PENGENALAN *CHORD* OTOMATIS MENGGUNAKAN JARINGAN SYARAF TIRUAN *LEARNING VECTOR QUANTIZATION*

SKRIPSI

oleh

YUSRI ALFIANSYAH

0310960081 – 96

PROGRAM STUDI ILMU KOMPUTER

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS BRAWIJAYA

MALANG

2008

UNIVERSITAS BRAWIJAYA

PENGENALAN *CHORD* OTOMATIS MENGGUNAKAN JARINGAN SYARAF TIRUAN *LEARNING VECTOR QUANTIZATION*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

oleh
YUSRI ALFIANSYAH
0310960081 – 96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2008


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

PENGENALAN *CHORD* OTOMATIS MENGGUNAKAN JARINGAN SYARAF TIRUAN *LEARNING VECTOR QUANTIZATION*

oleh
YUSRI ALFIANSYAH
0310960081 – 96

Setelah dipertahankan di depan Majelis Penguji pada 31 Juli 2008 dan dinyatakan memenuhi syarat untuk memperoleh gelar sarjana dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Drs. Marji, MT.	**Reza Andria Siregar, ST.**
NIP. 131 993 386	NIP. 132 318 426

Mengetahui
Ketua Jurusan Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Brawijaya,

Dr. Agus Suryanto, MSc.
NIP. 132 126 049

UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama	:	Yusri Alfiansyah
NIM	:	0310960081
Jurusan	:	Matematika
Penulis skripsi berjudul	:	Pengenalan *Chord* Otomatis Menggunakan Jaringan Syaraf Tiruan *Learning Vector Quantization*

Dengan ini menyatakan bahwa:

1. Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.
2. Apabila di kemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan,maka saya bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 31 Juli 2008
Yang menyatakan,

Yusri Alfiansyah
NIM. 0310960081

PENGENALAN CHORD OTOMATIS MENGGUNAKAN JARINGAN SYARAF TIRUAN LEARNING VECTOR QUANTIZATION

ABSTRAK

Pengenalan *chord* adalah salah satu varian dari bidang pengenalan suara. Sistem pengenalan *chord* otomatis pada komputer mencoba menirukan kemampuan manusia mengenali *chord* suatu musik dengan mendengarkan suara musik tersebut. Pada penelitian ini pengenalan dilakukan terhadap file musik dalam format wav. Keluaran pengenalan berupa rangkaian *chord* penyusun musik yang ditampilkan tiap potongan musik secara berurutan. File musik dalam format wav yang merupakan sinyal dalam domain waktu dibagi ke dalam *frame-frame* kecil berukuran 16384 (2^{14}) sampel suara. Tiap *frame* tersebut ditransformasi menggunakan metode FFT sehingga dihasilkan sinyal dalam domain frekuensi. Tiap koefisien hasil FFT dipetakan menurut nada yang bersesuaian menjadi PCP yang kemudian diskala dalam rentang 0–1. Hasil penskalaan ini yang dijadikan input jaringan syaraf tiruan. Pembelajaran dan pengenalan dalam jaringan syaraf tiruan dilakukan dengan metode *Learning Vector Quantization*, dengan laju pembelajaran 0,01 dan iterasi maksimum 10000. Hasil pengenalan yang dilakukan terhadap 10 lagu dengan berbagai aliran musik menunjukkan tingkat akurasi mencapai 94,248% untuk pengenalan musik solo, 74,374% untuk pengenalan musik *full*, dan 77,704% untuk musik *full* yang dikonversi dari midi.

AUTOMATIC CHORD RECOGNITION USING LEARNING VECTOR QUANTIZATION NEURAL NETWORK

ABSTRACT

Chord recognition is one of sound recognition subject. Automatic music recognition system on computer simulate human ability for retrieve chord of music just by hear it. In this research, recognition applied for music file in wav format. Output of recognition is chord sequence that retrieved from input music and displayed frame by frame. Music file in wav format, as time window signal, divide into several frames where each frame include 16384 sound sample. Each frame then transform by FFT method into frequency domain signal. Each FFT coefficient mapped into PCP based on suitable pitch class. This PCP become input for neural network after scaled at 0–1 range. Learning and recognizing process of neural network perform by Learning Vector Quantization algorithm with learning rate 0,01 and maximum iteration 10000 for learning process. Recognition result of 10 songs with various genre show the accuracy of recognition reach 94,248% for solo music, 74,374% for full music, and 77,704% for full music converted from midi file.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *SOURCECODE*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam kehidupan saat ini, musik telah menjadi suatu hal yang tidak terpisahkan dari kehidupan manusia. Musik merupakan salah satu sarana yang dijadikan manusia sebagai hiburan, bahkan musik bagi sebagian orang dijadikan sebagai sumber penghasilan. Bagi kebanyakan orang musik sekedar sebagai media rekreasi, namun ada juga yang berpendapat bahwa musik dapat mempengaruhi idealisme dan aktivitas seseorang dalam kehidupannya.

Ada berbagai cara bagi setiap manusia untuk menikmati musik. Ada yang cukup dengan mendengarkan saja, namun ada juga yang memilih cara menikmati musik dengan memainkannya sendiri. Ada sebagian manusia yang memiliki kemampuan untuk menerjemahkan musik yang didengar ke dalam nada-nada sehingga dapat dimainkan menggunakan alat musik sendiri. Dengan hanya mendengarkan musik saja, seseorang bisa mengetahui *chord* atau kunci nada yang dipakai dalam musik tersebut.

Syaraf manusia memiliki kemampuan untuk menerima dan membedakan respon yang ditangkap dari lingkungan luar, termasuk respon berupa suara. Syaraf dalam telinga yang tersusun sedemikian rupa mampu menerima rangsang dari luar berupa suara, yang kemudian diteruskan ke otak dan diolah, sehingga bisa dibedakan suara dengan frekuensi rendah dan tinggi.

Ada berbagai bagian yang berperan dalam telinga manusia sehingga bisa mendengarkan suara. Menurut McCuskey (2003), telinga manusia seperti *spectrum analyzer* biologis. Dalam telinga terdapat bagian yang disebut *cochlea*. Dalam *cochlea* tersebut terdapat sebuah membran. Membran yang berlapis syaraf tersebut mempunyai bagian yang berbeda-beda ukuran dan ketebalannya. Ada yang berbentuk tipis dan kaku, yang dapat meresonansi suara dalam frekuensi tinggi. Ada juga yang berbentuk tebal dan lentur yang dapat meresonansi suara dalam frekuensi rendah. Apabila ada suara dengan frekuensi tertentu masuk ke dalam telinga akan mengakibatkan resonansi pada suatu bagian dari membran tersebut, kemudian akan diteruskan oleh syaraf pada bagian yang beresonansi menuju otak, hingga manusia dapat membedakan suara menurut tinggi rendahnya frekuensi.

Komputer merupakan suatu alat yang memiliki kemampuan penghitungan dengan ketelitian tinggi. Kemampuan menghitung itu dapat dimanfaatkan manusia untuk membantu permasalahan-permasalahan yang dihadapi. Termasuk permasalahan dalam pencarian *chord* dari sebuah musik.

Salah satu metode dalam ilmu komputer untuk menangani masalah pengenalan pola adalah jaringan syaraf tiruan. Sesuai dengan namanya, jaringan syaraf tiruan adalah suatu metode yang menirukan jaringan syaraf manusia. Jaringan syaraf tiruan memungkinkan sistem komputer melakukan pembelajaran terhadap dirinya sendiri untuk mengenali pola. *Learning Vector Quantization* (LVQ) seringkali digunakan sebagai metode pembelajaran jaringan syaraf tiruan untuk pengenalan pola. Dengan menggunakan metode ini, komputer bisa mengenali pola suara, tidak terbatas untuk pengenalan suara manusia, namun juga bisa juga diterapkan untuk pengenalan *chord*.

Penelitian mengenai pengenalan *chord* sebelumnya telah dilakukan oleh Alexander Sheh dan Daniel Ellis (2003). Mereka melakukan segmentasi dan pengenalan *chord* menggunakan metode *Hidden Markov Model* dengan pembelajaran *Expectation Maximation algorithm*. Dari penelitian ini didapatkan hasil keakuratan pengenalan mencapai 76% untuk segmentasi, dan 22% untuk pengenalan. Penelitian ini dilakukan terhadap 20 lagu *The Beatles*.

Kyogu Lee dan Malcolm Slaney melakukan penelitian pada tahun 2006, yang didasarkan pada penelitian sebelumnya yang dilakukan oleh Alexander Sheh dan Daniel Ellis. Dengan memperbaiki *preprocessing* pada data input dan dengan pembatasan pengenalan serta penggunaan sampel musik yang berbeda, penelitian ini menghasilkan keakuratan pengenalan mencapai 93%. Namun perlu dicatat bahwa dalam penelitian ini, Lee dan Slaney melakukan penelitian terhadap data suara hanya berupa suara piano dan pengenalan dilakukan terbatas pada 36 tipe *chord*, yaitu *chord* major, minor dan major7.

Dalam skripsi ini akan diterapkan jaringan syaraf tiruan dengan metode pembelajaran *Learning Vector Quantization* (LVQ) untuk menyelesaikan masalah pengenalan *chord*. Dengan masukan sebuah musik dalam format wav, keluaran yang dihasilkan berupa *chord* yang menyusun musik tersebut. Proses pengenalan *chord* ini secara

garis besar dapat dibedakan menjadi 2 tahap, yaitu tahap pengolahan sinyal digital dan tahap pengenalan pola. Dari penelitian ini diharapkan dapat diketahui keakuratan dari pengenalan *chord* dengan menggunakan metode jaringan syaraf tiruan *Learning Vector Quantization* (LVQ).

1.2 Rumusan Masalah

Dari uraian pada latar belakang dapat dibuat suatu rumusan masalah yaitu: bagaimana keakuratan pengenalan *chord* otomatis dengan menggunakan jaringan syaraf tiruan *Learning Vector Quantization*?

1.3 Batasan Masalah

Ruang lingkup yang membatasi permasalahan yang akan dibahas pada skripsi ini antara lain :

- Masukan bagi program hanya berupa file musik dalam format wav dengan *sample rate* 44100.
- Pengenalan dibatasi hanya untuk 24 tipe *chord*, yaitu 12 *chord* major dan 12 *chord* minor.
- Keluaran sistem pengenalan berupa rangkaian *chord* penyusun sebuah lagu.
- Keakuratan hasil pengenalan dilakukan dengan menghitung persentase kesesuaian tiap *frame* rangkaian *chord* hasil pengenalan dengan rangkaian *chord* yang sebenarnya.

1.4 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah untuk mengetahui keakuratan pengenalan *chord* otomatis dengan menggunakan metode jaringan syaraf tiruan *Learning Vector Quantization* (LVQ).

1.5 Manfaat

Hasil penulisan skripsi ini diharapkan dapat bermanfaat bagi praktisi ilmu komputer maupun musik yang ingin mengadakan analisis dalam bidang yang berkenaan dengan musik. Selain itu diharapkan hasil penulisan skripsi ini dapat bermanfaat juga untuk dikembangkan lebih lanjut menjadi aplikasi yang lebih kompleks,

misalnya untuk mengenali komposisi musik secara otomatis, dimana yang dikenali tidak hanya *chord*, namun notasi musik yang menyusun suatu lagu dalam bentuk not balok.

BAB II
TINJAUAN PUSTAKA

Pengenalan *chord* adalah salah satu implementasi dari pengenalan pola (*pattern recognition*). Lee (2006) mengemukakan bahwa pengenalan *chord* otomatis, sering pula disebut *Automatic Chord Recognition*, merupakan proses untuk mendapatkan informasi berupa urutan *chord* dari suara musik. Pengenalan *chord* ini sangat berguna bagi yang akan melakukan analisis harmonik dari sebuah musik.

2.1 Musikologi

2.1.1 Tangga Nada

Dalam seni musik dikenal istilah tangga nada yang berisikan kumpulan nada-nada yang harmonis. Keharmonisannya terjadi karena ada aturan di balik itu semua. Hasugian (2007) mengemukakan bahwa dalam teori musik, tangga nada menunjuk pada persepsi atas frekuensi suatu nada. Sebagai contoh, nada A memiliki tinggi nada yang ekuivalen dengan 440 Hz. Nada A dengan tinggi nada demikian dikenal sebagai nada konser, sekalipun tidak selalu demikian.

Nada selalu berulang untuk tiap oktaf yang ada, maka istilah tangga nada kromatik sering dipakai untuk 12 nada dari tiap oktaf. Selisih frekuensi suatu nada pada suatu oktaf dengan nada tersebut pada oktaf yang lain adalah dua kali frekuensi nada tersebut. Sebagai contoh nada A dengan frekuensi 440 Hz pada suatu oktaf, maka pada satu oktaf di atasnya nada A mempunyai frekuensi 880 Hz. Tabel 2.1 menunjukkan frekuensi dari 12 nada kromatik pada suatu oktaf.

2.1.2 *Chord*

Chord adalah kumpulan tiga nada atau lebih yang dimainkan secara bersamaan dan terdengar harmonis. *Chord* bisa dimainkan secara terputus-putus ataupun secara bersamaan. Beberapa contoh alat musik yang bisa memainkan *chord* adalah gitar, piano, dan organ.

Tabel 2.1 Frekuensi Tangga Nada Kromatik (Hasugian, 2007)

Nada	Frekuensi
A	440,00 Hz
A# / Bb	466,16 Hz
B	493,88 Hz
C	523,25 Hz
C# / Db	554,37 Hz
D	587,33 Hz
D# / Eb	622,25 Hz
E	659,25 Hz
F	698,46 Hz
F# / Gb	739,99 Hz
G	783,99 Hz
G# / Ab	830,61 Hz
A	880,00 Hz

Jenis-jenis *chord* ditentukan oleh interval nada-nada penyusunnya. Interval adalah jarak/selang antara 2 buah nada. Hasugian (2007) menyatakan bahwa *chord* disusun dengan mengombinasikan interval yang ada dan memainkan 3 atau lebih nada secara bersamaan. Interval diukur dari nada awal tipe *chord* yang akan disusun. Misalkan disusun *chord* D, maka nada D dihitung sebagai nada awal (dasar) dalam perhitungan interval.

Contoh *chord* sederhana adalah tipe *chord* triad, yaitu *chord* yang disusun dengan 3 nada penyusun. *Chord-chord* yang lain merupakan kelanjutan dari tipe triad ini. Tipe triad major dan minor adalah tipe-tipe *chord* yang paling dasar yang termasuk dalam tipe *chord* triad.

Triad major disusun dengan kombinasi interval 2 – 1½ , artinya interval antara nada ke-1 dengan nada ke-2 adalah 2, dan interval nada ke-2 dengan nada ke-3 adalah 1½. Sedangkan triad minor disusun dengan kombinasi interval 1½ – 2, artinya interval antara nada ke-1 dengan nada ke-2 adalah 1½, dan interval nada ke-2 dengan nada ke-3 adalah 2.

Misalkan akan disusun *chord* C major dan C minor. Maka dari tangga nada C didapat urutan tangga nada sebagai berikut:

Nada : C – D – E – F – G – A – B – C

Interval : 1 1 1/2 1 1 1 1/2

Karena C major disusun dengan kombinasi interval 2 – 1½, maka *chord* C major disusun oleh nada C-E-G. Sedangkan *chord* C minor yang memiliki kombinasi interval 1½ – 2, disusun oleh nada C-Eb-G atau C-D#-G.

2.2 Pengolahan Sinyal Digital

Pengolahan sinyal digital atau *Digital Signal Processing* (DSP) adalah proses mendapatkan atau mengekstrak informasi yang dibawa oleh sinyal. Terdapat beberapa proses yang termasuk di dalam pengolahan sinyal digital, diantaranya yaitu *sampling, quantization, filtering, windowing,* transformasi sinyal. Dalam pengenalan *chord* otomatis ini, proses pengolahan sinyal digital yang digunakan adalah proses *windowing* dan transformasi sinyal.

2.2.1 Sinyal Digital

Sinyal yang dikeluarkan oleh sumber bunyi diterima dalam bentuk sinyal analog. Sinyal analog adalah sinyal kontinyu dalam domain waktu dan amplitudo (Pelton, 1993).

Untuk dapat dianalisis dalam komputer, maka sinyal analog perlu diubah terlebih dahulu menjadi sinyal digital. Sinyal digital adalah sinyal diskrit dan masih dalam domain waktu. Proses mengubah sinyal analog menjadi sinyal digital disebut dengan proses ADC (*Analog to Digital Convertion*).

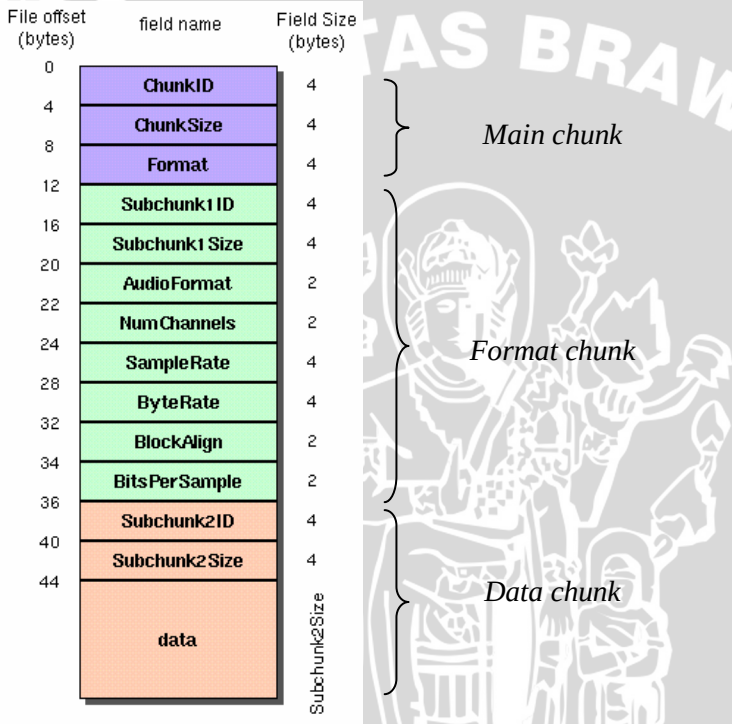
2.2.2 Struktur File Wav

Suara dalam bentuk digital diwujudkan dalam berbagai macam format file, diantaranya wave, mp3, wma, ogg, dan lain sebagainya. Salah satu format file yang sering digunakan dalam pengolahan sinyal digital adalah file dalam format wav (*wave file*).

Menurut McCuskey (2003), format file wav merupakan bagian dari spesifikasi RIFF milik Microsoft yang digunakan untuk penyimpanan file-file multimedia. File wav dimulai dengan bagian *header* dan diikuti oleh rentetan *data chunk*. File wav terdiri dari 3 bagian, yaitu *main chunk, format chunk, dan data chunk*.

File wav menyimpan data sinyal suara dalam bentuk diskrit, yaitu berupa deret bilangan yang merepresentasikan amplitudo dalam

domain waktu. Pada bagian *file header* terdapat informasi tentang file wav tersebut, diantaranya menyatakan nilai *sample rate*, jumlah *channel*, dan *bit per sample*. Dari keterangan pada *file header* tersebut dapat diketahui berapa sampel yang dicuplik dari sinyal analog tiap detik. Gambar 2.1 menjelaskan lebih rinci tentang struktur file wav.



Gambar. 2.1 Struktur File Wav (Wilson, 2003)

Sebagaimana telah dijelaskan di muka, bahwa file wav terdiri dari 3 bagian. Bagian-bagian tersebut dijelaskan lebih rinci sebagai berikut (McCuskey, 2003):

a. Bagian *Main Chunk*
   ChunkID : berisi kata "RIFF" dalam format ASCII
   ChunkSize : berisi informasi ukuran *chunk*
   Format : berisi kata "*WAVE*"
b. Bagian *Format Chunk (SubChunk 1)*

SubChunk1ID : berisi kata “fmt”
SubChunk1Size : berisi informasi ukuran *subchunk1*
AudioFormat : informasi jenis kompresi data *chunk*
Misalnya bernilai 1 untuk kompresi PCM
NumChannels : banyaknya *channel*.
Misal: Mono=1, Stereo=2
SampleRate : *sample rate* dari file wav, misal 8000 untuk 8000Hz, 44100 untuk 44100 Hz.
ByteRate : banyaknya byte tiap detik.
ByteRate = *SampleRate* x *NumChannels* x *BitPerSample* / 8
BitsPerSample : ukuran bits untuk tiap sampel.
Misal: 8 bit = 8

c. Bagian *Data Chunk (SubChunk 2)*
SubChunk2ID : berisi kata “data”
SubChunk2Size : berisi informasi ukuran *subchunk2*.
SubChunk2Size = *NumSamples* x *NumChannels* x *BitsPerSample* / 8
Data : Data suara aktual dalam byte, merepresentasikan amplitudo tiap sampel dari sinyal.

Inti dari struktur file wav adalah pada bagian *Data*. Pada bagian tersebut tersimpan data suara aktual, yaitu data amplitudo dari suara yang tersimpan pada file wav tersebut. Nilai-nilai pada bagian *Data* apabila digambar akan membentuk suatu gelombang suara yang merupakan sinyal diskrit dalam domain waktu. Ukuran file wav sangat tergantung pada ukuran bagian *Data* ini. Biasanya untuk suara dengan durasi satu menit, dengan *sample rate* 44100 Hz, dibutuhkan ukuran file wav sekitar 10 MB.

2.2.3 *Windowing*

Sinyal suara dari sebuah lagu berukuran panjang, sehingga akan sulit untuk melakukan pengenalan sekaligus. Oleh karena itu pengenalan dilakukan bagian per bagian. Proses pembagian sinyal suara menjadi bagian-bagian kecil ini dinamakan *windowing*.

Pada proses *windowing*, sinyal dengan ukuran panjang dipecah-pecah menjadi bagian-bagian (*frames*) kecil. Setelah didapat bagian-bagian kecil, dilakukan kuantisasi atau pembobotan. Metode kuantisasi yang ada dalam proses *windowing* adalah *rectangular*

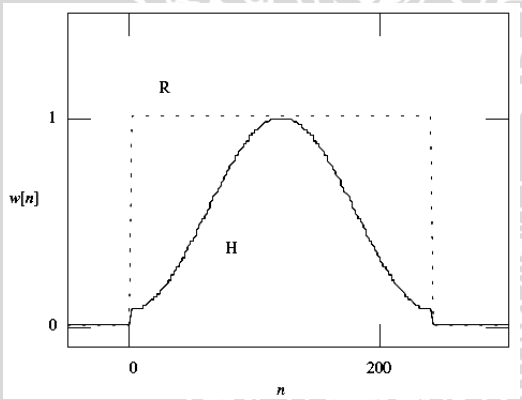
window, *Bartlett window*, *Hanning window*, *Gaussian window*, *Hamming window*, *Blackman window*, *Blackman-Harris window*, *Blackman-Nuttall window*, dan *User-defined wéighting window*. Untuk pengolahan sinyal suara, metode yang sering digunakan adalah *Hamming Window*, yang merupakan penyempurnaan dari metode *Hanning Window*. Metode *Hamming window* dapat dirumuskan (Chu, 2003):

$$w(n) = 0{,}54 - 0{,}46\cos\left(\frac{2\pi n}{N-1}\right) \qquad (2.1)$$

dimana

$$0 \le n \le N-1$$

dengan N merupakan lebar *window*. Dengan menggunakan metode ini, maka bagian awal dan bagian akhir sinyal akan diberi bobot mendekati nol. Gambar 2.2 menunjukkan pola pembobotan dengan metode *hamming window*.



Gambar 2.2 Pembobotan dengan metode *hamming window* (Chu, 2003).

2.2.4 Transformasi *Fourier*

Sinyal yang terekam pada sinyal diskrit merupakan sinyal pada domain waktu. Sedangkan untuk melakukan analisis terhadap sinyal, akan lebih mudah jika diubah terlebih dahulu menjadi sinyal dalam domain frekuensi. Proakis dan Manolakis (1995) mengemukakan bahwa proses transformasi akan mengubah sinyal dalam domain waktu menjadi sinyal dalam domain frekuensi. Pada

tahap ini masing-masing *frame* ditransformasi ke dalam domain frekuensi. Salah satu metode transformasi yang sering digunakan untuk mengubah sinyal dalam domain waktu ke domain frekuensi adalah transformasi fourier, yaitu transformasi yang memanfaatkan deret fourier.

Setiap sinyal dari kelas periodik merupakan kumpulan sejumlah sinusoida-sinusoida dengan magnitudo dan frekuensi yang berbeda. Oleh karena itu sinyal tersebut selalu dapat dinyatakan oleh penjumlahan atas komonen-komponen sinyal sinusoida dengan magnitudo dan frekuensi yang berbeda satu dengan yang lain yang merupakan kelipatan bulat dari frekuensi dasar (Tjokronegoro, 2005).

2.2.4.1 *Discrete Fourier Transform* (DFT)

Untuk mendapatkan sinyal dalam domain frekuensi dari sebuah sinyal diskrit, salah satu metode transformasi fourier yang digunakan adalah transformasi *fourier* diskrit (DFT). DFT dilakukan terhadap masing-masing *frame* dari sinyal yang telah di-*windowing*.

Misalkan diberikan N deret data $\{f(kT)\}\ \forall\ 0 \le k \le N-1$, dengan N adalah jumlah data (genap), T adalah periode *sampling*, $(N-1)T$ adalah panjang data, dan $\Omega = \frac{\omega_s}{N} = \frac{2\pi}{NT}$ adalah satuan frekuensi diskrit, maka set transformasi *Fourier* diskrit (DFT) didefinisikan sebagai (Tjokronegoro, 2005):

$$\begin{aligned} F(n\Omega) &= F\left[f(kT)\right] \\ &= \sum_{k=0}^{N-1} f(kT)e^{-jn\Omega kT}\ ,\ \text{n} = 0,1,2,\ldots,N-1 \end{aligned} \qquad (2.2)$$

Transformasi di atas adalah periodik, yang dapat ditunjukkan dari sifat fungsi $e^{-jn\Omega kT}$ yang periodik, yaitu $e^{-jn\Omega kT} = e^{j(n+N)kT}$, $k,n = 0,\pm1,\pm2,\ldots$. Dalam notasi umum $F(n\Omega)$ adalah kompleks, sehingga dapat dituliskan sebagai:

$$F(n\Omega) = |F(n\Omega)|e^{j\phi(n\Omega)} \qquad (2.3)$$

Selanjutnya, untuk menyingkat penulisan, transformasi *Fourier* diskrit dapat ditulis sebagai:

$$F(n) = \sum_{k=0}^{N-1} f(k) W^{nk} \quad , \; n = 0,1,2,\ldots,N-1 \qquad (2.4)$$

dengan

$$W = e^{-j\Omega T} = e^{-j\frac{2\pi}{N}} \qquad (2.4.1)$$
$$= \cos\left(\frac{2\pi}{N}\right) - j \sin\left(\frac{2\pi}{N}\right)$$

Menurut Brigham (1988), sebuah magnitudo sinyal bisa didapat dari akar kuadrat penjumlahan sinyal real dan imajinernya. Jika *re*(*n*) adalah bagian real transformasi dan *im*(*n*) adalah bagian imajiner transformasi, maka sinyal *x*(*n*) akan mempunyai spektrum magnitudo atau spektrum fourier $\sqrt{re(n)^2 + im(n)^2}$. Sedangkan Oppenheim (2000) mengemukakan bahwa bila *x*(*n*) adalah sinyal real, maka *im*(*n*) adalah nol sehingga magnitudonya adalah *x*(*n*) itu sendiri.

2.2.4.2 *Fast Fourier Transform* (FFT)

Discrete Fourier Transform (DFT) adalah suatu instrumen matematika yang efektif dan penting untuk analisis sinyal dengan menggunakan komputer. Namun yang menjadi persoalan adalah bahwa transformasi *Fourier* diskrit tersebut memerlukan waktu komputasi yang sangat panjang. Oleh karena itu diperlukan suatu teknik komputasi yang efisien, baik dari sisi waktu maupun dari sisi penggunaan memori.

Banyak teknik yang telah dikembangkan untuk tujuan efisiensi algoritma komputasi transformasi *Fourier*. Teknik *Fast Fourier Transform* (FFT) pertama kali dikemukakan oleh Cooley-Tukey pada tahun 1965 (Tjokronegoro, 2005). Sejak itu, hampir semua teknik FFT didasarkan pada sifat simetri dan sifat periodik dari komponen W^{nk}:

a. Complex Conjugate Symetry

$$W^{k(N-n)} = W^{(N-k)n} = W^{-kn} = \left(W^{kn}\right) \qquad (2.5)$$

b. *Periodicity* dalam n dan k

$$W^{kn} = W^{k(N+n)} = W^{(N+k)n} \tag{2.6}$$

Inti dari FFT adalah memecah persamaan DFT dengan ukuran N menjadi persamaan DFT yang terdiri dari 2 titik saja. Dari sifat-sifat yang ditunjukkan persamaan 2.5 dan 2.6, persamaan DFT dapat disederhanakan menjadi FFT. Misalkan untuk N=8, diperoleh formula FFT sebagai berikut:

$$n = 4n_2 + 2n_1 + n_0 \text{ dan } k = 4k_2 + 2k_1 + k_0 \tag{2.7}$$

sehingga dihasilkan:

$$F(n_2, n_1, n_0) = \sum_{k_0=0}^{1}\sum_{k_1=0}^{1}\sum_{k_2=0}^{1} f(k_2, k_1, k_0) W^{(4n_2+2n_1+n_0)(4k_2+2k_1+k_0)} \tag{2.8}$$

dimana

$$W = \left(e^{-j\frac{2\pi}{N}} \right)$$

Hasil transformasi ini dipengaruhi oleh beberapa parameter, yaitu *sample rate* sinyal suara dan FFT *size*. *Sample rate* sinyal suara berpengaruh pada besarnya jangkauan frekuensi dari koefisien hasil FFT. Jangkauan frekuensi hasil FFT adalah setengah dari *sample rate* sinyal suara yang ditransformasi. Artinya apabila terdapat sinyal suara dengan *sample rate* 44100 Hz, maka koefisien-koefisien hasil transformasi dari sinyal suara tersebut berkisar dari 0 Hz sampai 22050 Hz.

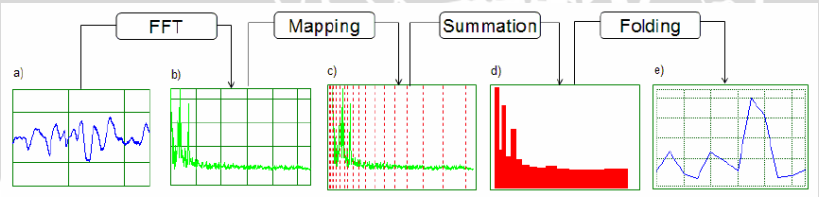
Sedangkan besar FFT *size* berpengaruh terhadap ketelitian tiap koefisien FFT. Semakin besar FFT *size*, maka tiap koefisien hasil FFT akan mewakili rentang frekuensi yang semakin kecil, sehingga ketelitiannya semakin tinggi. Sebaliknya apabila ukuran sampel FFT semakin kecil, maka tiap koefisien hasil FFT akan mewakili rentang frekuensi yang semakin besar, sehingga ketelitiannya semakin rendah.

2.3 *Pitch Class Profile*

Pitch Class Profile (PCP) adalah salah satu fitur yang paling sering digunakan dalam pengenalan *chord*, sejak diperkenalkan pertama kali oleh Fujishima pada tahun 1999. PCP merupakan sebuah vektor dengan 12 koefisien yang merepresentasikan intensitas relatif magnitudo dari tiap nada dalam tangga nada kromatik (Lee dan Slaney, 2006).

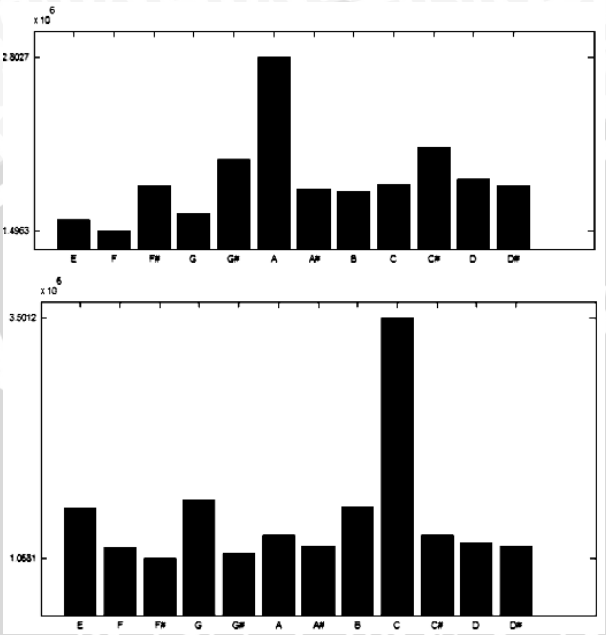
Intensitas nada pada PCP merupakan representasi intensitas nada tersebut pada semua oktaf yang ada pada data suara. Jadi nada A pada suatu oktaf dan nada A pada semua oktaf yang lain, baik oktaf lebih tinggi atau lebih rendah, diwakili oleh satu data pada PCP.

Penyusunan PCP mengacu pada hasil transformasi sinyal digital dalam domain frekuensi. Tiap data hasil dari transformasi sinyal dalam domain frekuensi dipetakan ke satu nada (yang memiliki frekuensi terdekat) dari 12 nada pada PCP. PCP merupakan kombinasi *magnitude* dari frekuensi-frekuensi yang bersesuaian dengan tiap nada pada tangga nada kromatik dari sinyal dalam domain frekuensi.



Gambar 2.3 Langkah-langkah penyusunan PCP (Cabral, dkk., 2005)

Pada gambar 2.3 Dapat dilihat langkah-langkah penyusunan PCP. Data suara hasil transformasi dengan metode FFT dipetakan dalam daerah yang sesuai dengan frekuensi dari 12 nada kromatik. Kemudian magnitudo tiap data pada suatu region dijumlahkan. Langkah selanjutnya setiap region yang mewakili suatu nada kromatik digabung dan dijumlahkan dengan region lain yang mewakili nada yang sama, hingga akhirnya didapat 12 keofisien yang mewakili 12 nada kromatik. Gambar 2.4 menunjukkan contoh *Pitch Class Profile*.



Gambar 2.4 Contoh Pitch Class Profiles (PCP) untuk chord Amaj7 (atas) dan Cmaj7 (bawah) (Cabrall dkk, 2005)

PCP digunakan sebagai input untuk sistem pengenalan *chord*. Untuk dapat digunakan sebagai input sistem pengenalan *chord* PCP bisa langsung digunakan, ataupun dengan cara memetakannya dalam suatu skala (Cabrall dkk, 2005). Misalnya dipetakan dengan bilangan 0 – 1.

2.4 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan adalah paradigma pemrosesan suatu informasi yang terinspirasi oleh sistem sel syaraf biologi, sama seperti otak yang memproses suatu informasi. Elemen mendasar dari paradigma tersebut adalah struktur yang baru dari sistem pemrosesan informasi (Yani, 2005). Jaringan syaraf tiruan, seperti manusia, belajar dari suatu contoh. Jaringan syaraf tiruan dibentuk untuk memecahkan suatu masalah tertentu seperti pengenalan pola atau klasifikasi melalui proses pembelajaran.

Jaringan syaraf tiruan telah dikembangkan dengan menggunakan model matematis untuk menirukan cara kerja jaringan syaraf biologis, dengan berdasarkan asumsi (Kusumadewi, 2003):

a. Pengolah informasi terdiri dari elemen-elemen sederhana yang disebut neuron.
b. Sinyal dilewatkan dari satu neuron ke neuron yang lain melalui hubungan koneksi.
c. Tiap hubungan koneksi mempunyai nilai bobot tersendiri.
d. Tiap neuron mempergunakan fungsi aktivasi (biasanya tidak linear) terhadap masukan yang diterimanya untuk menentukan sinyal keluarannya.

Jaringan syaraf tiruan memiliki pendekatan yang berbeda untuk memecahkan masalah dibandingkan dengan sebuah komputer konvensional. Umumnya komputer konvensional menggunakan pendekatan algoritma (komputer konvensional menjalankan sekumpulan perintah untuk memecahkan masalah). Jika suatu perintah tidak diketahui oleh komputer konvensional maka komputer konvensional tidak dapat memecahkan masalah yang ada.

Sangat penting mengetahui bagaimana memecahkan suatu masalah pada komputer konvensional dimana komputer konvensional akan sangat bermanfaat jika dapat melakukan sesuatu dimana pengguna belum mengatahui bagaimana melakukannya. Dalam hal ini penggunaan jaringan syaraf tiruan dibutuhkan untuk mengenali pola *chord* baik yang sudah dilatihkan maupun yang belum pernah dilatihkan.

2.4.1 Arsitektur Jaringan Syaraf Tiruan

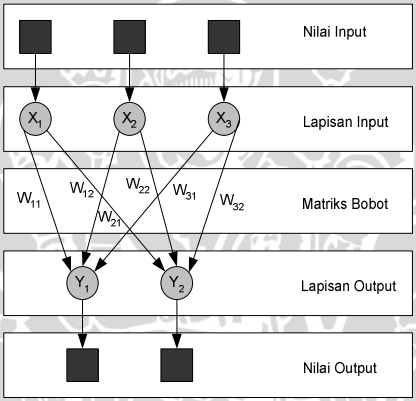
Jaringan syaraf tiruan terdiri dari neuron-neuron yang dikelompokkan dalam berbagai lapisan. Pada umumnya, lapisan-lapisan yang berada pada lapisan yang sama akan memiliki keadaan yang sama. Faktor yang menentukan keadaan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Untuk neuron pada lapisan yang sama akan memiliki fungsi aktivasi yang sama pula. Apabila neuron-neuron yang terletak pada lapisan yang sama akan dihubungkan dengan neuron-neuron pada lapisan yang lain, maka setiap neuron yang terletak pada lapisan tersebut juga harus dihubungkan dengan setiap lapisan pada lapisan lainnya. Misalkan akan dihubungkan

neuron-neuron pada lapisan tersembunyi dengan neuron-neuron pada lapisan output, maka setiap neuron pada lapisan tersembunyi harus dihubungkan dengan setiap neuron pada lapisan output.

Ada beberapa arsitektur jaringan syaraf tiruan, diantaranya (Kusumadewi, 2003):

a. Jaringan dengan lapisan tunggal (*single layer net*)

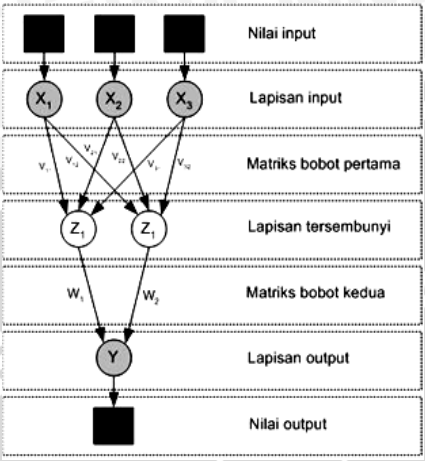
Sesuai dengan namanya, jaringan ini hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima input kemudian secara langsung akan mengolahnya menjadi output tanpa melalui lapisan tersembunyi. Gambar 2.5 menunjukkan arsitektur jaringan syaraf tiruan dengan lapisan tunggal.



Gambar 2.5 Jaringan syaraf dengan lapisan tunggal

b. Jaringan dengan banyak lapisan (*multilayer net*)

Jaringan dengan banyak lapisan mempunyai satu atau lebih lapisan yang terletak diantara lapisan input dengan lapisan output. Lapisan ini disebut sebagai lapisan tersembunyi (*hidden layer*). Jaringan yang mempunyai banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan tunggal, tentu dengan pembelajaran yang lebih rumit. Akan tetapi, pada beberapa kasus, pembelajaran pada jaringan yang mempunyai banyak lapisan ini lebih sukses dalam menyelesaikan masalah. Gambar 2.6 menunjukkan arsitektur jaringan syaraf tiruan dengan banyak lapisan.



Gambar 2.6 Jaringan syaraf dengan banyak lapisan

2.4.2 Proses Pembelajaran

Jaringan syaraf akan mencoba untuk mensimulasikan kemampuan otak manusia untuk belajar. Jaringan syaraf tiruan juga tersusun atas neuron-neuron dan dendrit. Tidak seperti model biologis, jaringan syaraf memiliki struktur yang tidak dapat diubah, dibangun oleh sejumlah neuron, dan memiliki nilai tertentu yang menunjukkan seberapa besar koneksi antar neuron (yang dikenal dengan nama bobot). Perubahan yang terjadi selama proses pembelajaran adalah perubahan nilai bobot. Nilai bobot akan bertambah jika informasi yang diberikan oleh neuron yang bersangkutan tersampaikan, sebaliknya jika informasi tidak disampaikan oleh neuron ke neuron yang lain, maka nilai bobot yang menghubungkan keduanya akan dikurangi. Pada saat pembelajaran dilakukan pada input yang berbeda, maka nilai bobot akan diubah secara dinamis hingga mencapai suatu nilai yang cukup seimbang. Apabila nilai ini telah tercapai, mengindikasikan bahwa tiap-tiap input telah berhubungan dengan output yang diharapkan.

Ada dua jenis proses pembelajaran yaitu (Kusumadewi, 2003):

a. Pembelajaran terawasi (*supervised learning*)
   Metode pembelajaran pada jaringan syaraf disebut terawasi jika target output yang diharapkan telah diketahui sebelumnya.
b. Pembelajaran tak terawasi (*unsupervised learning*)

Pada metode pembelajaran yang tak terawasi ini tidak memerlukan target output. Pada metode ini tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran.

Di dalam proses pembelajaran, *learning rate* adalah sebuah variabel yang nilainya diantara 0 sampai 1 yang berguna untuk menentukan laju pembelajaran. Sedangkan *max epoch* adalah batas maksimum perulangan untuk melakukan proses pembelajaran.

2.5 Learning Vector Quantization

Learning Vector Quantization (LVQ) merupakan salah satu metode dalam jaringan syaraf tiruan yang telah digunakan secara luas dalam proses pengenalan pola. LVQ pertama kali diperkenalkan oleh Teuvo Kohonen sebagai algoritma *clustering* yang didasarkan pada sekumpulan prototipe. Pemikiran tersebut dikombinasikan dengan *self-organizing learning* namun menggunakan pelatihan yang terawasi. Kegunaan metode LVQ sering dijumpai pada aplikasi-aplikasi *data-mining*, robotik, maupun pengenalan linguistik (Kohonen, dkk., 1996).

LVQ merupakan metode yang mudah dipahami dan diterapkan. Penentuan klasifikasi suatu dari sekumpulan vektor dilakukan dengan membandingkan jarak *euclidean* vektor-vektor tersebut dengan sejumlah vektor-vektor prototipe yang merepresentasikan kelas dari sekumpulan data dalam klasifikasi yang sama.

LVQ adalah suatu metode untuk melakukan pembelajaran pada lapisan kompetitif yang terawasi. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor input. Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitif ini hanya tergantung pada jarak antara vektor-vektor input. Jika 2 vektor input mendekati sama maka lapisan kompetitif akan meletakkan kedua vektor input tersebut ke dalam kelas yang sama (Kusumadewi, 2003).

Pada proses pembelajaran, suatu vektor prototipe akan mengalami perubahan yang akan semakin memperjelas klasifikasi. Adapun vektor prototipe yang mengalami perubahan tersebut adalah vektor pemenang, yaitu vektor prototipe yang terdekat (jarak *Euclidean*-nya) terhadap vektor input. Dalam hal ini berlaku skema *Winner Takes All (WTA)*, karena perubahan signifikan hanya dialami

oleh vektor prototipe pemenang dengan jarak Euclidean terkecil (Biehl, 2005).

2.5.1 *LVQ* 1

LVQ 1 ini merupakan algoritma dasar LVQ. Inti dari algoritma ini adalah pelatihan yang bertujuan untuk mengklasifikasikan vektor input ke dalam kelas prototipe yang tersedia.

Proses pembelajaran pada algoritma LVQ 1 diawali dengan inisialisasi secara acak untuk memilih vektor-vektor yang berlaku sebagai prototipe. Kemudian dilakukan perulangan hingga ditemukan kondisi berhenti, dimana dalam setiap perulangan dilakukan penentuan prototipe yang terdekat dari vektor input. Kemudian vektor protitipe tersebut akan mengalami perubahan yang bersifat mendekati atau menjauh dari vektor input, tergantung pada kecocokan terhadap target pelatihan. Apabila target pelatihan sesuai dengan vektor prototipe terdekat dari vektor input, maka vektor prototipe akan berubah mendekati vektor input. Sebaliknya apabila target pelatihan tidak sesuai dengan vektor prototipe terdekat dari vektor input, maka vektor prototipe akan berubah menjauhi vektor input. Penentuan kedekatan antara vektor input ke vektor prototipe ditentukan dengan menggunakan jarak *euclidean.*

Persamaan 2.5, 2.6. 2.7.a, dan 2.7.b menjelaskan proses dasar yang terjadi pada algoritma LVQ 1.

$$c = \arg\min \{\| x - m_i \|\} \tag{2.9}$$

$$d = \sum_{i=1}^{n} | x_i - w_i | \tag{2.10}$$

$m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)]$ (2.11.a)
Jika x dan m_c termasuk ke dalam kelas yang sama

$m_c(t+1) = m_c(t) - \alpha(t)[x(t) - m_c(t)]$ (2.11.b)
Jika x dan m_c tidak termasuk ke dalam kelas yang sama

$m_i(t+1) = m_i(t)$ (2.11.c)
Untuk vektor-vektor lain dimana $i \neq c$.

Dimisalkan terdapat sejumlah *i* vektor prototipe, m_i (disebut juga *codebook vector*). Vektor m_c merupakan vektor prototipe dengan jarak *Euclidean d* terkecil, dimana *c* didapatkan dari persamaan 2.9, dan jarak Euclidean *d* didapat dengan persamaan 2.10. Pada persamaan 2.11.a dan 2.11.b, m_c merupakan vektor prototype yang terdekat dengan vektor input *x*. Sehingga m_c diperbarui (*update*) sebelum digunakan pada perulangan berikutnya. Besarnya *update* yang dialami vektor m_c tergantung pada nilai laju pelatihan α. Nilai laju pelatihan α dapat berupa konstanta ataupun variabel yang berubah secara monoton terhadap iterasi (Kohonen, 1995).

2.5.2 *LVQ* 2.1

Algoritma LVQ 2.1 digunakan untuk memperbaiki algoritma sebelumnya, LVQ 1. Perbaikan dilakukan pada percepatan pelatihan dengan melibatkan 2 vektor prototipe pada *update* vektor. Jika pada LVQ 1 vektor yang diperbarui adalah vektor yang paling dekat dengan vektor input, maka pada LVQ 2.1 vektor yang diperbarui adalah 2 vektor dari kelas berbeda yang merupakan vektor terdekat yang sesuai dengan kelas target pelatihan, dan vektor terdekat yang tidak sesuai dengan kelas target pelatihan.

Vektor prototipe dengan kelas yang sesuai dengan target diperbarui mendekati vektor input. Sedangkan vektor protipe dengan kelas yang tidak sesuai dengan target yang diperbarui menjauh dari vektor input. Proses perbaruan bobot untuk vektor-vektor prototipe pada LVQ 2.1 berdasarkan persamaan 2.12.

$$m_i(t+1) = m_i(t) + \alpha(t)[x(t) - m_j(t)] \quad (2.12.a)$$

$$m_j(t+1) = m_j(t) - \alpha(t)[x(t) - m_j(t)] \quad (2.12.b)$$

Vektor m_i dan m_j adalah dua vektor prototipe yang terdekat dengan vektor input x(t), dimana m_i adalah vektor yang berada dalam kelas yang sama dengan kelas target, sedangkan m_j adalah vektor yang berada dalam kelas yangberbeda dengan kelas target (Kohonen, 1995).

2.5.3 Kondisi Berhenti

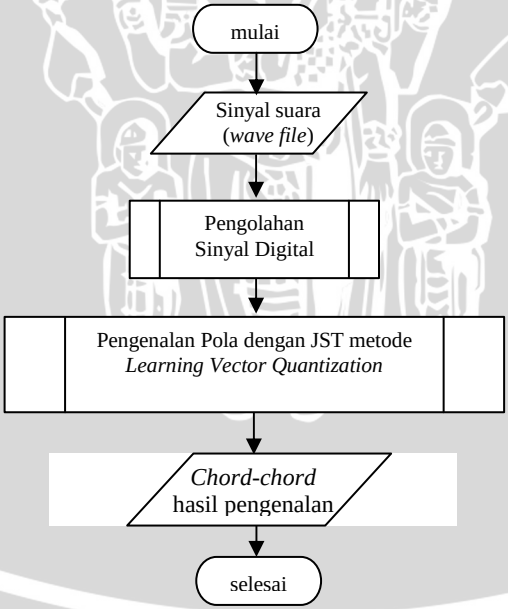
Seringkali dalam praktek terjadi terlalu banyak pembelajaran yang dilakukan. Hal ini tidak selamanya berpengaruh baik terhadap akurasi pengenalan. Ketika perulangan yang terjadi pada pembelajaran mencapai suatu titik tertentu, maka akurasi yang didapatkan mencapai tingkat optimal. Apabila perulangan pembelajaran tetap dilanjutkan melewati titik optimal tersebut, maka akurasi pengenalan akan mengalami penurunan akurasi secara lambat. Untuk mendapatkan titik dimana akurasi mencapai tingkat optimal, sangat tergantung pada kasus dan arsitektur jaringan syaraf, oleh karena itu, titik optimal tersebut hanya dapat ditentukan dari hasil percobaan (Kohonen, 1995).

BAB III
METODOLOGI DAN PERANCANGAN

3.1 Deskripsi Umum Sistem

Sistem pengenalan yang akan dibuat sebagai alat penelitian adalah sistem yang digunakan untuk mengenali *chord* dari sebuah lagu. Output dari sistem ini berupa *chord* hasil pengenalan, merupakan urutan *chord* yang mengiringi sebuah lagu.

Secara garis besar sistem ini terdiri dari dua tahap, seperti digambarkan pada Gambar 3.1. Tahap pertama yaitu tahap pengolahan sinyal digital. Inti dari tahap ini adalah untuk mengolah sinyal berbentuk musik dalam format file wav menjadi sebuah pola yang akan digunakan sebagai input dalam pengenalan dengan metode jaringan syaraf tiruan LVQ. Pola yang terbentuk merupakan pola magnitudo dari 12 frekuensi yang bersesuaian dengan frekuensi 12 nada kromatik. Tahap kedua adalah proses pengenalan *chord* dengan menggunakan LVQ. Pada tahap ini output pada proses tahap pertama menjadi input yang kemudian akan diproses dan menghasilkan pola *chord* yang sesuai dengan input.



Gambar 3.1 Diagram alir sistem pengenalan *chord*

3.2 Batasan Sistem

Sistem yang dibuat memiliki batasan-batasan:

- Input sistem hanya berupa file musik dalam format wav dengan *sampel rate* 44100 Hz.
- Pengenalan dibatasi hanya untuk 24 tipe *chord*, yaitu 12 *chord* major dan 12 *chord* minor.
- Output sistem berupa rangkaian *chord* penyusun sebuah lagu.

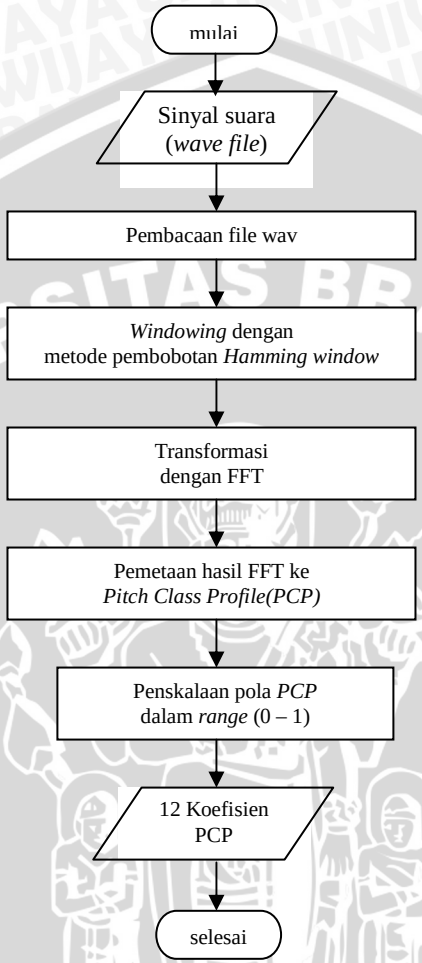
3.3 Pengolahan Sinyal Digital

Proses pengolahan sinyal digital yang akan dilakukan dapat dibagi lagi menjadi beberapa tahap, seperti digambarkan pada Gambar 3.2. Pada pengolahan sinyal digital ini, sinyal suara yang diolah sudah berbentuk digital, berupa sinyal suara dalam format file wav.

3.3.1 Pembacaan File Wav

Sinyal suara dalam bentuk wav merupakan sekumpulan data dalam format bilangan heksadesimal, dengan ketentuan-ketentuan sebagaimana informasi yang ditunjukkan pada bagian *header* file wav. Ketentuan-ketentuan tersebut diantaranya berupa *sample rate*, *bit per second*, *byte per sample*. Informasi-informasi tersebut akan sangat dibutuhkan dalam proses pembacaan data file wav untuk kemudian disimpan dalam variabel global yang lebih terstruktur dan memudahkan untuk pemrosesan selanjutnya.

Pada penelitian ini pembacaan file wav dilakukan dengan menggunakan *filestream*, kemudian data suara pada bagian *data chunk* dari file wav disimpan dalam sebuah variabel array bertipe `smallint`. Data file wav dalam variabel tersebut sudah dalam bentuk bilangan bulat, dan merupakan representasi dari amplitudo tiap sampel data suara, termasuk data dari dua *channel* apabila data suara merupakan data suara stereo.



Gambar 3.2 Proses pengolahan sinyal digital

Untuk data suara stereo dengan 2 *channel* (*left channels* dan *right channel*). Sampel suara ke-*n* dari *left channel* disimpan pada Data(0,*n)* dan sampel suara ke-*n* dari *right channel* disimpan pada Data(1,*n*). Sedangkan untuk data suara mono dengan hanya 1 *channel* suara, sampel suara ke-*n* dianggap sebagai *left channel* dan disimpan ke dalam array Data(0, *n*), dan sampel suara dari *right channel* dianggap tidak ada sehingga array Data(1,*n*) dibiarkan

kosong. Gambar 3.3 menunjukkan representasi data dalam variabel hasil pembacaan file wav.

index		0	1	2	...	*n*
left sample	0	12	15	28	...	292
right sample	1	13	6	-8	...	-120

Gambar 3.3 Representasi data file wav dalam variabel

3.3.2 *Windowing*

Sinyal suara dari sebuah lagu berukuran panjang, sehingga akan sulit untuk melakukan pengenalan sekaligus. Oleh karena itu pengenalan dilakukan bagian per bagian. Proses pembagian sinyal suara menjadi bagian-bagian atau blok-blok kecil ini dinamakan *windowing*.

Langkah pertama dalam *windowing* adalah menentukan lebar tiap *frame* sehingga akan didapat N sampel tiap *frame*. Karena hasil proses *windowing* ini akan digunakan untuk transformasi menggunakan metode FFT, dimana metode tersebut mengharuskan lebar *frame* merupakan sebuah bilangan pangkat dari 2, maka nilai N juga harus merupakan bilangan pangkat dari 2.

Dalam penelitian ini digunakan nilai N=16384 atau $N=2^{14}$. Dengan demikian setiap *frame* akan melibatkan 16384 titik sampel. Lebar *frame* dalam *windowing* ini disebut pula sebagai FFT *size*, karena juga merupakan ukuran panjang sampel yang akan ditransformasikan dengan metode FFT.

Pembagian sinyal ke dalam *frame* dilakukan dari awal bagian sinyal. Dengan nilai N=16384, maka *frame* pertama akan berisi data sampel ke-1 hingga ke-16384. Untuk *frame* berikutnya, awal *frame* ke-*t* berikutnya diisi oleh data sampel ke-*n* dengan:

$$n = \mathrm{N} \times (t\text{-}1) + 1 \qquad (3.1)$$

Pembagian sinyal seperti ini dilakukan pada tiap *channel* sinyal suara. Artinya, untuk sinyal stereo, antara *left channel* dengan *right channel* mendapatkan perlakuan yang sama, sehingga didapatkan 2 *frame* dengan indeks data yang sama, namun dengan data yang sangat mungkin berbeda.

Langkah berikutnya adalah pembobotan masing-masing *frame*. Pembobotan dilakukan dengan metode *Hamming window*. Tiap data

dari sinyal yang telah terbagi ke dalam *frame-frame* pada array Data(c, n) dikalikan dengan *w*(*n*) dengan:

$$w(n) = 0{,}54 - 0{,}46\cos\left(\frac{2\pi n}{N-1}\right) \qquad (3.2)$$

dimana N=FFT *size*, dan *n* adalah urutan data sampel pada suatu *frame*.

Hasil dari proses *windowing* ini berupa sinyal yang telah terbagi dalam *frame-frame* kecil dengan tiap *frame* menyimpan nilai amplitudo dari sinyal suara, dimana akibat proses pembobotan dengan metode *Hamming window* maka nilai amplitudo pada awal dan akhir *frame* mendekati nol.

3.3.3 Transformasi

Selanjutnya untuk mendapatkan koefisien dalam domain frekuensi dari sebuah sinyal digital dalam domain waktu, sinyal yang telah terboboti pada tiap *frame* ditransformasi dengan metode *Fast Fourier Transform* (FFT).

Dalam FFT, data sampel pada suatu *frame* harus dipecah-pecah lagi menjadi *n* bagian sehingga tiap-tiap bagian kecil tersebut terdiri dari 2 sampel suara. Bagian-bagian kecil ini nantinya akan dihitung untuk menghasilkan koefisien FFT dengan menggunakan persamaan DFT:

$$F(n) = \sum_{k=0}^{1} f(k) W^{nk} \quad , \; \mathrm{n} = 0,1,2,\ldots,N-1 \qquad (3.3)$$

dengan

$$W = e^{-j\Omega T} = e^{-j\frac{2\pi}{N}}$$
$$= \cos\left(\frac{2\pi}{N}\right) - j\sin\left(\frac{2\pi}{N}\right) \qquad (3.3.1)$$

Proses pemecahan *frame* FFT menjadi 2-point DFT memiliki aturan tersendiri. Misalkan nilai FFT *size* adalah 8, sehingga didapat *frame* FFT dengan urutan sample dalam index *0, 1, 2, 3, 4, 5, 6, 7* (indeks mulai nol). Setelah proses pemecahan maka akan didapatkan 4 pasang sampel data dengan indeks (0,4), (2,6), (1,5), (3,7).

Urutan hasil pemecahan di atas sekilas seperti tidak terpola. Untuk mendapat urutan seperti di atas, akan lebih mudah

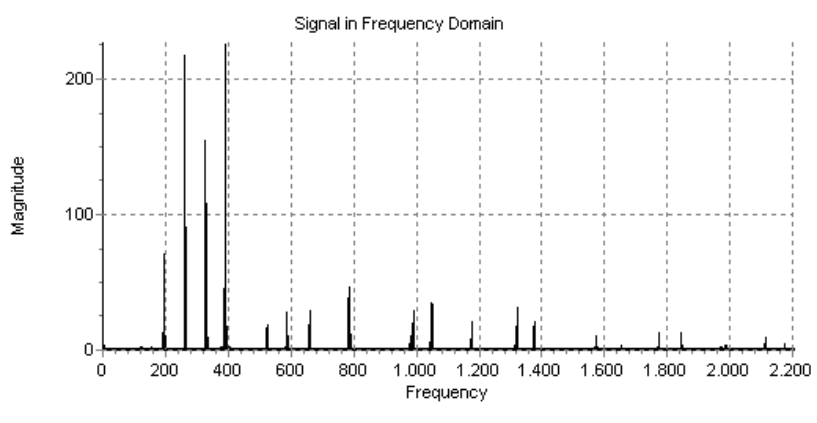
mengubahnya dahulu ke dalam format bilangan biner. Kemudian membaliknya dan mengubahnya kembali ke format bilangan desimal.

Urutan awal			Urutan hasil	
desimal	Biner		biner	desimal
0	000	→	000	0
1	001	→	100	4
2	010	→	010	2
3	011	→	110	6
4	100	→	001	1
5	101	→	101	5
6	110	→	011	3
7	111	→	111	7

Gambar 3.4 Representasi pemecahan *frame* menjadi 2-*point* DFT

Hasil dari perhitungan tiap-tiap bagian kecil *frame* dengan persamaan DFT diatas berupa deret dari pasangan bilangan real dan imajiner. Kemudian deret pasangan bilangan itu dikalkulasi lebih lanjut sehingga didapatkan koefisien magnitudo yang mewakili rentang frekuensi tertentu.

Dalam penelitian ini, karena nilai FFT *size* sebesar 16384 dengan masukan berupa file wav dengan *sample rate* 44100 Hz, maka akan dihasilkan sinyal dalam domain frekuensi dengan lebar frekuensi hasil setengah dari *sample rate* yaitu 0 Hz sampai dengan 22050 Hz. Sinyal dalam domain frekuensi tersebut berupa rangkaian koefisien-koefisien magnitudo sejumlah setengah kali FFT *size* yaitu 8192 koefisien. Setiap koefisien tersebut mewakili rentang frekuensi sepanjang 22050/8192 atau 2,692 Hz. Dengan demikian secara berturut-turut, koefisien pertama dari hasil FFT merupakan representasi dari magnitudo dari sinyal untuk frekuensi 0 Hz sampai 2,692 Hz, koefisien kedua merupakan representasi magnitudo sinyal untuk frekuensi 2,693 Hz sampai 5,383 Hz, demikian seterusnya sampai koefisien terakhir (koefisien ke-8192) yang merupakan representasi dari magnitudo sinyal untuk frekuensi sampai 22047,308 Hz - 22050 Hz.

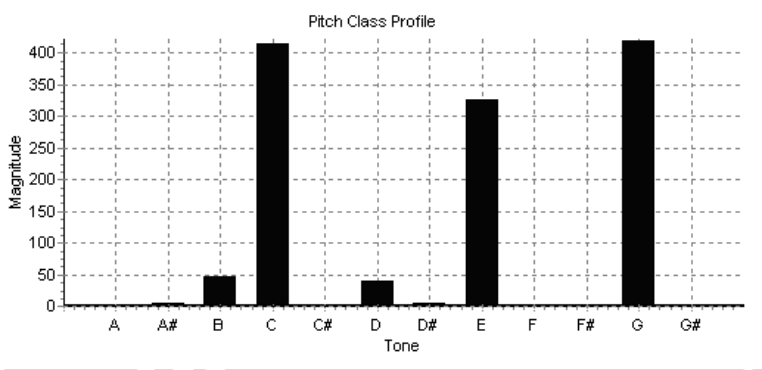


Gambar 3.5 Hasil FFT dari sinyal sampel *chord* C major dengan ukuran sampel 16384, *sample rate* 44100 Hz (dipotong dari frekuensi 0 Hz – 2200 Hz)

3.3.4 Pemetaan *Pitch Class Profile*

Koefisien dalam domain frekuensi tiap *frame* hasil FFT kemudian dipetakan menurut *pitch class* yang sama (termasuk *pitch* yang sama walaupun dalam oktaf yang berbeda). Proses ini akan menghasilkan koefisien *Pitch Class Profil (PCP)*, yang merupakan representasi magnitudo berdasarkan frekuensi yang bersesuaian dengan 12 macam nada dalam tangga nada kromatik.

Untuk menghasilkan output yang siap digunakan sebagai input pada tahap selanjutnya, yaitu proses pengenalan jaringan syaraf tiruan, maka 12 koefisien dalam *Pitch Class Profile* diskala ke dalam bilangan 0 sampai 1. Hasil dari proses ini berupa 12 koefisien dalam rentang 0 sampai 1 yang merupakan representasi dari *Pitch Class Profile* yang telah diskala. Pola tersebut digunakan sebagai input untuk jaringan syaraf tiruan.



Gambar 3.6 *Pitch Class Profile* dari sinyal suara sampel *chord* C Major sebelum diskala

	index
0,02	1
0,03	2
0,12	3
0,96	4
0,01	5
0,09	6
0,03	7
0,77	8
0,01	9
0,01	10
1,00	11
0,01	12

Gambar 3.7 Representasi *Pitch Class Profile* dari sinyal suara sampel *chord* C Major setelah diskala dalam variabel array PCP

3.4 Pengenalan dengan Jaringan Syaraf Tiruan

Tahap pengenalan dengan menggunakan jaringan syaraf tiruan ini terdiri dari 2 macam proses. Proses pertama yaitu proses pelatihan/pembelajaran. Pada proses ini, dilakukan pelatihan untuk tiap-tiap *chord* yang akan digunakan sebagai data pengenalan. Sedangkan proses kedua pada tahap ini yaitu proses pengenalan *chord*.

3.4.1. Kelas Pengenalan

Pada sistem pengenalan *chord* otomatis ini, output dari sistem diklasifikasikan ke dalam 24 kelas. Daftar 24 kelas tersebut dapat dilihat pada tabel 3.1.

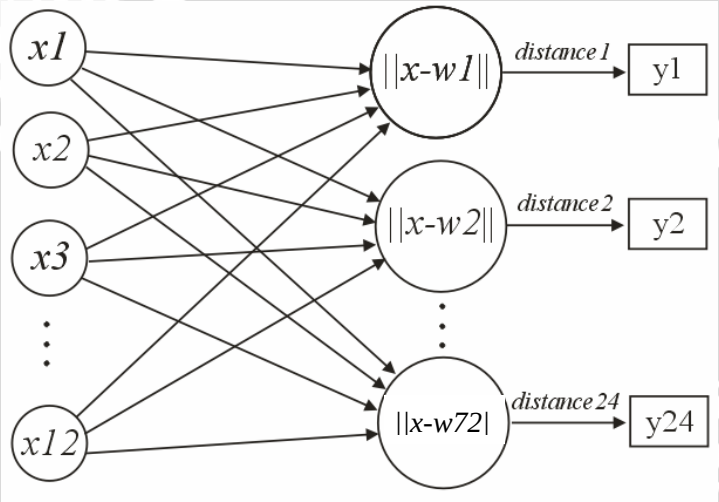
Tabel 3.1 Daftar Kelas Pengenalan *Chord*

Index	Kelas	Nada Penyusun	Keterangan
0	Am	A, C, E	A minor
1	A	A, C#, E	A Major
2	A#m	A#, C#, F	A# minor
3	A#	A#, D, F	A# Major
4	Bm	B, D, F#	B minor
5	B	B, D#, F#	B Major
6	Cm	C, D#, G	C minor
7	C	C, E, G	C Major
8	C#m	C#, E, G#	C# minor
9	C#	C#, F, G#	C# Major
10	Dm	D, F, A	D minor
11	D	D, F#, A	D Major
12	D#m	D#, F#, A#	D# minor
13	D#	D#, G, A#	D# Major
14	Em	E, G, B	E minor
15	E	E, G#, B	E Major
16	Fm	F, G#, C	F minor
17	F	F, A, C	F Major
18	F#m	F#, A, C#	F# minor
19	F#	F#, A#, C#	F# Major
20	Gm	G, A#, D	G minor
21	G	G, B, D	G Major
22	G#m	G#, B, D#	G# minor
23	G#	G#, C, D#	G# Major

3.4.2. Arsitektur Jaringan Syaraf Tiruan

Arsitektur jaringan syaraf tiruan dengan metode *Learning Vector Quantization* yang digunakan pada penelitian ini merupakan arsitektur jaringan dengan tipe jaringan lapis-tunggal umpan-maju (*Single Layer Feedforward*) yang terdiri atas unit masukan ($x_1, x_2, \ldots, x_{12}$) dan unit keluaran (Y_m). Gambar 3.8 menunjukkan arsitektur jaringan syaraf tiruan LVQ yang digunakan pada penelitian ini.

Unit masukan (x_n) berupa 12 neuron yang merepresentasikan nilai-nilai 12 koefisien *Pitch Class Profile* sebagai vektor masukan, dimana setiap vektor masukan tersebut mempunyai nilai dalam rentang 0 sampai 1. Unit masukan tersebut dalam implementasi disimpan dalam variabel array *x* bertipe real.



Gambar 3.8 Jaringan syaraf *Learning Vector Quantization*

Sedangkan unit keluaran terdiri dari 72 neuron (y_1, y_2, ... , y_{24}) yang merepresentasikan vektor-vektor prototipe yang merujuk pada kelas-kelas yang berbeda dalam pengenalan *chord*. Setiap kelas diwakili oleh 3 unit keluaran, hal ini untuk mengakomodasi variasi *chord* yang terjadi.

3.4.3 Proses Pembelajaran Jaringan dengan LVQ

Proses pembelajaran pada jaringan syaraf tiruan berlangsung secara otomatis, namun karena merupakan pembelajaran terawasi (*supervised*) maka target dari pembelajaran ditentukan sebelumnya. Inti proses pembelajaran jaringan syaraf tiruan ini adalah pembaruan bobot dari unit prototipe yang menjadi target pembelajaran.

Algoritma LVQ untuk proses pembelajaran pada sistem pengenalan *chord* dalam penelitian ini secara umum adalah sebagai berikut:

1. Ditetapkan :
   - Bobot awal (*w*) untuk setiap unit prototipe secara acak
   - Maksimum iterasi (*MaxEpoch*)
   - *Learning Rate* (LR), *Epsilon*

2. Dipilih :
   - Unit input *x*(*n*)
   - Target *T*
   - Tetapkan kondisi awal :
     - *Epoch* ← 0
     - *E* ← 100000 (E = perubahan bobot awal, diisi bilangan sangat besar)

3. Selama (*Epoch*<*MaxEpoch*) atau (*E*>*Epsilon*) dikerjakan langkah *a* sampai langkah *e* :
   a. *Epoch* ← *Epoch* + 1
   b. Ditentukan jarak minimum (*minDist*) dan index unit output dengan jarak minimum (*minIdx*)
      - *minDist* ← 10000
      - *minIdx* ← 0
   c. Untuk tiap-tiap unit output w_j dikerjakan:
      - Dihitung $||x-w_j||$
      - Apabila $||x-w_j||<minDist$, informasi jarak minimum diperbarui:
        - minDist ← $|| x-w_j ||$
        - minIdx ← *j*
   d. Bobot w_{minIdx} diperbarui dengan ketentuan
      - Jika $T = minIdx$ maka $w_{minIdx} = w_{minIdx} + \text{LR} (x-w_{minIdx})$
      - Jika $T \neq minIdx$ maka $w_{minIdx} = w_{minIdx} - \text{LR} (x-w_{minIdx})$
   e. $E \leftarrow \sum_{i=1}^{12} [\text{LR} (x_i-w_{minIdx})]$

3.4.4 Proses Pengenalan dengan LVQ

Proses pengenalan lebih sederhana dibandingkan proses pembelajaran. Pada proses ini unit input dibadingkan dengan unit

output, kemudian dicari unit output dimana jarak Euclidean dari unit input dan unit output tersebut adalah yang terkecil (minimum).

Algoritma LVQ pada proses pengenalan sistem pengenalan *chord* dalam penelitian ini secara umum adalah sebagai berikut:

1. Ditetapkan :
   - Bobot awal (*w*) untuk setiap unit prototipe
2. Dipilih unit input *x*(*n*)
3. Ditentukan jarak minimum (*minDist*) dan indeks unit output dengan jarak minimum (*minIdx*)
   - *minDist* ← 10000
   - *minIdx* ← 0
4. Untuk tiap-tiap unit output ke-j dikerjakan:
   - *Distance* ← $\| x\text{-}w_j \|$
   - Apabila *Distance*| < *minDist*, informasi minimum diperbarui:
     - minDist ← *Distance*
     - minIdx ← *j*
5. Jika semua unit telah dihitung jarak Euclidean-nya, maka output dari pengenalan telah diketahui yaitu unit output ke-*minIdx.*

3.5 Contoh Penerapan Algoritma

3.5.1 Contoh Penerapan Proses Pembelajaran

Berikut ini adalah contoh penerapan algoritma *Learning Vector Quantization* pada proses pembelajaran. Dimisalkan terdapat dua vektor prototipe A dan B masing-masing berukuran 4, dan suatu vektor masukan *x* yang berukuran 4 pula. Langkah-langkah penghitungan dalam dua iterasi adalah sebagai berikut:

1. Ditetapkan :
   - Bobot awal (*w*) untuk unit prototipe w_1 (1,0,1,0) dan unit prototipe w_2 (0,1,0,1)
   - *MaxEpoch*, misalkan 10000
   - *Learning Rate* (LR) : 0,1
   - *Epsilon* : 0,0000001
2. Dipilih :
   - Unit input *x*(*n*) = (0,7; 0,1;0,8;0)
   - Target *T=A*
   - Tetapkan kondisi awal : *Epoh* ← 0, *Eps* ← 1
3. Iterasi ke-1
   a. *Epoh* 1

b. Untuk tiap-tiap unit output ditentukan jarak Euclidean-nya.

$$d_1 = \sum_{i=1}^{4} | x_i - A_i |$$

= |0,7-1| + |0,1-0| + |0,8-1| + |0-0|
= 0,3 + 0,1 + 0,2 + 0
= 0,6

$$d_2 = \sum_{i=1}^{4} | x_i - B_i |$$

= |0,7-0| + |0,1-1| + |0,8-0| + |0-1|
= 0,7 + 0,9 + 0,8 + 1
= 3,4

d. Jarak minimum pada prototipe A, sedangkan T=A sehingga :
- $A_1 = A_1 + LR\ (x_1 - A_1) = 1 + 0,1(-0,3) = 0,97$
- $A_2 = A_2 + LR\ (x_2 - A_2) = 0 + 0,1(0,1) = 0,01$
- $A_3 = A_3 + LR\ (x_3 - A_3) = 1 + 0,1(-0,2) = 0,98$
- $A_4 = A_4 + LR\ (x_4 - A_4) = 0 + 0,1(0) = 0$

Sehingga bobot A setelah iterasi ke-1 (0,97; 0,01; 0,98; 0)

e. E = 0,06
LR = LR – 0,1(LR)
= 0,1 – 0,1(0,1)
= 0,09

4. Iterasi ke-2
   a. *Epoh* = 2
   b. Ditentukan :
      - *minDist* ← 10000
      - *minIdx* ← 0
   c. Untuk tiap-tiap unit output ditentukan jarak Euclidean-nya.

$$d_1 = \sum_{i=1}^{4} | x_i - A_i |$$

= |0,7-0,97| + |0,1-0,01| + |0,8-0,98| + |0-0|
= 0,27 + 0,09 + 0,18 + 0
= 0,54

$d_2 = \sum_{i=1}^{4} | x_i - B_i |$

= |0,7-0| + |0,1-1| + |0,8-0| + |0-1|

= 0,7 + 0,9 + 0,8 + 1

= 3,4

d. Jarak minimum pada prototipe A, sedangkan T=A sehingga :

- $A_1 = A_1 + LR (x_1 - A_1) = 0,97 + 0,09(-0,27) = 0,9457$
- $A_2 = A_2 + LR (x_2 - A_2) = 0,01 + 0,09(0,09) = 0,0181$
- $A_3 = A_3 + LR (x_3 - A_3) = 0,98 + 0,09(-0,18) = 0,9638$
- $A_4 = A_4 + LR (x_4 - A_4) = 0 + 0,09(0) = 0$

Sehingga bobot A setelah iterasi ke-1 (0,9457; 0,0181; 0,9638; 0)

e. E = 0,0486

LR = LR – 0,1(LR)

= 0,09 – 0,1(0,09)

= 0,081

5. Demikian seterusnya selama Epoh<MaxEpoh atau E>Epsilon

3.5.2 Contoh Penerapan Proses Pengenalan

Berikut ini adalah contoh penerapan algoritma *Learning Vector Quantization* pada proses pengenalan. Dimisalkan terdapat dua vektor prototipe A dan B dan satu vektor masukan, dengan ukuran dan bobot sebagaimana hasil proses pembelajaran pada di atas. Langkah-langkah penghitungan dalam proses pengenalan adalah sebagai berikut:

1. Untuk tiap-tiap unit output ditentukan jarak Euclidean-nya.

$d_1 = \sum_{i=1}^{4} | x_i - A_i |$

= |0,7-9457| + |0,1-0,0181| + |0,8-0,9638| + |0-0|

= 0,2457 + 0,0819 + 0,1638 + 0

= 0,4914

$d_2 = \sum_{i=1}^{4} | x_i - B_i |$

= |0,7-0| + |0,1-1| + |0,8-0| + |0-1|

= 0,7 + 0,9 + 0,8 + 1

= 3,4

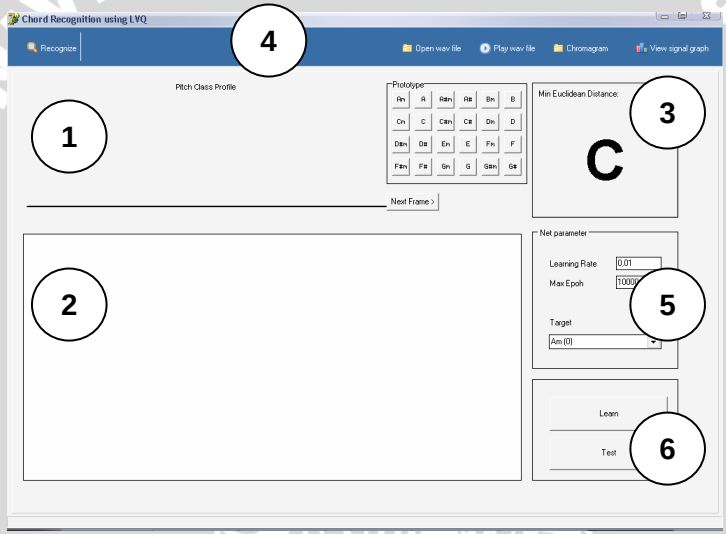
2. Jarak minimum adalah jarak vektor input terhadap prototipe A sehingga output dari proses pengenalan adalah A.

3.6 Rancangan Antarmuka Sistem

Antarmuka sistem pengenalan *chord* ini terdiri dari 2 tampilan utama, yaitu pembelajaran dan pengenalan

3.6.1 Form Pembelajaran

Untuk bagian pembelajaran, rancangan antarmuka sistemnya seperti digambarkan pada gambar 3.9.



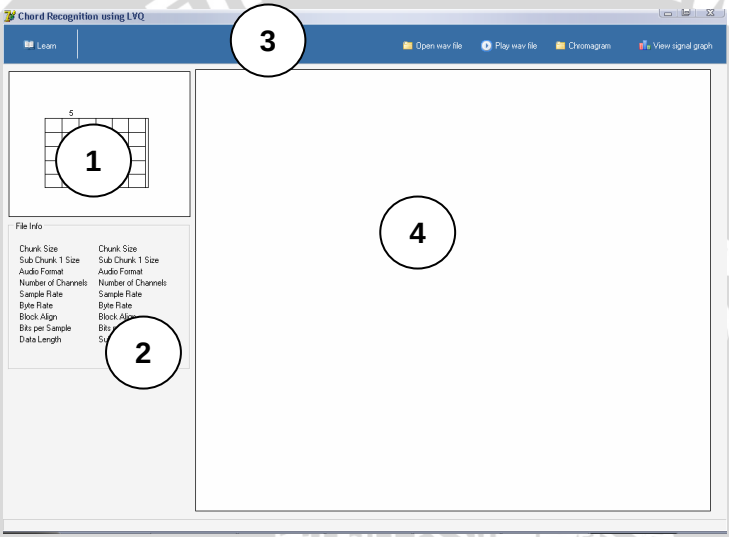
Gambar 3.9 Rancangan antarmuka untuk proses pembelajaran

Keterangan:
1 : Tampilan dalam bentuk PCP dari data sampel
2 : Memo untuk menampilkan informasi pembelajaran jaringan
3 : Panel untuk menampilkan prototipe terdekat
4 : Panel navigasi utama
5 : Panel untuk *setting* parameter jaringan
6 : Panel navigasi untuk pembelajaran

Proses pembelajaran dilakukan dengan memilih file suara (file contoh *chord*). Kemudian ditentukan target pembelajaran, kemudian dilakukan *setting* parameter jaringan syaraf tiruan. Setelah semua parameter diset, maka ditekan tombol *'Learn'* untuk melakukan pembelajaran.

3.6.2 Form Pengenalan

Untuk proses pengenalan, rancangan antarmuka sistemnya seperti digambarkan pada gambar 3.10.



Gambar 3.10 Rancangan antarmuka untuk proses pengenalan

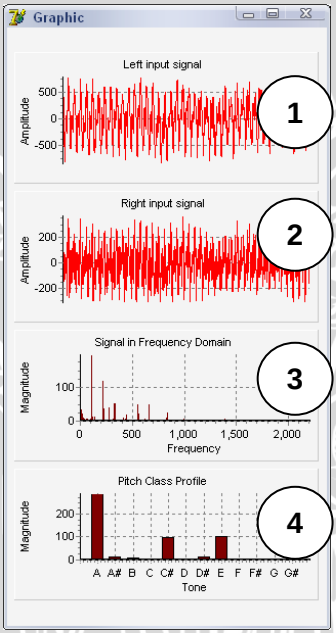
Keterangan:
1 : Panel untuk menampilkan gambar *chord diagram* hasil pengenalan
2 : Informasi tentang file wav yang dikenali
3 : Panel navigasi utama
4 : Panel memunculkan *chord* hasil pengenalan

Proses pengenalan dilakukan dengan melakukan *setting* parameter jaringan syaraf tiruan. Setelah semua parameter diset, maka ditekan tombol *'Open'* untuk memilih file suara (misalnya file

lagu), sekaligus melakukan pengenalan. Hasil pengenalan akan muncul di panel nomor 4.

3.6.3 Form Tampilan Grafik Sinyal Suara

Form tampilan grafik sinyal suara ini digunakan sebagai fasilitas tambahan yang memungkinkan *user* mengetahui bentuk sinyal suara beserta hasil transformasi FFT dan hasil *mapping* ke PCP yang dilakukan oleh sistem pengenalan *chord*. Rancangan tampilan form grafik sinyal ini seperti ditunjukkan pada gambar 3.11.



Gambar 3.11 Rancangan antarmuka untuk tampilan grafik sinyal

Keterangan:
1 : Panel untuk menampilkan grafik input chanel *left* dari sinyal
2 : Panel untuk menampilkan grafik input chanel *right* dari sinyal
3 : Panel untuk menampilkan grafik hasil FFT
4 : Panel untuk menampilkan grafik PCP

3.6.4 Form Tampilan Chromagram

Sistem pengenalan chord otomatis ini natinya juga akan dilengkapi dengan sebuah form yang menampilkan chromagram. Chromagram dapat digunakan untuk analisis hasil pengenalan lebih lanjut. Form tampilan chromagram suara ini digunakan sebagai fasilitas tambahan untuk mengetahui secara rinci intensitas tiap nada pada tiap *frame* pengenalan. Keseluruhan *frame* ditampilkan secara horizontal, dan tiap *frame* digambarkan secara vertikal dengan intensitas nada yang digambarkan dengan warna dalam skala keabuan.

3.7 Perancangan Uji Coba

Uji coba yang dilakukan dimaksudkan untuk mengetahui keakuratan pengenalan *chord* dengan metode jaringan syaraf tiruan *Learning Vector Quantization*. Uji coba akan dilakukan dengan menggunakan parameter jaringan syaraf tiruan, seperti *learning rate dan iterasi maksimum*, sedemikian hingga hasil pengenalan yang didapatkan optimal. Selain itu jenis suara musik yang akan dikenali bervariasi. Suara musik yang akan diteliti akan dibedakan menjadi 3 kelompok yaitu :

1. Suara musik solo, yaitu suara dari satu jenis alat musik saja. Misalkan piano atau gitar.
2. Suara musik *full*, yaitu suara musik dengan vokal dan musik lengkap.
3. Suara musik midi, yaitu suara musik dengan nada vokal dan musik lengkap yang dikonversi dari format midi menjadi wav.

Untuk masing-masing jenis suara musik akan diketahui tingkat keakuratan dan kemudian hasil analisis diarahkan untuk mengetahui apakah penyebab perbedaan hasil pengenalan yang mungkin terjadi.

Dalam penelitian ini, untuk tahap pembelajaran diambil sampel untuk masing-masing *chord* berupa sampel suara dari alat musik piano dan gitar, sehingga jumlah keseluruhan sampel adalah 48 sampel suara. Tiap sampel suara bisa dibagi menjadi beberapa *frame*, sehingga setiap sampel suara bisa dilakukan 2–6 kali pembelajaran.

Selanjutnya, untuk tahap pengenalan diujicobakan sejumlah suara musik untuk diketahui hasil keakuratannya dengan cara membandingkan *chord* hasil pengenalan dengan *chord* sebenarnya. Data suara musik yang diujicobakan adalah:

1. *Uji coba tahap pertama*
   Pada uji coba tahap pertama, musik yang diujicobakan adalah musik solo yang direkam dalam format wav dengan *sample rate* 44100 dan *bitrate* 16Kb, proses *recording* menggunakan perangkat lunak Sony Sound Forge 7.0.
2. *Uji coba tahap kedua*
   Pada uji coba tahap kedua, musik yang diujicobakan adalah musik *full*, yaitu musik dalam format mp3 yang banyak beredar di pasar, yang kemudian dikonversi ke dalam format wav dengan *sample rate* 44100 dan *bitrate* 16Kb, proses konversi menggunakan perangkat lunak Sony Sound Forge 7.0.
3. *Uji coba tahap ketiga*
   Pada uji coba tahap ketiga, musik yang diujicobakan adalah musik midi, yaitu musik dalam format midi yang diputar kemudian direkam ke dalam format wav dengan *sample rate* 44100 dan *bitrate* 16Kb, proses *recording* menggunakan perangkat lunak Sony Sound Forge 7.0.

Data suara musik yang diujicobakan pada ketiga tahap adalah 10 lagu dari berbagai macam *genre* musik. Hal ini dilakukan untuk memperluas cakupan penelitian sehingga tidak terbatas pengenalan *chord* untuk satu jenis musik saja. Tabel 3.2 menunjukkan daftar lagu yang akan diujikan pada penelitian ini.

Hasil pengenalan dari lagu yang diinputkan berupa rangkaian *chord* yang ditampilkan dalam blok-blok dengan tiap blok merupakan hasil pengenalan chord dari satu *frame* analisis. Dari 10 lagu yang akan diujicobakan, dihitung keakuratannya dengan menghitung jumlah *frame* yang sesuai dengan *chord* sebenarnya dibagi dengan jumlah *frame* keseluruhan.

Tabel 3.2 Daftar lagu yang akan diuji coba

No	Judul lagu	Artis	Genre
1	12.51	The Strokes	New wave
2	Angie	The Rolling Stones	Pop Balad
3	As tears go by		Balad
4	Hey Jude	The Rolling Stones	Pop
5	Hey there Delilah		Country
6	Imagine	The Beatles	Classic Pop
7	Let it be	Plain T. White	Classic Pop
8	Look what you've done	John Lennon	Pop
9	Stop crying your heart out	The Beatles	Britpop
10	Wonderwall	Jet	Rock
		Oasis	
		Oasis	

Hasil dari uji coba nantinya akan disimpan dalam tabel dengan format sebagaimana ditampilkan pada Tabel 3.3.

Tabel 3.3 Tabel hasil uji coba pengenalan *chord*

No	Judul lagu	Jumlah *Frame*	Jumlah *Frame* Benar	Keakuratan (%)

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi Sistem

Secara umum, sistem pengenalan chord otomatis ini dibuat dengan menggunakan Borland Delphi 7 di bawah *operating system* Windows XP Home Edition Service Pack 2. Sistem ini dibuat dan telah dicoba pada komputer dengan berbagai tipe *processor* serta memori.

4.1.1 Implementasi Pengolahan Sinyal

4.1.1.1 Struktur data untuk pengolahan sinyal

Sinyal suara dalam file wav masih dalam format heksadesimal. Untuk memudahkan proses pengolahan suara, data suara tersebut diubah terlebih dahulu ke dalam format desimal. Untuk menyimpan data dalam format desimal tersebut diperlukan struktur data buatan yang dikhususkan untuk menampung informasi dan data dari file wav. *Sourcecode* 4.1 menunjukkan struktur data yang digunakan untuk pembacaan file wav.

```
type
  TWaveHeader = packed record
    Marker_RIFF       : array [0..3] of char;
    ChunkSize         : cardinal;
    Marker_WAVE       : array [0..3] of char;
    Marker_fmt        : array [0..3] of char;
    SubChunkSize      : cardinal;
    FormatTag         : word;
    NumChannels       : word;
    SampleRate        : longint;
    BytesPerSecond    : longint;
    BytesPerSample    : word;
    BitsPerSample     : word;
    Marker_data       : array [0..3] of char;
    DataBytes         : longint;
  end;
  TChannel = record
    Data              : array of smallint;
  end;

//Deklarasi variable
  wavehdr    : TWaveHeader;
  wavedata   : array [0..1] of TChannel;
  numsamples : integer;
```

Sourcecode 4.1 Struktur Data Pembacaan File Wav

Untuk menampung data file wav, dibuat suatu tipe data buatan bertipe `record` yang berisi variabel-variabel di dalamnya. `TWaveHeader` merupakan *packed record* berisi variabel-variabel yang digunakan untuk menyimpan informasi *header* file wav. Sedangkan record `TChannel` berisi variable untuk menyimpan data aktual suara untuk setiap *channel* suara. Tabel 4.1 menunjukkan variable-variabel pada `TWaveHeader` dan `TChannel.`

Tabel 4.1 Variabel pada TWaveHeader

Nama Variabel	Keterangan
`Marker_RIFF`	Variable array bertipe char berukuran 4 untuk menyimpan karakter 'RIFF'
`ChunkSize`	Variabel bertipe cardinal untuk menyimpan ukuran chunk file wav
`Marker_WAVE`	Variable array bertipe char berukuran 4 untuk menyimpan karakter 'WAVE'
`Marker_fmt`	Variable array bertipe char berukuran 4 untuk menyimpan karakter 'fmt '
`SubChunkSize`	Variabel bertipe cardinal untuk menyimpan ukuran *subchunk*
`FormatTag`	Varabel bertipe word untuk menyimpan format suara file wav, misalnya nilai 1 untuk format suara PCM
`NumChannels`	Variable bertipe word untuk menyimpan jumlah chanel file suara, 1 untuk mono dan 2 untuk stereo
`SampleRate`	Variabel bertipe longint untuk menyimpan nilai *Sample Rate*
`BytesPerSecond`	Variabel bertipe longint untuk menyimpan jumlah byte tiap detik
`BytesPerSample`	Variabel bertipe word untuk menyimpan jumlah byte tiap sample suara
`BitPerSample`	Variabel bertipe word untuk menyimpan jumlah bit tiap sample suara
`Marker_data`	Variable array bertipe char berukuran 4 untuk menyimpan karakter 'data'
`DataBytes`	Variabel bertipe longint untuk menyimpan ukuran data chunk

4.1.1.2 Pembacaan file wav

Pembacaan file wav dilakukan dengan cara memasukkan file wav yang dikehendaki ke dalam *filestream*. Kemudian dilakukan pembacaan data sekaligus dimasukkan ke dalam variabel sesuai struktur yang telah dibuat. *Sourcecode* 4.2 menunjukkan prosedur yang digunakan untuk membaca file wav.

Pembacaan file wav dimulai dari informasi *header* file wav. Prosedur `Read` yang diimplementasikan pada *filestream* `stream` untuk membaca *header* file wav. Prosedur memungkinkan pembacaan data sekaligus memasukkannya ke dalam variable-variabel `TWaveHeader` secara otomatis sesuai dengan struktur *packed record* yang telah dibuat.

Setelah informasi header file wav didapatkan, proses dilanjutkan dengan pembacaan data aktual hingga bagian akhir data suara. Proses ini dilakukan dengan perulangan hingga seluruh bagian data aktual dari file wav terbaca.

4.1.1.3 *Windowing* dan FFT

Pada sistem pengenalan *chord* ini, fungsi untuk *windowing* dan transformasi fourier cepat (FFT) dilakukan oleh komponen DSPLab. Prosedur yang dibuat dalam implementasi digunakan untuk penentuan parameter-parameter untuk *windowing* dan FFT serta pemanggilan fungsi FFT dan kalkulasi hasil FFT.

Pada proses *windowing*, ditentukan batas awal dan batas akhir *frame* yang akan ditransformasi. Kemudian proses pembobotan akan dilakukan oleh komponen DSPLab. *Sourcecode* 4.3 menunjukkan prosedur yang menangani proses *windowing*.

Setelah batas awal dan batas akhir *frame* ditentukan, tiap sampel pada *frame* tersebut dijadikan input transformasi sebagai bagian real input transformasi. Sedangkan untuk bagian imajiner input transformasi ditentukan 0. Kemudian proses FFT dilakukan dengan memanggil prosedur `FFT` dan `CalculateMagnitude` dari komponen DSPLab. Hasil dari proses transformasi ini tersimpan dalam variabel array `RealOut[i]` untuk bagian real dan `ImagOut[i]` untuk bagian imajiner. *Sourcecode* 4.4 menunjukkan prosedur yang menangani proses transformasi.

```
//*******************************************************
//  LOADING WAV FILE
//*******************************************************

procedure TfMain.LoadFromFile(filename: string);
var
  Stream : TFileStream;
  i : integer;
begin
  statusbar1.SimpleText := 'Load wav file... ';
  Stream := TFileStream.Create(filename, fmOpenRead);

  FillChar(wavehdr, sizeof(wavehdr), 0);
  Stream.Read(wavehdr, sizeof(wavehdr));

  lbCS.Caption   := ': ' + IntToStr(wavehdr.ChunkSize);
  lbSC1S.Caption := ': ' + IntToStr(wavehdr.SubChunkSize);
  lbAU.Caption   := ': ' + IntToStr(wavehdr.FormatTag);
  lbNC.Caption   := ': ' + IntToStr(wavehdr.NumChannels);
  lbSR.Caption   := ': ' + IntToStr(wavehdr.SampleRate);
  lbBR.Caption   := ': ' + IntToStr(wavehdr.BytesPerSecond);
  lbBA.Caption   := ': ' + IntToStr(wavehdr.BytesPerSample);
  lbBPS.Caption  := ': ' + IntToStr(wavehdr.BitsPerSample);
  lbSC2S.Caption := ': ' + IntToStr(wavehdr.DataBytes);

  numsamples := (wavehdr.DataBytes div wavehdr.NumChannels) div
wavehdr.BytesPerSample;

  case wavehdr.NumChannels of
    1 : begin
          SetLength(wavedata[0].Data,
numsamples*wavehdr.NumChannels);
          Stream.Read(wavedata[0].Data[0],
numsamples*wavehdr.NumChannels);
        end;

    2 : begin
          SetLength(wavedata[0].Data,
numsamples*wavehdr.NumChannels);
          SetLength(wavedata[1].Data,
numsamples*wavehdr.NumChannels);
          for i := 0 to high(wavedata[0].Data) do
          begin
            Stream.Read(wavedata[0].Data[i], 2);
            Stream.Read(wavedata[1].Data[i], 2);
          end;
        end;
  end;
  Stream.Free;
end;
```

Sourcecode 4.2 Prosedur pembacaan file wav

```
procedure TfMain.doFFT;
var
  itr, maxitr, StartWindow, EndWindow : integer;
begin
  statusbar1.SimpleText := 'Analyze wav file... ';
  itr := 0;
  maxitr := numsamples*wavehdr.NumChannels div
dspFFT.BufferSize;
  dspFFTR.BufferSize := dspFFT.BufferSize;


  EndWindow := 0;
  while EndWindow+dspFFT.BufferSize<Length(wavedata[0].Data) do
  begin
    inc(itr);

    if (itr mod 3)=0 then
      statusbar1.SimpleText := 'Analyze wav file '+
IntToStr(100*itr div maxitr)+'%...';

    StartWindow := dspFFT.BufferSize*(itr-1);
    EndWindow := (dspFFT.BufferSize*itr)-1;

    Transform(StartWindow, EndWindow);
    SetPCP;
    RecognizeChord;

  end;
  statusbar1.SimpleText := 'Done.'
end;
```

Sourcecode 4.3 Prosedur untuk inisialisasi transformasi dan penentuan *frame* (*windowing*)

Proses transformasi FFT yang dilakukan adalah transformasi terhadap tiap *frame* yang telah di-*windowing*. Adapun lebar *frame* yang dipilih adalah 16384 sampel, atau 2^{14} sampel. Pemilihan nilai tersebut dikarenakan untuk sampel suara dengan *sample rate* 44100 Hz, nilai 2^{14} cukup merepresentasikan komposisi suara dari potongan file wav. Dengan lebar *frame* sebesar 2^{14}, maka tiap *frame* akan mewakili 0,317 detik sampel suara, dan nilai ini adalah yang paling optimal dari hasil percobaan.

Untuk suara dengan 2 *channel* (stereo) proses FFT dilakukan secara terpisah untuk suara dari *left channel* dan chanel *right channel*. Hasil transformasi dari kedua chanel tersebut nantinya akan digabung pada saat *mapping* ke PCP.

```
procedure TfMain.Transform(StartWindow, EndWindow : integer);
var
  i : integer;
  freqRange : real;
begin
  DrawTheWave(StartWindow,EndWindow);

  //FFT PROCESSING USING DSPLAB
  for i:=0 to dspFFT.BufferSize -1 do
  begin
    dspFFT.RealIn[i] := wavedata[0].Data[startWindow+i];
    dspFFT.ImagIn[i]:= 0;
  end;

  dspFFT.FFT;
  dspFFT.CalculateMagnitudes;

  if wavehdr.numChannels=2 then   //if stereo, get right channel
  begin
    for i:=0 to dspFFTR.BufferSize -1 do
    begin
      dspFFTR.RealIn[i] := wavedata[1].Data[startWindow+i];
      dspFFTR.ImagIn[i]:= 0;
    end;

    dspFFTR.FFT;
    dspFFTR.CalculateMagnitudes;
  end;

  //SETTING FOR FFT GRAPH DISPLAY
  freqRange := wavehdr.SampleRate/dspFFT.BufferSize;

  for I:= 0 to (dspFFT.BufferSize2 - 1)div 10 do
    freqScale[i] := Round(i*freqRange);

  fGraph.Series2.Clear;
  for I:= 0 to (dspFFT.BufferSize2 - 1)div 10 do
    fGraph.Series2.AddXY(freqScale[i], dspFFT.RealOut[I], '',
clMaroon);
end;
```

Sourcecode 4.4 Prosedur untuk FFT

4.1.1.4 *Pitch Class Profile Mapping*

Koefisien hasil transformasi dengan metode FFT berupa koefisien-koefisien yang merepresentasikan magnitudo yang mewakili rentang frekuensi tertentu. Hasil transformasi berupa deret bilangan real dan imajiner, namun yang dapat digunakan sebagai analisis *chord* nantinya hanya deret bilangan real.

Koefisien-koefisien tersebut kemudian dikelompokkan menurut frekuensi yang bersesuaian dengan frekuensi tiap nada dalam tangga nada kromatik. Bentuk demikian disebut *Pitch Class Profile (PCP)*. Kemudian 12 koefisien PCP diskala ke dalam rentang nilai 0 sampai 1. *Sourcecode* 4.5 menunjukkan prosedur pemetaan hasil FFT menjadi PCP sekaligus penskalaannya.

```
procedure TfMain.SetPCP;
var
  tone,i : integer;
  curFreq, temp : real;
begin
  // SET PCP, BEGIN FROM A Tone (110 HZ)
  for tone:=1 to 12 do
  begin
    PCP[tone] := 0;
    curFreq := PCPFreq[tone];
    for i:=0 to ((dspFFT.BufferSize2 - 1)div 10)-2 do
      if curFreq<freqScale[i+1] then
      begin
        PCP[tone] := PCP[tone] + dspFFT.RealOut[i];
        if wavehdr.numCHannels=2 then
          PCP[tone] := PCP[tone] + dspFFTR.RealOut[i];
        curFreq := curFreq*2;
      end;
  end;

  //scalling
  temp := PCP[1];
  for i:=2 to 12 do
    if PCP[i]>temp then
      temp := PCP[i];
  if temp<>0 then
    for i:=1 to 12 do
      PCP[i] := PCP[i]/temp;

  end;
end;
```

Sourcecode 4.5 Prosedur untuk PCP *Setting*

4.1.2 Implementasi Jaringan Syaraf Tiruan

4.1.2.1 Struktur Data Implementasi Jaringan Syaraf Tiruan

Dalam implementasi proses jaringan syaraf tiruan, ada sebuah konstanta `KLASS` yang berfungsi untuk menyimpan jumlah kelas pengenalan yang digunakan. Selain itu ada beberapa variabel global terdiri dari `MaxEpoh` untuk menyimpan iterasi maksimum pembelajaran, `weight` yang merupakan array dua dimensi bertipe real yang berfungsi untuk menyimpan nilai bobot semua kelas pengenalan, dan `distance` yang merupakan array 1 dimensi bertipe real yang digunakan untuk menyimpan jarak Euclidean tiap kelas ke vektor input *x*.

```
const
  KLASS=24;
var
  LearnStat, first : boolean;
  MaxEpoh : real;
  Weight  : array [0..3*KLASS-1, 1..12] of real;
  Distance : array [0..3*KLASS-1] of real;
```

Sourcecode 4.6 Struktur data jaringan syaraf tiruan

4.1.2.2 Proses Pembelajaran

Pada proses pembelajaran, prosedur pembelajaran yang digunakan dalam implementasi sistem pengenalan *chord* tampak pada *sourcecode* 4.7. Prosedur pembelajaran ini diawali proses inisialisasi input dan pembacaan weight yang tersimpan dalam file teks. Kemudian dilakukan perulangan selama kondisi berhenti belum ditemui, dimana tiap-tiap perulangan dilakukan *update* terhadap bobot yang tersimpan dalam file teks dimana bobot yang di-*update* adalah bobot dari kelas yang ditargetkan dalam pembelajaran. Pembelajaran menggunakan *learning rate* konstan

Perulangan berlangsung selama belum ditemui kondisi berhenti. *Sourcecode* 4.8 menunjukkan fungsi yang menangani kondisi berhenti. Kondisi berhenti ditemui jika iterasi mencapai nilai yang melebihi iterasi maksimum (*MaxEpoch*). Kondisi berhenti juga akan ditemui jika bobot sudah stabil, atau perubahan bobot kurang dari *Epsilon*. Apabila salah satu dari kedua kondisi tersebut ditemui, maka perulangan pada proses pembelajaran berhenti. Bobot yang digunakan adalah bobot terakhir dari target pembelajaran.

```
procedure TfMain.LearnChord;
var
  i,j,Epoh,minIdx,T,Target : integer;
  minDist, LR, E : real;
  F : textfile;
begin
  //init NetParameter
  T := cbTarget.ItemIndex;
  Target := T div 3;  LR := LR0;
  MaxEpoh := StrToFloat(eMaxEpoh.Text);

  //init Input
  for i:=1 to 12 do Xinput[i] := PCP[i];

  //init weight
  for i:=0 to (3*KLASS)-1 do
  begin
    assignFile(F, DIR + '\LVQ\' +IntToStr(i) + '.txt');
    Reset(F);
    for j:=1 to 12 do
      readln(F,weight[i,j]);
    CloseFile(F);
  end;

  //looping for training
  Epoh:=1;
  Repeat begin
    minDist := 1000000;  minIdx  := 0;

    //get euclidean distance for each prototype
    for i:=0 to (3*KLASS)-1 do
    begin
      distance[i]:=0;
      for j:=1 to 12 do
        distance[i] := distance[i] + (abs(Xinput[j] -
weight[i,j]));
      if distance[i] < minDist then
      begin
        minDist := distance[i];
        minIdx:= i;
      end;
    end;

    lbMinED.Caption := Chord(minIdx div 3);
    //weight update
    E:=0;
    if (minIdx div 3)=Target then
    begin
      for i:=1 to 12 do
      begin
        weight[minIdx,i]:=weight[minIdx,i] +
LR*(Xinput[i]-weight[minIdx,i]);
        if weight[minIdx,i]<0 then weight[minIdx,i]:=0;
        E := E + abs(LR*(Xinput[i]-weight[minIdx,i]));
      end;
    end
```

```
    else
    begin
      for i:=1 to 12 do
      begin
        weight[minIdx,i]:=weight[minIdx,i] -
LR*(Xinput[i]-weight[minIdx,i]);
        if weight[minIdx,i]<0 then weight[minIdx,i]:=0;
        E := E + abs(LR*(Xinput[i]-weight[minIdx,i]));
      end;
    end;
    Inc(Epoh);
  end;
  until StopCondition(Epoh, LR, E);
```

Sourcecode 4.7 Prosedur pembelajaran LVQ

```
function TfMain.StopCondition(Epoh : integer; E : real) :
boolean;
begin
  if (Epoh>MaxEpoh) OR (E<Epsilon) then
    result := TRUE
  else
    result := FALSE;
end;
```

Sourcecode 4.8 Fungsi Kondisi Berhenti untuk Pembelajaran LVQ

4.1.2.3 Proses Pengenalan

Proses penenalan *chord* yang dijalankan oleh prosedur seperti tampak pada *sourcecode* 4.9 merupakan proses yang lebih sederhana daripada proses pembelajaran. Proses ini hampir sama dengan proses pembelajaran hanya saja tidak memerlukan perulangan dan tidak memerlukan target. Vektor input dibandingkan dengan kelas-kelas prototipe yang ada kemudian ditentukan vektor prototipe dengan jarak terdekat dari vektor input. Vektor terdekat itulah yang menjadi output pengenalan sebagai *chord*.

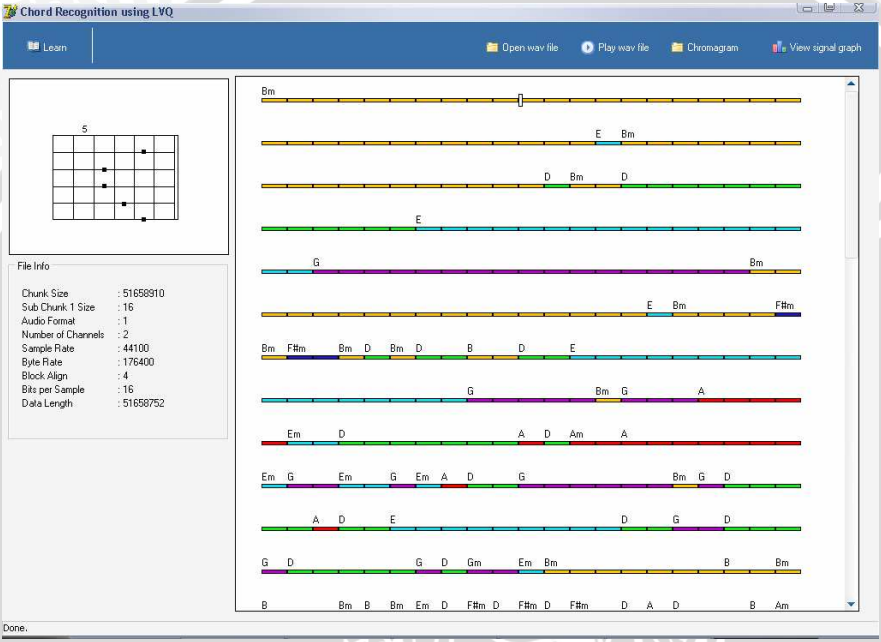
```
procedure TfMain.RecognizeChord;
var
  i,j : integer;
  minDist : real;
  minIdx : integer;
  F : textfile;
begin
  //init Input
  for i:=1 to 12 do
    Xinput[i] := PCP[i];

  //init weight
  for i:=0 to KLASS-1 do
  begin
    assignFile(F, DIR + '\LVQ\' + IntToStr(i) + '.txt');
    Reset(F);
    for j:=1 to 12 do
      readln(F,weight[i,j]);
    CloseFile(F);
  end;

  //looping for training
  minDist := 1000000;
  minIdx  := 0;

  //hitung jarak tiap2 bobot
  for i:=0 to KLASS-1 do
  begin
    distance[i]:=0;
    for j:=1 to 12 do
      distance[i] := distance[i] + ((Xinput[j] -
weight[i,j])*(Xinput[j] - weight[i,j]));

    distance[i] := sqrt(distance[i]);
    if distance[i] < minDist then
    begin
      minDist := distance[i];
      minIdx:= i;
    end;
  end;

  if LearnStat then
    lbMinED.Caption := Chord(minIdx)
  else
    DisplayResult(minIdx);
end;
```

Sourcecode 4.9 Prosedur pengenalan *chord*

4.2 Hasil dan Analisis Penelitian

4.2.1 Tampilan sistem pengenalan *chord*

Tampilan dari hasil implementasi sistem pengenalan *chord* ini tampak pada gambar 4.1.

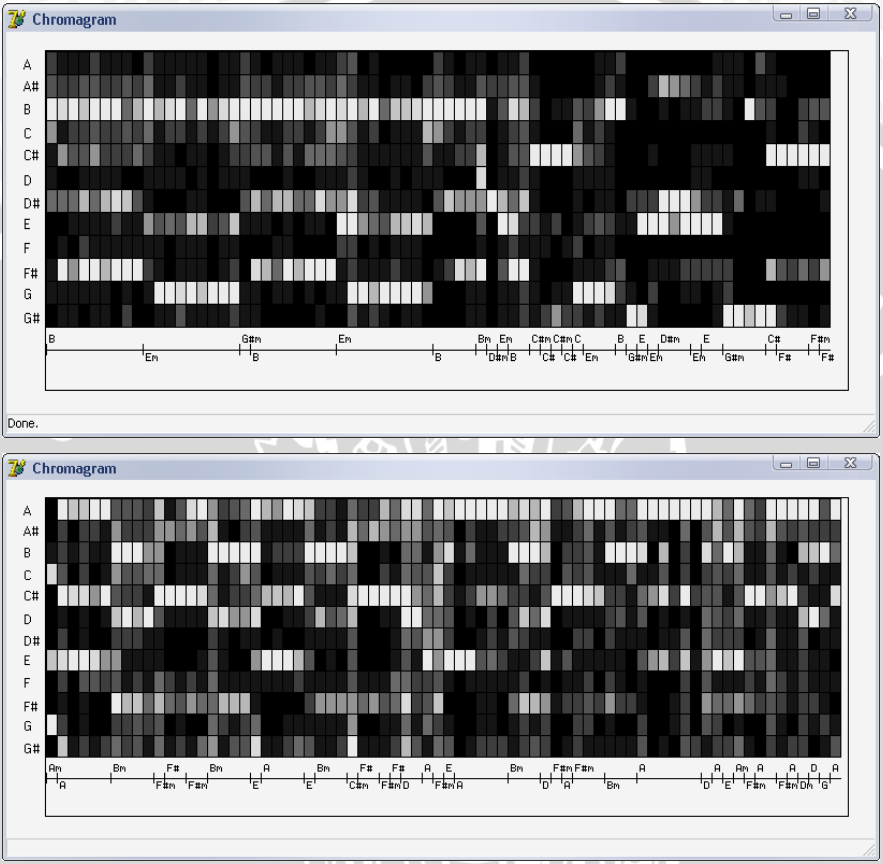


Gambar 4.1 Tampilan sistem pengenalan *chord*

Pada tampilan sistem penegnalan *chord* ini, *chord* hasil pengenalan ditampilkan dalam balok-balok memanjang, dimana tiap balok tersebut mewakili *frame* suara yang ditransformasi kemudian dikenali sebagai *chord*. Di sebalah kiri layar terdapat diagram *chord* dimana pada saat musik diputar (*play*) akan menunjukkan gambar *chord* dari suara yang sedang terdengar dan ditunjukkan pula oleh kursor yang akan bergerak sepanjang balok *chord* hasil pengenalan. Selain itu terdapat pula informasi mengenai file wav yang dikenali.

Hasil pengenalan dapat dianalisis pula melalui sebuah tampilan chromagram, yaitu sebuah tampilan grafik yang menunjukkan intensitas magnitudo tiap nada yang direpresentasikan dengan warna dalam skala keabuan. Chromagram menampilkan intensitas magnitudo tiap nada tiap

frame dari keseluruhan lagu yang dikenali. Gambar 4.2 menunjukkan tampilan dari chromagram.



Gambar 4.2 Tampilan chromagram

4.2.2 Proses Pembelajaran

Sebelum digunakan untuk pengenalan, sistem membutuhkan pembelajaran terlebih dahulu. Pembelajaran dilakukan dengan membaca file sampel suara *chord* yang terdiri dari 2 macam suara untuk tiap-tiap *chord* dalam sistem pengenalan. Total terdapat 72 sampel suara *chord* yang terdiri dari 24 file suara untuk 24 *chord* dalam bentuk suara gitar, dan 24 file suara untuk 24 *chord* dalam bentuk suara piano.

Pembelajaran dilakukan menggunakan parameter laju pembelajaran (*learning rate*) sebesar 0,01 dan batas maksimum iterasi 10000. Bobot hasil pembelajaran oleh 72 sampel suara ini kemudian digunakan sebagai bobot pada proses uji coba pengenalan *chord*.

4.2.3 Hasil Uji Coba Pengenalan *Chord*

Uji coba pengenalan *chord* dilakukan dalam tiga tahap. Pada kedua tahap tersebut, materi yang diujicobakan berbeda namun merupakan suara musik dari judul lagu yang sama. Tahap pertama dilakukan uji coba terhadap musik solo (satu macam alat musik). Tahap kedua dilakukan uji coba terhadap musik *full* yang diconvert dari format mp3 ke format wav. Sedangkan tahap ketiga dilakukan uji coba terhadap music midi yang telah diconvert ke format wav.

Hasil ujicoba pertama tampak pada tabel 4.2. Uji coba pertama adalah uji coba dengan menggunakan objek pengenalan berupa 10 lagu dengan musik solo, yaitu musik yang terdiri dari satu macam alat musik tanpa vokal. Sampel pada uji coba ini dibuat dengan merekam suara gitar ke dalam format wav dengan *sample rate* 44100 dan bitrate 16kbps.

Pada uji coba kedua yang dilakukan pengenalan terhadap 10 lagu, dengan komposisi musik lengkap termasuk vokal. Hasil uji coba tahap kedua yang telah dilakukan tampak pada table 4.3.

Sedangkan untuk uji coba tahap ketiga, dilakukan terhadap 5 lagu dari suara midi yang dikonversi ke dalam format wav dengan *sample rate* 44100 dan bitrate 16kbps. Pengkonversian dari midi ke format wav dilakukan semata-mata untuk mencoba variasi jenis suara karena sebenarnya format suara midi adalah format yang telah terstruktur dan dapat dikenali *chord* penyusunnya dengan cara yang jauh lebih mudah.

Tabel 4.2 Hasil uji coba tahap pertama (musik solo)

No	Judul Lagu / Artis	Jumlah *Frame*	Jumlah frame benar	Akurasi (%)
1	Angie The Rolling Stones	722	699	96,81%
2	As tears go by The Rolling Stones	451	393	87,14%
3	Hey Jude Beatles	688	665	96,66%
4	Hey there Delilah Plain T. White	561	527	93,94%
5	Imagine John Lennon	488	461	94.47%
6	Let it be Beatles	614	570	92,83%
7	Look what you've done Jet	591	571	96,62%
8	Stop crying your heart out Oasis	716	696	97,21%
9	Wonderwall Oasis	669	638	95,37%
10	12.51 The Strokes	389	364	93,57%
	Rata-rata			94,462%

Tabel 4.3 Hasil uji coba tahap (musik *full*)

No	Judul Lagu / Artis	Jumlah Frame	Jumlah frame benar	Akurasi (%)
1	Angie The Rolling Stones	729	609	83,54%
2	As tears go by The Rolling Stones	432	310	71,76%
3	Hey Jude Beatles	892	566	63,45%
4	Hey there Delilah Plain T. White	626	429	68,53%
5	Imagine John Lennon	491	405	82,48%
6	Let it be Beatles	612	504	82,35%
7	Look what you've done Jet	620	458	73,87%
8	Stop crying your heart out Oasis	788	591	75,00%
9	Wonderwall Oasis	707	443	62,66%
10	12.51 The Strokes	392	314	80,10%
	Rata-rata			74,374%

Tabel 4.4 Hasil uji coba tahap ketiga (musik *full* midi)

No	Judul Lagu / Artis	Jumlah Frame	Jumlah frame benar	Akurasi (%)
1	Angie The Rolling Stones	768	494	64,32%
2	As tears go by The Rolling Stones	647	609	94,13%
3	Hey Jude Beatles	727	348	85.28%
4	Hey there Delilah Plain T. White	631	620	85,26%
5	Imagine John Lennon	500	538	55,60%
6	Let it be The Beatles	573	278	82,199%
7	Look what you've done Jet	575	471	89,04%
8	Stop Crying your heart out Oasis	591	512	58,88%
9	Wonderwall Oasis	103	86	83,49%
10	12.51 The Strokes	378	298	78,84%
	Rata-rata			77,704%

4.2.4 Analisis Akurasi Hasil Pengenalan

1. Analisis Uji coba Musik Solo

Dari hasil uji coba tahap pertama, rata-rata akurasi pengenalan di atas 90% (94,248%). Akurasi pengenalan yang tinggi disebabkan karena karakteristik musik solo memang sederhana, hanya terdiri dari satu macam alat musik (gitar dan piano) yang berperan sebagai *rhythm* dari sebuah lagu.

Dari 10 lagu yang diujicobakan, hanya ada satu lagu yang memiliki akurasi pengenalan kurang dari 90% (87,14%). Hal ini disebabkan karena karakteristik lagu tersebut yang menggunakan teknik gitar petik dan dengan tempo musik yang lambat, sehingga suara dalam 1 *frame* yang terdengar seringkali kurang mewakili seluruh komposisi nada yang seharusnya menyusun *chord* pada suatu *frame*. Misalnya pada suatu bagian lagu yang sebenarnya disusun oleh *chord* C, karena suara musik dihasilkan dengan teknik petik dengan tempo lambat, maka ada kemungkinan pada suatu *frame* terjadi nada E yang sangat dominan dibandingkan nada C atau G (C, E dan G adalah nada-nada penyusun *chord* C). Hal ini akan meningkatkan kemungkinan kesalahan pengenalan terhadap *chord*.

2. Analisis Uji Coba Musik *Full*

Dari hasil uji coba tahap kedua, didapatkan rata-rata akurasi hasil pengenalan sebesar 74,374%. Akurasi terendah terjadi pada pengenalan lagu Wonderwall (Oasis) dengan akurasi 62,66%. Hal ini disebabkan karena *chord* yang sebenarnya lebih kompleks dari cakupan pengenalan dalam penelitian ini, di mana batasan pengenalan pada penelitian ini adalah pada *chord* major dan minor, sedangkan komposisi *chord* pada lagu tersebut banyak terdapat *chord* selain major dan minor.

Secara umum, akurasi pengenalan hasil uji coba pada tahap kedua lebih rendah dibanding akurasi pengenalan pada uji coba tahap pertama. Hal ini disebabkan karena musik pada uji coba tahap pertama lebih sederhana, karena hanya terdiri 1 suara alat musik dan berfungsi sebagai *rhytm section*. Sedangkan pada uji coba tahap kedua, musik yang dikenali jauh lebih kompleks, karena terdiri dari berbagai macam alat musik dan tidak saja terdiri dari *rythm section*, namun juga terdapat suara vokal, *lead melody* dan juga suara perangkat drum. Hal ini mengakibatkan terjadinya nada-nada yang sebenarnya bukan penyusun *chord* yang seharusnya,

namun mempunyai nilai yang cukup signifikan. Karena itu resiko kesalahan pengenalan menjadi lebih besar.

3. Analisis Uji Coba Musik Midi

Hasil uji coba tahap ketiga, didapatkan rata-rata akurasi hasil pengenalan sebesar 77,704%. Akurasi sebesar ini masih lebih baik dibanding pada pengenalan tahap, namun masih jauh lebih rendah dibanding pengenalan terhadap musik solo.

Selain itu untuk tiap-tiap lagu jika dibandingkan dengan uji coba pada tahap kedua terjadi ketidakkonsistenan. Sebagai contoh, lagu Hey Jude pada pengenalan tahap ketiga jauh lebih tinggi akurasi pengenalannya dibandingkan pada pengenalan tahap kedua. Namun sebaliknya pada lagu Stop crying your heart out, pengenalan tahap ketiga menunjukkan akurasi yang jauh lebih rendah dibandingkan pengenalan pada tahap kedua.

Dari uraian di atas, tampak bahwa pada dasarnya musik yang diujicobakan pada tahap kedua dan ketiga tidak jauh beda tingkat kesulitan pengenalannya. Walaupun sekilas tampak bahwa musik hasil konversi dari midi lebih 'jernih' di banding musik hasil konversi dari mp3, namun jika musik tersebut dilihat sebagai kumpulan nada, maka kedua jenis musik di atas adalah sama. Hasil pengenalan yang menunjukkan perbedan akurasi lebih disebabkan oleh karakter suara musik itu sendiri yang memang berbeda (pencipta suara musik midi berbeda dengan suara musik hasil konversi dari mp3).

4. Analisis Keseluruhan

Dari kedua tahap uji coba yang telah dilakukan, tampak bahwa uji coba tahap pertama, dimana objek yang dikenali berupa musik solo tanpa vokal memiliki akurasi pengenalan yang lebih tinggi. Sedangkan pada uji coba tahap kedua dimana objek yang dikenali berupa musik utuh memiliki akurasi lebih rendah.

Rendahnya akurasi pada uji coba pada tahap kedua dan ketiga disebabkan karena komposisi nada pada musik utuh lebih kompleks. Dengan kata lain progresi nada pada lagu tersebut lebih kompleks, dimana seringkali nada vokal tidak sesuai dengan nada-nada penyusun *chord*, walaupun nada tersebut masih tetap terdengar harmonis. Selain itu semakin banyaknya jenis alat musik yang dimainkan dengan hasil suara tiap alat musik yang berbeda-beda frekuensi suaranya, menyebabkan

range frekuensi semakin lebar. Hal ini dapat memperbesar kesalahan pengenalan *chord*.

Kesalahan pengenalan sering pula terjadi pada saat pergantian *chord*. Hal ini disebabkan sinyal pada *frame* yang diambil dari bagian lagu saat terjadi perubahan *chord*, memiliki struktur frekuensi yang lebih kompleks, karena merupakan gabungan dari pola dua macam *chord.* Kesalahan pengenalan akibat hal ini bisa diminimalisir dengan teknik *beat detection*, yaitu teknik untuk mendeteksi ketukan (*beat*) dari suatu musik, sehingga apabila dalam suatu *frame* terjadi dua macam *chord* akibat perpindahan nada lagu, maka *frame* itu akan dianalisis sebagai dua *frame* yang berbeda.

Untuk memperoleh akurasi hasil pengenalan yang lebih tinggi, diperlukan pengolahan lebih jauh terhadap sinyal suara sebelum sinyal suara tersebut dijadikan input pengenalan jaringan syaraf tiruan. Diantara metode pengolahan yang mungkin bisa dilakukan yaitu dengan deteksi nada dasar bass, karena suara nada dasar bass cukup merepresentasikan chord itu sendiri, namun masih perlu dikelompokkan lagi apakah nada dasar bass itu mewakili *chord* major atau *chord* minor.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil penelitian yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Jaringan syaraf tiruan dengan pembelajaran Learning Vector Quantization dapat diterapkan dalam pengenalan *chord* dengan hasil yang cukup baik. Sebelum digunakan untuk pengenalan, terhadap jaringan syaraf tiruan dilakukan pembelajaran dengan parameter laju pembelajaran 0,01 dan iterasi maksimal 10000 iterasi. Sampel suara sejumlah 72 suara yang digunakan untuk melatih 24 kelas pengenalan yang masing-masing kelas diwakili oleh 3 prototipe.
2. Hasil uji coba pengenalan *chord* yang telah dilakukan terhadap 30 sampel suara yang terdiri dari 10 lagu dengan tiap lagu terdiri dari 3 macam jenis sampel suara, diperoleh akurasi pengenalan sebesar 94,248% untuk pengenalan terhadap musik solo, 74,374% untuk pengenalan terhadap musik *full*, dan 77,704% untuk pengenalan terhadap musik *full* hasil konversi dari musik midi.
3. Dari hasil chromagram, dapat dilihat bahwa ketidakakuratan lebih disebabkan oleh kondisi sinyal suara yang dijadikan masukan tahap pengenalan, artinya kesalahan lebih disebabkan kurangnya pengolahan pada sinyal suara sebelum menjadi input pengenalan pada jaringan syaraf tiruan.

5.2. Saran

Beberapa sran yang dapat diberikan setelah pengerjaan skripsi ini adalah sebagai berikut:

1. Untuk meningkatkan akurasi, dapat digunakan beberapa metode pengolahan sinyal suara lebih lanjut, misalnya *beat detection* dan, *bass detection*.
2. Kesalahan pengenalan pada pengenalan terhadap musik *full* sering pula disebabkan karena adanya nada dari suara vokal yang terkadang berbeda dengan nada penyusun *chord*. Untuk itu pada proses pengolahan sinyal dapat dilakukan pemisahan suara vokal terlebih dahulu sebelum digunakan sebagai input pengenalan.

3. Selain itu perlu diperhitungkan pula apakah hasil pengenalan *chord* dari suatu *frame* mempunyai nilai yang signifikan terhadap *frame-frame* di sekitarnya. Karena sering terjadi *chord* yang hanya tampil sekali dalam sederet *frame* adalah *chord* yang salah.
4. Sampel untuk pelatihan jaringan perlu diperbanyak, dan dengan kualitas suara yang lebih baik. Hal ini perlu dilakukan untuk menghasilkan bobot pada jaringan yang lebih stabil.

DAFTAR PUSTAKA

Amimaya. 2006. *Pengenalan Suara menggunakan Jaringan Syaraf Tiruan Multilayer dengan Metode Backpropagation*. Universitas Brawijaya. Malang.

Bernsee, Stephan M. 2005. The DFT "a Pied": Mastering The Fourier transform in One Day. http://dspdimension.com. Diakses pada 21 September 2007.

Biehl, Michael, dkk. 2005. The Dynamics of Learning Vector Quantization. European Symposium on Artificial Neural Networks. Bruges.

Brigham, E. Oran. 1988. *Fast Fourier Transform*. Prentice Hall International, Inc. Canada.

Cabral, Giordano, dkk. 2005. *Automatic X Traditional Descriptor Extraction: TheCase of Chord Recognition*. University of London.

Cabral, Giordano, dkk. 2005. *Impact of Distance of Pitch Class Profile Computation*. Universite Pierre et Marie Curie. Paris.

Chu, Wai C. 2003. *Speech Coding Algorithms, Foundation and Evolution of Standardized Coders*. John Wiley & Sons, Inc. Ney Jersey.

Hasugian, Jimmy. 2007. *Teori Musik*. http://jimmyhasugian.com/ministry/. Diakses pada 7 Desember 2007.

Keyser. 1998. *Music Theory for Flamenco*. http://members.aol.com/BuleriaChk/private/flamenco.html. Diakses pada 20 Juni 2008.

Kohonen, Teuvo., dkk. 1996. *LVQ_PAK: The Learning Vector Quantization Program Package*. HelsinkiUniversity of Technology.

Kusumadewi, Sri. 2003. *Artificial Intelligence (Teknik dan Aplikasinya)*. Graha Ilmu. Yogyakarta.

Lee, Kyogu, dan Malcolm Slaney. 2006. *Automatic Chord Recognition from Audio Using an HMM with Supervised Learning*. University of Victoria.

McCuskey, Mason. 2003. *Beginning Game Audio Programming*. Premier Press.

Oppenheim, Alan V and Willsky, Alan S, 2000. *Sinyal dan Sistem Jilid 1 Edisi kedua*. ed. Wibi Hardani. Erlangga. Jakarta.

Pelton, Gordon E, 1993. *Voice Proccesing, International edition.* McGraw. Hill Inc. Singapore.

Proakis, John G and Dimitris Manolakis G, 1995. *Pemrosesan Sinyal Digital (Prinsip-prinsip, Algoritma, dan Aplikasi).* Prentice-Hall International, Inc. Singapore.

Sheh, Alexander, and Daniel Ellis P.W. 2003. *Chord Segmentation and Recognition using EM-Trained Hidden Markov Models.* John Hopkins University.

Tjokronegoro, Harijono A. 2001. *Pengolahan Sinyal*. Penerbit ITB. Bandung.

Wilson, Scott. 2003. *Microsoft Wave Soundfile Format*. http:// ccrma.stanford.edu/courses/422/projects/WaveFormat/. Diakses pada 15 September 2007.

Yani, Eli. 2005. *Pengantar Jaringan Syaraf Tiruan*. MateriKuliah.com.