639 … 0.42083041 -0.90743801]	Jeremy-Unlock
[6.77213962 -0.23793197 … -0.45510832 -0.19442151]	Reno-Buka
[6.91848108 4.57864334 … 0.46757498 0.34370023]	Reno-Kunci
[9.63143105 1.87307437 … -0.49800077 -0.28081273]	Reno-Lock
[9.35613503 5.58733726 … 2.68769036 5.19910091]	Reno-Unlock
[6.34739219 0.72058586 … 0.72317123 1.42381377]	Riana-Buka
[6.01671717 4.41989241 … -0.63084828 -0.62228372]	Riana-Kunci
[6.36786419 1.57238906 … -6.52003005 2.54428280]	Riana-Lock
[7.54966032 -0.93785496 … -0.32084432 -0.53231088]	Riana-Unlock
[4.60854177 0.57698191 … 0.04430529 -0.53585502]	Rifky-Buka
[1.60116666 2.40669639 … -6.20795451 -4.03477484]	Rifky-Kunci
[5.81439128 3.23660446 … 0.2300168 -0.06537708]	Rifky-Lock
[5.34966244 3.84089734 … -0.61600214 -0.65322604]	Rifky-Unlock

Tabel 4.7 Hasil Uji pelatihan ke-1

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

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

Pada pelatihan ke-2 parameter *Learning Rate* yang digunakan adalah 0.05, *Learning Rate Decay* adalah 0.097 dan *Maximum Epoh* adalah 100. Akurasi yang didapatkan dari pengujian dengan pelatihan ke-2 adalah 60% (dari total 20 data uji, terdapat 12 data uji yang teridentifikasi dengan benar).

Tabel 4.8 Bobot akhir pelatihan ke-2

Bobot Akhir	Kelas
[8.04061971 -1.07149743 … -0.91964407 1.45645593]	Hulda-Buka
[0.778622 3.40990689 … -1.86640863 1.65736337]	Hulda-Kunci
[8.59611925 1.38604826 … -0.22516135 0.20121064]	Hulda-Lock
[6.11945245 1.7415953 … 0.17716473 0.64686684]	Hulda-Unlock
[4.27371099 3.36798508 … 0.37061779 0.20087292]	Jeremy-Buka

Bobot Akhir	Kelas
[7.59357298 -0.83033471 … -0.77891964 -0.87868353]	Jeremy-Kunci
[7.58954619 2.71928204 … -0.07882085 0.4717338]	Jeremy-Lock
[10.41144177 -1.79648792 … -0.21137828 -0.71232375]	Jeremy-Unlock
[10.7303445 -1.02535408 … 0.18368819 0.07391555]	Reno-Buka
[5.95131505 -7.16593480 … -8.08258166 9.62258368]	Reno-Kunci
[-1.26936212 1.87307437 … -7.10587591 2.51831464]	Reno-Lock
[6.51905385 3.58723954 … -0.09836992 0.13564988]	Reno-Unlock
[5.28217774 5.09195419 … -0.00583131 0.09389015]	Riana-Buka
[5.5315545 3.87283717 … 0.19166328 -0.02597601]	Riana-Kunci
[6.04931207 2.7561733 … -0.45567543 -0.03230751]	Riana-Lock
[5.68156466 -0.56652186 … 0.30652316 -0.9298022]	Riana-Unlock
[1.31333202 2.19861974 … -0.38779215 -0.38002437]	Rifky-Buka
[0.87487089 1.15232731 … 0.11432986 -0.07742834]	Rifky-Kunci
[7.10304693 3.66838321 … 0.08801771 0.66401709]	Rifky-Lock
[4.74517446 3.52553202 … 0.04205641 -0.61596216]	Rifky-Unlock

Tabel 4.9 Hasil pengujian pelatihan ke-2

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

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

Pada pelatihan ke-3 parameter *Learning Rate* yang digunakan adalah 0.05, *Learning Rate Decay* adalah 0.1 dan *Maximum Epoh* adalah 100. Akurasi yang didapatkan dari pengujian dengan pelatihan ke-3 adalah 40% (dari total 20 data uji, terdapat 8 data uji yang teridentifikasi dengan benar).

Tabel 4.10 Bobot akhir pelatihan ke-3

Bobot Akhir	Kelas
[8.04061971 -1.07149743 … -0.91964407 1.45645593]	Hulda-Buka
[0.778622 3.40990689 … -1.86640863 1.65736337]	Hulda-Kunci
[8.59611925 1.38604826 … -0.22516135 0.20121064]	Hulda-Lock
[6.11945245 1.7415953 … 0.17716473 0.64686684]	Hulda-Unlock
[4.27371099 3.36798508 … 0.37061779 0.20087292]	Jeremy-Buka
[7.59357298 -0.83033471 … -0.77891964 -0.87868353]	Jeremy-Kunci
[7.58954619 2.71928204 … -0.07882085 0.4717338]	Jeremy-Lock

Bobot Akhir	Kelas
[10.41144177 -1.79648792 … -0.21137828 -0.71232375]	Jeremy-Unlock
[10.7303445 -1.02535408 … 0.18368819 0.07391555]	Reno-Buka
[5.95131505 -7.16593480 … -8.08258166 9.62258368]	Reno-Kunci
[-1.26936212 1.87307437 … -7.10587591 2.51831464]	Reno-Lock
[6.51905385 3.58723954 … -0.09836992 0.13564988]	Reno-Unlock
[5.28217774 5.09195419 … -0.00583131 0.09389015]	Riana-Buka
[5.5315545 3.87283717 … 0.19166328 -0.02597601]	Riana-Kunci
[6.04931207 2.7561733 … -0.45567543 -0.03230751]	Riana-Lock
[5.68156466 -0.56652186 … 0.30652316 -0.9298022]	Riana-Unlock
[1.31333202 2.19861974 … -0.38779215 -0.38002437]	Rifky-Buka
[0.87487089 1.15232731 … 0.11432986 -0.07742834]	Rifky-Kunci
[7.10304693 3.66838321 … 0.08801771 0.66401709]	Rifky-Lock
[4.74517446 3.52553202 … 0.04205641 -0.61596216]	Rifky-Unlock

Tabel 4.11 Hasil pengujian pelatihan ke-3

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

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

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. Agustus 2013. “Vulnerability evaluation of speake rverification under voice conversion spoofing: the effect of text constraints”. ISCA