PERANCANGAN PROTOTYPE REAL-TIME AKOR PIANO DENGAN METODE HARMONIC-FFT

DESIGN OF REAL-TIME PROTOTYPE OF PIANO CHORD WITH HARMONIC-FFT METHOD

Melati Wahyutami[1], Gelar Budiman, ST., MT[2]., Inung Wijayanto, ST., MT[3]

[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]melatiw@gmail.com, [2]gelarbudiman@telkomuniversity.ac.id, [3]iwijayanto@telkomuniversity.ac.id

Abstrak

Musik adalah untaian nada yang dapat dinikmati semua orang. Seiring berkembangnya jaman, banyak pendengar musik yang ingin mempelajari bagaimana cara memainkan alat musik, baik nada tunggal maupun akornya. Tetapi dalam mempelajarinya, banyak pemula yang mengalami kesulitan dalam menentukan akor dari suatu lagu. Oleh karena itu, dibutuhkan aplikasi untuk membantu dalam mempelajarinya.

Dalam Tugas Akhir ini dirancang suatu sistem yang dapat menentukan akor yang tepat dari suatu lagu. Hal ini dilakukan dengan mendeteksi nada tunggal yang dimasukkan dan mencocokkan hasil deteksi dengan *database* untuk menentukan akor yang sesuai. Sistem ini menggunakan metode *Harmonic-FFT* dan *JST Backpropagation. Harmonic-FFT* digunakan untuk mengambil ciri *file* audio yang dimasukkan sedangkan *JST Backpropagation* digunakan untuk mengenali atau mengklasifikasi suara masukan.

Presentase akurasi rata-rata yang didapat dari haasil pengujian pada sistem ini adalah 61,49%. Dengan presentase paling baik yang diperoleh adalah 78,57% pada lagu aura dengan jarak 4. Sedangkan dari hasil MOS, nilai yang paling baik adalah 3,50 pada lagu Clementine dengan jarak 3.

Kata kunci : *Real-Time* , Akor, *Harmonic-FFT, JST Backpropagation*

Abstract

Music is series of tone which can be enjoyed by all people. Along the development era, many music listeners want to learn how to play music instruments, either single tones or chords. But in studying it, a lot of beginners confused in determining chords in a song. Therefore, the application is necessary to helps in learn it.

In this Final Project, designed a system that can determining right chords in the songs. It is done by detecting single tone which is entered and match the result of detection with database to determine appropriate chord. This system is using Harmonic-FFT and JST Backpropagation methods. Harmonic-FFT is use to extract feature (audio file) dan JST Backpropagation use to identify or classify the input.

Average of the accuracy percentage of the test results obtained in this system is 61.49%. With most good percentage of 78.57% was obtained in the song Aura with distance 4. While the results of the MOS, the best value was 3.50 in the song Clementine with distance 3.

Keywords : Real-Time , Chord, Harmonic-FFT, JST Backpropagation

2.1 Pendahuluan

Musik adalah untaian nada yang dapat dinikmati oleh semua orang.Seiring berkembangnya jaman, industri musik pun turut berkembang menjadi industri yang menjanjikan.Banyak pendengar musik yang ingin belajar memainkan musik, baik mempelajari nada tunggal maupun akornya.Namun dalam mempelajarinya, banyak pemula yang mengalami kesulitan dalam menentukan akor yang tepat dalam sebuah lagu.Oleh karena itu, dibutuhkan aplikasi untuk mempermudah dalam bermain musik.

Pada Tugas Akhir terdahulu[7] pernah dibuat sistem yang dapat menentukan akor dari nada tunggal piano. Dalam sistem ini digunakan metode *FFT* dan Jaringan Syaraf Tiruan ART 2. Namun, sistem tersebut masih terbatas proses *yang non-real time*.

Untuk mengatasi masalah di atas, maka dalam Tugas Akhir ini akan dibuat aplikasi yang dapat menentukan akor dari suatu lagu secara *real-time*. Hal ini dilakukan dengan cara mendeteksi nada tunggal yang dimasukkan dan mencocokkan hasil deteksi dengan *database* untuk menentukan akor yang sesuai. Sistem mendeteksi nada tunggal piano dengan menggunakan metode *Harmonic-FFT* dan akan memberikan keluaran berupa akor dari nada tunggal yang dimasukkan.

Pada Tugas Akhir ini, akan dianalisis bagaimana akurasi sistem dalam menetukan nada tunggal dan akor serta dianalisis pula kualitas penentuan akor berdasarkan tingkat akurasi dan MOS. Sistem ini menggunakan *file* .wav sebagai masukan. *File* masukan tersebut berisikan lagu yang berupa nada tunggal piano. Jumlah lagu yang

dipakai adalah empat lagu, yaitu Aura, Balonku, Clementine, dan Twinkle. Simulasi sistem ini akan dilakukan menggunakan *software* Matlab 2010a.

2.2 Dasar Teori dan Perancangan

2.1 Nada[11]

Nada adalah bunyi yang beraturan, yaitu memiliki frekuensi tunggal tertentu. Dalam teori musik, setiap nada memiliki tinggi nada atau tala tertentu menurut frekuensinya ataupun menurut jarak relatif tinggi nada tersebut terhadap tinggi nada patokan. Nada dasar suatu karya musik menentukan frekuensi tiap nada dalam karya tersebut. Nada dapat diatur dalam tangga nada yang berbeda-beda. Istilah nada sering dipertukarkan dengan "not" walaupun kedua istilah tersebut memiliki perbedaan arti.

Notasi musik adalah sistem penulisan karya musik. Dalam notasi musik, nada dilambangkan oleh not. Notasi musik standar saat ini adalah notasi balok, yang didasarkan pada paranada dengan lambang untuk tiap nada menunjukkan durasi dan ketinggian nada tersebut. Tinggi nada digambarkan secara vertikal sedangkan waktu atau ritme digambarkan secara horizontal. Terdapat pula notasi bentuk lain misalnya notasi angka yang juga digunakan di negara-negara Asia termasuk Indonesia.

2.2 Akor[7]

Akor adalah kumpulan tiga nada atau lebih yang bila dimainkan secara bersamaan terdengar harmonis. Akor bisa dimainkan secara terputus-putus ataupun secara bersamaan sehingga terbentuk suatu iringan atau backsound yang jika diisi nada yang tepat akan membuat suatu lagu lebih enak didengar dan dapat menimbulkan nuansa pada lagu tersebut, mungkin bisa senang, sedih, ramai, lesu dan sebagainya tergantung pendengar

Akor terdiri dari beberapa jenis, antara lain akor mayor, akor minor, akor dominan septim, akor *diminished*, akor *augmented*, akor minor 6, akor mayor 7, akor *suspended* dan masih banyak yang lainnya. Akor yang digunakan dalam tugas akhir ini adalah akor mayor dan akor minor. Penyisipan akor yang berbeda akan memberikan efek nuansa yang berbeda dalam iringan suatu lagu.

2.3 Fast Fourier Transform[1][6]

Fast Fourier Transform (FFT) adalah sebuah algoritma untuk menghitung penjumlahan dengan cepat. Algoritma ini dikembangkan dari algoritma Discrete Fourier Transform (DFT). Rumus dari FFT adalah sebagai berikut :

$$X(k) = \sum_{n=0}^{N-1} x(n).\, e^{\frac{-j2\pi(k-1)(n-1)}{N}} ;\ 0 \leq k \leq N\text{-}1 \qquad (1)$$

Keterangan :

X(k) = Hasil FFT, sinyal dalam domain frekuensi

x(n) = sampel ke-n dalam suatu frame

N = N titik FFT, nilai N akan sama dengan nilai nframe pada proses segmentasi

Dalam matematika, terdapat bilangan yang dinamakan root of unity. Nilai bilangan ini diangkat ke dalam bentuk integer adalah kurang lebih 1. Namun, bilangan ini menjadi sangat penting bila digunakan dalam teori bilangan ataupun juga teori transformasi fourier.

FFT dibagi menjadi dua metode yaitu decimation in time/DIT (FFT penyingkat waktu) bila dilakukan pada data kawasan waktu dan metode decimation in frequency (DIF) yang sama-sama berfungsi mentransformasi sinyal menjadi spectrum frekuensi yang pertama kali dikenalkan pada tahun 1965 oleh Cooley dan Tukey. Decimation adalah proses pembagian sinyal menjadi beberapa bagian yang lebih kecil yang bertujuan untuk memperoleh waktu proses yang lebih cepat. Langkah awal yang dilakukan adalah memisahkan dua bagian titik data, deret data nomor-nomor genap dan deret data nomor-nomor ganjil. Semakin besar nilai titik FFT, maka hasil deteksi frekuensi oleh FFT akan semakin mendekati benar. Namun di lain sisi semakin banyak nilai titik FFT akan semakin lama waktu proses.

2.3 *Harmonic-FFT*

*Harmonic-FF*T merupakan pengembangan dari FFT (*Fast Fourier Transform*).Karena pada FFT masih ada kekurangan dalam menganalisis sinyal yang harmonik, maka dikembangkanlah *Harmonic-FFT*.

Harmonic-FFT adalah metode yang mengadopsi kemampuan persepsi pendengaran manusia. *Harmonic-FFT* bekerja dengan memodifikasi nilai mutlak dari FFT dengan cara mengambil nilai frekuensi *harmonic* data. Hal ini dilakukan dengan cara menyamakan nilai-nilai frekuensi *harmonic* terhadap nilai maksimum dari frekuensi *harmonic* yang ada dan meredam nilai-nilai di luar frekuensi *harmonic*. Kemudian nilai-nilai tersebut dirata-ratakan agar jumlah ciri yang didapat tidak terlalu banyak. Untuk menyamakan perlakuan data yang satu dengan yang lain dilakukan normalisasi nilai sehingga masuk ke dalam rentang nilai 0 sampai 100. Tahap terakhir dilakukan penajaman nilai dengan cara meredam nilai yang berada di bawah *threshold*.

2.5 Jaringan Saraf Tiruan[5][9]

Jaringan saraf tiruan atau biasa disingkat JST, adalah sistem komputasi dimana arsitektur dan operasi diilhami dari pengetahuan tentang sel saraf biologis di dalam otak. JST dapat digambarkan sebagai model matematis dan komputasi untuk fungsi aproksimasi non-linier, klasifikasi data kluster dan regresi non-parametrik atau sebuah simulasi dari koleksi model saraf biologi.

Pada jaringan saraf tiruan juga terdapat istilah neuron atau sering juga disebut dengan unit, sel atau node. Setiap neuron terhubung dengan neuron-neuron lain melalui layer dengan bobot tertentu. Bobot disini melambangkan informasi yang digunakan oleh jaringan untuk menyelesaikan persoalan. Pada jaringan saraf biologis, bobot ini dapat dianalogikan dengan aksi pada proses kimia yang terjadi pada *sypnatic gap.* Sedangkan setipa neuron sendiri mempunyai *internal state* yang disebut aktivasi dimana aktivasi ini merupakan fungsi dari *input* yang diterima.

Suatu neuron akan mengirim sinyal ke neuro-neuron lain, tetapi dalam suatu saat hanya ada satu sinyal yang dapat dikeluarkan walaupun sinyal tersebut ditransmisikan pada beberapa neuron lain.

2.6 Backpropagation[5][9]

Secara garis besar, mengapa algoritma ini disebut sebagai propagasi balik, dapat dideskripsikan sebagai berikut, ketika jaringan diberikan pola masukan sebagai pola pelatihan maka pola tersebut menuju ke unit-unit pada lapisan tersembunyi (*hidden layer*) untuk diteruskan ke unit-unit lapisan keluaran. Kemudian unit-unit lapisan keluaran memberikan tanggapan yang disebut keluaran jaringan. Saat keluaran jaringan tidak sama dengan keluaran yang diharapkan maka keluaran akan menyebar mundur (*backward*) pada lapisan tersembunyi diteruskan ke unit pada lapisan masukan. Oleh karenanya maka mekanisme pelatihan tersebut dinamakan *backpropagation* / propagasi balik.

Pada intinya, pelatihan dengan metode backpropagation terdiri dari tiga langkah, yaitu fase umpan maju (*feedforward*), fase perhitungan dan propagasi balik dari *error* yang bersangkutan dan fase pembobotan.

Secara detail, langkah-langkah dan fase pelatihan *backpropagation* adalah sebagai berikut :

a) Langkah 0 : insialisasi bobot dan bias
   Baik bobot maupun bias dapat diset dengan sembarang angka (acak) dan biasanya angka di sekitar 0 dan 1 atau -1 (bias positif atau negatif).
b) Langkah 1 : jika *stopping condition* masih belum terpenuhi, jalankan langkah 2-9.
c) Langkah 2 : untuk *data training*, lakukan langkah 3-8.

Fase Umpan Maju (*feedforward*)

d) Langkah 3 : Setiap unit *input* (Xi, i=1,..,n) menerima sinyal dan menyebarkan sinyal tersebut pada seluruh unit pada *hidden units*. Perlu diketahui bahwa *input* Xi yang dipakai adalah *input training* data yang diskalakan. Pertama, *input* yang mungkin dipakai dalam sistem dicari nilai terendah dan tertingginya. Kemudian skala yang digunakan tergantung dari fungsi aktivasinya.
e) Langkah 4 : Setiap *hidden* unit (Zj, j=1,..,p) akan menjumlahkan sinyal-sinyal *input* yang sudah terbobot, termasuk biasnya.

$$z_{in_j} = v_{0j} + \sum^{n} x_i v_{ij} \quad (2)$$

Dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dari *hidden* unit yang bersangkutan,

$$z_j = f(z_{in_j}) \quad (3)$$

Lalu mengirim sinyal *output* ini ke seluruh unit pada unit *output.*

f) Langkah 5 : setiap unit *output* (Yk, k=1,..,m) akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot, termasuk biasnya,

$$y_{in_k} = w_{0k} + \sum^{p} z_j w_{jk} \quad (4)$$

Dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dari unit *output* yang bersangkutan

$$y_k = f(y_{in_k}) \quad (5)$$

Propagasi Error (*Backpropagation of Error*)

g) Langkah 6 : Setiap unit *output* (Yk, k=1,..,m) menerima suatu pola target (*desired output*) yang sesuai dengan *input training pattern* untuk menghitung kesalahan (*error*) antara target dengan *output* yang dihasilkan jaringan,

$$\delta_k = (t_k - y_k) f'(y_{in_k}) \quad (6)$$

Sebagaimana *input training* data, *output training* data t_k juga telah diskalakan menurut fungsi aktivasi yang dipakai. Faktor δ_k ini digunakan untuk menghitung koreksi *error* ($\delta_{w_{jk}}$) yang nantinya akan dipakai untuk memperbaharui w_{jk}, dimana :

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (7)$$

Selain itu juga dihitung koreksi bias Δw_{0k} yang nantinya dipakai untuk memperbaharui w_{0k}, dimana :

$$\Delta w_{0k} = \alpha \delta_k \quad (8)$$

Faktor δ_k ini kemudian dikirimkan ke layer yang berada pada langkah 7.

h) Langkah 7 : Setiap *hidden* unit (Zj, j=1,..,p) menjumlah *input* delta (yang dikirim dari layer langkah 6) yang sudah berbobot.

$$\delta_{in_j} = \sum^m \delta_k w_{jk} \quad (9)$$

Kemudian hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi *error* delta j, dimana :

$$\delta_j = \sum^m \delta_{in_j} f'(z_{in_j}) \quad (10)$$

Faktor δ_j ini digunakan untuk menghitung koreksi *error* (Δv_{ij}) yang nantinya akan dipakai untuk memperbaharui v_{ij}, dimana :

$$\Delta v_{ij} = \alpha \delta_j x_i \quad (11)$$

Selain itu juga dihitung koreksi bias Δv_{0j} yang nantinya akan dipakai untuk memperbaharui v_{0j}, dimana :

$$\Delta v_{0j} = \alpha \delta_j \quad (12)$$

Pembaharuan (*adjustment*) bobot dan bias

i) Langkah 8 : Setiap unit *output* (Yk, k=1,..,m) akan memperbaharui bias dan bobotnya dari setiap *hidden* unit (j=0,..,p),

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (13)$$

Demikian pula untuk setiap *hidden* unit (Zj, j=1,..,p) akan memperbaharui bias dan bobotnya dari setiap unit *input* (i=0,..,n),

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad (14)$$

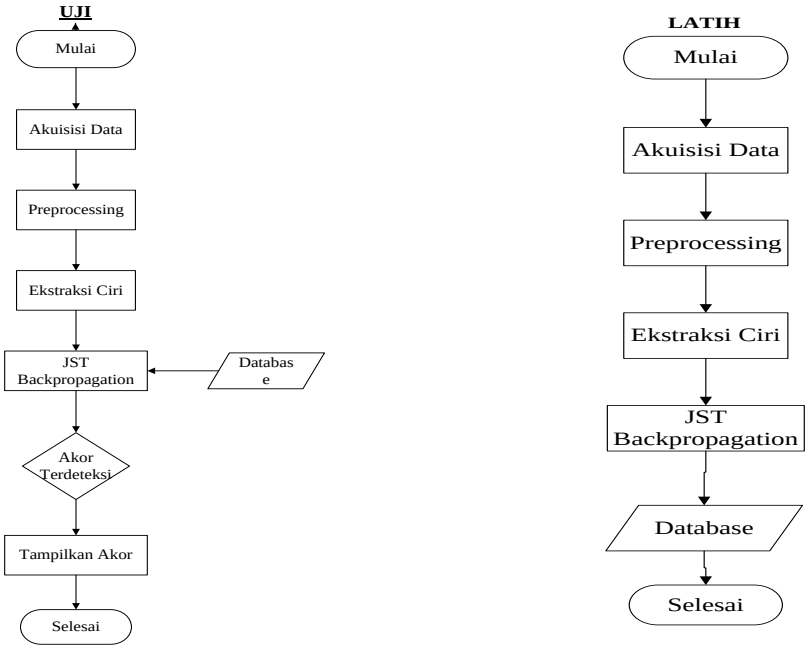
j) Langkah 9 : Memeriksa *stopping condition*. Jika *stop condition* telah terpenuhi, maka pelatihan jaringan dapat dihentikan.

Untuk menentukan *stopping condition* terdapat dua cara yang biasa dipakai :

1. Pertama, dengan membatasi iterasi yang ingin dilakukan. Misalnya jaringan akan dilatih sampai pada iterasi ke-500. Yang dimaksud satu iterasi disini adalah perulangan langkah ke-3 sampai ke-8 untuk semua *training* data yang ada.
2. Cara kedua adalah membatasi *error*. Untuk metode *backpropagation*, dipakai *Mean Square Error* untuk menghitung rata-rata *error* antara *output* yang dikehendaki pada *training* data dengan *output* yang dihasilkan oleh jaringan.

Selain kedua cara di atas, ada sebuah pertimbangan lagi yang dapat dipakai untuk menghentikan pelatihan. Ada kalanya sebelum mencapai kondisi seperti yang diinginkan, *error* justru semakin besar. Kejadian seperti ini disebut *overtraining*. Kondisi *error* yang diperhatikan disini tidak hanya *error* dari *training set*, tapi juga dari *test set*. Jika salah satu saja dari *training set error* atau *test set error* bertambah besar, maka pelatihan harus dihentikan.

2.7 Perancangan Sistem



Gambar 1 Blok Diagram Sistem Uji dan Sistem Latih

Cara kerja dari sistem ini adalah masukan data berupa rekaman nada tunggal piano. Data audio yang telah dimasukan akan masuk ke dalam proses *preprocessing*. Sinyal masukan akan disaring atau di-*filter* pada proses *filtering*. Kemudian akan dilakukan pemotongan komponen yang dianggap bukan sinyal pada proses *cropping*. Setelah itu, dilakukan normalisasi dan penentuan onset dari sinyal keluaran *cropping*. Pada ekstraksi ciri, akan diambil ciri dari sinyal keluaran blok *preprocessing*. Hasil keluaran dari ekstraksi ciri akan masuk ke blok *JST Backpropagation* untuk menentukan nada *input* apa yang sedang dimainkan. Setelah menentukan jenis nada tunggalnya, akor akan ditentukan dengan melihat nada tunggal terakhir yang dimainkan. Keluaran berupa akor piano yang tepat untuk nada tunggal yang dimasukkan.

3. Pembahasan

3.1 Analisis Hasil Output Sistem

3.1.1 Berdasarkan MOS

Nilai MOS didapatkan dari hasil survey yang melibatkan 30 orang mahasiswa yang menilai secara subjektif tingkat akurasi dari setiap lagu. Data MOS ini didapat dengan menggunakan lembar kuisioner terlampir. Berikut adalah dasar kualitas hasil yang diajukan

$4{,}5 < \text{nilai MOS} \leq 5$: kualitas hasil Sangat Akurat / *Excellent*
$3{,}5 < \text{nilai MOS} \leq 4{,}5$: kualitas hasil Akurat / *Good*
$2{,}5 < \text{nilai MOS} \leq 3{,}5$: kualitas hasil Cukup Akurat / *Pair*
$1{,}5 < \text{nilai MOS} \leq 2{,}5$: kualitas hasil Kurang Akurat / *Poor*
$1 \leq \text{nilai MOS} \leq 1{,}5$: kualitas hasil Tidak Akurat / *Bad*

Tabel 1 Hasil MOS

Lagu	Jarak	Jumlah Score	Rata-rata	Kategori
Aura lee	2	84	2,80	Cukup Akurat
	3	98	3,27	Cukup Akurat
	4	83	2,77	Cukup Akurat
Balonku	2	85	2,83	Cukup Akurat
	3	100	3,33	Cukup Akurat
	4	84	2,80	Cukup Akurat
Clementine	2	90	3,00	Cukup Akurat
	3	105	3,50	Cukup Akurat
	4	83	2,77	Cukup Akurat
Twinkle	2	86	2,87	Cukup Akurat
	3	101	3,37	Cukup Akurat
	4	94	3,13	Cukup Akurat

Berdasarkan tabel nilai MOS didapatkan bahwa hasil penentuan akor yang dianggap paling baik adalah Clementine dengan jarak 3 yaitu dengan nilai 3,50 dengan kategori cukup akurat.

3.1.2 Berdasarkan Hasil Program

Pada skenario ini, pengujian dilakukan untuk menguji ketepatan hasil akor terhadap akurasi output dengan menggunakan panduan literatur akor.

Tabel 2 Hasil jumlah akor terdeteksi

Lagu	Jarak	Akor Benar	Jumlah Akor	Persentasi (%)
Balonku	2	17	28	60,71
	3	13	19	68,42
	4	11	14	78,57
Aura	2	17	24	70,83
	3	10	16	62,5
	4	9	12	75
Twinkle	2	9	21	42,85
	3	6	14	42,85
	4	6	11	54,54
Clementine	2	18	30	60

	3	11	20	55
	4	10	15	66,67

Dengan hasil di atas, maka dapat diketahui bahwa nilai akor terdeteksi rata-rata adalah 61,49%. Dengan perhitungan sebagai berikut

Rata-rata akor terdeteksi

$$= \frac{70,83+62,5+75+60,71+68,42+78,57+60+55+66,67+42,85+42,85+54,54}{12} \; x \; 100\% = 61,49\%$$

Pada semua lagu yang diujikan, presentasi akor benar yang paling baik adalah pada jarak 4 nada. Pada lagu balonku didapatkan 78,57%, pada lagu aura didapatkan 75%, pada lagu clementine didapatkan 66,67%, dan pada lagu twinkle didapatkan 54,54%.

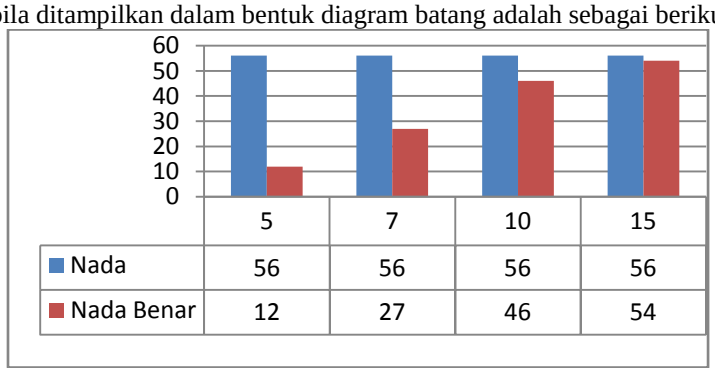
3.2 Pengaruh Jumlah Layer Pada JST Backpropagation

Pada skenario ini, pengujian dilakukan untuk menguji ketepatan metode klasifikasi dalam menentukan nada masukkan dengan menggunakan variasi jumlah layer pada JST Backpropagation. Variasi jumlah layer yang digunakan adalah 5, 7, 10, dan 15. Dalam skenario pengujian ini, nilai epoch yang digunakan adalah 1000. Hasil dari skenario pengujian ini adalah sebagai berikut.

Tabel 3 Hasil Pengujian Variasi Jumlah Layer pada lagu Balonku

Jumlah Layer	Jumlah Nada	Nada Terdeteksi (benar)	Persentasi (%)
5	56	12	21,43
7	56	27	48,21
10	56	46	82,14
15	56	54	96,43

Data pengujian diatas bila ditampilkan dalam bentuk diagram batang adalah sebagai berikut :



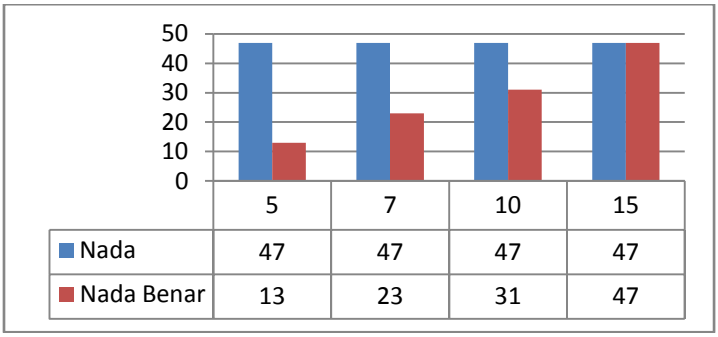
Gambar 2 Hasil Deteksi Nada dengan Variasi Jumlah Layer pada lagu Balonku

Pada hasil pengujian di atas dapat dilihat bahwa ketika jumlah layer 5, maka persentasi nada benar yang terdeteksi adalah 21,43%. Sedangkan ketika jumlah layer 7 dan 10, maka persentasi nada benar yang terdeteksi adalah 48,21% dan 82,14%. Persentasi jumlah nada benar yang terdeteksi adalah ketika jumlah layer 15, yaitu 96,43%

Tabel 4 Hasil Pengujian Variasi Jumlah Layer pada lagu Aura

Jumlah Layer	Jumlah Nada	Nada Terdeteksi (benar)	Persentasi (%)
5	47	13	27,66
7	47	23	48,94
10	47	31	65,96
15	47	47	100

Data pengujian diatas bila ditampilkan dalam bentuk diagram batang adalah sebagai berikut :



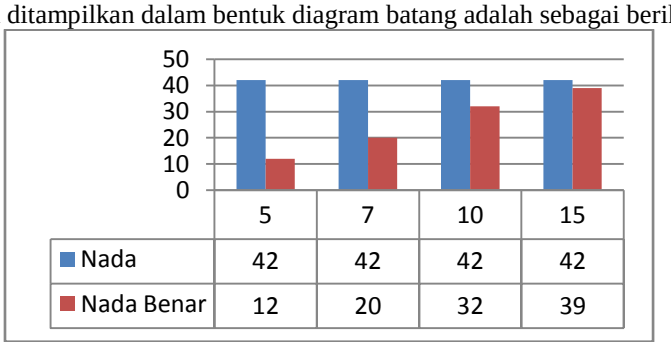
Gambar 3 Hasil Deteksi Nada dengan Variasi Jumlah Layer pada lagu Aura

Pada hasil pengujian di atas dapat dilihat bahwa ketika jumlah layer 5, maka persentasi nada benar yang terdeteksi adalah 27,66%. Sedangkan ketika jumlah layer 7 dan 10, maka persentasi nada benar yang terdeteksi adalah 48,94% dan 65,96%. Persentasi jumlah nada benar yang terdeteksi adalah ketika jumlah layer 15, yaitu 100%

Tabel 5 Hasil Pengujian Variasi Jumlah Layer pada lagu Twinkle

Jumlah Layer	Jumlah Nada	Nada Terdeteksi (benar)	Persentasi (%)
5	42	12	28,57
7	42	20	47,62
10	42	32	76,19
15	42	39	92,86

Data pengujian diatas bila ditampilkan dalam bentuk diagram batang adalah sebagai berikut :



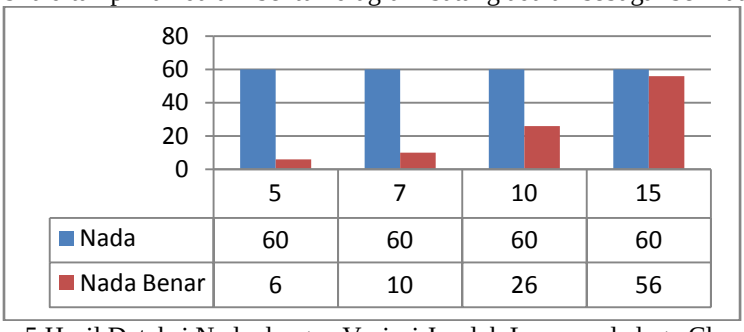
Gambar 4 Hasil Deteksi Nada dengan Variasi Jumlah Layer pada lagu Twinkle

Pada hasil pengujian di atas dapat dilihat bahwa ketika jumlah layer 5, maka persentasi nada benar yang terdeteksi adalah 28,57%. Sedangkan ketika jumlah layer 7 dan 10, maka persentasi nada benar yang terdeteksi adalah 47,67% dan 76,19%. Persentasi jumlah nada benar yang terdeteksi adalah ketika jumlah layer 15, yaitu 92,86%.

Tabel 6 Hasil Pengujian Variasi Jumlah Layer pada lagu Clementine

Jumlah Layer	Jumlah Nada	Nada Terdeteksi (benar)	Persentasi (%)
5	60	6	10
7	60	10	16,67
10	60	26	43,33
15	60	56	93,33

Data pengujian diatas bila ditampilkan dalam bentuk diagram batang adalah sebagai berikut :



Gambar 5 Hasil Deteksi Nada dengan Variasi Jumlah Layer pada lagu Clementine

Pada hasil pengujian di atas dapat dilihat bahwa ketika jumlah layer 5, maka persentasi nada benar yang terdeteksi adalah 10%. Sedangkan ketika jumlah layer 7 dan 10, maka persentasi nada benar yang terdeteksi adalah 16,67% dan 43,33%. Persentasi jumlah nada benar yang terdeteksi adalah ketika jumlah layer 15, yaitu 93,33%

Jumlah layer yang digunakan dalam pelatihan JST Backpropagation sangat mempengaruhi hasil klasifikasi nada dalam sistem. Untuk mendapatkan hasil yang optimal, harus dilakukan beberapa kali percobaan. Semakin besar jumlah layer yang digunakan maka semakin besar pula tingkat akurasi yang didapatkan. Tetapi ada batas optimum jumlah layer yang digunakan, yaitu ketika jumlah layer ditingkatkan, akurasi yang didapatkan tidak terlalu berbeda. Peningkatan jumlah layer berbanding lurus dengan lamanya waktu pelatihan JST Backpropagation tersebut. Pada Tugas Akhir ini, didapatkan nilai optimum ketika layer yang digunakan berjumlah 15.

4. Kesimpulan

Berdasarkan perancangan dan analisis yang dilakukan, dapat diperoleh kesimpulan sebagai berikut

1. Sistem penentu akor piano dibuat dengan merancang sistem yang terdiri dari akuisisi data, preprocessing, ekstraksi ciri, dan klasifikasi menggunakan JST Backpropagation.
2. Untuk menentukan jumlah layer yang digunakan pada JST Backpropagation perlu dilakukan percobaan beberapa kali sehingga mendapatkan jumlah layer yang tepat. Berdasarkan hasil pengujian, hasil paling optimal diperoleh ketika jumlah layer 15.
3. Berdasarkan hasil program, persentasi ketepatan penentuan akor untuk lagu balonku, aura, twinkle, dan clementine yang paling baik didapatkan pada jarak 4 nada yaitu 78,57%, 75%, 54,54%, dan 66,67%. Sedangkan berdasarkan hasil MOS, pada lagu aura, balonku, Clementine dan twinkle yang mendapatkan nilai paling tinggi adalah pada saat jarak 3 nada yaitu 3,27; 3,33; 3,50; 3,37.

Daftar Pustaka

[1] Brigham, E. Organ. 1988. *The Fast Fourier Transform And Its Application*. Singapore : Prentice Hall, Inc

[2] Chaerun, Rianda. 2011.*Perancangan Sistem Konversi Nada Tunggal Gitar ke Dalam Not Balok Menggunakan Fast Fourier Tansform (FFT).* Bandung:ITTelkom

[3] Fajar, Galih Ahmad. 2011. *Pengenalan dan Analisis Kualitas Penalaan Nada Tunggal Piano Secara Real Time Menggunakan Metode JST-SOM*. Bandung:ITTelkom

[4] H. B. Kekre, Vaishali Kulkarni, Prashant Gaikar Nishant Gupta, "*Speaker Identification using Spectrograms of Varying Frame Sizes*" International Journal of Computer Applications (0975 – 8887), Volume 50– No.20, July 2012

[5] Hermawan, Arif. 2006. Jaringan *Saraf Tiruan : Teori dan Aplikasi.* Yogyakarta:Penerbit Andi

[6] Lindsey, Ken. 2006. *Optimization Strategies for FFT Use in Musical Audio Analysi*s. Oregon USA : Ashland

[7] Lyana. 2012. *Aplikasi Sistem Penentu Akor Pada Audio dengan Fast Fourier Transform dan Jaringan Syaraf Tiruan ART 2*. Bandung:ITTelkom

[8] Nurunnadifah, Liliek. 2013. *Perancangan Aplikasi Mesin Pencari Judul Lagu MP3 dengan Input Suara Piano Menggunakan Metode JST-SOM.* Bandung:ITTelkom

[9] Setiawan, Kuswara. 2001. *Paradigma Sistem Cerdas*. Surabaya:Sekolah Tinggi Teknik Surabaya

[10] Simatupang, Dhita Maya Roselyn. 2013. *Perancangan dan Analisis Sistem Pendeteksi Nada Piano Secara Real Time Menggunakan IC Frequency to Voltage Converter Berbasis Mikrokontroler.* Bandung:ITTelkom

[11] Soeharto, M. 1978. *Belajar Notasi Balok.* Jakarta:Gramedia

[12] Sulistyowati, Edi Winarko, "*Peramalan KLB Campak Menggunakan Gabungan Metode JST Backpropagation dan CART*", IJCCS, Vol.8, No.1, pp. 49~58, January 2014

[13] Suyanto. 2007. *Artificial Intelligence : Searching, Reasoning, Planning, and Learning.* Bandung:Penerbit Informatika

[14] Yudha, Indrajit Prawira. 2012. *Sistem Identifikasi Jenis Suara Manusia Berdasarkan Jangkauan Vokal Menggunakan Jaringan Saraf Tiruan Backpropagation*. Bandung:ITTelkom