KONVERSI NADA NADA AKUSTIK MENJADI CHORD MENGGUNAKAN PITCH CLASS PROFILE DAN NEURAL NETWORK BACKPROPAGATION

Febrianzah Junaidy Permana[1], Rizky Yuniar Hakkun, S.Kom, M.Kom[2], Drs. Miftahul Huda, MT [2]
Mahasiswa Jurusan Teknologi Informasi[1] , Dosen Pembimbing [2]
Politeknik Elektronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember
Kampus PENS-ITS Keputih Sukolilo Surabaya 60111
Telp (+62)31-5947280, 5946114, Fax. (+62)31-5946114
Email : pens@eepis-its.edu or junjp@student.eepis-its.edu or junjp@yahoo.co.id
Homepage : http://www.eepis-its.edu

ABSTRAK

Chord recognition atau pengenalan chord adalah sebuah transkripsi dari suara menjadi chord, dimana sebuah masukan yang berupa file audio akan diklasifikasikan sesuai dengan taraf-taraf ketentuan yang berbeda-beda. Chord merupakan beberapa nada yang dibunyikan sehingga menciptakan suatu suara yang harmonis. Chord yang akan dikenali dalam aplikasi ini adalah chord standar yaitu chord mayor dan chord minor. Proses kerja aplikasi diawali dengan proses sampling file audio berformat wave. Langkah selanjutnya adalah membagi data wave menjadi frame-frame (frame blocking). Dari frame frame tersebut kemudian ditranformasikan dengan Fast Fourier Transform (FFT). Proses dilanjutkan dengan deteksi puncak untuk mengambil nilai-nilai puncak dari FFT. Nilai-nilai deteksi puncak tersebut diklasifikasi berdasarkan frekuensi nadanya melaui Pitch Class Profile (PCP). Hasil nilai PCP tersebut digunakan sebagai input data pada algoritma neural network backpropagation untuk proses pelatihan.

Kata kunci: *Chord, Fast Fourier Transform, Pitch Class Profile, Backpropagation*

1. PENDAHULUAN

1.1 Latar Belakang

Perkembangan musik saat ini telah sangat berkembang dengan pesatnya, hal ini disebabkan karena banyaknya minat atau kegemaran dalam bermain musik. Di dalam dunia musik dikenal istilah *chord*, *chord* inilah sebagai acuan nada yang terdapat pada musik atau lagu. *Chord* merupakan beberapa nada yang dibunyikan sehingga menciptakan suatu suara yang harmonis. *Chord – chord* musik sangat beragam dan bervariasi, mulai dari *standar chord*, *minor chord, mayor chord, seventh chord, suspended chord, diminished chord, augmented chord* dll. [1,2]

Untuk memainkan sebuah lagu pemain musik harus mengerti dan mencari *chord-chord* apa saja yang ada pada lagu tersebut. Bagi para musisi atau ahli musik, pencarian *chord* pada suatu lagu bisa jadi suatu hal yang mudah baginya, karena seorang ahli musik telah terlatih pendengaran dan perasaannya dalam mencari dan memainkan nada – nada pada suatu lagu atau musik. Namun sebaliknya, bagi seorang pemula dalam belajar musik, hal ini akan terasa sangat sulit dalam pencarian chord – chord dalam suatu lagu. Karena seorang pemula, pendengarannya belum terlatih dalam mencari dan memainkan nada – nada chord atau kurang memahami karakteristik dari chord – chord dalam musik.

Fungsi untuk mengenali *chord* secara otomatis sangat penting dalam beberapa aplikasi, diantaranya seperti sistem musik interaktif, maupun aplikasi edukasi. *Chord recognition* atau pengenalan *chord* adalah sebuah transkripsi dari suara menjadi *chord*, dimana dapat di klasifikasikan sesuai dengan taraf-taraf ketentuan yang berbeda-beda, dari sebuah perbedaan yang sederhana antara *chord major* dan *minor* dengan sekompleks tipe *chord (maj, min, dim, aug, 7^{th}, dll).*

Pada proyek akhir ini, akan di bangun suatu sistem aplikasi untuk merubah nada nada menjadi chord pada suatu lagu, yang dapat membantu seorang pemain musik atau pemula dalam belajar musik atau mencari dan memainkan chord pada suatu lagu.

1.2 Rumusan Permasalahan

Berdasarkan uraian tersebut di atas, dalam pengerjaan proyek akhir ini timbul beberapa masalah diantaranya adalah :

1. Merekam sample chord – chord untuk pengekstrakan fitur frekuensi.

2. Pembacaa data dari file audio.
3. Pencarian awal dan akhir suatu sinyal.
4. Merubah sinyal dalam dimensi waktu menjadi dimensi frekuensi.
5. Mengelompokkan setiap nilai frekuensi yang ada ke dalam kelas nada.
6. Menentukan arsitektur jaringan saraf tiruan yang optimal.
7. Membuat interface yang dapat menempatkan chord yang telah dikenali menjadi tepat saat dimainkan, dan user friendly atau mudah dioperasikan.

1.3 Batasan Permasalahan

Adapun batasan masalah dalam pembuatan proyek akhir ini, diantaranya adalah sebagai berikut :

1. File input atau lagu yang digunakan adalah berformat .wav.
2. File lagu menggunakan single instrument.
3. Chord yang dikenali adalah chord mayor dan minor.
4. Aplikasi tidak Realtime.

1.4. Tujuan

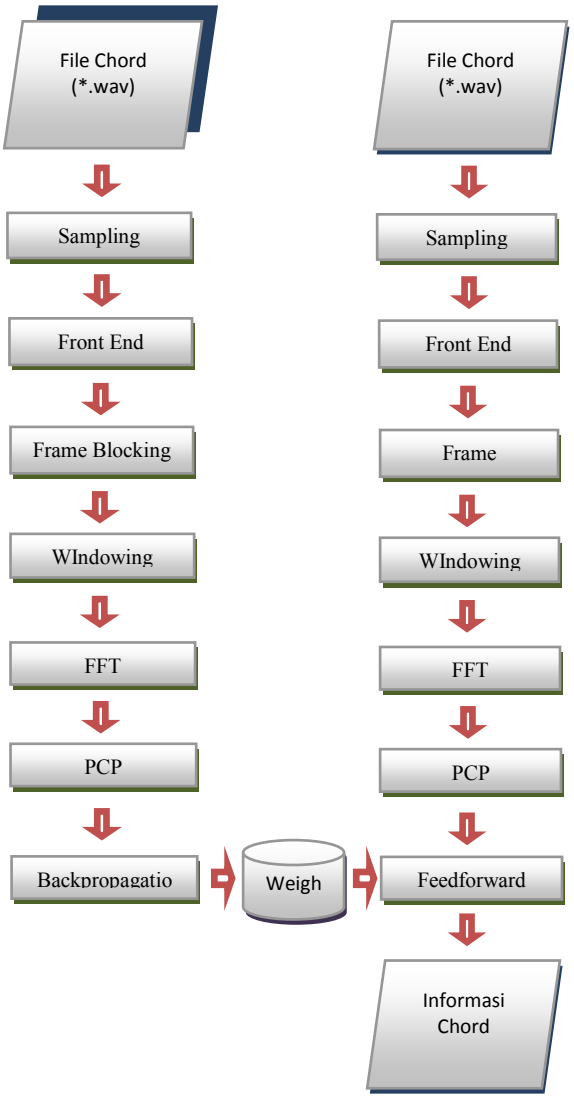
Proyek akhir ini bertujuan untuk membangun suatu perangkat lunak yang dapat mendeteksi *Chord* secara otomatis dengan input file audio, yang mana didalamnya terjadi pemrosesan sinyal audio dan proses learning pada backpropagation. Sehingga audio input akan dikenali oleh sistem, dan menghasilkan output berupa informasi baris-baris chord dari file audio tersebut yang nantinya akan membantu pemain musik dalam mencari chord dari sebuah lagu.

2. PERANCANGAN SISTEM

Pada bab ini, pembahasan materi difokuskan pada perencanaan dan pembuatan sistem yang merupakan pokok pembahasan dari tugas akhir ini. Pembuatan perangkat lunak ini menggunakan software Delphi 7.

Proses kerja sistem pada aplikasi ini meliputi sampling, front end detection, frame blocking, windowing, FFT, PCP, perancangan arsitektur backpropagation, proses learning, dan penyajian informasi chord.

Input sistem yaitu berupa kumpulan chord mayor dan minor yang direkam satu per satu untuk diambil fitur frekuensinya dengan tahapan proses seperti diatas, kemudian di kelompokkan dalam kelas nada, dan hasil tersebut dijadikan sata training backpropagation pada proses learning. Setelah mendapatkan bobot yang tepat dengan error yang kecil, sistem dapat menerima input chord yang akan dikenali informasi chordnya.



Gambar 2.1 *Proses dalam perancangan sistem*

2.1 Sampling

Proses sampling ini adalah proses pembacaan dari input file audio wav. Pembacaan akan disimpan ke dalam array, sehingga file wav dapat diproses melalui penyimpanan ini. Langkah pertama yang harus dilakukan adalah pembuatan tipe data dasar untuk mengambil informasi yang ada pada header file WAV.

2.2 Front-End Detection

Front-end Detection digunakan untuk menentukan batasan suatu sinyal dalam hal ini letak sinyal awal dan akhir dari suatu frame sehingga bentuk sinyal asli tidak berubah. Hal ini dilakukan agar didapatkan sinyal suara tanpa noise karena pada proses pengambilan sampel sinyal suara sering terdapat noise yang mengakibatkan perubahan bentuk sinyal asli.[7]
Untuk menentukan letak awal dan akhir suatu sinyal dapat dilihat berdasarkan power. Sinyal suara terlebih dahulu dibagi menjadi beberapa frame dan tiap frame mempunyai power tertentu. Perhitungan power adalah sebagai berikut

$$P = \sum_{i=0}^{i=N} \overline{X_i^2}$$

Keterangan:
P = Power sinyal suara (dB)
Xi = Frame ke-i (dB)
N = Jumlah sampel per frame
i = Sampel ke-i

Nilai power digunakan untuk membedakan voice atau bukan. Dari nilai power diatas didapatkan nilai rata-rata, dengan menambahkan standart deviasi maka didapatkan nilai awal dan akhir dari suatu frame.
Dalam Statistika dan Probabilitas standard deviasi adalah perkiraan deviasi atau penyimpangan standar dari data-data yang ada. Rata-rata adalah perkiraan nilai rata-rata dari semua data yang mncul. Misalnya, di set data(3, 5), diperoleh rata-rata 3 dan standard deviasi adalah 1.

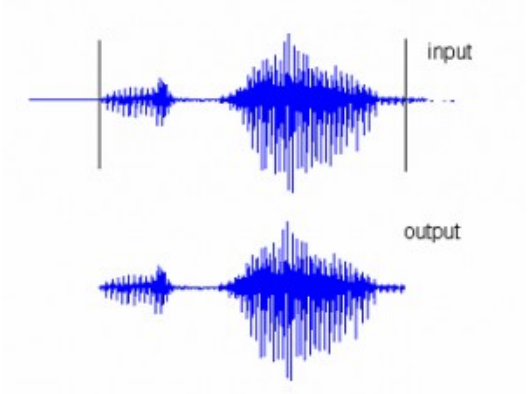
Hubungan dari nilai antara standard deviasi dengan nilai rata-rata adalah jika data-data yang ada mendekati nilai rata-rata maka hasil standart deviasinya keciil(mendekati nol), jika data-data yang ada berbeda jauh dengan nilai rata-rata maka hasil standard deviasinya besar(jauh dari nol), dan jika semua datanya sama maka rata-rata adalah data itu sendiri dan standard deviasinya adalah nolDari data power yang ada akan dicari nilai standard deviasi & rata-rata dengan rumus sebagai berikut

$$\text{Standar Deviasi} = \sum_{i=0}^{n} \frac{x_i^2}{n}$$

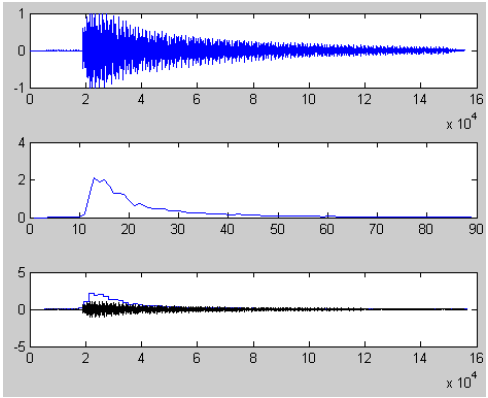
$$Rata-rata = \sum_{i=0}^{n} \frac{x_i}{n}$$

Sinyal dengan nilai power diatas standart deviasi diambil sebagai awal dan akhir sinyal serta dianggap sebagai suatu voice.

voice > mean + standart deviasi



Gambar 2.2 Sinyal pada proses Front-End detection



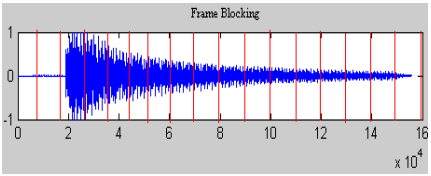
Gambar 2.3 *sinyal yang telah di Power*

Pada proses di atas ***Gambar 2.2*** terdapat tiga grafik sinyal. Grafik 1 proses pembacaan sinyal suara dari sebuah input, grafik 2 sinyal yang telah di Powering, grafik 3 perbedaan antara sinyal yang telah di power dengan sinyal asli

2.3 Frame Blocking

Frame Blocking adalah pembagian sinyal audio menjadi beberapa Frame yang nantinya dapat memudahkan dalam perhitungan dan analisa sinyal, satu frame terdiri dari beberapa sampel

tergantung tiap berapa detik suara akan disampel dan berapa besar frekuensi samplingnya.
Pada proses ini dilakukan pemotongan sinyal dalam slot-slot tertentu agar memenuhi 2 syarat yaitu linear dan time invariant.
Dalam proyek akhir dilakukan percobaan mengambil sampel tiap 65 ms dengan frekuensi sampling sebesar 44100Hz. Ini berarti tiap satu frame terdiri dari 2866 sampel dengan perhitungan sebagai berikut:
Fs = 44100 Hz berarti terdapat 44100 sampel tiap 1 detik
Disampling tiap 65 ms = 0.065 detik
Jumlah sampel tiap frame = (44100 x 0,065) = 2866 sampel
Proses pembentukan frame sinyal audio bisa dilakukan tanpa atau dengan overlapping antara 1 frame dengan frame berikutnya.
Berikut ini contoh frame yang tidak overlapping :

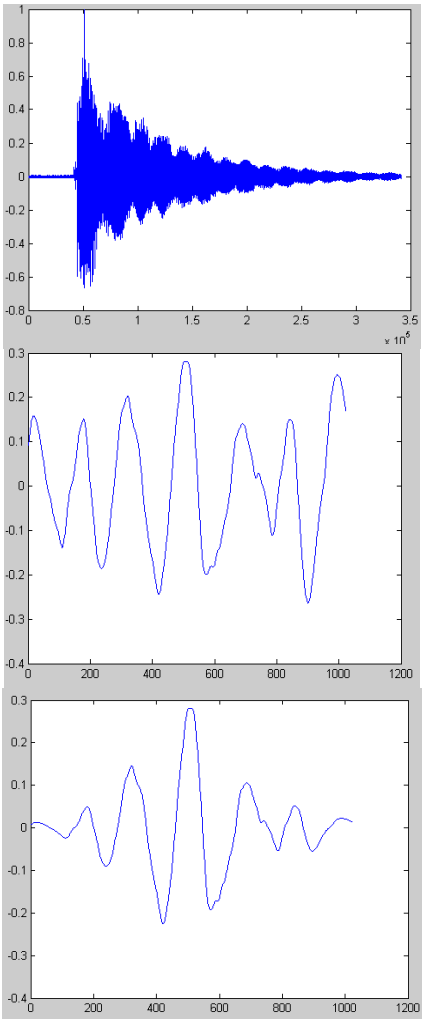


Gambar 2.4 *Frame Blocking pada Chord C pada instrument Piano*

Pada ***Gambar 2.4*** dilakukan proses frame blocking dimana sinyal di sampling tiap 65 ms sehingga ditemukan 2866 sampel dalam Frekuensi Sampling 44100Hz.

2.4 Windowing

Pada waktu pembentukan frame pada suatu sinyal audio secara ideal, diharapkan pada bagian awal dan akhir memiliki nilai magnitude yang mendekati nol, tetapi pada kenyataanya, system yang real menunjukkan magnitude yang bervariasi pada awal dan akhir frame tersebut. Untuk mereduksi puncak pada setiap segmen (awal & akhir suatu frame), kita perlu untuk mengaplikasikan suatu window penghalus pada setiap frame. Untuk meningkatkan kualitas penghalusan ini, kita bisa melakukan overlapping satu frame dengan yang lain, sehingga dapat membangkitkan suatu feature yang lebih halus sepanjang durasi waktu tersebut.
Jenis windowing ada beberapa macam yaitu Hamming, Hanning, Bartlet, Rectanguler dan Blackman.



Gambar 2.5 *Perbedaan sinyal sebelum dan sesudah di hamming pada chord*

Pada gambar terdapat tiga grafik sinyal, grafik 1 sinyal input, grafik 2 sinyal dibaca pada frekuensi 100000 - 101023, dan grafik 3 hasil windowing dari grafik 2.

2.5 FFT

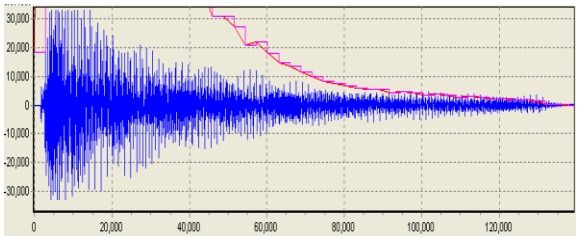
Transformasi Fourier cepat atau Fast Fourier Transform, biasa disingkat FFT adalah suatu algoritma untuk menghitung transformasi Fourier diskrit atau Discrete Fourier Transform, DFT dengan cepat dan efisien. Transformasi Fourier Cepat diterapkan dalam beragam bidang, mulai dari pengolahan sinyal digital, memecahkan persamaan diferensial parsial, dan untuk algoritma untuk mengalikan bilangan bulat besar.

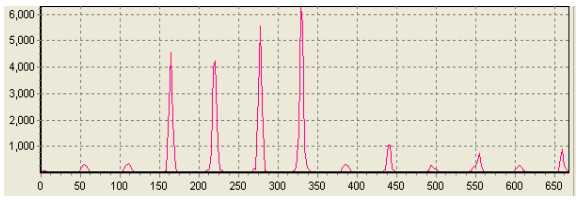
Misalkan "x_0, ..., x_{N-1} merupakan bilangan kompleks. Transformasi Fourier Diskret didefinisikan oleh rumus:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk} \qquad k = 0, \ldots, N -$$

Menghitung deret ini secara langsung memerlukan operasi aritmetika sebanyak O(N^2). Sebuah algoritma FFT hanya memerlukan operasi sebanyak O(N log N) untuk menghitung deret yang sama. Secara umum algoritma tersebut tergantung pada pemfaktoran N.

Setiap algoritma FFT, dengan penyesuaian, dapat diterapkan pula untuk menghitung DFT invers. Ini karena DFT invers adalah sama dengan DFT, namun dengan tanda eksponen berlawanan dan dikalikan dengan faktor 1/N.

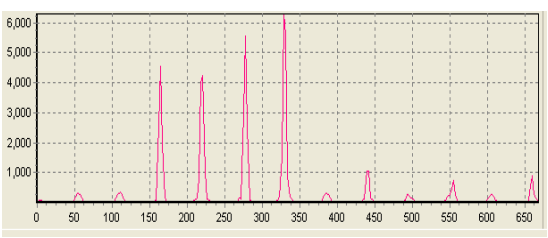


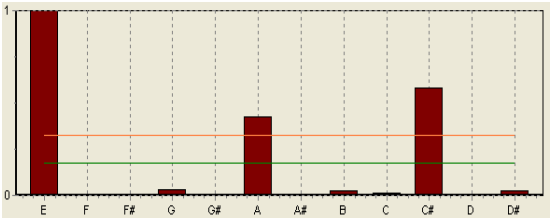


Gambar 2.6 *Sinyal sebelum dan sesudah dilakukan FFT*

2.6 PCP

PCP adalah kepanjangan dari Pitch Class Profile. Sebuah nada dalam music mempunyai standard pitch, yang di identifikasi dengan sebuah nama dan satu oktaf (misal C4). Dengan teknik PCP maka dapat di deteksi komponen dari sebuah chord berdasarkan kelas Pitch dari nada. PCP berawal dari sebuah penyajian frekuensi antara lain Transformasi Fourier, setelah di frekuensi di hitung melalui Transformasi Fourier maka frekuensi dipetakan ke dalam 12 pitch kelas(C, C#, D, D#, E, F, F#, G, G#, A, A#, B).





Gambar 2.7 *FFT dan Hasil PCP dari chord A major(E, A, C#)*

2.7 BACKPROPAGATION

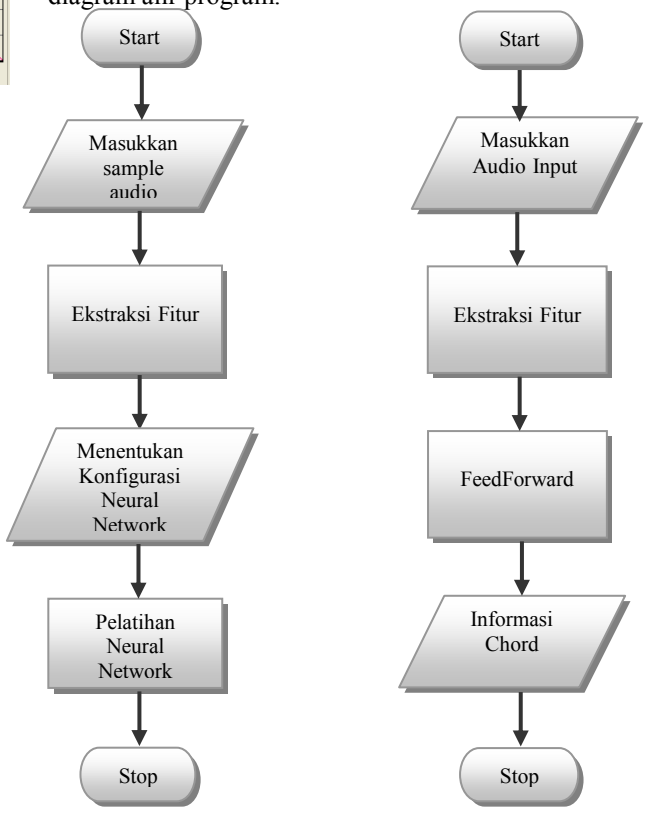
Dari hasil PCP yang telah didapatkan digunakan sebagai data training pada proses pelatihan backpropagation. Sebelum proses pelatihan dilakukan, perlu ditentukan arsitektur jaringan saraf yang digunakan. Berikut ini arsitektur yang digunakan :

- 12 neuron masukan
- *n* lapisan tersembunyi dengan masing masing *n* neuron dan 1 bias.
- 5 neuron keluaran

Selain menentukan arsitekture, diperlukan juga nilai pembelajaran (learning rate), nilai error maksimum, dan jumlah epoch maksimum.

3. PEMBUATAN PROGRAM

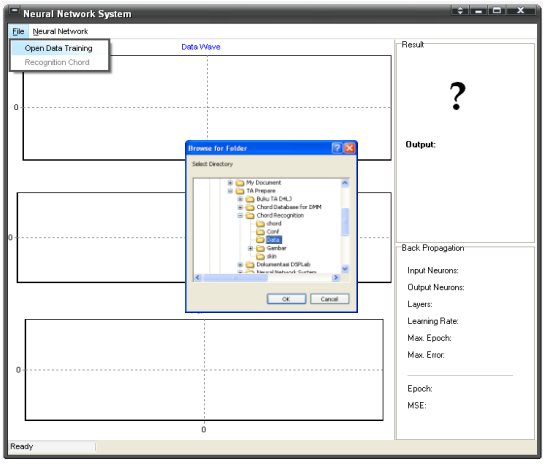
Pembuatan program ditulis dalam bahasa pemrograman pascal dengan menggunakan software developer Delphi 7. Berikut ini adalah diagram alir program.



Gambar 3.1 *Diagram alir program*

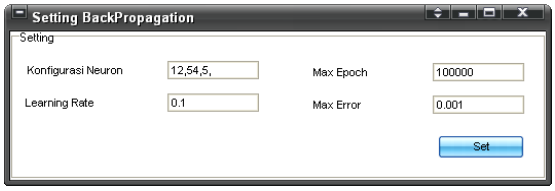
Berdasarkan diagram alir di atas, secara garis besar, prorgam terbagi atas 2 proses, yaitu persiapan program dan pengenalan chord masukan. Proses persiapan program adalah tahap dimana program akan menerima audio sampel berupa chord dan melakukan pelatihan jaringan saraf tiruan. Setelah proses tersebut, maka program telah siap untuk menerima chord masukan yang akan dikenali.

Sedangkan proses yang kedua adalah proses pengenalan chord masukan dengan menggunakan jaringan saraf tiruan yang telah siap. Program pertama kali akan me-load direktori yang berisi audio sampel berupa chord dan memproses setiap chord yang ada di dalamnya.

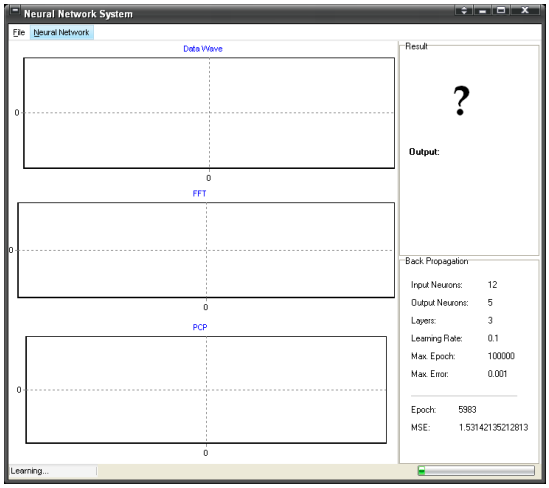


Gambar 3.2 *Proses pemasukan data training*

Setelah proses tersebut, program akan menerima masukan dari pengguna untuk konfigurasi jaringan saraf tiruan dan program mulai melakukan proses pelatihan.

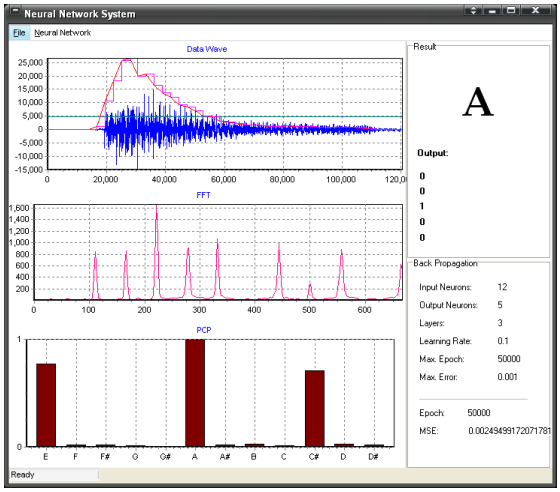


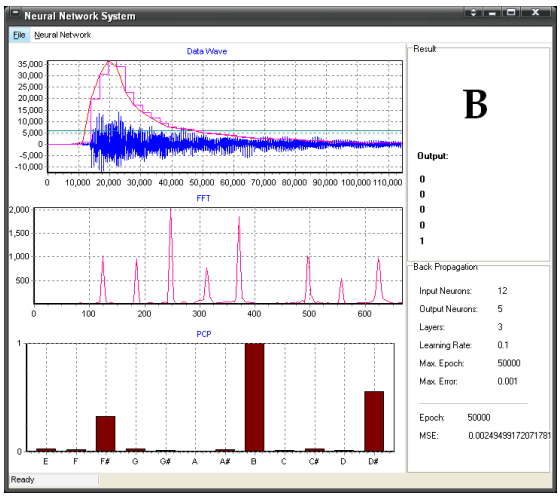
Gambar 3.3 *Proses pemasukan konfigurasi backpropagation*



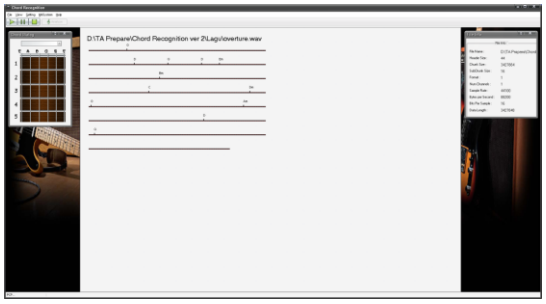
Gambar 3.4 *Proses learning*

Setelah program selesai melakukan pelatihan, maka program telah siap untuk melakukan pengenalan chord. Pengguna hanya me-load 1 data file chord yang akan dikenali. Berikut ini adalah tampilan program setelah berhasil mengenali informasi chord pada file audio.

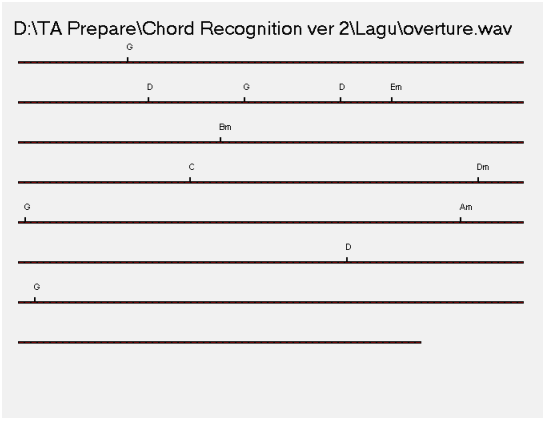




Gambar 3.5 *Tampilan program pada proses pengenalan*



Gambar 3.6 *Tampilan program pengenalan chord pada lagu*



Gambar 3.7 *Bar chord pada lagu*

4. UJI COBA DAN ANALISA

Beberapa percobaan dilakukan untuk mengetahui akurasi dari sistem dengan arsitektur jaringan saraf tiruan yang berbeda, yaitu berdasarkan jumlah layer dan jumlah neuron pada 1 hidden layer.

Untuk percobaan jumlah layer, arsitektur yang digunakan adalah :

- 12 neuron pada input layer.
- *n* hidden layer.
- 54 neuron pada setiap hidden layer.
- 1 bias pada input dan hidden layer.
- 5 neuron pada output layer.

Sedangkan untuk percobaan jumlah neuron, arsitektur yang digunakan adalah :

- 12 neuron pada input layer.
- 1 hidden layer.
- *n* neuron pada setiap hidden layer.
- 1 bias pada input dan hidden layer.
- 5 neuron pada output layer.

4.1 Percobaan I

Percobaan pertama menggunakan data – data sebagai berikut,

- 72 chord sebagai data training, yang tiap tiap chord terdapat 3 buah.
- 72 chord yang sama dengan data training sebagai data percobaan.

Percobaan Jumlah Layer

Percobaan ini menggunakan jumlah layer pada jaringan backpropagation yang diubah ubah. Berikut ini adalah hasilnya :

Jumlah layer	Chord	Jumlah	Akurat	Error
4	Mayor	36	33	3
	Minor	36	31	5
	Jumlah	**72**	**64**	**8**
Jumlah layer	**Chord**	**Jumlah**	**Akurat**	**Error**
5	Mayor	36	32	4
	Minor	36	34	2
	Jumlah	**72**	**66**	**6**
Jumlah layer	**Chord**	**Jumlah**	**Akurat**	**Error**
6	Mayor	36	32	4
	Minor	36	30	6
	Jumlah	**72**	**62**	**10**
Jumlah layer	**Chord**	**Jumlah**	**Akurat**	**Error**
7	Mayor	36	33	3
	Minor	36	31	5
	Jumlah	**72**	**64**	**8**

Tabel 4. 1 *Hasil percobaan jumlah layer pada percobaan 1*

Percobaan Jumlah Neuron

Percobaan ini menggunakan jumlah neuron pada satu hidden layer sebagai parameter yang diubah-ubah. Hasilnya disajikan pada tabel di bawah ini

Jumlah neuron	Chord	Jumlah	Akurat	Error
12	Mayor	36	18	18
	Minor	36	14	22
	Jumlah	**72**	**32**	**40**
Jumlah neuron	**Chord**	**Jumlah**	**Akurat**	**Error**
24	Mayor	36	24	12
	Minor	36	28	8
	Jumlah	**72**	**52**	**20**

Jumlah neuron	Chord	Jumlah	Akurat	Error
36	Mayor	36	14	22
	Minor	36	22	14
	Jumlah	**72**	**36**	**36**
Jumlah neuron	**Chord**	**Jumlah**	**Akurat**	**Error**
48	Mayor	36	20	16
	Minor	36	11	25
	Jumlah	**72**	**31**	**41**
Jumlah neuron	**Chord**	**Jumlah**	**Akurat**	**Error**
54	Mayor	36	35	1
	Minor	36	34	2
	Jumlah	**72**	**69**	**3**

Tabel 4. 2 *Hasil percobaan junlah neuron pada percobaan 1*

4.2 Percobaan II

Pada percobaan kedua, data yang digunakan sebagai berikut :

- 72 chord mayor sebagai data training, yang tiap tiap chord terdapat 3 buah.
- 168 chord mayor minor dan 72 chord mayor minor dari data training yang sama sebagai data testing.

Percobaan Jumlah Layer

Percobaan ini menggunakan jumlah layer pada jaringan backpropagation yang diubah ubah. Berikut ini adalah hasilnya :

Jumlah layer	Chord	Jumlah	Akurat	Error
4	Mayor	120	96	24
	Minor	120	101	19
	Jumlah	**240**	**197**	**43**
Jumlah layer	Chord	Jumlah	Akurat	Error
5	Mayor	120	102	18
	Minor	120	110	10
	Jumlah	**240**	**212**	**28**
Jumlah layer	Chord	Jumlah	Akurat	Error
6	Mayor	120	95	25
	Minor	120	101	19
	Jumlah	**240**	**196**	**44**
Jumlah layer	Chord	Jumlah	Akurat	Error
7	Mayor	120	105	15
	Minor	120	100	20
	Jumlah	**240**	**205**	**35**

Tabel 4. 3 *Hasil percobaan jumlah layer pada percobaan 2*

Percobaan Jumlah Neuron

Percobaan ini menggunakan jumlah neuron pada satu hidden layer sebagai parameter yang diubah-ubah. Hasilnya disajikan pada tabel di bawah ini.

Jumlah neuron	Chord	Jumlah	Akurat	Error
12	Mayor	120	94	26
	Minor	120	103	17
	Jumlah	**240**	**197**	**43**
Jumlah neuron	Chord	Jumlah	Akurat	Error
24	Mayor	120	69	51
	Minor	120	79	41
	Jumlah	**240**	**148**	**92**
Jumlah neuron	Chord	Jumlah	Akurat	Error
36	Mayor	120	81	39
	Minor	120	61	59
	Jumlah	**240**	**142**	**98**
Jumlah neuron	Chord	Jumlah	Akurat	Error
48	Mayor	120	93	27
	Minor	120	87	33
	Jumlah	**240**	**180**	**60**
Jumlah neuron	Chord	Jumlah	Akurat	Error
54	Mayor	120	108	12
	Minor	120	114	6
	Jumlah	**240**	**222**	**18**

Tabel 4. 4 *Hasil percobaan junlah neuron pada percobaan 2*

Banyak faktor yang mempengaruhi keakuratan hasil pengenalan. Beberapa diantaranya yaitu, inisialisasi penimbang, adaptasi penimbang, parameter laju pelatihan dan penentuan jumlah lapis tersembunyi. Banyak studi empiris membuktikan bahwa meneruskan pelatihan pada saat galat mencapai nilai yang kecil dan stabil atau datar, akan menghasilkan nilai – nilai penimbang yang tidak diinginkan. Dan pada banyak penelitihan menunjukkan bahwa konvergensi tidak akan dicapai bila penimbang kurang bervariasi. Sedangkan parameter laju pelatihan sangat berpengaruh pada intensitas proses pelatihan, efektivitas, dan kecepatan mencapai konvergensi dari pelatihan.

5. KESIMPULAN DAN SARAN

a. Kesimpulan

Berdasarkan pada hasil pengujian dan analisa terhadap hasil yang didapatkan, maka dapat diambil suatu kesimpulan yaitu :

1. Aplikasi ini mengasilkan informasi – informasi *chord* dari input sebuah lagu dimana informasi yang disajikan lebih cepat dari pada pencarian secara manual oleh pemain musik.
2. Tingkat keberhasilan atau keakuratan tergantung dari proses pembelajaran, jumlah data training dan kualitas data training tersebut.
3. Aplikasi ini dapat menyimpan dan meload hasil bobot akhir.

b. Saran

Mengingat masih banyaknya hal-hal yang belum dapat diimplementasikan pada proyek akhir ini, maka mempertimbangkan beberapa saran untuk perbaikan-perbaikan proyek akhir ini dalam hal:

1. Perekaman file data training dan pengenalan sebaiknya memperhatikan kualitas, baik dari permainan gitar maupun proses perekaman.
2. Data pelatihan kira-kira 60% dari data total, hal ini dapat menambah keakuratan sistem dalam proses pengenalan .
3. Percobaan sebaiknya dilakukan lebih banyak, agar dapat menemukan konfigurasi yang tepat dan akurat.

6. DAFTAR PUSTAKA

[1] Artikel tentang “Pengenalan Chord”, http://ratdix.wordpress.com, 2008.

[2] Artikel tentang“Akord”, http://id.wikipedia.org/wiki/Akord

[3] Audio Sampling, http://www.elektroindonesia.com/elektro/elek35a.html

[4] Artikel dan Tutorial tentang “Front-End Detection”, http://agusslamet.wordpress.com/2008/09/front-end-detection/, 2008.

[5] Pengolahan Sinyal Digital, Adit, http://adhit8.blogspot.com/2010/04/pengolahan-sinyal-digital-menggunakan.html

[6] Window Function, http://en.wikipedia.org/wiki/Window_function

[7] Fast Fourier Transform, Wikipedia, http://en.wikipedia.org/wiki/Fast_Fourier_transform

[8] Realtime Chord Recognition of Musical Sound : a System Using Common Lisp Music, Takuya Fujishima, CCRMA, Stanford University.

[9] Kusumadewi; Sri. 2003. Artificial Intelligence (Teknik & Aplikasinya). Yogyakarta: Graha Ilmu

[10] Mauridhi Hery P dan Agus Kurniawan,”SUPERVISED NEURAL NETWORKS dan aplikasinya”, Graha Ilmu,Yogyakarta:2006

[11] Febrianzah Junaidy Permana, “Pembuatan Database Software Musik Digital Mentor”, Politeknik Elektronika Negeri Surabaya : 2009