BAB 2

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Iman Hermadi et al (2010) dalam penelitiannya menggunakan *Self Organizing Maps* untuk *clustering* data dari pelamar program sarjana di IPB meliputi data akademik, data penilaian serta data pribadi. Dari penelitian tersebut disimpulkan bahwa peluang untuk diterima dari masing-masing kategori bergantung kepada nilai dan juga kategori SMA juga berkontribusi terhadap diterima atau tidaknya pelamar.

Ardi Pujiyanta (2008) dalam penelitiannya membandingkan antara *Self Organizing Maps* dengan *Boltzmannn Machine* untuk mencari jalur terpendek, menyimpulkan bahwa Jaringan Syaraf Tiruan Boltzmann Machine memberikan jalur dengan jarak yang lebih optimal daripada *Self Organizing Maps.* Dan juga *Boltzmann Machine* memerlukan waktu yang lebih sedikit dalam pencarian jalur terpendek.

Shekar (2009) menggunakan *Self Organizing Maps* untuk mengklasifikasikan dokumen dan menyimpulkan beberapa hal, dengan lebih banyak dokumen maka *mapping time* akan berkurang. Tingkat akurasi dari pengetesan data akan meningkat jika jumlah dokumen semakin banyak.

Asmadin et al (2012) menggunakan *Self Organizing Maps* untuk mengelompokan habitat dasar perairan dangkal berbasis data Satelit *QuickBird.* Dalam penelitian tersebut menyimpulkan bahwa *Self Organizing Maps* dapat mengklasifikasi citra *Quickbird* dari berbagai sumber kanal, dan juga menunjukan hasil yang relatif baik.

Defit (2010) melakukan penelitian untuk penambangan informasi tersembunyi dari database tidak terstruktur menggunakan *Self Organizing Maps*

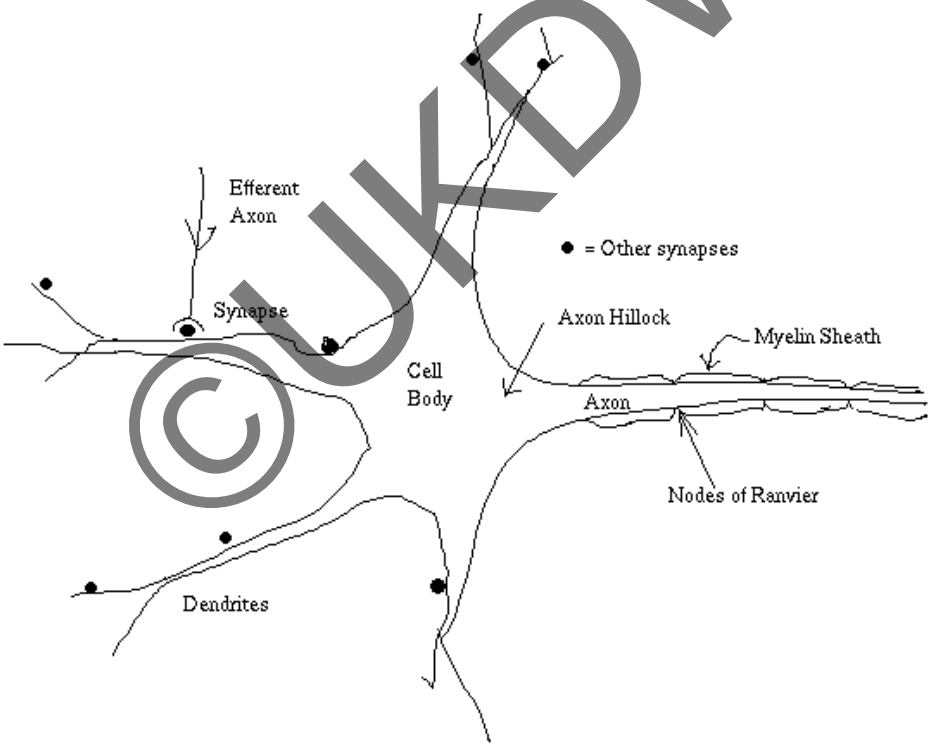
(SOM). Dalam penelitiaannya *Self Organizing Maps* sangat baik dalam mengelompokkan dokumen-dokumen.

Dalam penelitian ini saya akan menggunakan metode *Self Organizing Maps* untuk mengklasifikasikan suara manusia kedalam enam jenis suara yang ada yaitu *Sopran, Mezzosopran, Alto, Tenor, Bariton, dan Bass.*

2.2 Landasan Teori

2.2.1 Jaringan Syaraf Tiruan (*Artificial Neural Network)*

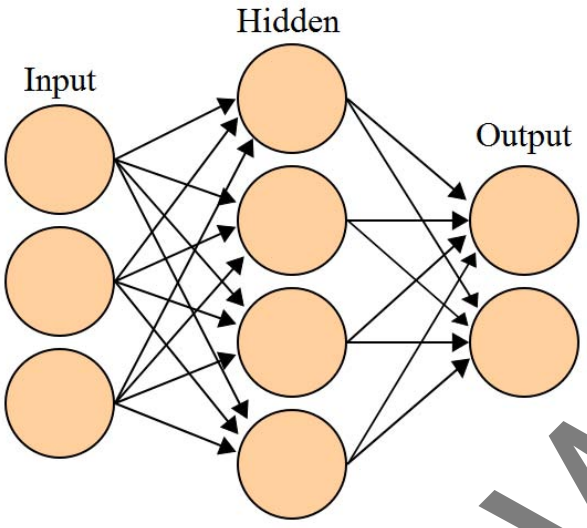
[1]Jaringan syaraf tiruan merupakan suatu sistem pengolahan informasi yang memiliki karakteristik yang sama dengan jaringan syaraf manusia.



Gambar 2.1 Susunan Syaraf Manusia

[1] Fausett L, Fundamental of Neural Networks, Hal. 3

Di dalam jaringan syaraf tiruan, setiap *neuron* mempunyai prinsip kerja yang sama. Setiap *neuron* selalu menerima *input* dan selalu menghasilkan *output*.



Gambar **2.2 Jaringan Syaraf Tiruan**

2.2.2 Self Organizing Maps

Menurut Fusett (1994) *Self Organizing Map* merupakan salah satu model Jaringan Syaraf Tiruan yang menggunakan metode unsupervised dibekali dengan pengetahuan dasar (parameter-parameter jaringan). Proses pelatihan SOM adalah proses pencocokan bobot setiap *neuron output* yang ada.

Algoritma SOM :

Step 0. Inisialisasi bobot (bisa random)

Tentukan topologi serta parameter

Tentukan Learning rate

Step 1. Bila kondisi terpenuhi lakukan step 2-8

Step 2. Untuk setiap input vector lakukan step 3-5

Step 3. Untuk setiap j hitung jarak dengan rumus

$$D(j) = \sum_i (w_{ij} - x_i)^2$$

Step 4. Pilih index J dengan D(J) paling kecil

Step 5. Update bobot j dan tetangganya dengan

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$$

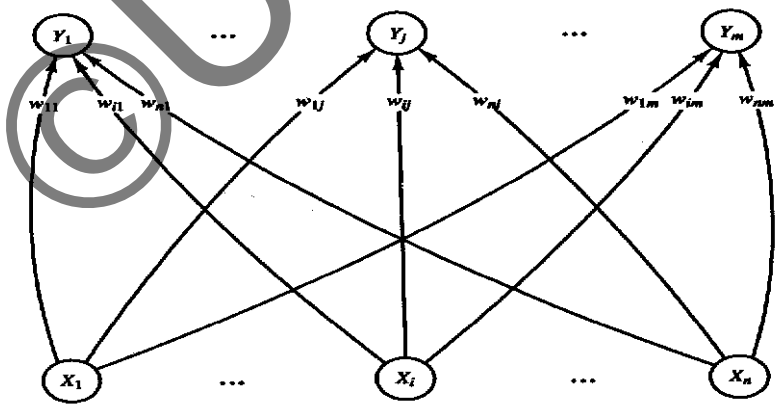
Step 6. Update Learning Rate

Step 7. Kurangi radius tetangga topologi

Step 8. Berhenti

Arsitektur SOM

Jaringan ini terdiri dari dua lapisan (*layer*), yaitu lapisan *input* dan *output.* Setiap *neuron* dalam lapisan input terhubung dengan setiap *neuron* pada lapisan *output.* Setiap *neuron* dalam lapisan output merepresentasikan kelas (*cluster*) dari input yang diberikan.



Gambar 2.3 Arsitektur SOM

Dikutip dari : Fausett, Laurence (1994). *Fundamental of Neural Networks, Architectures, Algorithms, and Applications*, hlm 170

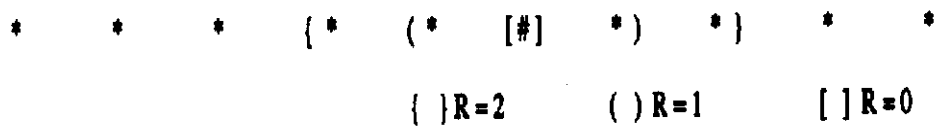
Keterangan Gambar 2.1 :

$X_{1\text{-}n}$: neuron *input*

Y_{1-n} : neuron *output*

W_{ij} : bobot hubungan neuron *input* dan neuron *output.*

Self Organizing Map memiliki 3(tiga) topologi yaitu topologi 1 dimensi, 2 dimensi dan hexagonal. Dalam penelitian kali ini digunakan topologi 1 dimensi, yang bisa dilihat pada Gambar 2.4.



Gambar 2.4 Topologi 1 dimensi

Dikutip dari : Fausett, Laurence (1994). Fundamental of Neural Networks, Architectures, Algorithms, and Applications, hlm 170.

2.2.3. WAV (Waveform Audio Format)

Wave (.WAV)* merupakan *file audio* yang tidak terkompresi dan berukuran besar karena seluruh sampel audio disimpan semuanya di media penyimpanan dalam bentuk digital. Format *wave* merupakan format kasar (*raw format)*. Format dasarnya dikenal dengan nama *Pulse Code Modulation* (PCM).

2.2.4. Fast Fourier Transform (FFT)

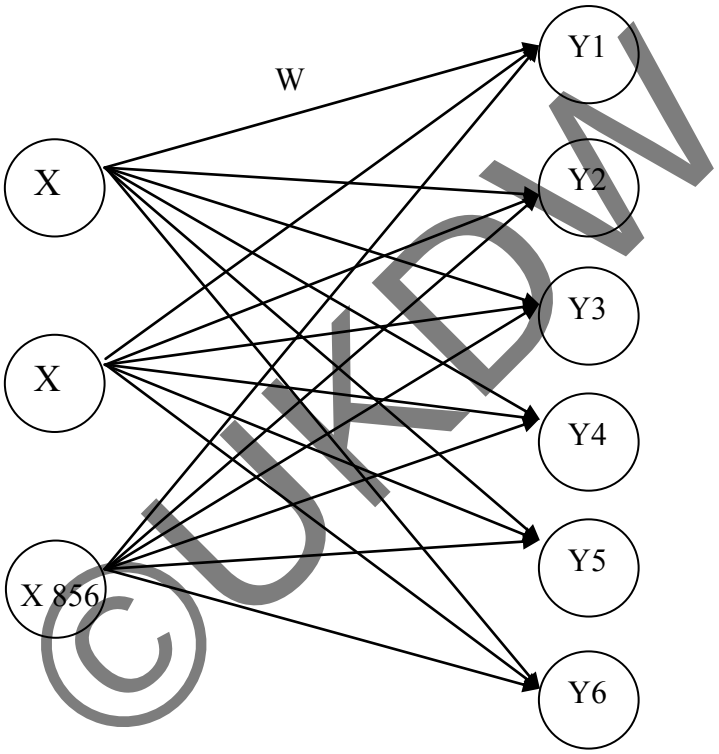
FFT adalah suatu algoritma untuk menghitung *Discrete Fourier Transform* (DFT) dengan lebih cepat dan efisien, dimana algoritma ini memiliki fungsi untuk mengubah sinyal berbasis waktu menjadi berbasis frekuensi.

2.2.5. Implementasi Self Organizing Map

Dalam penelitian ini akan diterapkan metode *Self Organizing Map* untuk mengklasifikasikan suara manusia melalui *file wav* sebagai data masukan dalam sistem. Kasus ini akan menerapkan topologi 1 dimensi, akan ada 6 *Cluster* yang

disediakan dan merupakan representasi dari jumlah jenis suara manusia yang ada yaitu *Sopran, Mezzo sopran, Alto, Tenor, Bariton,* dan *Bass*.

Untuk mendapatkan *input* yang tepat bagi *Self Organizing Map,* maka sebelumnya data rekaman suara akan melalui proses awal, antara lain *sampling* untuk mendapatkan sampel suara berupa amplitude, kemudian dilakukan proses *preprocessing* untuk mendapatkan sinyal suara yang lebih baik dan tahap terkhir adalah FFT yang akan merubah data sampel yang berupa amplitudo menjadi frekuensi. Pada Gambar 2.5. dibawah ini menunjukan arsitektur dari penelitian ini.



Gambar 2.5. Arsitektur Sistem

Keterangan :

X1,X2…..X856 : Input data yang diproses dalam SOM

W : bobot setiap cluster yang ada

Y1,Y2,Y3,Y4,Y5,Y6 : output

BAB 3

PERANCANGAN SISTEM

3.1. Spesifikasi Sistem

3.1.1. Perangkat Keras

Pembuatan sistem ini menggunakan spesifikasi sebagai berikut :

a. Processor Intel Core 2 Duo CPU T5750 2.00GHZ

b. RAM 1GB DDR2

c. 120 GB HDD

d. Realtek High Definition Audio

3.1.2. Perangkat Lunak

Dalam pembuatan sistem ini dibutuhkan perangkat lunak sebagai berikut:

a. Sistem Operasi Windows 7 Professional 32 bit

b. Borland Delphi 7

c. Microsoft Access 2007

3.2. Perancangan Program

Dalam pembuatan program ini penulis menggunakan bahasa pemrograman Borland Delphi 7. Dalam Delphi juga sudah tersedia komponen atau *plugin* gratis dalam mendukung sistem yang akan penulis buat, komponen tersebut adalah *Wave audio package v1.09.* Sementara untuk suara yang dipakai akan berformat

Wav dengan *sample rate* 44.100 Hz, dan durasi perekaman sekitar 10 detik. Kemudian akan melalui proses *sampling* dengan mengambil nilai amplitudo. Setelah mendapatkan nilai amplitudo setiap sampel akan melalui tahap *preprocessing* yaitu *pre-emphasis, Frame blocking,* dan *windowing* dengan *Hamming window* kemudian akan melalui tahap kuatisasi untuk mendapatkan nilai frekuensinya dengan *Fast Fourier Transform* (FFT) dalam penelitian ini fungsi digunakan komponen Delphi yaitu DSPLab yang bisa di *download,* dan pada komponen ini fungsi dari *hamming window dan* FFT sudah dihandle secara langsung. Kemudian dalam tahap klasifikasi akan digunakan *Self Organizing Map* (SOM). Dimana dalam tahap ini akan di kelompokan ke dalam 6 (enam) cluster, yang nantinya akan menentukan 6 (enam) kelompok jenis suara yaitu *Sopran, Mezzo sopran, Alto, Tenor, Bariton, dan Bass.*

3.2.1. Pre-Emphasis

Proses ini adalah tahap awal setelah akuisi data atau proses *sampling* terhadap *input.* Tujuan dari proses ini untuk memperbaiki sinya dari gangguan *noise.* Dalam hal ini maka tingkat akurasi akan meningkat. Rumus yang didapat dari Rabinet (1993) dan digunakan dalam sistem ini adalah sebagai berikut :

$$Y(n) = S(n) - \alpha S(n-1) \qquad [3.1]$$

Keterangan :

Y(n) = Hasil *pre-emphasis* ke-n dari sinyal.

S(n) = input sinyal ke-n.

α = alpha dimana nilai *default* yang digunakan adalah 0.97.

n = jumlah data sinyal *input.*

3.2.2. Frame Blocking

Frame Blocking adalah proses yang akan diterapkan pada sinyal atau data suara yang sebelumnya sudah dilakukan proses *preemphasis.* Dalam proses ini sinyal atau data akan dibagi menjadi beberapa potongan untuk memudahkan proses selanjutnya, potongan tersebut disebut *frame.* Setiap *frame* memiliki beberapa sampel tergantung berapa detik dan besar frekuensi samplingnya. Dalam kasus ini akan digunakan *1024* data untuk masing-masing *frame.*

3.2.3. Windowing

Dalam proses sebelumnya yaitu *frame blocking* akan menghasilkan efek sinyal *discontinue.* Oleh sebab itu perlu untuk menjadikan sampel suara tersebut menjadi *continue,* digunakanlah proses windowing. Jenis *window* yang akan digunakan adalah *Hamming Window*. Rumus yang didapat dari Rabinet (1993) dan digunakan dalam penelitian ini adalah:

$$w(n) = 0.54-0.46*\cos(2*\pi*n/(N-1)) \quad [3.2]$$

Keterangan :

w(n) : hasil ke-n dari *hamming window.*

n : sampel ke-n.

N : jumlah sampel tiap *frame.*

3.2.4. Fast Fourier Transform (FFT)

Sampel yang dihasilkan dari proses-proses sebelumnya adalah spectrum suara dalam domain waktu, sehingga harus dirubah menjadi sinyal frekuensi dengan menggunakan *Fast Fourier Transform.* Hasil dari proses FFT merupakan

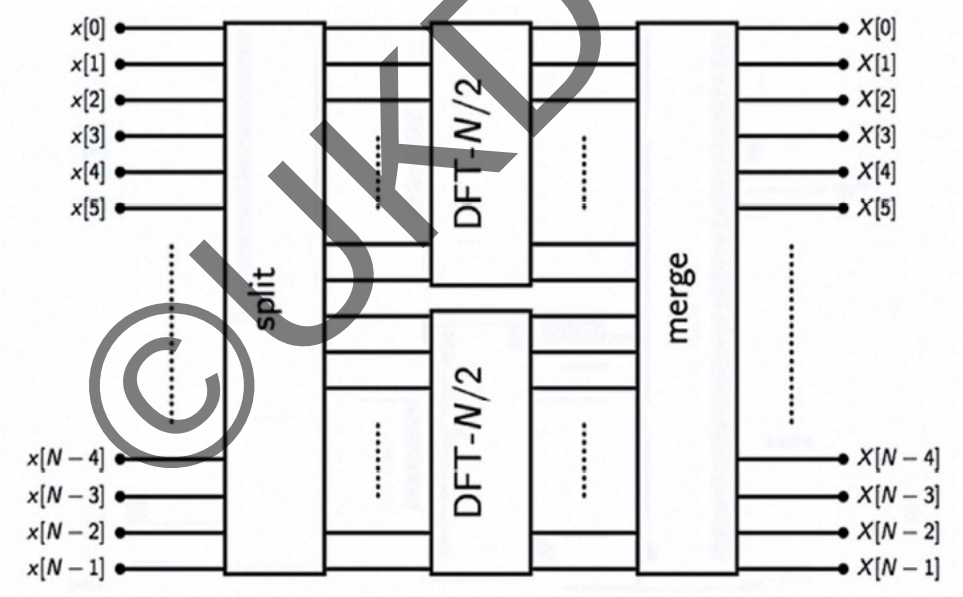
pendeteksian gelombang frekuensi domain dalam bentuk diskrit. Rumus FFT didapat dari Prandoni (2008) adalah sebagai berikut :

$$X[k] = \sum_{n=0}^{N-1} X[n] W_N^{nk} \qquad k = 0, \dots, N-1 \qquad [3.3]$$

dengan

$$W = e^{-\frac{j2\pi}{N}} \qquad [3.4]$$

Fast Fourier Transform merupakan proses cepat perhitungan *Discrete Fourier Transform* (DFT), [2]dengan mengubah bentuk DFT yang memiliki panjang *N* menjadi penjumlahan dari dua buah bentuk DFT dengan panjang masing-masing *N/2,* satu bagian terdiri dari elemen bernomor ganjil dan yang lain genap . FFT memiliki contoh skema seperti Gambar 3.1. dibawah ini:



Gambar 3.1. skema FFT

Dikutip dari : Chrisantyo, L. Pengolahan Sinyal Digital.

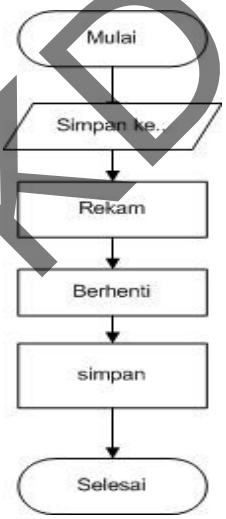
[2] http://anggieprasetiyo.blogspot.com/2013/05/cooley-dan-tukey.html

3.3. Flowchart

Dalam Flowchart ini dijelaskan proses yang ada dalam program atau sistem ini. Dan juga dijelaskan bahwa ketika belum dilakukan pelatihan maka pengujian tidak bisa dilakukan.

3.3.1. Flowchart Rekam Suara

Dalam Rekam Suara ini ditunjukan alur dari proses perekaman suara dan menyimpannya. Pada Rekam suara ini digunakan komponen yang sudah disediakan gratis seperti yang telah sebelumnya dijelaskan. Flowchart Rekam Suara dapat dilihat pada Gambar 3.2.

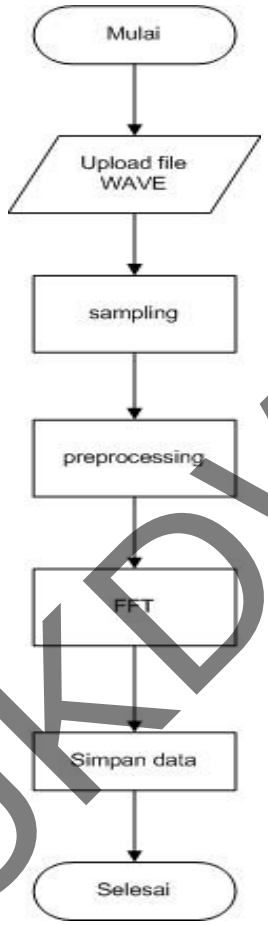


Gambar 3.2. Flowchart Rekam Suara

3.3.2. Flowchart Preprocessing

Disini digambarkan proses akuisi data untuk mendapatkan *input* yang nanti akan digunakan dalam proses Klasifikasi. Dalam kasus ini akan dilakukan proses *preprocessing* dahulu dengan proses *pre-emphasis, Frame blocking, Hamming Window* dan kemudian baru diproses dengan FFT untuk mendapatkan

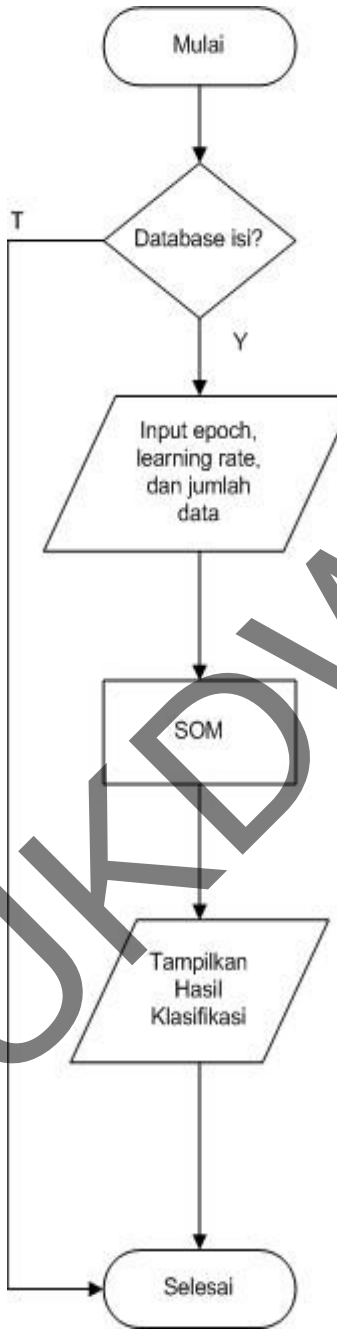
rata-rata *magnitude* sebagai nantinya untuk input proses klasifikasi. Flowchart proses *Preprocessing* dapat dilihat pada Gambar 3.3.



Gambar 3.3. Flowchart Preprocessing

3.3.3. Flowchart Klasifikasi

Dalam *flowchart* ini digambarkan proses Klasifikasi menggunakan Jaringan Syaraf Tiruan *Self Organizing Map*. Proses klasifikasi ini akan menyediakan input *epoch, learning rate,* dan jumlah data. Flowchart dapa dilihat pada Gambar 3.4.



Gambar 3.4. Flowchart Klasifikasi

3.4. Rancangan Database

Pada penelitian ini akan menggunakan database Microsoft Access. Tabel yang akan digunakan ada 1 (satu). Tabel tersebut adalah tabel suara.

Tabel 3.1. Tabel Suara

Nama	Tipe data
ID	Autonumber
Nama	Text
Bit	Number
Sample Rate	Text
Channel	Number
Data FFT	Memo
Cluster	Number

Tabel 3.1. memuat informasi tentang *file wave.* Berikut adalah keterangan untuk table pertama.

1. ID

digunakan untuk membedakan tiap data, tipe data yang digunakan autonumber supaya tidak ada yang sama.

2. Nama

Akan mengambil Nama tiap file suara yang di inputkan dan dimasukan ke dalam table.

3. Bit

Pada field ini akan menyimpan nilai bit dari file *wave* yang diinputkan.

4. Sample Rate

Field ini akan menyimpan nilai berupa *Sample Rate* dari file suara.

5. Channel

Field ini berisi *Channel* dari file *wave* yang hanya berisi nilai 1 atau 2 dan menunjukan *Mono* atau Stereo.

6. Data FFT

Field ini berisi nilai *sample* berupa rata-rata *magnitude* yang didapatkan dari proses *Fast Fourier Transform* (FFT).

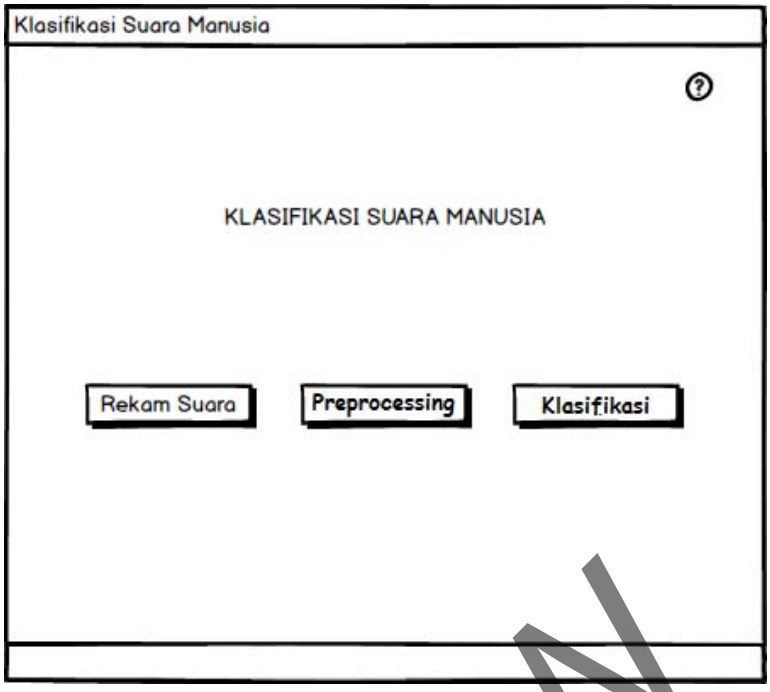
7. Cluster

Field ini berisi nomor *cluster* yang akan diperoleh setelah tahap klasifikasi selesai dilakukan. Fungsi dari field ini untuk menunjukan data yang bersangkutan masuk kedalam kelompok mana.

3.5 Rancangan Tampilan

3.5.1 Form Awal Program

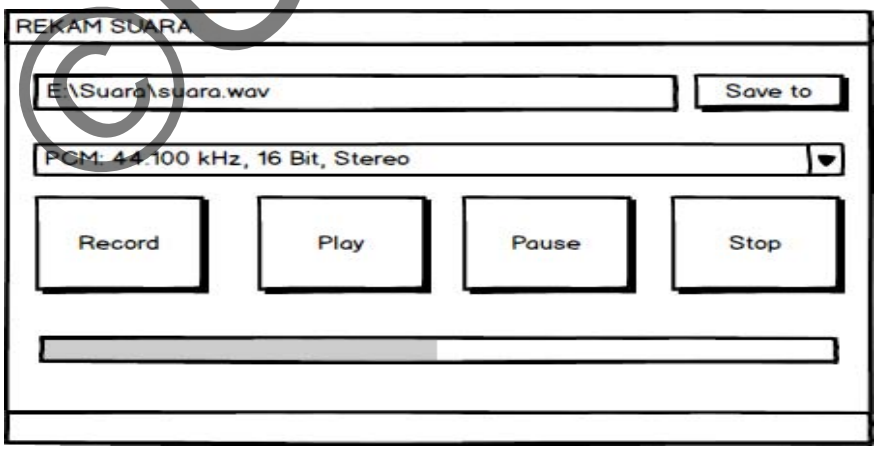
Form awal adalah tampilan awal program ini. Dan akan ada tiga (3) pilihan Rekam suara, Latih, atau Pengujian. Selain itu ada tombol *Help* untuk bantuan atau panduan dalam menggunakan program. Form Utama program bisa dilihat pada Gambar 3.5.



Gambar 3.5 Form Utama Program

3.5.2 Form Rekam Suara

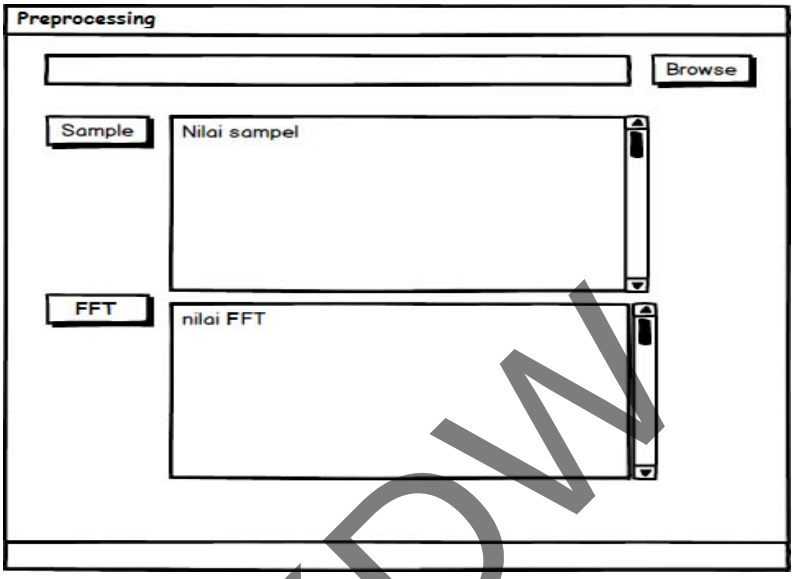
Pada form ini ditampilkan fungsi-fungsi untuk proses perekaman suara, yang komponennya didapat dari *Wave Audio Package v1.09.II.* Form perekaman dapat dilihat pada Gambar 3.6.



Gambar 3.6. Form Rekam Suara

3.5.3 Form Preprocessing

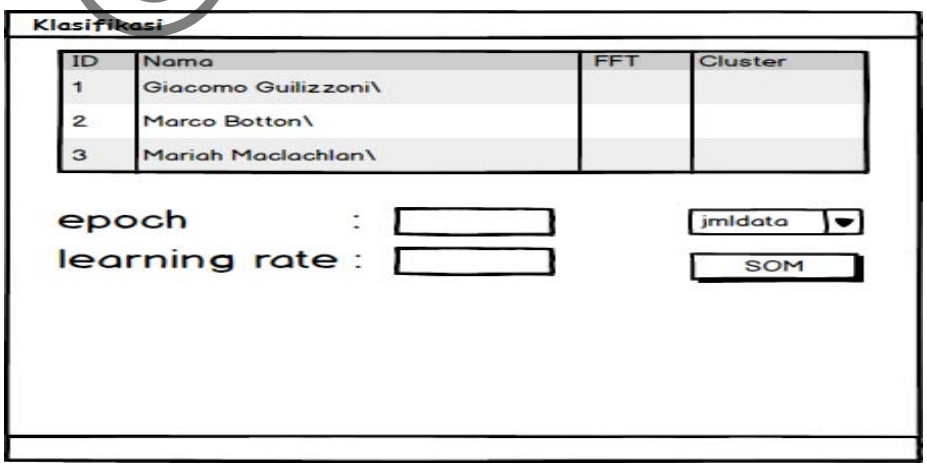
Form ini difungsikan untuk melakukan proses awal untuk mendapatkan *input* yang diperlukan untuk proses klasifikasi. Proses *preprocessing* dapat dilihat di Gambar 3.7.



Gambar 3.7. Form Preprocessing

3.5.4 Form Klasifikasi

Form ini difungsikan untuk melakukan proses pelatihan suara input agar diperoleh bobot yang nanti akan digunakan dalam proses pengujian. Dalam pelatihan ini diperlukan parameter berupa *epoch* dan *Learning rate* dan jumlah data. Proses Klasifikasi dapat dilihat di Gambar 3.8.



Gambar 3.8. Form Klasifikasi.

BAB 4

IMPLEMENTASI DAN ANALISIS SISTEM

4.1. Implementasi Sistem

4.1.1. Implementasi Form Tampilan Sistem

Pada Gambar 4.1 dibawah ini merupakan tampilan form awal sistem, dimana terdapat 4 (empat) tombol yang disediakan. Tombol-tombol tersebut adalah tombol *Help, Rekam suara, Pre processing, dan Klasifikasi.*

Gambar 4.1 Form Tampilan Awal Sistem.

Fungsi dari setiap tombol adalah sebagai berikut :

a. Help

Tombol ini berfungsi hanya untuk menampilkan bantuan yaitu berupa petunjuk dan kegunaan masing-masing tombol ataupun menu yang ada pada sistem.

b. Rekam Suara

Tombol *Rekam suara* jika di klik maka akan menampilkan Form Perekaman Suara.

c. Pre Processing

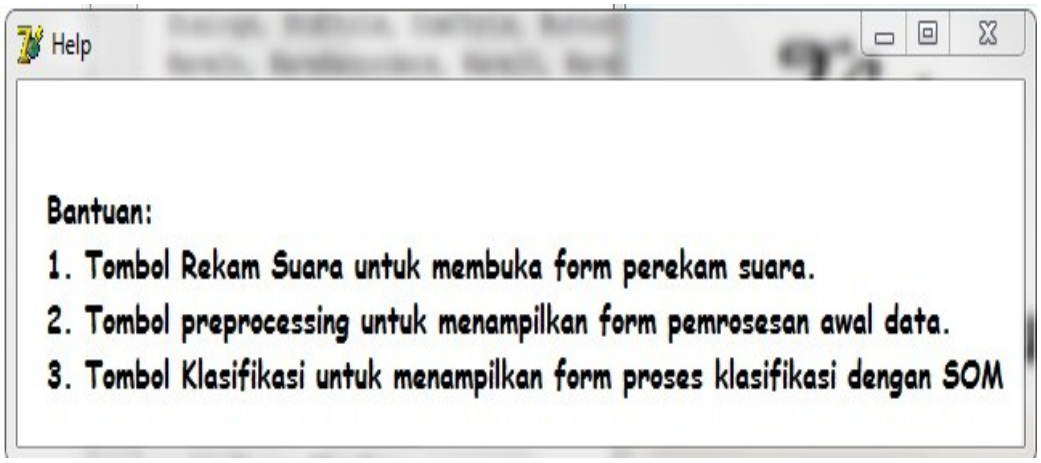
Ketika tombol ini di klik akan menampilkan Form untuk memulai pemrosesan awal file *wav*.

d. Klasifikasi

Pada tombol ini memiliki fungsi untuk menampilkan menu atau Form Proses terakhir dari pengklasifikasian suara yang sudah ada dalam *database*.

4.1.2. Form Help

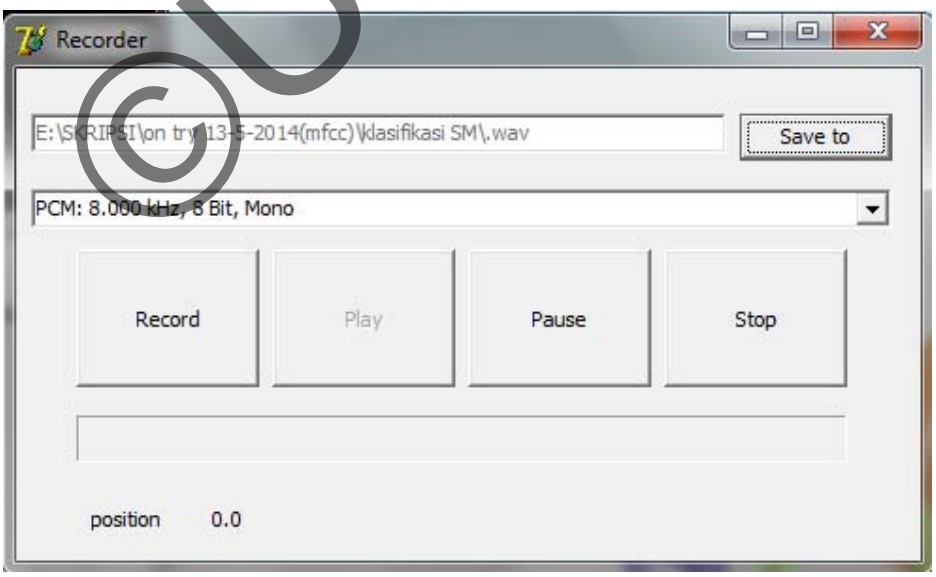
Pada Gambar 4.2. dibawah ini menunjukan tampilan dari Form Help, dimana pada form ini hanya berisi tentang panduan dan petunjuk penggunaan sistem, mulai dari fungsi-fungsi tombol yang ada dan juga langkah-langkah untuk memproses suara untuk diklasifikasikan.



Gambar 4.2 Form Bantuan

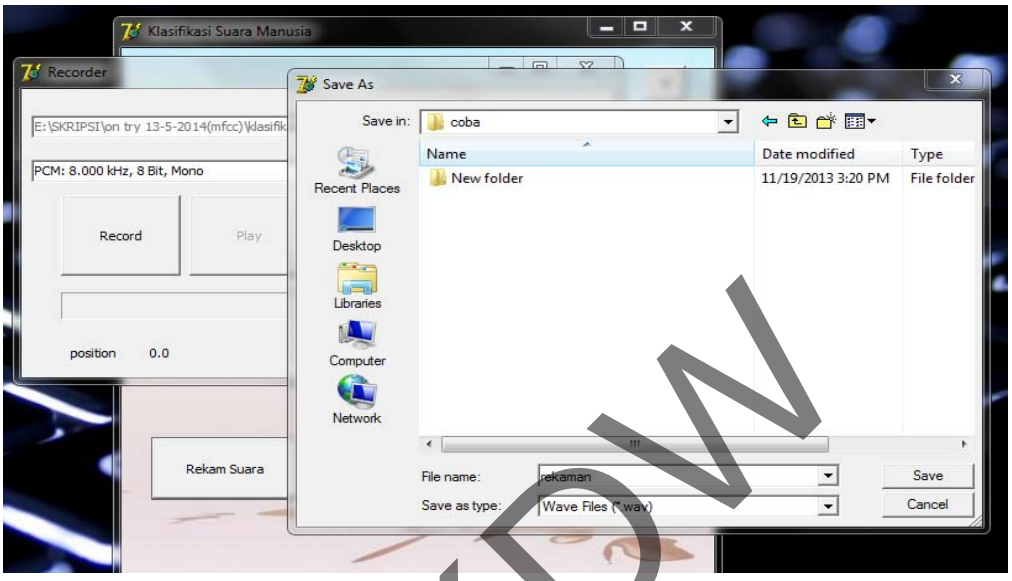
4.1.3. Form Rekam Suara

Form perekaman suara dapat dilihat pada Gambar 4.3. dibawah ini. Pada Form rekam suara kita bisa melakukan perekaman suara serta menyimpan *file* rekaman tersebut, selain itu juga tersedia pilihan untuk mengatur berapa *sample rate, Bit rate dan juga channel mono* maupun *stereo* untuk file suara yang akan kita rekam.



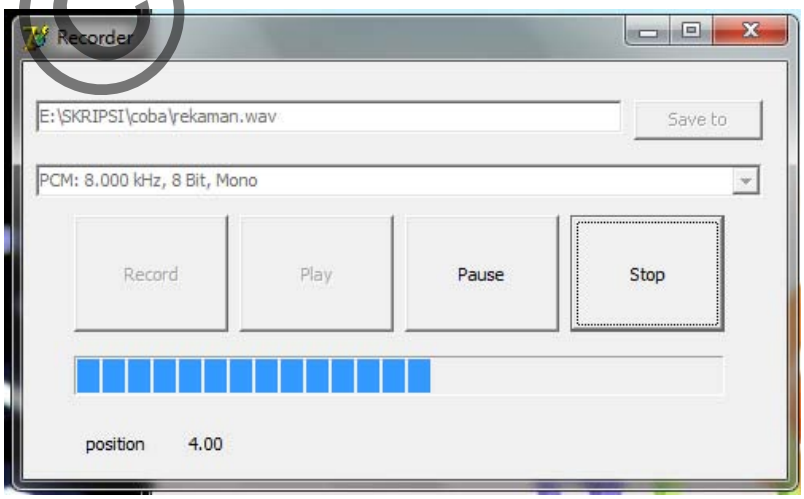
Gambar 4.3. Form Rekam suara atau *Recorder.*

Pada Gambar 4.4. menjelaskan tentang proses awal dari perekaman suara yaitu memilih tombol *save to* yang berada pada pojok kanan atas. Tombol ini berfungsi untuk memilih destinasi folder untuk menyimpan suara rekaman dan menentukan nama *file* rekaman.



Gambar 4.4. Form penyimpanan file rekaman suara

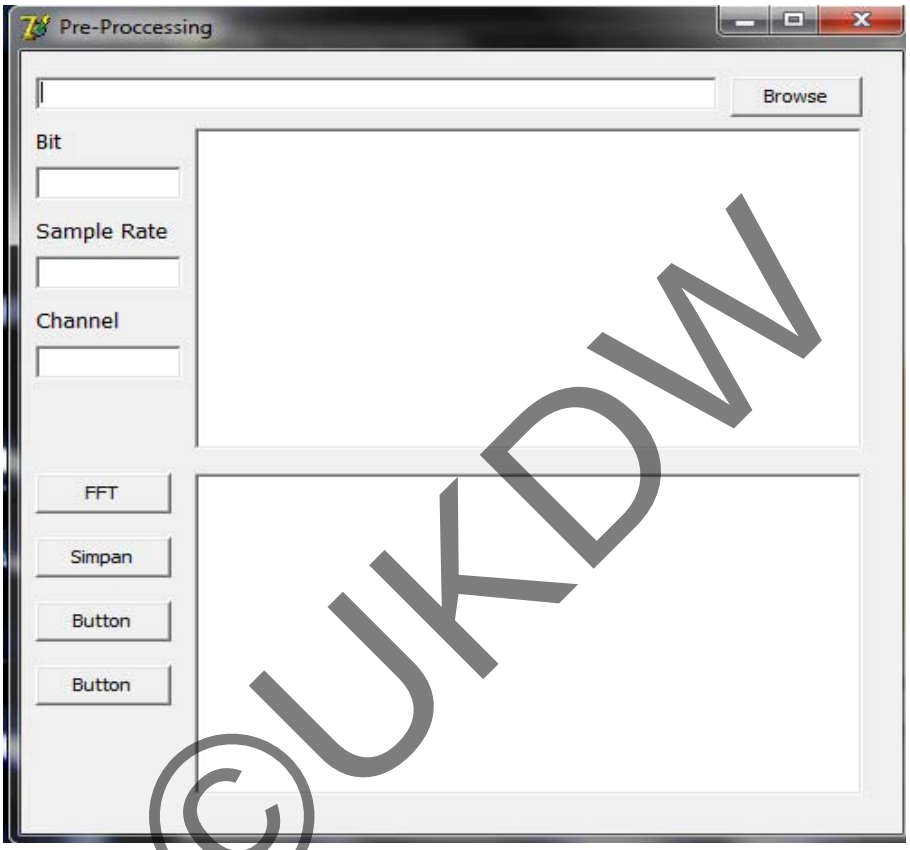
Pada Gambar 4.5. dibawah ini ditunjukan proses dalam perkaman suara dan secara langsung tersimpan di komputer sesuai dengan folder dan nama *file* yang sudah kita tentukan sebelumnya melalui tombol *save to* yang ada pada Form ini.



Gambar 4.5. Form proses perekaman suara.

4.1.4. Form Preprocessing

Form ini adalah tampilan yang akan muncul ketika kita mengklik tombol *pre-processing* pada form tampilan awal sistem. Pada Form ini ada 3 (tiga) tombol yaitu *Browse,* FFT, dan Simpan. Form ini dapat dilihat pada Gambar 4.6. dibawah ini.

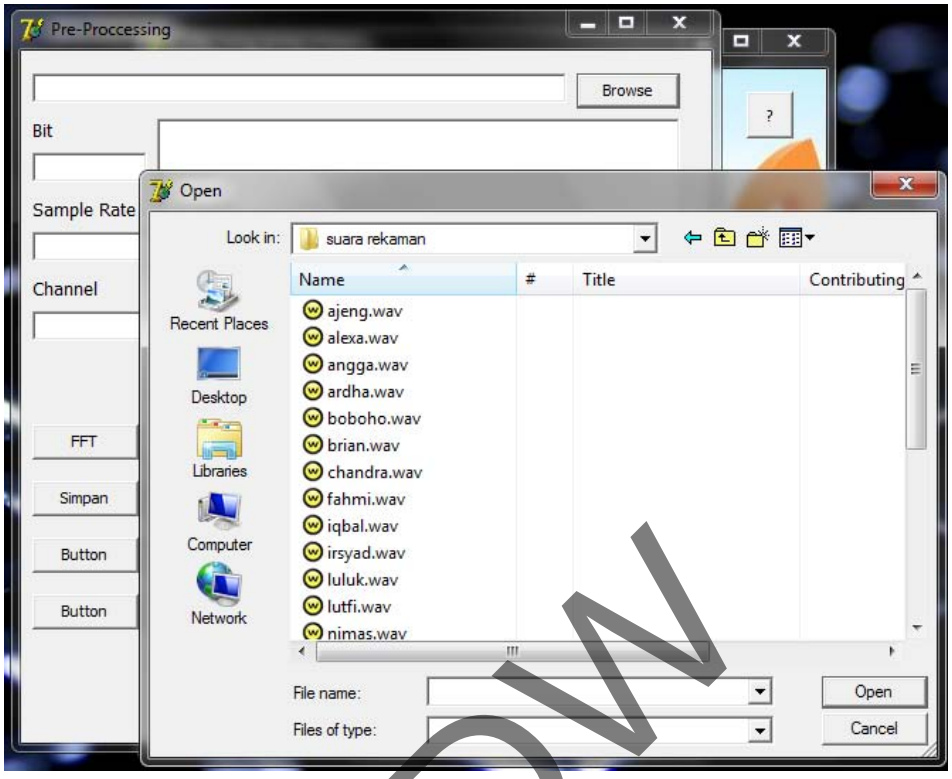


Gambar 4.6. Tampilan Form Pre-processing.

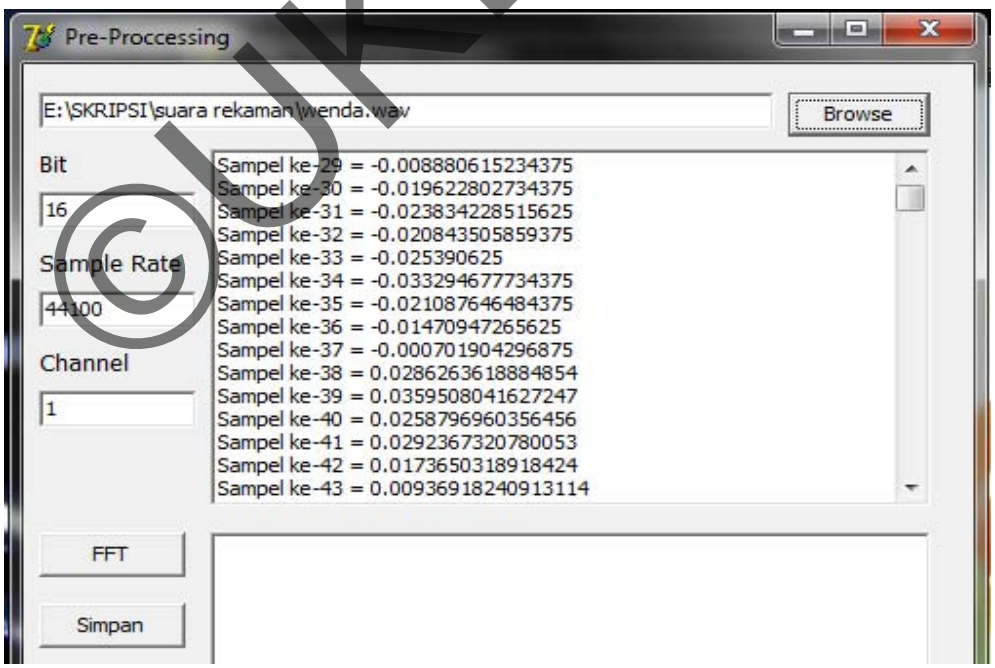
Fungsi tombol dalam form ini adalah sebagai berikut:

a. Browse

Tombol ini berfungsi untuk mengambil *file* suara yang sudah tersimpan dalam *hard disk* untuk digunakan dalam proses selanjutnya sekaligus akan menampilkan nilai amplitude dari *file wav* tersebut. Tampilan untuk tombol ini dapat dilihat pada Gambar 4.7. dan Gambar 4.8. dibawah ini.



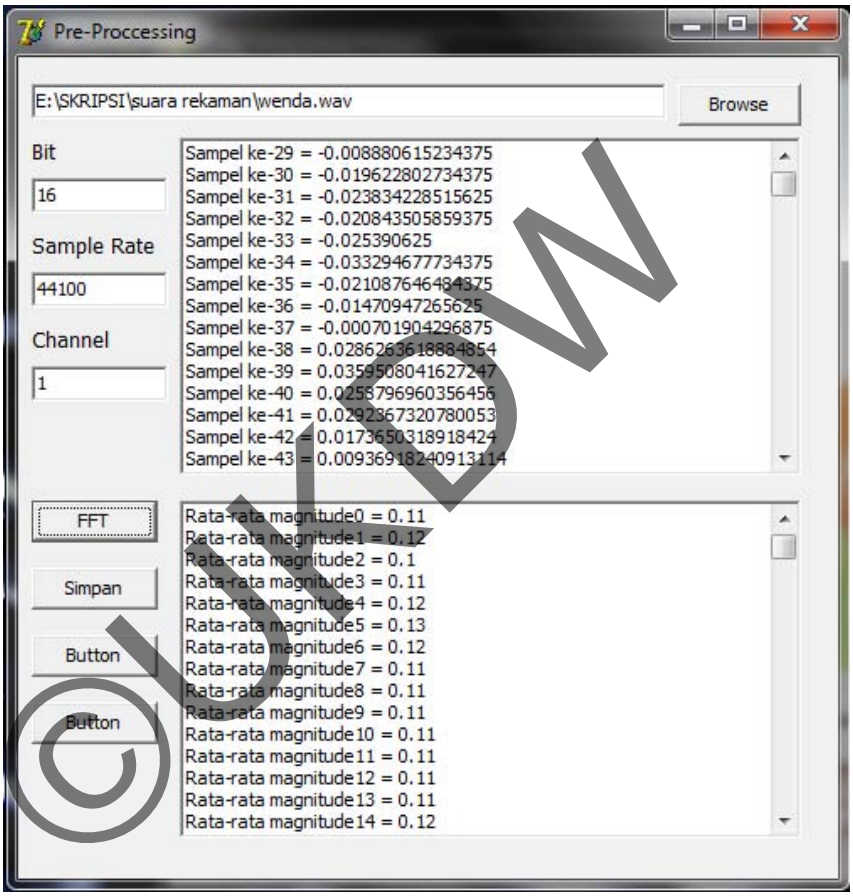
Gambar 4.7. Tampilan fungsi tombol Browse.



Gambar 4.8. Tampilan hasil amplitude

b. FFT

Tombol ini memiliki fungsi untuk memproses *file* yang sudah kita ambil sebelumnya dengan tombol *Browse.* Proses yang dilakukan adalah *pre-emphasis, frame blocking,* dan *hamming window* setelah itu kemudian dilakukan proses FFT (*Fast Fourier Transform).* Proses ini dapat dilihat pada Gambar 4.9. dibawah ini.



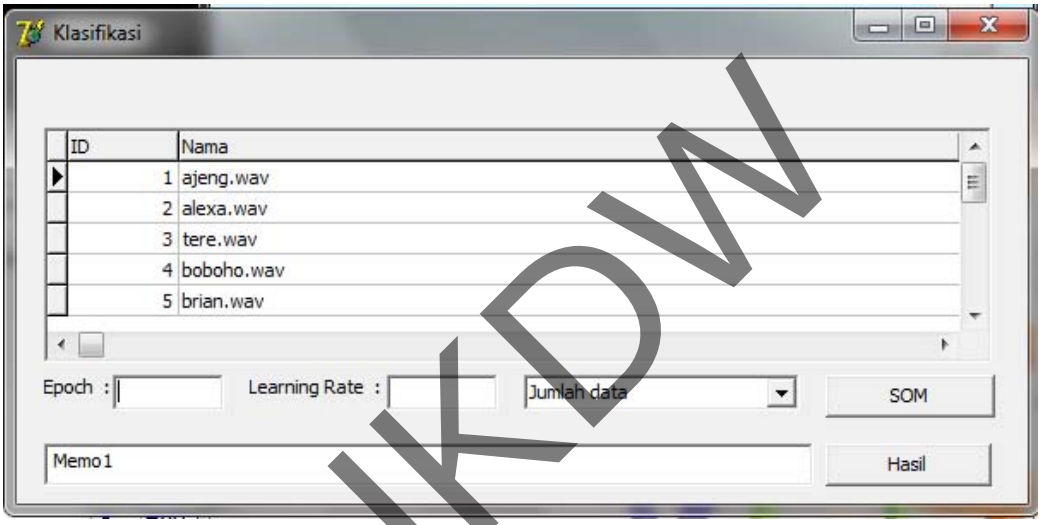
Gambar 4.9. Tampilan rata-rata magnitude dengan FFT.

c. Simpan

Tombol simpan hanya berfungsi untuk menyimpan data yang diperlukan dalam proses peng-klasifikasian selanjutnya. Ketika tombol ini di klik akan muncul pesan bahwa penyimpanan telah berhasil dilakukan. Dapat dilihat pada Gambar 4.10. dibawah ini.

4.1.5. Form Klasifikasi

Form ini adalah proses terakhir dari sistem Klasifikasi ini, dimana dalam Form ini terdapat 2 (dua) buah tombol, yaitu tombol SOM dan Hasil. Selain itu juga ada 2 (dua) kolom *input* yang digunakan untuk memasukan nilai untuk dipakai pada proses klasifikasi dengan SOM (*Self Organizing Map).* Dalam Form ini ada combobox untuk memilih jumlah data yang akan diproses dalam SOM, dan *database* ditampilkan pada tabel. Tampilan Form ini dapat dilihat pada Gambar 4.10. dibawah ini.



Gambar 4.10. Form Klasifikasi

4.2. Analisis Sistem

Sistem ini memiliki tujuan untuk mengimplementasikan *Self Organizing Map* (SOM) untuk mengklasifikasikan data suara yang sudah sebelumnya dimasukan ke dalam *database.* Dimana data suara tersebut di pre-processing terlebih dahulu dan kemudian dilakukan *Fast Fourier Transform* (FFT) untuk mendapatkan data berupa rata-rata *magnitude.*

4.2.1 Analisis Data

Dalam sistem ini yang digunakan adalah *file wave* dengan format PCM 44.100 KHZ 16 Bit Mono, dengan durasi perekaman 10 detik. Data pada *database* bisa dilihat pada Gambar 4.11. dibawah ini. Dimana dari setiap *file wav* diambil

nilai amplitudonya untuk kemudian dilakukan *preprocessing* yang kemudian akan diperoleh rata-rata *magnitude* yang akan digunakan sebagai *input* proses klasifikasi *Self Organizing Map* (SOM).

ID	Nama	Bit	Sample rate	Channel	FFT	Add New Field
1	ajeng.wav	16	44100	1	0.01;0.02;0.04;0.04;0.05;0.06;0.06;0.07;0.	
2	alexa.wav	16	44100	1	0.17;0.28;0.32;0.42;0.43;0.28;0.22;0.28;0.	
3	tere.wav	16	44100	1	0.13;0.21;0.19;0.25;0.3;0.25;0.22;0.3;0.28	
4	boboho.wav	16	44100	1	0.01;0.02;0.03;0.04;0.05;0.06;0.07;0.09;0.	
5	brian.wav	16	44100	1	0.01;0.05;0.07;0.08;0.08;0.21;0.17;0.17;0.	
6	chandra.wav	16	44100	1	0.03;0.04;0.05;0.06;0.07;0.09;0.12;0.09;0.	
7	angga.wav	16	44100	1	0.17;0.15;0.15;0.15;0.13;0.12;0.12;0.13;0.	
8	ardha.wav	16	44100	1	0.01;0.04;0.04;0.05;0.06;0.08;0.09;0.08;0.	
9	fahmi.wav	16	44100	1	0.01;0.04;0.04;0.05;0.06;0.09;0.13;0.15;0.	
10	luluk.wav	16	44100	1	0.03;0.03;0.04;0.05;0.06;0.07;0.08;0.09;0.	
11	nimas.wav	16	44100	1	0.01;0.04;0.05;0.05;0.09;0.08;0.08;0.11;0.	
12	selvi.wav	16	44100	1	0.01;0.04;0.05;0.06;0.07;0.07;0.09;0.11;0.	

Gambar 4.11 Isi database

Dalam pengambilan nilai amplitudo dilakukan proses seperti pada Gambar 4.12. dibawah ini.

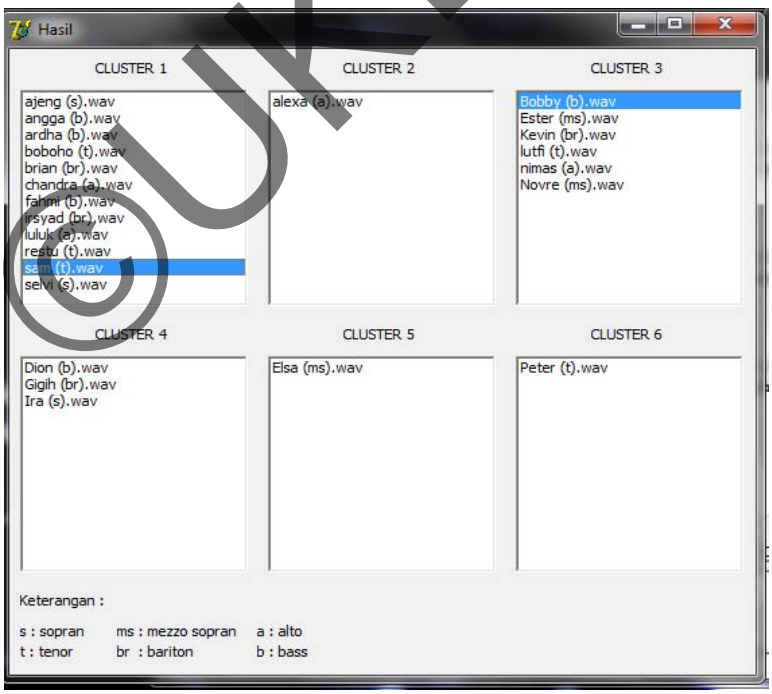
```
case Integer(Buffer[34]) of
8 : begin
          a[k] := Inttohex(Integer(Buffer[i+y]),2)+a[k];
          b[k] := (strtoint('$'+a[k]))-128;
          if b[k] <= 0 then
            begin
              b[k] := b[k]/127;
            end else
          if b[k] > 0 then
            begin
              b[k] := b[k]/128;
            end;
            //max := 127;
        end;//for
 16 : begin
        a[k] := Inttohex(Integer(Buffer[i+y]),2) +a[k];
        b[k] := strtoint('$' + a[k]);
        if b[k] < 32768 then
        begin
           b[k] := (b[k])/32767;
        end else
        if b[k] > 32767 then
        begin
            b[k] := (b[k]-65536)/32768;
        end;
        //max := 32767;
    end;
end;
```

Gambar 4.12. Potongan code pembacaan nilai amplitudo

Dalam kasus ini digunakan fungsi case dengan kondisi 8 Bit dan 16 Bit. Sistem akan memeriksa berapa ukuran Bit yang dipakai oleh *file wave* dimana sistem mendapatkan Bit tersebut dari fungsi *Buffer* yang sudah ada di Delphi.

4.2.2. Analisis Proses Klasifikasi

Dalam penelitian ini proses klasifikasi menggunakan Jaringan Syaraf Tiruan yaitu *Self Organizing Map* (SOM). Setelah melakukan penelitian terhadap data suara *wav* dengan format PCM 44100 Hz 16 Bit Mono, dimana dalam prosesnya dilakukan terlebih dahulu pengambilan nilai sampel melalui *sampling,* kemudian dilakukan *preprocessing* berupa *preemphasis, frame blocking, dan windowing* sehingga didapatkan nilai sampel untuk diubah menjadi frekuensi dengan FFT. Setelah itu rata-rata magnitude yang didapatkan menjadi *input* untuk *Self Organizing Map.* Pada penelitian kali ini tingkat akurasi yang didapatkan tidak terlalu baik dengan kondisi jumlah data 12 buah.

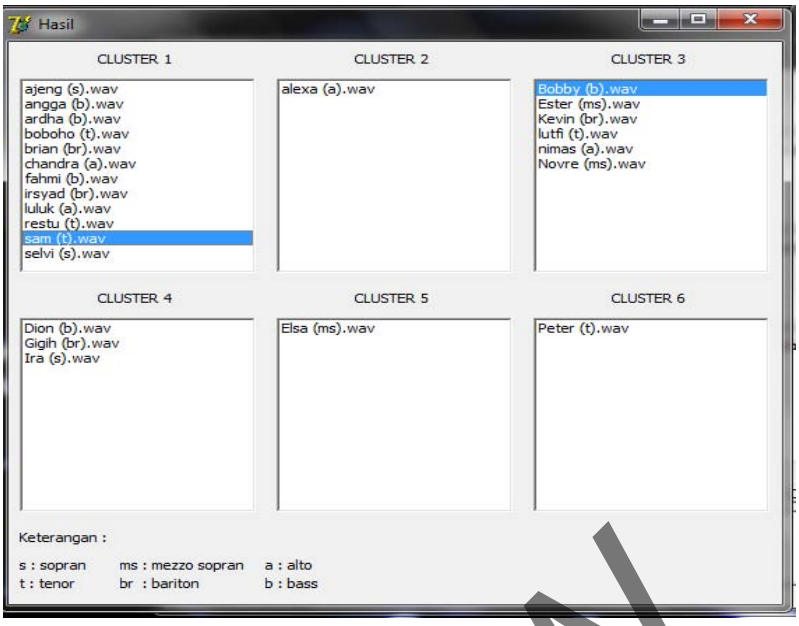


Gambar 4.12. Hasil Klasifikasi 24 data suara, epoch 100 dan learning rate 0.6.

Tabel 4.1. Tabel hasil penelitian dengan learning rate 0.6

Nama	Cluster sesuai epoch			Jenis suara asli
	100	200	1000	
Ajeng	1	1	1	Sopran
Alexa	3	4	4	Alto
Tere	1	3	3	Sopran
Boboho	1	1	1	Tenor
Brian	2	2	3	Bariton
Chandra	1	1	1	Alto
Angga	2	3	3	Bass
Ardha	1	1	1	Bass
Fahmi	1	1	1	Bass
Luluk	1	1	1	Alto
Nimas	4	5	6	Alto
Selvi	2	3	4	sopran

Dalam kasus ini pengelompokan tidak akurat dan masih banyak data yang mengumpul pada kelas yang sama. Pada Tabel 4.1. diatas dipakai learning rate yang sama yaitu 0.6.



Gambar 4.13. Hasil Klasifikasi 24 data suara, epoch 200 dan learning rate 0.6.

Dari hasil yang didapatkan pada Gambar 4.12. dan Gambar 4.13. menujukan bahwa tidak ada perubahan yang terjadi begitu juga sampai dengan epoch ke 500.

LISTING PROGRAM

1. Form Awal

```
unit Awal;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, jpeg;

type
  TFormAwal = class(TForm)
    Image1: TImage;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    BitBtn4: TBitBtn;
    Image2: TImage;
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormAwal: TFormAwal;

implementation

uses Utama, Lanjut, Klass, Help;

{$R *.dfm}

procedure TFormAwal.BitBtn2Click(Sender: TObject);
begin
  FormUtama.ShowModal;
end;

procedure TFormAwal.BitBtn3Click(Sender: TObject);
begin
  FormLanjut.ShowModal;
end;
```

```
procedure TFormAwal.BitBtn4Click(Sender: TObject);
begin
  FormAkhir.ShowModal;
end;

procedure TFormAwal.BitBtn1Click(Sender: TObject);
begin
  FormBantuan.ShowModal;
end;

end.
```

2. Form Rekam Suara

```
unit Utama;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, Buttons, mmSystem, WaveUtils, WaveStorage,
  WaveIn, WaveRecorders, WaveIO, WaveOut, WavePlayers;

type
  TFormUtama = class(TForm)
    WaveFile: TEdit;
    btnSimpan: TButton;
    Format: TComboBox;
    btnRecord: TBitBtn;
    btnPlay: TBitBtn;
    btnPause: TBitBtn;
    btnStop: TBitBtn;
    AudioLevel: TProgressBar;
    SaveDialog: TSaveDialog;
    StockAudioPlayer: TStockAudioPlayer;
    StockAudioRecorder: TStockAudioRecorder;
    Position: TLabel;
    PositionLabel: TLabel;
    WaveCollection1: TWaveCollection;
    WaveStorage1: TWaveStorage;
    procedure FormCreate(Sender: TObject);
    procedure FormatChange(Sender: TObject);
    procedure btnSimpanClick(Sender: TObject);
    procedure btnRecordClick(Sender: TObject);
    procedure btnPlayClick(Sender: TObject);
    procedure btnPauseClick(Sender: TObject);
    procedure btnStopClick(Sender: TObject);
    procedure StockAudioPlayerActivate(Sender: TObject);
```

```
    procedure StockAudioPlayerDeactivate(Sender: TObject);
    procedure StockAudioPlayerLevel(Sender: TObject; Level: Integer);
    procedure StockAudioRecorderActivate(Sender: TObject);
    procedure StockAudioRecorderDeactivate(Sender: TObject);
    procedure StockAudioRecorderLevel(Sender: TObject; Level: Integer);



  private
    { Private declarations }
  public

    { Public declarations }
  end;

var
  FormUtama: TFormUtama;
implementation

{$R *.dfm}



procedure TFormUtama.FormCreate(Sender: TObject);
  var
  pcm: TPCMFormat;
  WaveFormatEx: TWaveFormatEx;
begin
  for pcm := Succ(Low(TPCMFormat)) to High(TPCMFormat) do
  begin
    SetPCMAudioFormatS(@WaveFormatEx, pcm);
    Format.Items.Append(GetWaveAudioFormat(@WaveFormatEx));
  end;
  Format.ItemIndex := Ord(StockAudioRecorder.PCMFormat) - 1;

  WaveFile.Text := ChangeFileExt(ExtractFilePath(Application.ExeName) +
    SaveDialog.Filename, '.' + SaveDialog.DefaultExt);
  btnRecord.Enabled := True;
  btnPlay.Enabled := FileExists(WaveFile.Text);
end;

procedure TFormUtama.FormatChange(Sender: TObject);
begin
  StockAudioRecorder.PCMFormat := TPCMFormat(Format.ItemIndex + 1);
end;
```

```
procedure TFormUtama.btnSimpanClick(Sender: TObject);
begin
  if SaveDialog.Execute then
  begin
    WaveFile.Text := SaveDialog.FileName;
    btnRecord.Enabled := True;
    btnPlay.Enabled := FileExists(SaveDialog.FileName);
  end;
end;

procedure TFormUtama.btnRecordClick(Sender: TObject);
begin
    StockAudioRecorder.RecordToFile(WaveFile.Text);
end;

procedure TFormUtama.btnPlayClick(Sender: TObject);
begin
  StockAudioPlayer.PlayFile(WaveFile.Text);
end;

procedure TFormUtama.btnPauseClick(Sender: TObject);
begin
    if StockAudioPlayer.Active then
    StockAudioPlayer.Paused := not StockAudioPlayer.Paused
  else
    StockAudioRecorder.Paused := not StockAudioRecorder.Paused;
end;

procedure TFormUtama.btnStopClick(Sender: TObject);
begin
   if StockAudioPlayer.Active then
    StockAudioPlayer.Active := False
  else
    StockAudioRecorder.Active := False;
end;

procedure TFormUtama.StockAudioPlayerActivate(Sender: TObject);
begin
  Format.Enabled := False;
  btnSimpan.Enabled := False;
  btnPause.Enabled := True;
  btnStop.Enabled := True;
  btnRecord.Enabled := False;
  btnPlay.Enabled := False;
  btnStop.SetFocus;
```

```
end;

procedure TFormUtama.StockAudioPlayerDeactivate(Sender: TObject);
begin
  Format.Enabled := True;
  btnSimpan.Enabled := True;
  btnRecord.Enabled := True;
  btnPlay.Enabled := True;
  btnPause.Enabled := False;
  btnStop.Enabled := False;
  btnPlay.SetFocus;
end;

procedure TFormUtama.StockAudioPlayerLevel(Sender: TObject;
  Level: Integer);
begin
  Position.Caption := MS2Str(StockAudioPlayer.Position, msAh);
  AudioLevel.Position := Level;
end;

procedure TFormUtama.StockAudioRecorderActivate(Sender: TObject);
begin
  Format.Enabled := False;
  btnSimpan.Enabled := False;
  btnPause.Enabled := True;
  btnStop.Enabled := True;
  btnRecord.Enabled := False;
  btnPlay.Enabled := False;
  btnStop.SetFocus;
end;

procedure TFormUtama.StockAudioRecorderDeactivate(Sender: TObject);
begin
  Format.Enabled := True;
  btnSimpan.Enabled := True;
  btnRecord.Enabled := True;
  btnPlay.Enabled := True;
  btnPause.Enabled := False;
  btnStop.Enabled := False;
  btnPlay.SetFocus;
  Position.Caption := '0.00';
end;

procedure TFormUtama.StockAudioRecorderLevel(Sender: TObject;
  Level: Integer);
begin
```

```
   Position.Caption := MS2Str(StockAudioRecorder.Position, msAh);
  AudioLevel.Position := Level;
end;

end.
```

3. Form Preprocessing

```
unit Lanjut;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,ExtCtrls, math, mmSystem, WaveUtils, WaveStorage, DB,
  ADODB, FMTBcd, SqlExpr, dspFFT, dspIIRFilters;



type
    RealArray = Array of Real;
    TFormLanjut = class(TForm)
    Edit1: TEdit;
    Button1: TButton;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    WaveStorage1: TWaveStorage;
    ListBox1: TListBox;
    ListBox2: TListBox;
    ADOConnection1: TADOConnection;
    ADOTable1: TADOTable;
    DataSource1: TDataSource;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Button3: TButton;
    Button4: TButton;
    ADOQuery1: TADOQuery;
    SQLQuery1: TSQLQuery;
    dspFFTL: TdspFFT;
    dspIIRFilter1: TdspIIRFilter;
    dspFFTR: TdspFFT;
```

```
    Button2: TButton;


    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);


  private
    { Private declarations }
    sCurrentDir : string;
    sLokasiDB   : string;
    freqScale : array [1..20000] of real;
    function FloatToInt(pAngka : Double) : Integer;


  public
    { Public declarations }

  end;

var
  FormLanjut: TFormLanjut;

implementation

uses Loading;

{$R *.dfm}

var B, E       : RealArray;
    iJmlSample,ch,nFrame : Integer;
    sFrekuensi : String;
    B2         : array[0..440999] of Real;
    FB         : array[0..1000,0..512] of Real;
    avgMagnitude, magnitude, Rmagnitude, Lmagnitude : array of real;

//Uses Loadingx;

function TFormLanjut.FloatToInt(pAngka : Double) : Integer;
begin
    Result := StrToInt(FloatToStr(pAngka));
end;
```

```
procedure TFormLanjut.Button1Click(Sender: TObject);
var
  Buffer      : PChar;
  iFileHandle : Integer;
  iFileLength : Integer;
  iBytesRead  : Integer;
  i,k,y       : Integer;
  iJenis      : Integer;
  iJmlBit     : Integer;
  frekuensi   : Integer;
  fJmlBit     : Double;
  fJmlSample  : Double;
  A           : Array of String;

begin
  ListBox1.Clear;
  ListBox2.Clear;
  if OpenDialog1.Execute then
  begin
    iFileHandle := FileOpen(OpenDialog1.FileName, fmOpenRead);
    iFileLength := FileSeek(iFileHandle, 0, 2);
    FileSeek(iFileHandle, 0, 0);
    Buffer      := PChar(AllocMem(iFileLength + 1));
    iBytesRead  := FileRead(iFileHandle, Buffer^, iFileLength);
    FileClose(iFileHandle);
    Edit1.Text := OpenDialog1.FileName; // buka file

    //info wav
    iJenis  := Integer(Buffer[22]);
    ch      := iJenis;
    iJmlBit := Integer(Buffer[34]);
    sFrekuensi  := IntTohex((Integer(Buffer[26])),2)
    + IntTohex((Integer(Buffer[25])),2) + IntTohex((Integer(Buffer[24])),2);
    frekuensi := strtoint('$' +sFrekuensi);
    Edit2.Text := inttostr(iJmlBit);
    Edit3.Text := inttostr(frekuensi);
    Edit4.Text := inttostr(iJenis);

    //Listbox1.Items.Add( 'frekuensi = '+inttostr(frekuensi));
    //Listbox1.Items.Add( 'bit = '+inttostr(iJmlBit));
```

```
if (iJenis =1) then
begin

end else
begin

end;
 //Data chunk
for i := 0 to 50 do
begin
   if Buffer[i] = 'd' then
     iJenis := i + 8;
end;
i := iJenis;
k := 0;
fJmlBit   := iJmlBit/8;
fJmlSample  := Floor((iBytesRead -iJenis)/fJmlBit);
iJmlBit     := FloatToInt(fJmlBit);
iJmlSample := FloatToInt(fJmlSample);
//Listbox1.Items.Add( 'jml sample i= '+inttostr(iJmlSample));


  SetLength(A, iJmlSample+1);
  SetLength(B, iJmlSample+1);
  SetLength(E, iJmlSample+1);
  while (i <= iBytesRead-1) do
  begin
    a[k] := '';
     //Listbox1.Items.Add('ibytesread=' +IntTostr(iBytesRead));
    for y:=0 to iJmlBit-1 do
    begin
       case Integer(Buffer[34]) of
             8 : begin
       a[k] := Inttohex(Integer(Buffer[i+y]),2)+a[k];
       b[k] := (strtoint('$'+a[k]))-128;
       if b[k] <= 0 then
         begin
           b[k] := b[k]/127;
         end else
                    if b[k] > 0 then
                      begin
                              b[k] := b[k]/128;
                      end;
                      //max := 127;
      end;//for
16 : begin
```

```
          a[k] := Inttohex(Integer(Buffer[i+y]),2) +a[k];
          b[k] := strtoint('$' + a[k]);
          if b[k] < 32768 then
          begin
            b[k] := (b[k])/32767;
          end else
          if b[k] > 32767 then
          begin
             b[k] := (b[k]-65536)/32768;
          end;
          //max := 32767;
       end;
    end;
   end;
     //Listbox1.Items.Add('Sampel ke-'+IntToStr(k+1)+' = '+floattostr(b[k]));
       i := i +iJmlBit;
       k := k + 1;

   end;
   end;
   listbox1.Items.Add('hasil berhasil');
  end;



procedure TFormLanjut.Button3Click(Sender: TObject);
var
  x :integer;
  tampung : string;
begin
  for x := 1 to nFrame do
  begin
     tampung := tampung + FloatToStr(avgMagnitude[x])+';';
  end;

  ADOTable1.Open;
  ADOTable1.Append;
  ADOTable1.FieldByName('Nama').AsString :=
ExtractFileName(OpenDialog1.FileName);
  ADOTable1.FieldByName('Bit').AsString := Edit2.Text;
  ADOTable1.FieldByName('Sample rate').AsString := Edit3.Text;
  ADOTable1.FieldByName('Channel').AsString := Edit4.Text;
  ADOTable1.FieldByName('FFT').AsString := tampung;
  ADOTable1.Post;
  Showmessage('Data berhasil disimpan');
end;
```

```
procedure TFormLanjut.FormCreate(Sender: TObject);
begin
  sCurrentDir := ExtractFilePath(ParamStr(0));
  sLokasiDB   := sCurrentDir + '\Klasifikasi.mdb';
end;

procedure TFormLanjut.Button4Click(Sender: TObject);
var
  i,j   : integer;
  freqRange, nilai : real;
  str : string;
begin
  //pre-emphasis//
   B2[0] := B[0];
   for i := 1 to 440999 do
    begin
      B2[i] := B[i]-(0.97*B[i-1]);
      //listbox2.Items.Add('pre-emp['+IntToStr(i+1)+'] = '+floattostr(B2[i]));
    end;

    nFrame := Round(Length(B2)/dspFFTL.BufferSize); //Buffersize = 1024
    SetLength(avgMagnitude, nFrame);
    SetLength(Lmagnitude, dspFFTL.BufferSize2);  //Buffersize2 = 512
    SetLength(Rmagnitude, dspFFTL.BufferSize2);
    SetLength(magnitude, dspFFTL.BufferSize2);
    str := '';

    for i:= 0 to nFrame-1 do
       begin
          //<--input FFT-->//
          for j := 0 to dspFFTL.BufferSize-1 do
             begin
                case ch of
                1 : begin
                   dspFFTL.RealIn[j] := B2[dspFFTL.BufferSize*i+j];
                   dspFFTL.ImagIn[j] := 0;
                end;
                2 : begin
                   dspFFTL.RealIn[j] := B2[dspFFTL.BufferSize*i+j];
                   dspFFTL.ImagIn[j] := 0;

                   dspFFTR.RealIn[j] := B2[dspFFTR.BufferSize*i+j];
                   dspFFTR.ImagIn[j] := 0;
                end;
                end;// end case ch
```

```
            end; //for j dsp

          // FFT dgn komponen dsp
          dspFFTL.FFT;
          dspFFTR.FFT;
          // hitung magnitude
          dspFFTL.CalculateMagnitudes;
          dspFFTR.CalculateMagnitudes;
          //range frekuensi
          freqRange := (44100/2)/dspFFTL.BufferSize2;
          for j:= 0 to (dspFFTL.BufferSize2) do
               freqScale[j] := (j*freqRange);

          //proses awal nilai magnitude
          nilai := 0;
          for j:= 0 to (dspFFTL.BufferSize2-1) do
             begin
             Lmagnitude[j] :=
sqrt(power(dspFFTL.RealOut[j],2)+power(dspFFTL.ImagOut[j],2));
             Rmagnitude[j] :=
sqrt(power(dspFFTR.RealOut[j],2)+power(dspFFTR.ImagOut[j],2));
             magnitude[j]  := Lmagnitude[j]+Rmagnitude[j];
             FB[i][j] := magnitude[j];
             nilai := nilai + magnitude[j];
             end; // for dspfft buffersize2-1

          //rata2 magnitude tiap frame
          avgMagnitude[i] :=
StrtoFloat(Format('%0.3f',[nilai/dspFFTL.BufferSize2]));
          str := str +''+FloatToStr(avgMagnitude[i]);
          listbox2.Items.Add('Rata-rata magnitude'+ IntToStr(i)+' =
'+FloatToStr(avgMagnitude[i])+' ');


       end;//for nFrame
       //strpola := str;
       listbox2.Items.Add('Pola = ' +str);



end;



procedure TFormLanjut.Button2Click(Sender: TObject);
begin
```

```
  ADOConnection1.Execute('DELETE * FROM voicetab');
  ADOConnection1.Execute('ALTER TABLE voicetab ALTER COLUMN ID
Autoincrement(1,1)');
  showmessage('data terhapus');
end;

end.
```

4. Form Klasifikasi

```
unit Klass;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids, DBGrids, ADODB, DB, Math;

type
    //DoubleArray = Array of Double;
    TFormAkhir = class(TForm)
    ADOConnection1: TADOConnection;
    ADOTable1: TADOTable;
    ADOQuery1: TADOQuery;
    editep: TEdit;
    edital: TEdit;
    Button1: TButton;
    DataSource1: TDataSource;
    Button2: TButton;
    Memo1: TMemo;
    ComboBox1: TComboBox;
    DBGrid1: TDBGrid;
    Label1: TLabel;
    Label2: TLabel;
    ListBox1: TListBox;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
    coba : Integer;
  end;
```

```
var
  FormAkhir: TFormAkhir;

implementation

uses Hasil;

{$R *.dfm}

procedure TFormAkhir.Button2Click(Sender: TObject);
var
  bobot       : array[1..6, 1..856] of double;
  cluster   : array[1..100] of integer;
  epoch   : integer;
  learningRate : double;
  i,j,k,l,z,n : integer;
  input   : array[1..30, 1..856] of double;
  jarak   : array[1..6] of double;
  tmpBb : double;
  tampJrk : double;
  WinUnit, win_unit : integer;
  region : integer;
  combobox, jmlData : integer;
  tmp     : array [1..30] of string;
  List    : TStrings;
begin
  jmlData := 0;
  region:=1;
  epoch:=StrToInt(editep.Text);
  learningRate:=StrToFloat(edital.Text);
  WinUnit := 1;
  FormHasil.combx := FormAkhir.ComboBox1.ItemIndex;
  combobox := ComboBox1.ItemIndex;
  if combobox = 0 then
  begin
    jmlData := 6;
  end
  else if combobox = 1 then
  begin
    jmlData := 12;
  end
  else if combobox = 2 then
  begin
    jmlData := 18;
  end
  else if combobox = 3 then
```

```
  begin
    jmlData := 24;
  end
  else if combobox = 4 then
  begin
    jmlData := 30;
  end
  else
  begin
     jmlData := 0;
  end;
  //<---ambil data dari database-->//
  for z := 1 to jmlData do
  begin
    for n := 1 to 856 do
    begin
      ADOQuery1.SQL.Text :='select * from voicetab where ID ='+IntToStr(z);
      ADOQuery1.Active:=true;
      tmp[z] := ADOQuery1.FieldValues['FFT'];
      List := TStringList.Create;
        try
          ExtractStrings([';'], [],PChar(tmp[z]),List);
          input[z][n] := input[z][n] + strtofloat(List[n-1]);

        finally
          List.Free;
        end;  // end try-finally

    end;// for baru
  end; //end for ambil data


for i:=1 to 6 do
  begin
    for j:=1 to 856 do
    begin
      bobot[i][j] := Random;
    end;
  end;

//<--Klasifikasi SOM-->//
for z:=1 to 2 do
  begin
    for i:=1 to epoch do
    begin
      for j:=1 to jmlData do
```

```
    begin
      for k:=1 to 6 do    //cluster = 6
      begin
        for l:=1 to 856 do
        begin
          tmpBb:=sqr(bobot[k][l]-input[j][l]);
          jarak[k]:=jarak[k]+tmpBb;
        end;
      end;

      tampJrk:=jarak[1];
      for k:=2 to 6 do
      begin
        if jarak[k] < tampJrk then
        begin
          tampJrk:=jarak[k];
          WinUnit:=k;
        end
        else if jarak[1]=tampJrk then
          WinUnit:=1;
      end;

      for k:=1 to 6 do
      begin
        jarak[k]:=0;
      end;

      cluster[j]:=WinUnit;

      if region=0 then
      begin
        for k:=1 to 856 do
        begin
          bobot[WinUnit][k]:=((1-
learningRate)*bobot[WinUnit][k])+(learningRate*input[j][k]);
        end;
      end
      else
      if region=1 then
      begin
        if WinUnit=1 then
        begin
          for k:=1 to 856 do
          begin
            bobot[WinUnit][k]:=((1-
learningRate)*bobot[WinUnit][k])+(learningRate*input[j][k]);
```

```
            bobot[WinUnit+1][k]:=((1-
learningRate)*bobot[WinUnit+1][k])+(learningRate*input[j][k]);
          end;
        end
        else if WinUnit=6 then
        begin
          for k:=1 to 856 do
          begin
            bobot[WinUnit][k]:=((1-
learningRate)*bobot[WinUnit][k])+(learningRate*input[j][k]);
            bobot[WinUnit-1][k]:=((1-learningRate)*bobot[WinUnit-
1][k])+(learningRate*input[j][k]);
          end;
        end
        else
        begin
          for k:=1 to 856 do
          begin
            bobot[WinUnit][k]:=((1-
learningRate)*bobot[WinUnit][k])+(learningRate*input[j][k]);
            bobot[WinUnit+1][k]:=((1-
learningRate)*bobot[WinUnit+1][k])+(learningRate*input[j][k]);
            bobot[WinUnit-1][k]:=((1-learningRate)*bobot[WinUnit-
1][k])+(learningRate*input[j][k]);
          end;
        end;
      end;
    end;

    learningRate:=learningRate*0.5;
  end;
  region:=region-1;
 end;

 for i := 1 to jmlData do
 begin
    ADOQuery1.SQL.Text :='select * from voicetab where ID ='+IntToStr(i);
    ADOQuery1.Active:=true;
    ADOQuery1.Edit;
    ADOQuery1.FieldByName('Cluster').Text:=IntToStr(cluster[i]);
    ADOQuery1.Post;
 end;
 FormHasil.ShowModal;
end;
```

```
procedure TFormAkhir.Button1Click(Sender: TObject);
begin
   showmessage('Hore');
end;

end.
```

5. Form Hasil

```
unit Hasil;



interface



uses

  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

  Dialogs, StdCtrls, DB, ADODB;



type

  TFormHasil = class(TForm)

    ListBox1: TListBox;

    ListBox2: TListBox;

    ListBox3: TListBox;

    ListBox4: TListBox;

    ListBox5: TListBox;

    ListBox6: TListBox;

    Label1: TLabel;

    Label2: TLabel;

    Label3: TLabel;
```

```
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    ADOConnection1: TADOConnection;
    DataSource1: TDataSource;
    ADOQuery1: TADOQuery;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    combx : integer;
  end;

var
  FormHasil: TFormHasil;

implementation

uses Klass;

{$R *.dfm}

procedure TFormHasil.FormCreate(Sender: TObject);
```

```
var
  i : integer;
  jmldt   : integer;
  kelas : String;
  tmp : integer;
begin
  if combx = 0 then
  begin
    jmldt := 6;
  end
  else if combx = 1 then
  begin
    jmldt := 12;
  end
  else if combx = 2 then
  begin
    jmldt := 18;
  end
  else if combx = 3 then
  begin
    jmldt := 24;
  end
  else if combx = 4 then
  begin
```

```
  jmldt := 30;
end
else
begin
   jmldt := 0;
end;


ADOQuery1.SQL.Text :='select * from voicetab ';
ADOQuery1.Active:=true;
    for i := 1 to 30 do
      begin
        ADOQuery1.SQL.Text :='select * from voicetab where ID ='+IntToStr(i);
        ADOQuery1.Active:=true;
        tmp := ADOQuery1.FieldByName('Cluster').AsInteger;
        if tmp = 1 then
          begin
            kelas := ADOQuery1.FieldByName('Nama').AsString;
            ListBox1.Items.Add(kelas);
          end//if
        else if  tmp = 2 then
          begin
             kelas := ADOQuery1.FieldByName('Nama').AsString;
             ListBox2.Items.Add(kelas);
          end//else if
```

```
          else if  tmp = 3 then
            begin
              kelas := ADOQuery1.FieldByName('Nama').AsString;
              ListBox3.Items.Add(kelas);
            end//else if
          else if  tmp = 4 then
            begin
              kelas := ADOQuery1.FieldByName('Nama').AsString;
              ListBox4.Items.Add(kelas);
            end//else if
          else if  tmp = 5 then
            begin
              kelas := ADOQuery1.FieldByName('Nama').AsString;
              ListBox5.Items.Add(kelas);
            end//else if
          else
            begin
              kelas := ADOQuery1.FieldByName('Nama').AsString;
              ListBox6.Items.Add(kelas);
            end;
        end;// end for
end;


end.
```