

**SISTEM PENGENALAN CHORD MUSIK OTOMATIS DENGAN
ALGORITMA HIDDEN MARKOV MODEL DAN
PITCH CONTOUR EXTRACTION**

SKRIPSI

DITO ATHALLAH MAJID

201401133



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

**SISTEM PENGENALAN CHORD MUSIK OTOMATIS DENGAN
ALGORITMA HIDDEN MARKOV MODEL DAN
PITCH CONTOUR EXTRACTION**

SKRIPSI

**Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Ilmu Komputer**

DITO ATHALLAH MAJID

201401133



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

PERSETUJUAN

Judul : SISTEM PENGENALAN CHORD MUSIK OTOMATIS DENGAN ALGORITMA HIDDEN MARKOV MODEL DAN PITCH CONTOUR EXTRACTION

Kategori : SKRIPSI

Nama : DITO ATHALLAH MAJID

Nomor Induk Mahasiswa : 201401133

Program Studi : SARJANA (S-1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

Telah diuji dan dinyatakan lulus di Medan, 09 Januari 2025

Komisi Pembimbing :

Pembimbing 2

Dewi Sartika Br.Ginting S.Kom., M.Kom
NIP. 199005042019032023

Pembimbing 1

Desilia Selvida S.Kom., M.Kom.
NIP. 198912052020012001

Diketahui/Disetujui Oleh



NIP. 197812212014042001

PERNYATAAN

**SISTEM PENGENALAN CHORD MUSIK OTOMATIS DENGAN
ALGORITMA HIDDEN MARKOV MODEL DAN
PITCH CONTOUR EXTRACTION**

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil penelitian saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah dicantumkan sumbernya.

Medan, 23 Desember 2024



Dito Athallah Majid

201401133

PENGHARGAAN

Bismillahirrahmanirrahim, segala puji syukur dipanjatkan kepada Allah *Subhanahu Wa Ta ‘ala* atas segala nikmat iman dan islam, kesehatan serta karunia-Nya sehingga penulis mampu untuk menyusun skripsi sebagai syarat untuk mendapatkan gelar Sarjana Komputer di Program Studi S-1 Ilmu Komputer, USU. Tidak lupa shalawat serta salam tetap tercurahkan kepada Rasulullah SAW yang telah mengeluarkan umat manusia dari kegelapan menuju zaman terang benderang saat ini.

Pada kesempatan ini penulis ingin mengucapkan terima kasih kepada Ibu tercinta, drg. Yusmaniar atas segala bentuk perjuangan, kasih sayang, dan doa-doa yang dipanjatkan untuk penulis. Dan terima kasih juga kepada Papa saya, Budi Susanto. Penyusunan skripsi ini tidak terlepas dari banyaknya bantuan serta bimbingan dari banyak pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Bapak Prof. Dr. Muryanto Amin S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Bapak Dr. Mohammad Andri Budiman S.T., M.Comp.Sc., M.E.M. selaku Wakil Dekan I Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Ibu Dr. Amalia, S.T., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
5. Ibu Desilia Selvida S.Kom., M.Kom. selaku Dosen Pembimbing I yang telah memberi banyak masukan, arahan, motivasi, serta dukungan kepada penulis selama penyusunan skripsi ini.
6. Ibu Dewi Sartika Br.Ginting, S.Kom., M.Kom. selaku Dosen Pembimbing II yang telah memberi banyak bimbingan dan masukan yang berharga kepada penulis selama proses penyusunan skripsi ini.
7. Bapak Amer Sharif S.Si., M.Kom. selaku Dosen Pengaji I yang telah memberikan arahan dan masukan kepada penulis.

8. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku Dosen Pengaji II yang telah memberi saran serta masukan kepada penulis.
9. Seluruh Bapak dan Ibu dosen Program Studi S-1 Ilmu Komputer, yang telah meluangkan banyak waktu, tenaga, dan pikiran untuk mengajar dan memberi wawasan serta moral yang berharga, baik di bangku perkuliahan maupun setelah lulus.
10. Teristimewa kepada Orang Tua penulis Budi Susanto dan drg, Yusmaniar yang telah memberikan penulis semangat, kekuatan, kesabaran, ilmu yang bermanfaat, dan doa yang tiada hentinya untuk penulis sehingga penulis dapat menjalani perkuliahan sampai dengan penyusunan skripsi.
11. Sahabat seperjuangan “mujahiruddin” yang telah memberikan kenangan yang berharga dan berkesan kepada penulis selama perkuliahan sampai dengan penyusunan skripsi.
12. Teman-teman stambuk 2020 yang telah menjadi teman yang baik selama perkuliahan sampai di masa akhir perkuliahan.

Dan seluruh pihak yang telah memberi dukungan serta doa baik yang tidak dapat penulis sebutkan satu per-satu. Semoga Allah *Subhanahu Wa Ta’ala* senantiasa melimpahkan keberkahan serta kebaikan dan hasil penelitian ini dapat memberi manfaat maupun inspirasi untuk kedepannya.

Medan, 23 Desember 2024

Penulis,



Dito Athallah Majid

ABSTRAK

Musik merupakan salah satu bentuk seni yang diciptakan melalui rangkaian suara yang disatukan sehingga menciptakan irungan nada yang dapat di mengerti oleh manusia. Musik adalah seni yang melibatkan pengaturan suara yang berhubungan dengan nada, ritme, dan timbre. Pada masa sekarang, musik sudah mengalami perkembangan yang cukup pesat baik dari segi genre dan struktur musik itu sendiri. perkembangan teknologi khususnya dibidang komputer sudah semakin pesat. Banyak teknologi baru yang bermunculan pada saat ini. Kecerdasan buatan (*Artificial Intelligence*) adalah teknologi modern yang berfokus pada penciptaan sistem yang bereaksi seperti manusia. Penelitian ini memanfaatkan model dari *Hidden Markov Model* (HMM) dan Ekstraksi Fitur audio *Chromagram* yang di dapatkan menggunakan *Short Time Fourier Transfer* (STFT) untuk deteksi *Chord*. *Chord* yang dapat diklasifikasi terdiri 24 *Chord* dengan pembagian 12 *Major Chord* dan 12 *Minor Chord*. Dataset yang digunakan adalah hasil audio yang direkam oleh penulis yang berisikan 474 kumpulan rekaman *instrument* musik dengan berbagai macam gaya rekaman yang terdiri dari *instrument* piano, bass dan gitar. Penelitian ini memperoleh akurasi sebesar 79%, *precision* 85%, *recall* 78%, dan F1-score 80% dari seluruh kelas *Chord*.

Kata Kunci: *Chord*, Musik, Deteksi *Chord*, *Hidden Markov Model* (HMM), *Short Time Fourier Transfer* (STFT).

ABSTRACT

AUTOMATIC MUSIC CHORD RECOGNITION SYSTEM WITH HIDDEN MARKOV MODEL AND PITCH CONTOUR EXTRACTION

Music is a form of art that is created through a series of sounds that are put together so as to create a tonal accompaniment that can be understood by humans. Music is an art that involves the arrangement of sounds related to tone, rhythm, and timbre. At present, music has experienced a fairly rapid development both in terms of genre and structure of the music itself. technological developments, especially in the field of computers, are increasingly rapid. Many new technologies are emerging at this time. Artificial intelligence is a modern technology that focuses on creating systems that react like humans. This research utilizes the model of Hidden Markov Model (HMM) and Feature Extraction of Chromagram audio obtained using Short Time Fourier Transfer (STFT) for Chord detection. Chord that can be classified consists of 24 chords with a division of 12 Major Chord and 12 Minor Chord. The dataset used is the result of audio recorded by the author which contains 474 collections of musical instrument recordings with various recording styles using a piano, bass and guitar. This research obtained an accuracy of 79%, precision 85%, recall 78%, and F1-score 80% of all Chord classes.

Keywords: Chord, Music, Chord Detection, Hidden Markov Model (HMM), Short Time Fourier Transfer (STFT).

DAFTAR ISI

PERSETUJUAN	ii
PERNYATAAN	iii
PENGHARGAAN	iv
ABSTRAK	vi
ABSTRACT	vii
DAFTAR ISI	viii
DAFTAR TABEL	xi
DAFTAR GAMBAR	xii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	4
1.3 Batasan Masalah.....	4
1.4 Tujuan Penelitian.....	4
1.5 Manfaat Penelitian.....	5
1.6 Metodologi Penelitian	5
1.7 Penelitian Relevan.....	6
1.8 Sistematika Penulisan.....	7
BAB 2 LANDASAN TEORI	9
2.1 Automatic Chord Recognition.....	9
2.2 Machine Learning	9
2.3 Hidden Markov Model	10
2.4 Pitch Contour Extraction (PCE).....	12
2.5 Digital Audio Processing	13
2.5.1 <i>Audio Format</i>	13
2.5.2 <i>Sample Rate</i>	14
2.5.3 <i>Bit Rate</i>	15
2.6 Musical Background.....	16
2.7 Machine Learning	17
2.7.1 <i>Supervised Learning</i>	18

2.7.2	<i>Unsupervised Learning</i>	18
2.8	Framework	19
BAB 3 ANALISIS DAN PERANCANGAN		20
3.1	Analisis	20
3.1.1	<i>Analisis Masalah</i>	20
3.1.2	<i>Analisis Kebutuhan</i>	20
3.2	Gambaran Umum Sistem	21
3.3	Analisis Dataset	23
3.4	<i>Pre-Processing</i>	25
3.4.1	Normalisasi Audio	25
3.4.2	Ekstraksi Fitur Chromagram	26
3.4.3	<i>Smoothing</i>	27
3.4.4	Segmentasi	28
3.4.5	Optimasi Parameter	28
3.5	Representasi Fitur Audio dengan Chromagram	29
3.6	Klasifikasi dengan Model HMM dan Template Matching	31
3.6.1	Analisis Chromagram dan Template Matching	31
3.7	Evaluasi	32
3.8	Menyimpan Model	32
3.9	Perhitungan Ekstraksi Fitur Audio Menggunakan Chromagram	32
3.10	Perhitungan Deteksi Chord Menggunakan HMM dan Template Matching	35
3.11	Perancangan Sistem	39
3.11.1	Diagram Umum Sistem	40
3.11.2	<i>Use Case Diagram</i>	40
3.11.3	<i>Activity Diagram</i>	41
3.11.4	<i>Sequence Diagram</i>	42
3.11.5	<i>Flowchart</i> Sistem	43
BAB 4 IMPLEMENTASI DAN PENGUJIAN		45
4.1	Implementasi Sistem	45
4.1.1	<i>Halaman Utama Website</i>	45
4.1.2	Hasil Klasifikasi Sistem	45
4.2	<i>Pre-Processing</i> Dataset	46
4.3	Implementasi Ekstraksi Fitur Menggunakan <i>Chromagram</i>	52
4.4	Implementasi Klasifikasi Chord Menggunakan Template Matching dan	

HMM	55
4.5 Evaluasi	60
4.6 Menyimpan dan Memanggil Model	67
4.7 Hasil Pengujian Sistem.....	68
BAB 5 PENUTUP.....	70
5.1 Kesimpulan.....	70
5.2 Saran.....	71
DAFTAR PUSTAKA	72

DAFTAR TABEL

Tabel 3. 1 Contoh Labelisasi Dataset Chord	25
Tabel 3. 2 Contoh Normalisasi Audio	26
Tabel 3. 3 Contoh Ekstraksi Chromagram	26
Tabel 3. 4 Contoh Smoothing.....	27
Tabel 3. 5 Contoh proses Segmentasi.....	28
Tabel 4. 1 Tabel Nilai Parameter Uji Coba Pertama dan Kedua.....	55
Tabel 4. 2 Tabel Hasil Metrik Evaluasi Pada Uji Coba Pertama dan Kedua	55
Tabel 4. 3 TP, TN, FP, dan FP Semua Kelas Chord.....	65
Tabel 4. 4 Hasil Akurasi, Precision, Recall, dan F1-Score Semua Kelas Chord	66
Tabel 4. 5 Tabel Pengujian Deteksi Chord pada File Audio.....	68

DAFTAR GAMBAR

Gambar 2.1 Arsitektur Umum HMM	11
Gambar 2. 2 Perbandingan format .mp3 dengan .WAV	14
Gambar 2. 3 Perbandingan Sample Rate.....	15
Gambar 2. 4 Perbandingan Bit Rate.....	16
Gambar 2. 5 Circle Of Fifth Chords	17
Gambar 3. 1 Arsitektur Umum Sistem.....	22
Gambar 3. 2 Tahapan Pembuatan Model.....	23
Gambar 3. 3 Struktur Folder Dataset	23
Gambar 3. 4 Dataset dalam bentuk Waveform	24
Gambar 3. 5 Dataset Dengan Label Chord	24
Gambar 3. 6 Proses Normalisasi Audio	25
Gambar 3. 7 Proses Ekstraksi Chromagram	26
Gambar 3. 8 Proses Smoothing.....	27
Gambar 3. 9 Proses Segmentasi.....	28
Gambar 3. 10 Proses Optimasi Parameter.....	29
Gambar 3. 11 Proses Ekstraksi Chromagram	30
Gambar 3. 12 Contoh Visualisasi Chromagram	30
Gambar 3. 14 Diagram Umum Sistem.....	40
Gambar 3. 15 Use Case Diagram.....	41
Gambar 3. 16 Activity Diagram.....	41
Gambar 3. 17 Sequence Diagram	42
Gambar 3. 18 Flowchart Sistem.....	43
Gambar 4. 1 Tampilan Halaman Utama	45
Gambar 4. 2 Hasil Deteksi Chord	46
Gambar 4. 3 Normalisasi Audio	46
Gambar 4. 4 Hasil Normalisasi Audio	47
Gambar 4. 5 Ekstraksi Chromagram.....	47
Gambar 4. 6 Hasil Ekstraksi Chromagram	48
Gambar 4. 7 Smoothing	48
Gambar 4. 8 Hasil Smoothing.....	49
Gambar 4. 9 Segmentation.....	49
Gambar 4. 10 Hasil Segmentation	50
Gambar 4. 11 Proses Optimasi Parameter.....	51
Gambar 4. 12 Hasil Optimasi Parameter.....	52
Gambar 4. 13 Transformasi Data Audio	53
Gambar 4. 14 Pehitungan Parameter Adaptif	53
Gambar 4. 15 Ekstraksi Fitur Chromagram	53
Gambar 4. 16 Penggabungan dan Optimasi Fitur	53
Gambar 4. 17 Inialisasi Model Template Matching dan HMM	57

Gambar 4. 18 Proses Menghitung Metrik Evaluasi	60
Gambar 4. 19 Proses Pelatihan Model.....	61
Gambar 4. 20 Proses Evaluasi Model.....	62
Gambar 4. 21 Proses Membuat Grafik Metrik Evaluasi	63
Gambar 4. 22 Hasil Grafik Akurasi, F1-Score, Precision, dan Recall	63
Gambar 4. 23 Hasil Grafik Total Training Loss	64
Gambar 4. 24 Hasil Confusion Matrix.....	65
Gambar 4. 25 Proses Menyimpan dan Memanggil Model.....	68

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Musik merupakan salah satu bentuk seni yang diciptakan melalui rangkaian suara yang disatukan sehingga menciptakan irungan nada yang dapat di mengerti oleh manusia. Musik adalah seni yang melibatkan pengaturan suara yang berhubungan dengan nada, ritme, dan timbre. Pada masa sekarang, musik sudah mengalami perkembangan yang cukup pesat baik dari segi genre dan struktur musik itu sendiri. Musik dapat membantu orang yang mendengarkannya merasa nyaman dan merupakan salah satu cara paling efektif untuk menyampaikan emosi dengan cepat. Musik juga dapat membantu orang merasakan banyak emosi seperti kegembiraan, kegembiraan, kesedihan, dan masih banyak lagi (Rajesh & Nalini, 2020).

Musik secara umum dapat dimainkan dengan banyak cara salah satu caranya dengan menggunakan instrument musik untuk menghasilkan bunyi serta nada yang di inginkan. Instrument atau alat musik adalah alat yang dibuat atau dimodifikasi untuk tujuan menghasilkan musik. Pada prinsipnya segala sesuatu yang menghasilkan bunyi dan dapat diaransemen oleh seorang pemusik dapat dianggap sebagai instrument. Berdasarkan cara memainkannya, alat musik terbagi menjadi 5 jenis, yaitu alat musik tiup, alat musik perkusi, alat musik petik, alat musik gesek, dan alat musik tekan. Alat musik tiup dimainkan dengan cara ditiup sehingga menghasilkan bunyi melalui rongga bunyi yang terdapat pada alat. Contoh alat musik ini adalah suling, terompet. Alat musik perkusi dimainkan dengan cara di pukul sehingga menghasilkan bunyi. Contoh alat musik ini adalah drum, gendang, dan cajon. Berikutnya untuk alat musik petik adalah alat musik yang dimainkan dengan cara di petik. Contoh alat musiknya adalah gitar, bass dan harpa. Alat musik gesek adalah alat musik yang dimainkan dengan cara di gesek. Contoh alat musik yang di mainkan dengan cara di gesek adalah biola, cello. Yang terakhir adalah alat musik tekan yang dimainkan dengan cara di tekan. Contoh alat musik yang dimainkan dengan cara di tekan adalah piano (Lutfiah et al., n.d.).

Nada adalah rangkaian bunyi yang beraturan dan memiliki frekuensi tertentu. Nada memiliki susunan atau urutan yang dimulai dari nada dasar hingga nada oktaf

hal ini dinamakan dengan tangga nada. Terdapat tiga macam tangga nada yaitu tangga nada Pentatonic, Diatonic dan Chromatic. Nada dasar Diatonic juga terbagi menjadi dua jenis yaitu tangga nada mayor (Major Scale) dan tangga nada minor (Minor Scale). Tangga nada mayor (Major Scale) dimulai dari nada dasar Do dengan interval $1 - 1 - \frac{1}{2} - 1 - 1 - 1 - \frac{1}{2}$ dari nada Do awal sampai nada Do oktaf. Berbeda dengan tangga nada minor (Minor Scale) tangga nada ini dimulai dari nada dasar La dengan interval $1 - \frac{1}{2} - 1 - 1 - 1 - \frac{1}{2} - 1$ dari nada La awal sampai nada La oktaf (Ferdiawan et al., 2022).

Chord adalah nada yang dibunyikan secara tersendiri atau kumpulan dari beberapa nada yang dimainkan secara bersamaan sehingga menjadi satu kesatuan suara yang harmonis. Berdasarkan penyusunannya *Chord* terbagi menjadi beberapa jenis yaitu diantaranya *Major*, *Minor*, *Augmented* dan *Diminished*. *Chord* biasanya di gambarkan dengan huruf yaitu C, D, E, F, G, A, B dan terdapat *Chord* kres dan mol seperti C# atau Db, D# atau Eb, F# atau Gb, dan A# atau Bb. *Chord* juga memiliki progresi atau perpindahan yang berfungsi sebagai penjaga dinamika dari sebuah lagu yang dimainkan agar tidak berantakan (Ferdiawan et al., 2022).

Pada zaman sekarang, perkembangan teknologi khususnya dibidang komputer sudah semakin pesat. Banyak teknologi baru yang bermunculan pada saat ini. Kecerdasan buatan (*Artificial Intelligence*) adalah teknologi modern yang berfokus pada penciptaan sistem yang bereaksi seperti manusia. Kecerdasan buatan adalah bidang ilmu komputer yang sedang berkembang. Saat ini banyak penelitian yang dilakukan di bidang AI, termasuk pembuatan musik, sistem rekomendasi, dan klasifikasi. Di bidang kecerdasan buatan, berbagai algoritma dan teknologi untuk jaringan saraf terus berkembang dan dikembangkan (Solanki & Pandey, 2022).

Pembelajaran mesin (*Machine Learning*) adalah teknologi baru yang berkembang pesat yang memungkinkan komputer membaca dan menafsirkan data yang ada secara otomatis. *Machine Learning* menggunakan beberapa algoritma untuk membangun model yang bersifat matematis dan menggunakan data serta wawasan sebelumnya untuk memprediksi data baru. Baru-baru ini, telah digunakan untuk pengenalan teks, deteksi perkataan yang mendorong kebencian, sistem rekomendasi, pengenalan wajah, dan lainnya. Tanpa pemrograman yang

komprehensif, komputer dapat memprediksi dan memutuskan sendiri, karena mereka menggunakan model matematika yang dibangun olehnya algoritma pembelajaran mesin dengan bantuan data pelatihan yang merupakan kumpulan data sampel yang ada. *Machine Learning* terbagi menjadi 3 kelas yaitu *Supervised*, *Unsupervised*, *Reinforcement Learning* (Bansal et al., 2022).

Hidden Markov Model (HMM) adalah model statistik yang kuat untuk menggambarkan sistem kompleks yang keadaan spesifiknya tidak dapat diamati secara langsung. Ini adalah proses stokastik ganda yang terdiri dari proses Markov yang tidak dapat diobservasi dan serangkaian keluaran yang bergantung padanya. Sederhananya, HMM dirancang untuk menangkap hubungan probabilistik antara keadaan tersembunyi dan hasil yang dapat diamati. *Hidden Markov Model* juga menjadi alat serbaguna yang memungkinkan peneliti memodelkan dan menganalisis sistem di mana terdapat ketidakpastian atau informasi tersembunyi, dan dapat memberikan informasi penting ke dalam hubungan antara data yang dapat diamati dengan *Hidden State* (Tseng et al., 2020).

Istilah "Hidden" dalam *Hidden Markov Model* mengacu pada fakta bahwa keadaan dasar sistem tidak terlihat atau diketahui secara langsung. Kondisi tersembunyi ini mempengaruhi hasil yang dapat diobservasi dan observasi yang dapat diukur atau dicatat. Model ini mengasumsikan bahwa sistem beralih antara keadaan tersembunyi yang berbeda berdasarkan serangkaian probabilitas, dan setiap keadaan menghasilkan keluaran yang dapat diamati berdasarkan serangkaian probabilitas yang berbeda (Tseng et al., 2020).

Terdapat penelitian terdahulu terkait pendekripsi *Chord* musik dalam data audio seperti yang dilakukan oleh Taxiang Li yang berjudul “Study on a CNN-HMM Approach for Audio-Based Musical Chord Recognition”. Penelitian ini menggunakan penggabungan antara algoritma *Convolutional Neural Network* (CNN) dengan *Hidden Markov Model* (HMM). Penelitian ini menggunakan fitur chroma yang berbeda-beda yaitu menggunakan *Short-Time Fourier Transform* (STFT), *Constant-Q Transform* (CQT), dan *Chroma Energy Normalized Statistic* (CENS) yang membuat kinerja dari program yang dibangun meningkat. Namun pada penelitian ini, program yang dibuat tidak dapat mendekripsi musik dengan genre tertentu seperti Pop, Jazz, Blues, dan R&B's dikarenakan datasetnya yang

kecil serta dikarenakan kompleksitas sehingga penelitian ini hanya dapat mendeteksi lagu yang memiliki 24 *Chord* dasar (Li, 2021).

1.2 Rumusan Masalah

Di masa sekarang, Musik menjadi hal yang cukup diminati. Namun, dalam proses pembelajaran musik khususnya di bidang aransemennya masih banyak orang yang mengalami kesulitan untuk mempelajari lagu - lagu tertentu karena terbatasnya sumber informasi terkait lagu yang ada sehingga diperlukan program yang nantinya berfungsi untuk mendeteksi *Chord* dari lagu yang ingin dipelajari.

1.3 Batasan Masalah

Batasan-batasan masalah yang terdapat pada penelitian ini yaitu.

1. Sistem hanya bisa mendeteksi *Chord* musik dalam 2 bentuk akord yaitu Akord Mayor dan Akord Minor.
2. Dataset yang digunakan dalam penelitian ini adalah dataset berbentuk audio yang di rekam sendiri oleh penulis sebanyak 474 sampel yang terdiri dari rekaman suara alat musik piano, bass dan gitar.
3. Dataset yang digunakan adalah audio hasil rekaman Instrument Musik dengan format WAV dengan *sample rate* sebesar 44100Hz (44.1 Khz) dan *Bit Rate* sebesar 24 Bit.
4. Program yang dirancang menggunakan bahasa Python.
5. Program dirancang berbasis web dan dibangun menggunakan *framework* streamlit.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk menciptakan sistem *Automatic Chord Recognition* (Sistem Deteksi Chord Otomatis) yang akurat serta efektif dan berguna sebagai sarana pembelajaran dalam hal aransemennya dan pembelajaran akademis terkait musik. Fokus penelitian mencakup bagaimana kinerja dari Algoritma *Hidden Markov Model* (HMM) dalam melakukan rekognisi terhadap *Chord* yang ada pada data berbentuk *Audio* dengan format WAV dan *sample rate* sebesar 44100hz serta *bit rate* sebesar 24 Bit. Penelitian ini juga bertujuan sebagai sarana identifikasi potensi aplikasi praktis dari sistem pengenalan chord musik otomatis, seperti dalam industri musik, pemrosesan audio, atau pengembangan aplikasi musik digital.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini diharapkan mampu.

1. Sistem pengenalan *Chord* otomatis menggunakan algoritma *Hidden Markov Model* akan meningkatkan efisiensi dalam mendeteksi *Chord* dalam data audio. Dengan adanya sistem ini, sistem yang dirancang akan banyak membantu penggunaanya karena proses pembelajaran dan aransemen musik dapat menjadi lebih mudah dan cepat.
2. Dengan menggunakan Algoritma *Hidden Markov Model* (HMM) proses deteksi *Chord* akan menjadi efisien karena algoritma ini adalah algoritma yang cukup akurat serta dapat mendeteksi dengan *Chord* dalam data audio dengan baik.
3. Hasil penelitian ini berpotensi untuk menjadi sarana pendorong inovasi teknologi di bidang industri musik terkhususnya dalam bidang rekognisi *Chord* otomatis dalam data audio.

1.6 Metodologi Penelitian

Beberapa metode yang diterapkan dalam penelitian ini adalah sebagai berikut:

1. Studi Literatur

Pada tahap ini, penelitian dimulai dengan mencari referensi dari berbagai sumber terpercaya dan melakukan peninjauan pustaka melalui buku-buku, jurnal, *e-book*, artikel ilmiah, makalah ataupun situs internet yang berhubungan dengan penggunaan algoritma *Hidden Markov Model* (HMM), *Automatic Chord Recognition, Machine Learning*.

2. Analisis dan Perancangan Sistem

Berdasarkan ruang lingkup penelitian, penulis melakukan analisa terhadap apa saja yang akan dibutuhkan dalam penelitian untuk segera dirancang dalam sebuah diagram alir (*flowchart*).

3. Pencarian dan Pengumpulan Dataset

Pada tahap awal, dataset untuk sistem deteksi *chord* otomatis ini merupakan dataset yang direkam sendiri oleh penulis dengan 3 buah instrumen yaitu gitar, bass dan piano. Masing masing instrumen dimainkan dengan cara permainan yang berbeda meliputi perbedaan progresi, perbedaan dinamika permainan dan perbedaan tempo permainan.

4. *Pre-processing Data*

Tahap ini dilakukan untuk membersihkan dan mempersiapkan data audio agar dapat diproses oleh model HMM. Tahapan ini meliputi *resampling* untuk menyamakan seluruh *sample rate* audio yang digunakan, *normalize audio* untuk menyamakan amplitudo, *smoothing*, *trim silence*.

5. Membangun Model HMM

Model akan dibuat dengan bahasa python serta beberapa *libraries* seperti *librosa*, *numpy*, dan *hmmlearn*. Model ini akan dilatih menggunakan data audio yang telah diproses dan diekstraksi fiturnya untuk mendeteksi progresi *chord* secara akurat.

6. Implementasi Sistem dan Model

Implementasi pendedeksian *Chord* pada sumber data audio ini akan dilakukan menggunakan Bahasa pemrograman python dan menggunakan *Visual Studio Code*.

7. Pengujian dan Evaluasi

Pada tahap ini, sistem yang telah dirancang dilakukan uji coba untuk melakukan pendekripsi *Chord* melalui sumber data audio yang di inputkan ke dalam program menggunakan algoritma yang sudah ditentukan yaitu Hidden Markov Model (HMM).

8. Dokumentasi

Pada tahap ini, penelitian yang telah dilakukan, di dokumentasikan mulai dari tahap analisa sampai kepada pengujian dalam bentuk skripsi untuk menunjukkan hasil dari penelitian yang dilakukan.

1.7 Penelitian Relevan

Berikut adalah penelitian terdahulu yang relevan dengan penelitian yang dilakukan, yaitu:

1. Penelitian ini dilakukan oleh Taxiang Li yang berjudul “Study on a CNN-HMM Approach for Audio-Based Musical Chord Recognition” pada tahun 2021. Penelitian ini menggunakan penggabungan antara algoritma *Convolutional Neural Network* (CNN) dengan *Hidden Markov Model* (HMM). Penelitian ini menggunakan fitur chroma yang berbeda-beda yaitu menggunakan *Short-Time Fourier Transform* (STFT), *Constant-Q Transform* (CQT), dan *Chroma Energy Normalized Statistic* (CENS) yang membuat kinerja dari program yang

dibangun meningkat. Namun pada penelitian ini, program yang dibuat tidak dapat mendeteksi musik dengan genre tertentu seperti Pop, Jazz, Blues, dan R&B's dikarenakan datasetnya yang kecil serta dikarenakan kompleksitas sehingga penelitian ini hanya dapat mendeteksi lagu yang memiliki 24 *Chord* dasar.

2. Penelitian relevan berikutnya dilakukan oleh Ivanna K. Timotius dan Adhi Prayogo dengan penelitian yang berjudul “SISTEM PENGENALAN CHORD PADA FILE MUSIK DIGITAL DENGAN MENGGUNAKAN PITCH CLASS PROFILES DAN HIDDEN MARKOV MODEL” pada tahun 2010. Penelitian ini menggunakan gabungan antara algoritma *Hidden Markov Model* dengan ekstraksi fitur audio *Pitch Class Profile*. Sistem yang di desain pada penelitian kali ini hanya dapat mengenali 3 jenis *chord families* yaitu *Major chord*, *Minor Chord*, dan *Dominant seventh* yang pada 12 *roots* yaitu A, Bb, C, C#, D, Eb, E, F, F#, G, G#. Data yang di inputkan kedalam program adalah data lagu yang terdiri dari 14 musik dengan progresi chord yang di rekam dari MIDI, 14 file musik yang direkam menggunakan instrument gitar, 5 lagu yang didalamnya terdapat berbagai macam instrument dan direkam dari MIDI, dan 3 file lagu kompleks komersial yang didalamnya terdapat suara vokal manusia. Hasil yang di peroleh adalah tingkat akurasi yang diperoleh pada percobaan dengan jenis data pertama sebesar 100%, untuk jenis data kedua sebesar 97,97%, jenis data ketiga sebesar 85,35% dan yang terakhir sebesar 59,80%.

1.8 Sistematika Penulisan

Sistematika penulisan skripsi yang digunakan dalam penelitian ini adalah sebagai berikut:

BAB 1 PENDAHULUAN

Pada bab ini mencakup penjelasan tentang latar belakang pemilihan judul, rumusan dan batasan masalah, tujuan, manfaat, dan metodologi penelitian, penelitian relevan, dan sistematika penulisan skripsi.

BAB 2 LANDASAN TEORI

Bab ini menjelaskan teoritis dalam penelitian dan pengembangan sistem, seperti pendahuluan penjelasan tentang *Automatic chord recognition*, *Digital audio processing*, *Musical background*, PCE dan HMM.

BAB 3 ANALISIS DAN PERANCANGAN

Bab ini menjelaskan mengenai analisis dan perancangan pada sistem dan penelitian.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas tentang implementasi dan pengujian sistem yang sesuai dengan rancangan yang telah disiapkan sebelumnya.

BAB 5 KESIMPULAN DAN SARAN

Bab ini mencakup rangkuman dari setiap bab dan menarik kesimpulan yang telah dibahas, dan memberikan saran-saran yang dapat menjadi masukan untuk peneliti di masa mendatang.

BAB 2

LANDASAN TEORI

2.1 Automatic Chord Recognition

Adalah Pengenalan akord terprogram secara otomatis yang mengacu pada metode mengenali dan menerjemahkan tampilan akord secara konsekuensi dalam sepotong musik dari rekaman suara. Proses ini termasuk menganalisis substansi konsonansi bunyi untuk menentukan bunyi dasar akord yang dimainkan. Analis di bidang Pemulihan Data Musik telah membuat perhitungan dan model yang berbeda, menghitungnya berdasarkan pembelajaran mesin, untuk secara tepat mengenali dan menafsirkan akord dari rekaman suara (Lanz, n.d.).

Deteksi akord otomatis juga menjadi salah satu masalah penting dalam analisis musik dengan banyak untuk penerapannya seperti pengambilan informasi musik, identifikasi musik, dan transkripsi musik otomatis. Tugasnya adalah memperkirakan akord dari nada-nada yang diamati dalam bentuk simbolis atau akustik. Progresi akord atau harmoni yang merupakan salah satu elemen terpenting, memainkan peran dominan dalam menentukan struktur dan suasana musik. Saat kita mendengarkan suatu musik, bahkan tanpa mengetahui not-not individu dalam musik tersebut, kita dapat mendengar harmoninya. Dengan demikian, progresi akord musik dapat membantu tugas transkripsi musik otomatis (IEEE Signal Processing Society., 2010).

2.2 Machine Learning

Pembelajaran Mesin (*Machine Learning*) adalah bagian dari kecerdasan buatan yang mencakup algoritma dan model komputasi yang memungkinkan komputer mempelajari pola dari data dan membuat prediksi seta keputusan tanpa pemrograman eksplisit (Yadav Kasula, n.d.).

Pada *Machine Learning* terdapat 2 macam jenis pembelajaran yaitu *Supervised learning* dan *Unsupervised Learning*. *Supervised Learning* berfungsi dengan cara memetakan input dan output berdasarkan berdasarkan contoh pasangan input dan outputnya. Supervised learning menyimpulkan dari data pelatihan berlabel yang terdiri dari sekumpulan contoh pelatihan. Dataset input yang digunakan dalam Supervised Learning terbagi menjadi Dataset Latih (*Train Dataset*) dan Dataset tes (*Test Dataset*). Dataset latih memiliki variabel output yang nantinya perlu di

prediksi atau di klasifikasikan. Berbeda dengan *Supervised Learning*, *Unsupervised Learning* bekerja secara sendiri untuk menyajikan dan menemukan struktur datanya. Apabila mengenali data baru, maka ia akan mempelajari data ini dengan menggunakan fitur fitur pada data sebelumnya yang sudah di pelajari (Mahesh, 2018).

2.3 Hidden Markov Model

Hidden Markov Model (HMM) adalah model statistik yang kuat untuk menggambarkan sistem kompleks yang keadaan spesifiknya tidak dapat diamati secara langsung. Ini adalah proses stokastik ganda yang terdiri dari proses Markov yang tidak dapat diobservasi dan serangkaian keluaran yang bergantung padanya. Sederhananya, HMM dirancang untuk menangkap hubungan probabilistik antara keadaan tersembunyi dan hasil yang dapat diamati. *Hidden Markov Model* juga menjadi alat serbaguna yang memungkinkan peneliti memodelkan dan menganalisis sistem di mana terdapat ketidakpastian atau informasi tersembunyi, dan dapat memberikan informasi penting ke dalam hubungan antara data yang dapat diamati dengan *Hidden State* (Tseng et al., 2020).

Istilah "*Hidden*" dalam *Hidden Markov Model* mengacu pada fakta bahwa keadaan dasar sistem tidak terlihat atau diketahui secara langsung. Kondisi tersembunyi ini mempengaruhi hasil yang dapat diobservasi dan observasi yang dapat diukur atau dicatat. Model ini mengasumsikan bahwa sistem beralih antara keadaan tersembunyi yang berbeda berdasarkan serangkaian probabilitas, dan setiap keadaan menghasilkan keluaran yang dapat diamati berdasarkan serangkaian probabilitas yang berbeda (Tseng et al., 2020).

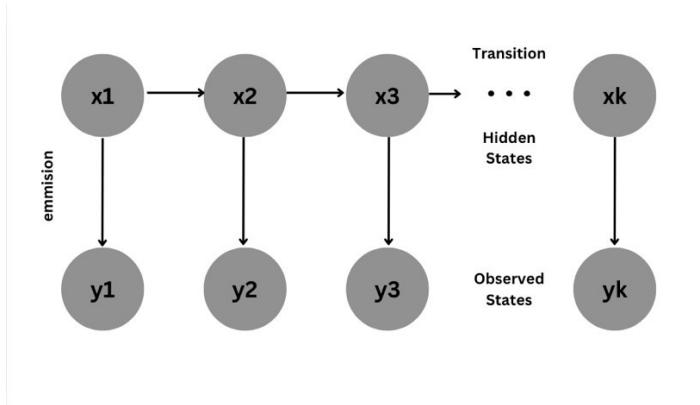
States Dari Algoritma HMM tidak terlihat oleh user, Melainkan outputnya yang terlihat. HMM Adalah tuple $\theta = (X, O, \pi, A, B)$ dimana $X = \{x_1, x_2, \dots, x_n\}$ adalah sebuah elemen yang disebut state. $O = \{o_1, o_2, \dots, o_m\}$ Adalah data yang di observasikan. $\pi = \pi_1, \pi_2, \dots, \pi_n$ Dimana $\pi_i = P(x_i)$, adalah vektor probabilitas awal yang mengacu pada distribusi keadaan awal, yang mana $0 \leq \pi_i \leq 1$, dan $\sum i \pi_i = 1$.

A adalah matriks probabilitas transisi. Dengan ukuran $n \times n$. Dan $a_{ij} = P(x_i | x_j)$ merepresentasikan kemungkinan transisi dari hidden state i ke hidden state j. notasi alternatif untuk a_{ij} adalah $a[x_i, x_j]$. B adalah Barisan probabilitas observasi.

Setiap $b_i(o_i) = P(o_i | x_i)$ mewakili probabilitas observasi x_i yang dihasilkan dari state x_i . Notasi alternatif untuk $b_i(o_i)$ adalah $b[x_i, o_i]$.

Terkadang, kita merujuk pada HMM dengan notasi yang lebih pendek $\theta = (\pi, A, B)$, karena X dan O bukan parameter yang dihitung, atau dengan notasi yang lebih pendek lagi $\theta = (A, B)$, dimana π telah dilipat menjadi A . Secara khusus, HMM menggambarkan distribusi probabilitas pada serangkaian observasi dengan panjang m . Misalnya, itu adalah rangkaian kata atau kalimat untuk tugas Pemrosesan Bahasa Alami. Diasumsikan bahwa setiap observasi dipancarkan oleh keadaan tersembunyi yang cocok x_i , yang nilainya secara langsung mengontrol matriks probabilitas (Perikos et al., 2021).

Berikut adalah arsitektur umum dari algoritma Hidden Markov Model :



Gambar 2.1 Arsitektur Umum HMM

Pada gambar di atas terdapat beberapa elemen yaitu *Hidden State*, *Observed State*, *Transition*, *Emissions*. x_1, x_2, x_3 adalah *Hidden State* dimana urutan state tersembuyi yang tidak langsung dapat di amati. Menurut probabilitas transisi, setiap state tersembunyi dapat berubah ke state lain pada titik waktu berikutnya. Urutan state yang dapat diamati disebut sebagai state teramat (Observed State). Setiap state teramat dihasilkan dari state tersembunyi berdasarkan kemungkinan emisi. Misalnya, ada kemungkinan state tersembunyi X_1 akan menghasilkan state teramat Y_1 dengan tingkat kemungkinan tertentu. panah transisi menunjukkan transisi antara state tersembunyi. Ada kemungkinan bahwa transisi dari satu state tersembunyi ke state tersembunyi lainnya terjadi, dan kemungkinan ini membentuk matriks transisi A .

Emisi menunjukkan kemungkinan bahwa suatu state tersembunyi akan menghasilkan atau memancarkan suatu state teramati. Ini menunjukkan hubungan antara state tersembunyi dan state teramati.

2.4 Pitch Contour Extraction (PCE)

Pitch Contour Extraction (PCE) adalah proses mengubah data rekaman musik polifonik menjadi rangkaian nada yang berkesinambungan dan mengidentifikasi serta mengekstraksi variasi frekuensi dasar (F0) dari sinyal suara dari waktu ke waktu. Kontur nada ini diklasifikasikan menggunakan isyarat pendengaran. Beberapa karakteristik yang menentukan kontur nada meliputi rata-rata nada, deviasi nada, signifikansi, durasi, dan vibrato. Dengan menganalisis distribusi dari berbagai properti kontur, kita dapat mengidentifikasi kontur melodi dan non-melodi. Tujuan utama dari ekstraksi kontur nada adalah untuk memperkirakan nada dominan dalam rekaman suara yang sangat penting untuk mendekripsi garis melodi dan menganalisis musik. Pada akhirnya, proses ini akan membantu mengekstraksi melodi utama dari data musik yang kompleks secara otomatis, dengan menangkap variasi nada (Frekuensi fundamental) dari waktu ke waktu yang mewakili struktur melodi atau harmonik dari sinyal suara (Salamon & Gomez, 2012).

Dalam implementasinya, metode PCE menggunakan *Short Time Fourier Transform* (STFT) dengan hitungan matematis seperti berikut :

$$X(f, t) = \sum_{n=-\infty}^{\infty} x(n)w(t-n)e^{-j2\pi fn}$$

Dimana :

$X(f, t)$ adalah nilai dari transformasi Fourier waktu pendek (f = frekuensi, t = waktu)

$x(n)$ adalah sinyal waktu kontinu yang ingin di analisis (n)

$w(t-n)$ adalah jendela waktu berguna untuk menmbatasi analisis ke jendela waktu tertentu saja.

$e^{-j2\pi fn}$ adalah bagian dari transformasi Fourier yang memberikan informasi tentang frekuensi komponen sinyal pada waktu tertentu

STFT digunakan untuk mengubah sinyal domain waktu menjadi domain frekuensi yang nantinya akan mempermudah dalam pengerjaan deteksi *Chord* otomatis.

2.5 Digital Audio Processing

adalah proses pengolahan sinyal audio dalam bentuk digital untuk berbagai tujuan, seperti peningkatan kualitas suara, analisis, dan transformasi audio. Dalam konteks *Machine Learning* hal ini melibatkan penggunaan algoritma pembelajaran mesin untuk menganalisis, memproses, dan menghasilkan hasil yang berguna dari data audio digital.

2.5.1 Audio Format

Format Audio atau *Audio Format* adalah bentuk jadi sebuah file audio yang tersimpan di dalam sistem komputer. Format audio secara garis besar terbagi menjadi 2 tipe yaitu Format Audio Terkompresi dan Format Audio tanpa kompresi. Format audio tanpa kompresi adalah format yang memungkinkan penggunanya menyimpan audio sambil menjaga integritas *bit* dan *Sample Rate* dari sumber pembuatannya. Misalnya, file yang tidak terkompresi yang diambil dari CD audio akan memiliki spesifikasi yang sama dengan sumbernya. Format tanpa kompresi tidak hanya digunakan untuk *ripping* CD, tetapi juga banyak digunakan sebagai format output yang dihasilkan dari perekaman pada software DAW (*digital audio workstation*). Sedangkan Format audio terkompresi adalah format audio yang dapat menyimpan data dalam ukuran lebih kecil. Proses kompresi menghapus data dan mengurangi ukuran file. Tujuan utama dari audio terkompresi adalah untuk menghilangkan informasi dari audio yang mungkin sulit didengar oleh telinga manusia. Namun, pada kenyataannya terdapat perbedaan yang cukup signifikan jika dibandingkan dengan sistem suara kelas atas. Semakin tinggi bit rate yang Anda pilih, semakin sedikit penurunan kualitas yang terjadi, dan sebaliknya.

Terdapat berbagai macam format audio terkompresi maupun tidak terkompresi. Antara lain adalah WAV (*Waveform Audio Format*), PCM, AIFF, AU dan BWF untuk format audio tanpa kompresi dan mp3, Vorbis (OGG), mid (MIDI) untuk format audio dengan kompresi.

MPEG-1 Layer 3 (MP3) adalah salah satu metode kompresi audio yang paling banyak digunakan. Format ini memiliki ukuran penyimpanan yang kecil sehingga memungkinkan pengiriman cepat melalui Internet dan kompatibilitas dengan pemutar audio apa pun menjadikannya ideal untuk berbagi audio, streaming, dan hiburan online. Layanan streaming besar seperti Amazon Music dan Google Play Musik menawarkan sebagian besar kontennya dalam format MP3. Platform buku audio seperti Audible, Amazon, dan Scribd hanya menjual buku dalam bentuk MP3. Analisis dan investigasi forensik juga menggunakan algoritma untuk mendeteksi hal-hal seperti produk palsu (Papadakis et al., 2022). WAV (*Waveform Audio Format*) adalah format audio yang dirilis oleh Microsoft. WAV adalah tipe format audio yang sangat baik dalam menjaga informasi suara di dalamnya. Format WAV biasanya sering digunakan sebagai Master pada records/lagu yang di produksi arena memiliki kualitas yang maksimal (Indrayani, 2020).

FITUR	mp3	WAV
Kompresi	Lossy	Tanpa Kompresi
Kualitas Audio	Bergantung pada tingkat kompresi	Kualitas audio yang tinggi
Ukuran File	Kecil	Besar
Kecocokan Penggunaan	Streaming Online	Produksi Audio, pengeditan, Mastering
Kompatibilitas	Umum didukung oleh berbagai perangkat	Umum didukung, tetapi ukuran file besar

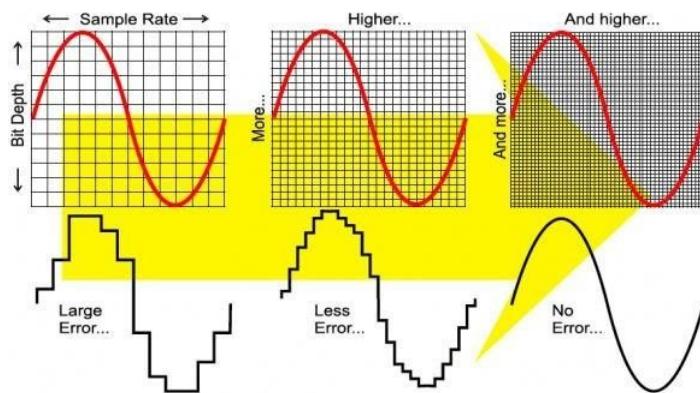
Gambar 2. 2 Perbandingan format .mp3 dengan .WAV

2.5.2 Sample Rate

Sample rate juga dikenal sebagai sampling rate adalah jumlah sampel audio yang dikumpulkan setiap detik dan dinyatakan dalam Hertz (Hz). Hal ini mengacu pada frekuensi pengambilan sampel suara untuk menghasilkan representasi digital. Semakin tinggi ukuran sampel maka semakin tinggi pula kualitas produk digital yang dihasilkan. Industri audio umumnya menggunakan banyak sample rate yang berbeda serta dapat beralih di antara satu dengan lainnya tanpa mengorbankan kualitas suara. Sample rate yang paling umum digunakan adalah 44,1 kHz, 44,1 kHz dipilih sebagai standar untuk sistem CD pada akhir tahun 1970an. Keputusan tersebut terkait dengan kompatibilitas dengan video yang

sebelumnya digunakan untuk menyimpan sinyal digital. Kemudian, laju sampel 44,1 kHz diadopsi untuk digunakan di beberapa sistem audio konsumen lainnya, termasuk pemutar Digital Audio Tape, dan pada masa kini di layanan streaming musik, seperti Spotify, Amazon Music, Apple Music, Deezer, dan Qobuz (Välimäki & Bilbao, 2023).

Namun, dalam audio dan video profesional, 48 kHz adalah laju pengambilan sampel yang disarankan. Ini tersedia untuk pemutar DAT dan juga digunakan dalam standar radio digital Eropa, file audio digital, dan soundtrack DVD. Menurut sejarah, frekuensi sampel pertama 32 kHz dipilih untuk penyiaran karena mendekati frekuensi radio 15 kHz. Frekuensi oversampled, seperti 96 dan 192 kHz, menjadi populer dalam produksi musik selama tahun 1990an. Selain itu, kelompok kelipatan 88,2 dan 176,4 kHz 44,1 kHz tersedia di beberapa sistem karena dimasukkan dalam standar audio MPEG-2. Sample rate tertinggi yang saat ini digunakan untuk sampel audio multi-bit adalah 352,8 dan 384 kHz (Välimäki & Bilbao, 2023)



Gambar 2.3 Perbandingan Sample Rate

(Sumber : <https://www.headphonesty.com>)

2.5.3 Bit Rate

Bit rate adalah istilah yang digunakan untuk mendeskripsikan jumlah data yang ditransfer ke audio. *Bitrate* yang lebih tinggi umumnya akan menghasilkan kualitas audio yang lebih baik. Dalam produksi musik modern, terdapat beberapa macam bitrate yang digunakan mulai dari 16 bit, 24 bit, 32 bit, hingga 64 bit. Jumlah bit atau digit biner (0 dan 1) yang digunakan untuk mewakili setiap sampel bentuk gelombang menentukan rentang dinamis maksimum suara yang direkam secara digital. Setiap bit setara dengan rentang dinamis sekitar 6,0 dB, jadi secara

teori, 16 bit menghasilkan rentang 96 dB (6×16) dan 24 bit menghasilkan rentang 144 dB (6×24). Selain itu, karena rentang dinamis rekaman ditentukan (dan ditetapkan) sebelum trek dirilis, rentang pemutaran nominal 93 dB berfungsi dengan baik untuk distribusi musik. Meskipun demikian, dalam perekaman dan produksi, 24 bit tidak hanya memberikan para produser audio tingkat *Noise floor* yang lebih rendah tetapi juga *Head Room* yang lebih luas (Scholarship@western & Toft, 2024).

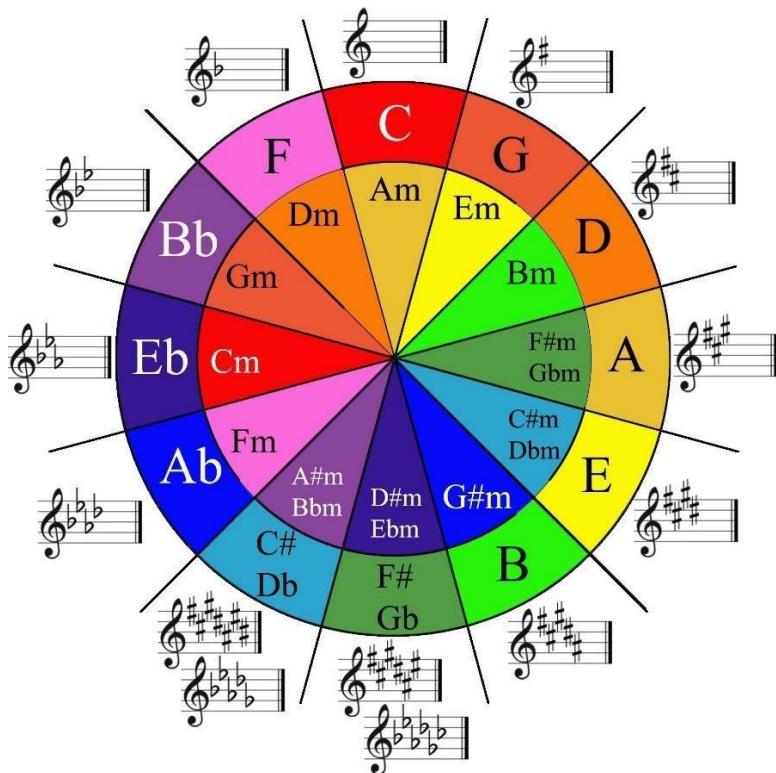
BIT DEPTH	SAMPLE RATE	KUALITAS
16 bit	44.1 kHz	Kualitas CD
24 bit	48 kHz	Kualitas Tinggi Audio Production
24 bit	96 kHz	Audio berkualitas sangat tinggi (untuk kebutuhan Arsip)

Gambar 2. 4 Perbandingan Bit Rate

2.6 Musical Background

Chord adalah salah satu fitur tingkat menengah yang paling menonjol dalam musik Barat seperti pop dan jazz. struktur akord menentukan maksud musical sebuah lagu sepanjang perkembangannya. Dalam improvisasi musik, biasanya musisi menyepakati urutan akord terlebih dahulu untuk mengembangkan ide musik seputar struktur menyeluruh musik itu sendiri. Oleh karena itu, karya musik secara kasar dapat digambarkan dalam bentuk progresi akord, yang biasa disebut struktur harmonik (Carsault et al., 2021).

Chord juga merupakan elemen dasar harmoni yang menentukan struktur harmonis sebuah karya. Secara khusus, akord dapat didefinisikan sebagai kemunculan beberapa nada musik secara bersamaan yang menghasilkan harmoni. Bergantung pada konvensi notasi dan anotasi, akord mungkin dikaitkan dengan nama atau label. Misalnya, akord G7 pada kunci C mayor (biasanya diucapkan "Akor ketujuh dominan G") berisi nada G-B-D-F (de Berardinis et al., 2023).



Gambar 2. 5 Circle Of Fifth Chords

(Sumber : <https://neelmodi.com>)

Nada adalah rangkaian bunyi yang beraturan dan memiliki frekuensi tertentu. Nada memiliki susunan atau urutan yang dimulai dari nada dasar hingga nada oktaf hal ini dinamakan dengan tangga nada. Terdapat tiga macam tangga nada yaitu tangga nada Pentatonic, Diatonic dan Chromatic. Nada dasar Diatonic juga terbagi menjadi dua jenis yaitu tangga nada mayor (Major Scale) dan tangga nada minor (Minor Scale). Tangga nada mayor (Major Scale) dimulai dari nada dasar Do dengan interval $1 - 1 - \frac{1}{2} - 1 - 1 - 1 - \frac{1}{2}$ dari nada Do awal sampai nada Do oktaf. Berbeda dengan tangga nada minor (Minor Scale) tangga nada ini dimulai dari nada dasar La dengan interval $1 - \frac{1}{2} - 1 - 1 - 1 - 1 - \frac{1}{2} - 1$ dari nada La awal sampai nada La oktaf (Ferdiawan et al., 2022).

2.7 Machine Learning

Pembelajaran Mesin (*Machine Learning*) adalah bagian dari kecerdasan buatan yang mencakup algoritma dan model komputasi yang memungkinkan

komputer mempelajari pola dari data dan membuat prediksi seta keputusan tanpa pemrograman eksplisit (Yadav Kasula, n.d.).

Pada *Machine Learning* terdapat 2 macam jenis pembelajaran yaitu *Supervised learning* dan *Unsupervised Learning*. *Supervised Learning* berfungsi dengan cara memetakan input dan output berdasarkan contoh pasangan input dan outputnya. Supervised learning menyimpulkan dari data pelatihan berlabel yang terdiri dari sekumpulan contoh pelatihan. Dataset input yang digunakan dalam Supervised Learning terbagi menjadi Dataset Latih (*Train Dataset*) dan Dataset tes (*Test Dataset*). Dataset latih memiliki variabel output yang nantinya perlu di prediksi atau di klasifikasikan. Berbeda dengan *Supervised Learning*, *Unsupervised Learning* bekerja secara sendiri untuk menyajikan dan menemukan struktur datanya. Apabila mengenali data baru, maka ia akan mempelajari data ini dengan menggunakan fitur-fitur pada data sebelumnya yang sudah dipelajari (Mahesh, 2018).

2.7.1 Supervised Learning

Supervised learning merupakan teknik pembelajaran mesin yang memanfaatkan data berlabel untuk melatih model. Dalam konteks penelitian deteksi chord musik, supervised learning digunakan untuk melatih model mengenali pola-pola chord dari data audio yang sudah memiliki label chord yang benar. Model akan menerima input berupa fitur-fitur audio seperti chroma features (x_t), kemudian menghasilkan prediksi chord (y^t). Model akan terus memperbarui parameternya untuk meminimalkan error antara prediksi dengan chord aktual. Dalam penelitian ini, template matching merupakan salah satu pendekatan supervised learning dimana template chord yang sudah didefinisikan digunakan sebagai acuan untuk mencocokkan dengan fitur audio input.

2.7.2 Unsupervised Learning

Unsupervised learning adalah teknik pembelajaran mesin yang tidak memerlukan data berlabel. Dalam konteks deteksi chord musik, Hidden Markov Model (HMM) dapat dianggap sebagai bentuk unsupervised learning karena mampu mempelajari pola transisi antar chord dan probabilitas emisi tanpa memerlukan label chord yang eksplisit. HMM dapat menemukan struktur

tersembunyi dari sekuens observasi chroma features untuk menentukan sekuens chord yang paling mungkin. Meskipun dalam implementasinya, kita tetap menggunakan beberapa parameter yang ditentukan berdasarkan pengetahuan musik seperti matriks transisi chord.

2.8 Framework

Framework adalah struktur yang digunakan untuk mengembangkan aplikasi yang memudahkan dalam menulis kode. Penggunaan framework membuat pembuatan program menjadi efisien, singkat, serta terorganisir. Framework memiliki fungsi untuk mempercepat pembuatan aplikasi, meningkatkan keamanan, dan mempermudah dalam perbaikan serta perawatan. Penelitian ini menggunakan framework streamlit untuk membangun antarmuka aplikasi deteksi chord musik.

Streamlit adalah kerangka kerja berbasis Python yang berfungsi untuk membuat aplikasi web interaktif. Framework ini memungkinkan pengembang menciptakan antarmuka pengguna dengan mudah tanpa harus menulis banyak baris kode. Dalam konteks penelitian ini, Streamlit digunakan untuk membangun antarmuka yang memungkinkan pengguna mengunggah file audio musik, memproses deteksi chord menggunakan algoritma HMM dan template matching, serta menampilkan hasil deteksi chord dalam format yang mudah dibaca. Streamlit dipilih karena kemudahannya dalam mengintegrasikan pemrosesan audio dan visualisasi hasil dengan tampilan web yang interaktif.

BAB 3

ANALISIS DAN PERANCANGAN

3.1 Analisis

Analisis dan perancangan sistem adalah suatu pendekatan yang sistematis untuk merancang dan mengembangkan sistem. Analisis mendefinisikan kebutuhan terkait sistem yang akan dikembangkan. Dengan pendekatan yang sistematis, diharapkan sistem yang dikembangkan dapat memenuhi kebutuhan pengguna secara efektif dan efisien.

3.1.1 *Analisis Masalah*

Musik memiliki kompleksitas yang tinggi dalam hal komposisi dan struktur nada yang membentuk chord. Chord dari suatu file audio dapat dikenali melalui karakteristik frekuensi dan pola harmonik yang terdapat dalam sinyal tersebut. Variasi dalam kualitas rekaman dan teknik permainan musik menambah kompleksitas dalam mendeteksi chord yang dimainkan. Beberapa rekaman audio mungkin memiliki noise atau distorsi yang dapat mempengaruhi akurasi deteksi. Selain itu, penggunaan berbagai jenis instrumen dan gaya permainan dapat menghasilkan karakteristik suara yang berbeda untuk chord yang sama. Kompleksitas bertambah ketika harus mendeteksi progresi chord, dimana multiple chord dimainkan secara berurutan dengan durasi yang bervariasi. Oleh karena itu, penelitian ini diharapkan dapat mengembangkan sistem yang mampu mendeteksi chord dan progresi chord secara akurat dari file audio musik.

3.1.2 *Analisis Kebutuhan*

Analisis kebutuhan adalah proses identifikasi, pemahaman, dan spesifikasi kebutuhan yang harus dipenuhi oleh sistem yang akan dikembangkan. Analisis kebutuhan dibagi menjadi dua bagian utama, yaitu fungsional dan non-fungsional.

1. Kebutuhan fungsional

Kebutuhan fungsional adalah spesifikasi fitur dan fungsi yang mesti dimiliki setiap sistem demi mencapai tujuan utama. Penelitian ini memiliki kebutuhan fungsional utama, yaitu:

- a. Menerima file audio (format WAV) sebagai input.

- b. Melakukan pre-processing pada file audio seperti normalisasi dan pembersihan noise.
- c. Mengekstrak fitur chromagram menggunakan librosa untuk analisis spektral.
- d. Menggunakan kombinasi Template Matching dan Hidden Markov Model (HMM) untuk mendeteksi chord.
- e. Melatih model HMM dengan dataset yang berisi rekaman chord instrumen gitar, bass dan piano (major dan minor).
- f. Menghasilkan keluaran berupa Deteksi single chord, Deteksi progresi 2 chord, Deteksi progresi 4 chord
- g. Menyediakan visualisasi hasil analisis dalam bentuk chromagram dan grafik.
- h. Menyediakan antarmuka pengguna yang interaktif menggunakan Streamlit.

2. Kebutuhan non-fungsional

Kebutuhan non-fungsional adalah cara di mana sistem berjalan dan beroperasi dengan seharusnya. Penelitian ini memiliki beberapa kebutuhan non-fungsional, yaitu:

- a. Performa Sistem mampu memproses file audio WAV dengan durasi hingga 1 menit. Waktu pemrosesan tidak lebih dari 30 detik per file audio. Penggunaan memori yang efisien dalam menganalisis file audio dan sistem dapat menangani multiple request dari pengguna.
- b. Antarmuka web yang intuitif dan mudah digunakan Tampilan visualisasi yang informatif (chromagram dan grafik hasil analisis) Feedback yang jelas untuk setiap aksi pengguna Responsif pada berbagai ukuran layar.
- c. Hasil prediksi yang menunjukkan seberapa akurat dalam memprediksi chord dalam file audio.

3.2 Gambaran Umum Sistem

Sistem ini disusun berdasarkan analisis mendalam terhadap kebutuhan deteksi chord musik. Tujuan utama yaitu untuk meningkatkan akurasi dan efisiensi dalam mengenali chord dan progresi chord dari file audio. User akan menggunakan sistem berbasis web yang dibangun dengan framework Streamlit, dimana pembuatan model dimulai dari pengumpulan dataset audio chord guitar

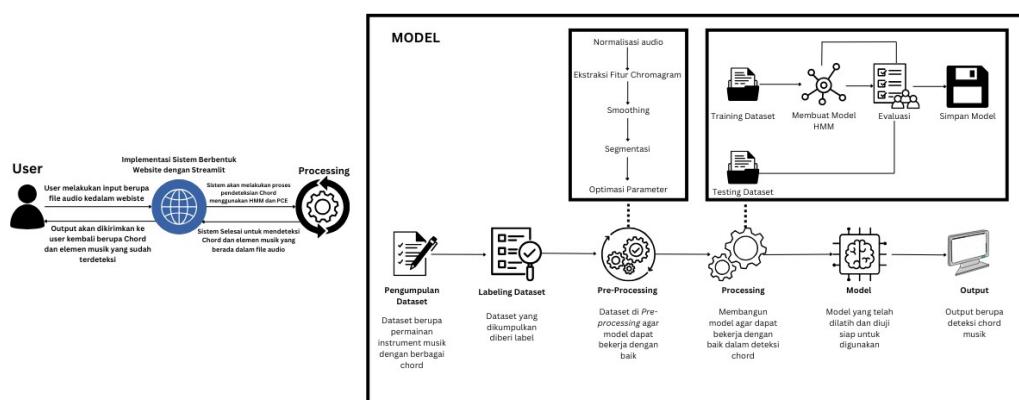
yang direkam secara terstruktur. Dataset terdiri dari berbagai jenis chord (major dan minor) yang diorganisir dalam folder-folder sesuai dengan nama chord.

File audio yang diunggah akan melalui serangkaian tahap pre-processing meliputi normalisasi audio, ekstraksi fitur chromagram menggunakan librosa, smoothing, dan segmentasi untuk analisis progresi chord. Selanjutnya, dilakukan implementasi dengan kombinasi metode Template Matching dan Hidden Markov Model (HMM) pada data untuk mendeteksi chord. Template Matching digunakan untuk analisis pola chord berdasarkan profil chroma, sementara HMM digunakan untuk memodelkan transisi antar chord dalam progresi.

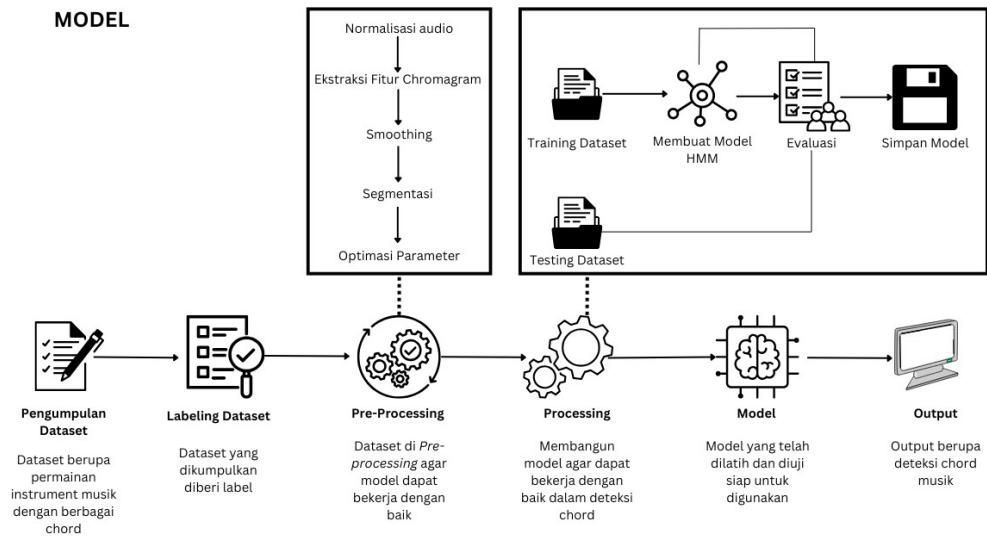
Sistem menyediakan tiga jenis analisis:

1. Deteksi single chord
2. Deteksi progresi 2 chord
3. Deteksi progresi 4 chord

Hasil analisis ditampilkan dalam bentuk visualisasi yang informatif, termasuk chromagram dan grafik profil chroma, disertai dengan tingkat kepercayaan (confidence score) untuk setiap deteksi chord. Model yang telah dilatih disimpan menggunakan pickle untuk penggunaan selanjutnya, dan user dapat melihat hasil deteksi chord beserta visualisasinya melalui antarmuka web yang interaktif.



Gambar 3. 1 Arsitektur Umum Sistem



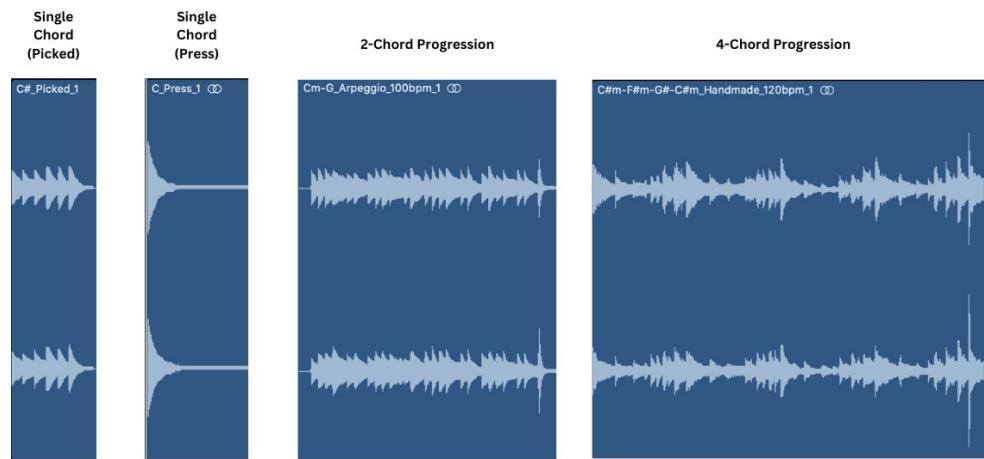
Gambar 3. 2 Tahapan Pembuatan Model

3.3 Analisis Dataset

Dataset yang digunakan pada penelitian ini merupakan kumpulan rekaman audio chord gitar yang diorganisir dalam struktur folder berdasarkan jenis chord. Dataset ini terdiri dari rekaman chord-chord dasar gitar yang mencakup chord major (C, D, E, F, G, A, B) dan chord minor (Cm, Dm, Em, Fm, Gm, Am, Bm) beserta variasinya dengan kunci yang dinaikkan setengah (C#, D#, F#, G#, A#). Setiap rekaman audio disimpan dalam format WAV (Waveform Audio File Format) dengan spesifikasi teknis yang konsisten, meliputi sample rate 44100 Hz, durasi 1-15 detik per sampel, bit depth 24-bit, dan channel stereo.

File Name	Date	Type
A_Picked_1.wav	9 December 2024 19.42	-- Folder
A_Press_1.wav	9 December 2024 19.46	-- Folder
A-D-E-A.FS_120bpm_1.wav	9 December 2024 14.08	986 KB Waveform audio
A-D-E-A.FS_150bpm_1.wav	9 December 2024 12.58	1,2 MB Waveform audio
A-D-E-A.Handmade_120bpm_1.wav	9 December 2024 15.01	2,2 MB Waveform audio
A-D-E-A.Handmade_160bpm_1.wav	9 December 2024 15.01	4,3 MB Waveform audio
A-D-E-A.Hard_Picked_80bpm_1.wav	9 December 2024 18.08	4,5 MB Waveform audio
A-D-E-A.Hard_Picked_120bpm_1.wav	9 December 2024 18.08	3,3 MB Waveform audio
A-D-E-A.Hard_Press_80bpm_1.wav	9 December 2024 14.31	9,3 MB Waveform audio
A-D-E-A.Hard_Press_120bpm_1.wav	9 December 2024 14.31	3,3 MB Waveform audio
A-D-E-A.Soft_Picked_80bpm_1.wav	9 December 2024 14.31	9,3 MB Waveform audio
A-D-E-A.Soft_Picked_120bpm_1.wav	9 December 2024 14.31	3,3 MB Waveform audio
A-D-G.PB_120bpm_1.wav	9 December 2024 15.01	9,1 MB Waveform audio
A-D-G.PB_150bpm_1.wav	9 December 2024 14.31	3,3 MB Waveform audio
A-E.Arpeggio_100bpm_1.wav	9 December 2024 18.08	9,3 MB Waveform audio
A-E.Arpeggio_120bpm_1.wav	9 December 2024 18.08	2,4 MB Waveform audio
A-E.Fretless_120bpm_1.wav	9 December 2024 15.01	2,2 MB Waveform audio
A-E.Fretless_150bpm_1.wav	9 December 2024 15.01	4,3 MB Waveform audio
A.wav	9 December 2024 15.01	1,2 MB Waveform audio

Gambar 3. 3 Struktur Folder Dataset



Gambar 3. 4 Dataset dalam bentuk Waveform

Dataset diorganisir dalam struktur folder yang sistematis, dimana setiap folder mewakili satu jenis chord dan berisi multiple sampel rekaman dari chord tersebut. Struktur ini memudahkan dalam proses pelatihan dan pengujian model. Total dataset mencakup 24 kelas chord yang terdiri dari 12 chord major dan 12 chord minor.

File Name	Chord	Note
A_Press_1.wav	A	Major
Am_Press_1.wav	Am	Minor
Am-E_Arpeggio_1.wav	Am - E	Major & Minor

Gambar 3. 5 Dataset Dengan Label Chord

Setiap file audio dalam dataset telah melalui proses validasi untuk memastikan kualitas rekaman yang konsisten, ketepatan chord yang dimainkan, minimal noise dan interferensi, serta durasi yang sesuai untuk analisis. Dataset ini kemudian digunakan sebagai dasar untuk melatih model HMM dalam pengenalan chord, membuat template chord untuk metode template matching, serta validasi dan pengujian akurasi sistem secara keseluruhan.

Tabel 3. 1 Contoh Labelisasi Dataset Chord

Nama File	Chord	Root Note	Tempo (bpm)
A_Picked_1.wav	A	A	-
B-F#_Arpeggio_100bpm.wav	B,F#	B	100
C-F-A#_PB_120bpm.wav	C,F,A#	C	120
D-G-A-B_Handmade_160bpm.wav	D,G,A,B	D	160

3.4 Pre-Processing

Pre-Processing merupakan tahapan penting untuk mempersiapkan sinyal audio agar siap diolah dalam proses deteksi chord. Tahapan pre-processing meliputi normalisasi audio, ekstraksi fitur chromagram, smoothing, segmentasi, dan optimasi parameter.

3.4.1 Normalisasi Audio

Normalisasi audio adalah proses menyeragamkan amplitudo sinyal audio untuk mengurangi variasi dalam volume dan intensitas suara. Proses ini menggunakan fungsi librosa.util.normalize() yang menghasilkan sinyal audio dengan amplitudo yang seragam. Normalisasi penting untuk memastikan konsistensi dalam analisis spektral dan mengurangi pengaruh variasi volume dalam rekaman.

```
# Normalize audio
y = librosa.util.normalize(y)
```

Gambar 3. 6 Proses Normalisasi Audio

Tabel 3.2 Contoh Normalisasi Audio

Amplitudo Sebelum	Amplitudo Sesudah
0.8234	0.9012
0.9123	0.9989
-0.6543	-0.7165
-0.4567	-0.5001
0.7890	0.8641

3.4.2 Ekstraksi Fitur Chromagram

Chromagram merupakan representasi visual dari konten musical yang menunjukkan distribusi energi di seluruh kelas pitch. Proses ekstraksi menggunakan fungsi librosa.feature.chroma_stft() dengan parameter yang disesuaikan seperti n_fft untuk ukuran window dan hop_length untuk pergeseran frame. Chromagram memungkinkan sistem untuk menganalisis komponen harmonik yang membentuk chord.

```
# Compute STFT
S = np.abs(librosa.stft(y, n_fft=frame_length, hop_length=hop_length))

# Compute chromagram
chroma = librosa.feature.chroma_stft(
    S=S,
    sr=sr,
    n_fft=frame_length,
    hop_length=hop_length,
    tuning=0.0
)
```

Gambar 3.7 Proses Ekstraksi Chromagram

Contoh pada tahap ini terdapat pada Tabel 3.3 di bawah ini.

Tabel 3.3 Contoh Ekstraksi Chromagram

Pitch Class	Frame 1	Frame 2	Frame 3
C	0.85	0.13	0.32
C#	0.21	0.90	0.43
D	0.40	0.33	0.88

3.4.3 Smoothing

Smoothing adalah proses penghalusan sinyal untuk mengurangi noise dan fluktuasi yang tidak diinginkan dalam chromagram. Teknik ini menggunakan kombinasi filter median dan metrik cosine melalui fungsi `librosa.decompose.nn_filter()`. Proses ini menghasilkan representasi yang lebih stabil dan mengurangi anomali dalam deteksi chord.

```
# Apply smoothing
chroma_smooth = np.minimum(chroma, librosa.decompose.nn_filter(
    chroma,
    aggregate=np.median,
    metric='cosine'
))
```

Gambar 3. 8 Proses *Smoothing*

Tabel 3. 4 Contoh *Smoothing*

Pitch Class	Chroma sebelum <i>Smoothing</i>	Chroma setelah <i>Smoothing</i>
C	0.85	0.87
C#	0.12	0.10
D	0.23	0.22
D#	0.08	0.05
E	0.45	0.48
F	0.15	0.13
F#	0.43	0.40
G	0.74	0.71
G#	0.32	0.35
A	0.87	0.90
A#	0.41	0.42
B	0.22	0.25

3.4.4 Segmentasi

Segmentasi adalah proses membagi audio menjadi bagian-bagian yang lebih kecil untuk analisis yang lebih detail. Sistem ini mengimplementasikan tiga jenis segmentasi: single chord (1 segmen), two chord progression (2 segmen), dan four chord progression (4 segmen). Setiap segmen dianalisis secara independen untuk mendeteksi chord yang dimainkan pada bagian tersebut.

```
def predict_chord(audio_path, models):
    """Predict chord from audio file"""
```

```
def detect_chord_progression_2(audio_path, models):
    """
    Mendeteksi progresi 2 chord berurutan dari file audio
    """
```

```
def detect_chord_progression_4(audio_path, models):
    """
    Mendeteksi progresi 4 chord berurutan dari file audio
    """
```

Gambar 3. 9 Proses Segmentasi

Tabel 3. 5 Contoh proses Segmentasi

Segmen	Waktu (detik)	Chord terdeteksi	<i>Confidence Score</i>
1	0.0 – 3.0	C	0.85
2	3.0 – 5.0	Am	0.78

3.4.5 Optimasi Parameter

Optimasi parameter melibatkan penyesuaian nilai-nilai kunci seperti `n_fft` dan `hop_length` untuk memaksimalkan akurasi deteksi. Nilai `n_fft` disesuaikan dengan panjang sinyal (minimal 512 sampel) dan `hop_length` diatur sebagai seperempat dari `n_fft`. Optimasi ini penting untuk menyeimbangkan antara resolusi frekuensi dan resolusi waktu dalam analisis spektral.

```

def extract_features_optimized(audio_path):
    try:
        # Load audio with fixed parameters
        y, sr = librosa.load(audio_path, sr=22050, duration=1.0)

        # Trim silence more aggressively
        y, _ = librosa.effects.trim(y, top_db=25)
        y = librosa.util.normalize(y)

        # Calculate adaptive n_fft based on signal length
        signal_length = len(y)
        n_fft = min(512, signal_length // 2) # Reduced from 2048 to 512
        hop_length = n_fft // 4

        # Multiple feature extraction
        # 1. Constant-Q chromagram
        chroma_cq = librosa.feature.chroma_cqt(
            y=y, sr=sr,
            hop_length=hop_length,
            bins_per_octave=36
        )

        # 2. STFT chromagram
        chroma_stft = librosa.feature.chroma_stft(
            y=y, sr=sr,
            n_fft=n_fft,
            hop_length=hop_length
        )

        # Combine features
        chroma_combined = (chroma_cq + chroma_stft) / 2.0

        # Get frame with highest energy
        frame_energy = np.sum(chroma_combined, axis=0)
        max_energy_frame = np.argmax(frame_energy)
        chroma_peak = chroma_combined[:, max_energy_frame]

        return chroma_peak, chroma_combined, sr, hop_length

    except Exception as e:
        print(f"Error in feature extraction: {e}")
        return None, None, None, None

```

Gambar 3. 10 Proses Optimasi Parameter

3.5 Representasi Fitur Audio dengan Chromagram

Proses pertama yang dilakukan yaitu dengan ekstraksi fitur audio menggunakan chromagram, di mana sinyal audio dikonversi menjadi representasi spektral yang menunjukkan distribusi energi pada 12 pitch class. Chromagram menggunakan Short-Time Fourier Transform (STFT) untuk menganalisis konten frekuensi audio dalam interval waktu tertentu. Setiap frame audio dianalisis untuk menghasilkan vektor 12-dimensi yang

merepresentasikan intensitas relatif dari setiap pitch class (C, C#, D, D#, E, F, F#, G, G#, A, A#, B).

Selanjutnya, setiap frame chromagram dinormalisasi untuk memastikan konsistensi dalam analisis. Parameter seperti n_fft dan hop_length dioptimalkan untuk mendapatkan resolusi waktu-frekuensi yang seimbang. Sebagai contoh, untuk file audio chord C major, chromagram akan menunjukkan intensitas tinggi pada pitch class C, E, dan G yang merupakan komponen dari chord tersebut.

```
def extract_features(audio_path, frame_length=2048, hop_length=512):
    """
    Ekstraksi fitur menggunakan STFT untuk mendapatkan chromagram
    """
    try:
        # Load audio
        y, sr = librosa.load(audio_path, sr=22050)

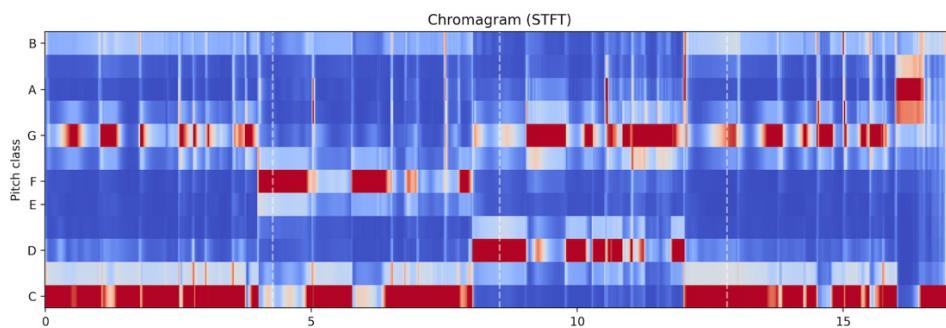
        # Normalize audio
        y = librosa.util.normalize(y)

        # Compute STFT
        S = np.abs(librosa.stft(y, n_fft=frame_length, hop_length=hop_length))

        # Compute chromagram
        chroma = librosa.feature.chroma_stft(
            S=S,
            sr=sr,
            n_fft=frame_length,
            hop_length=hop_length,
            tuning=0.0
        )

    return chroma, sr
```

Gambar 3. 11 Proses Ekstraksi Chromagram



Gambar 3. 12 Contoh Visualisasi Chromagram

Model HMM menerima vektor fitur chromagram sebagai input, kemudian menghasilkan probabilitas untuk setiap kemungkinan chord berdasarkan pola spektral yang telah dipelajari. Hasil ini dikombinasikan dengan template matching

untuk meningkatkan akurasi deteksi. Template matching membandingkan profil chromagram input dengan template predefinisi untuk setiap chord, menghasilkan skor kecocokan yang membantu dalam penentuan chord final.

Representasi fitur ini kemudian digunakan untuk analisis lebih lanjut, baik untuk deteksi single chord maupun progresi chord. Dengan pendekatan ini, sistem dapat mengidentifikasi karakteristik spektral yang membedakan berbagai jenis chord dan menghasilkan prediksi yang akurat.

3.6 Klasifikasi dengan Model HMM dan Template Matching

Setelah mendapatkan representasi fitur audio dengan chromagram, output vektor yang dihasilkan akan diproses menggunakan kombinasi Hidden Markov Model (HMM) dan Template Matching untuk melakukan klasifikasi chord.

Pada sistem ini, proses klasifikasi dapat dijelaskan dalam beberapa tahap. Pertama, chromagram yang dihasilkan dari proses ekstraksi fitur akan dianalisis menggunakan dua pendekatan paralel. Pendekatan pertama menggunakan HMM yang telah dilatih untuk mengenali pola transisi antar chord. HMM memiliki tiga komponen state yang merepresentasikan karakteristik chord, dengan matriks transisi yang dioptimalkan untuk menangkap pola perpindahan antar chord. Pendekatan kedua menggunakan Template Matching, dimana profil chroma dari input dibandingkan dengan template predefinisi untuk setiap chord.

3.6.1 Analisis Chromagram dan Template Matching

Untuk memproses sinyal audio secara menyeluruh, sistem menggunakan analisis chromagram yang bergerak sepanjang sinyal audio secara berurutan. Proses ini menggunakan Short-Time Fourier Transform (STFT) yang bergerak dalam satu dimensi waktu. Setiap frame analisis bergerak satu per satu sepanjang sinyal audio, menghasilkan representasi spektral yang menunjukkan intensitas 12 pitch class pada setiap titik waktu. Output yang dihasilkan adalah vektor fitur chromagram yang merepresentasikan karakteristik spektral dari audio input.

Untuk mengidentifikasi karakteristik chord yang signifikan, sistem menggunakan template matching yang tidak bergantung pada posisi spesifik dalam urutan waktu. Pendekatan ini membantu menghilangkan informasi yang kurang relevan dan fokus pada pola spektral yang penting. Template matching membandingkan

profil chroma dari input dengan template predefinisi untuk setiap chord, menghasilkan skor kecocokan yang mencerminkan karakteristik chord yang paling dominan.

Skor kecocokan yang dihasilkan dari proses template matching kemudian dikombinasikan dengan hasil analisis HMM untuk menghasilkan prediksi final. Kombinasi kedua metode ini memungkinkan sistem untuk mengidentifikasi chord dengan lebih akurat, dengan mempertimbangkan baik karakteristik spektral maupun pola transisi temporal dalam musik.

3.7 Evaluasi

Setelah sistem dilatih dengan kombinasi model HMM dan Template Matching, selanjutnya dilakukan evaluasi untuk mengetahui kemampuan sistem dalam mendeteksi chord dari file audio. Evaluasi menggunakan confusion matrix untuk menganalisis performa sistem dalam mengklasifikasikan chord. Confusion matrix memungkinkan kita melihat seberapa akurat sistem dalam mendeteksi setiap jenis chord dan mengidentifikasi pola kesalahan yang mungkin terjadi dalam klasifikasi. Evaluasi ini penting untuk memahami kekuatan dan keterbatasan sistem dalam mengenali berbagai jenis chord dan progresi chord.

3.8 Menyimpan Model

Setelah model HMM dilatih dan dievaluasi, tahap selanjutnya adalah menyimpan model. Penyimpanan dilakukan dengan menggunakan library pickle untuk menyimpan parameter dan state dari model HMM yang telah dilatih ke dalam file. Model yang tersimpan dapat dimuat kembali untuk digunakan dalam prediksi tanpa perlu melakukan proses pelatihan ulang. Proses ini meningkatkan efisiensi sistem karena model yang sudah dilatih dapat digunakan berulang kali.

3.9 Perhitungan Ekstraksi Fitur Audio Menggunakan Chromagram

Asumsikan kita akan melakukan deteksi chord pada data audio piano yang dimainkan dengan chord C. langkah pertama yang akan dilakukan oleh sistem adalah mengekstraksi fitur audio yang terkandung di dalam data menggunakan *Short Time Fourier Transfer* (STFT) dengan rumus sebagai berikut :

$$X(f, t) = \sum_{n=-\infty} x(n)w(t-n)e^{(-j2\pi fn)}$$

Dimana:

$X(f,t)$ = hasil STFT pada frekuensi f dan waktu t

$x(n)$ = sinyal input (sampel audio)

$w(t-n)$ = fungsi window

$e^{(-j2\pi fn)}$ = fungsi basis Fourier

n = indeks sampel

Σ = penjumlahan dari $-\infty$ sampai ∞

Data yang masuk akan melewati proses perhitungan bin frekuensi yang berfungsi untuk mengenali nada apa yang terkandung di dalam data audio yang kita inputkan dengan rumus berikut :

$$\text{Bin} = \text{ukuran window/sample rate}$$

Dalam kasus chord C, kita memiliki 3 nada utama yang membentuk chord C yaitu C, E, dan G. sehingga terjadilah perhitungan seperti ini :

1. Nada C4 (261.63 Hz) → bin = $261.63 \times 2048/44100 = 12$ (bin C)
2. Nada E4 (329.63 Hz) → bin = $329.63 \times 2048/44100 = 15$ (bin E)
3. Nada G4 (392.00 Hz) → bin = $392.00 \times 2048/44100 = 18$ (bin G)

Selanjutnya setiap frame akan dikalikan dengan fungsi window Hann untuk mengurangi efek diskontinuitas dengan perhitungan sebagai berikut :

$$w[n] = 0.5 * (1 - \cos(2\pi n/(N-1))), n = 0, 1, \dots, 2047$$

Contoh perhitungan :

$$N = 2048 \text{ (ukuran window)}$$

$$w[n] = 0.5 * (1 - \cos(2\pi n/(N-1)))$$

Contoh untuk 2 titik pertama:

$$n = 0:$$

$$\begin{aligned} w[0] &= 0.5 * (1 - \cos(2\pi \times 0/2047)) \\ &= 0.5 * (1 - 1) \\ &= 0 \end{aligned}$$

$$n = 1:$$

$$\begin{aligned} w[1] &= 0.5 * (1 - \cos(2\pi \times 1/2047)) \\ &= 0.5 * (1 - 0.9969) \\ &= 0.00153 \end{aligned}$$

Perhitungan ini dilakukan sampai window ke 2047. Setelah itu hasil window akan dikalikan dengan sampel audio. Misalkan kita memiliki sampel audio sebagai berikut :

$$x[0] = 0.82$$

$$x[1] = 0.79$$

Perkalian sampel dengan window:

$$\begin{aligned} xw[0] &= x[0] \times w[0] = 0.82 \times 0 = 0 \\ xw[1] &= x[1] \times w[1] = 0.79 \times 0.00153 = 0.00121 \end{aligned}$$

ini dilakukan sampai mencapai 44100hz (atau dilakukan sebanyak 44100 kali). Hasil perhitungan ini kemudian akan di gunakan oleh STFT untuk menentukan magnitude dari nada di dalam file inputan dengan perhitungan sebagai berikut :

Contoh untuk Bin 12 (C4) :

$$X(12,0) = \sum xw[n]e^{(-j2\pi \times 12 \times n / 2048)}$$

Untuk n = 0:

$$\begin{aligned} xw[0] \times e^{(-j2\pi \times 12 \times 0 / 2048)} \\ = (0.82 \times 0) \times 1 \\ = 0 \end{aligned}$$

Untuk n = 1:

$$\begin{aligned} xw[1] \times e^{(-j2\pi \times 12 \times 1 / 2048)} \\ = (0.79 \times 0.00153) \times (\cos(-2\pi \times 12 \times 1 / 2048) + j \times \sin(-2\pi \times 12 \times 1 / 2048)) \\ = 0.00121 \times (0.9963 - j0.0858) \\ = 0.00120 - j0.00010 \end{aligned}$$

Untuk n = 2:

$$\begin{aligned} xw[2] \times e^{(-j2\pi \times 12 \times 2 / 2048)} \\ = (0.75 \times 0.00613) \times (\cos(-2\pi \times 12 \times 2 / 2048) + j \times \sin(-2\pi \times 12 \times 2 / 2048)) \\ = 0.00460 \times (0.9853 - j0.1714) \\ = 0.00453 - j0.00079 \end{aligned}$$

Proses ini dilakukan sampai n = 2047, lalu semua hasil dijumlahkan :

$$\begin{aligned} X(12,0) &= (0) + (0.00120 - j0.00010) + (0.00453 - j0.00079) + \dots \\ \text{Magnitude } |X(12,0)| &= \sqrt{(\text{real}^2 + \text{imag}^2)} = 0.85 \end{aligned}$$

Proses ini dilakukan oleh ketiga bin yaitu bin C, E, dan G hingga menghasilkan magnitude terkuat untuk masing masing notasi dengan hasil sebagai berikut :

$$\begin{aligned} |X(12,0)| &= 0.85 // energi nada C4 \\ |X(15,0)| &= 0.70 // energi nada E4 \\ |X(18,0)| &= 0.75 // energi nada G4 \end{aligned}$$

Selanjutnya, hasil STFT akan di konversi menjadi chromagram dengan perhitungan sebagai berikut :

$$C(p,t) = \sum |X(f,t)|^2$$

Untuk *Pitch Class* C (p=0) :

$$\begin{aligned} C(0,t) &= |X(12,0)|^2 + |X(24,0)|^2 + |X(36,0)|^2 \\ &= 0.85^2 + 0.45^2 + 0.23^2 \end{aligned}$$

$$= 0.72 + 0.20 + 0.05 = 0.97$$

Angka 12, 24 dan 36 mewakili angka bin dari notasi C dari oktaf yang berbeda. Perhitungan ini kembali dilakukan untuk seluruh 24 notasi musik yang ada sehingga diperolehlah hasil chromagram sebagai berikut :

C = 1.00 // nada dasar (<i>root</i>)
C# = 0.11
D = 0.14
D# = 0.09
E = 0.79 // <i>third</i>
F = 0.18
F# = 0.13
G = 0.86 // <i>fifth</i>
G# = 0.11
A = 0.16
A# = 0.10
B = 0.12

Hasil menunjukkan notasi dengan *magnitude* terbesar terletak pada nada C, E dan G yang merepresentasikan akor C ketika di mainkan. Hasil ini akan digunakan sebagai input untuk proses deteksi chord menggunakan HMM dan *Template Matching*.

3.10 Perhitungan Deteksi Chord Menggunakan HMM dan Template Matching

Hasil yang diperoleh dari ekstraksi chromagram tersebut akan di cocokkan dengan template chord yang sudah kita definisikan.

Template C = [1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]

masing masing angka merepresentasikan notasi musik yaitu C C# D D# E F F# G G# A A# B

Penjelasan nilai:

1 = nada yang ada dalam chord (C, E, G)

0 = nada yang tidak ada dalam chord

Hasil yang diperoleh dari ekstraksi adalah :

Input = [1.00, 0.11, 0.14, 0.09, 0.79, 0.18, 0.13, 0.86, 0.11, 0.16, 0.10, 0.12]

Lalu masuk ke dalam perhitungan Korelasi Cosine dengan perhitungan sebagai berikut :

$$\text{correlation} = \text{dot(chroma, template)} / (\|\text{chroma}\| \times \|\text{template}\|)$$

1. Dot product:

$$\begin{aligned} \text{dot} &= (1.00 \times 1) + (0.11 \times 0) + (0.14 \times 0) + (0.09 \times 0) + (0.79 \times 1) + \\ &\quad (0.18 \times 0) + (0.13 \times 0) + (0.86 \times 1) + (0.11 \times 0) + (0.16 \times 0) + \\ &\quad (0.10 \times 0) + (0.12 \times 0) \\ &= 1.00 + 0.79 + 0.86 \\ &= 2.65 \end{aligned}$$

2. Magnitude chroma:

$$\begin{aligned} \|\text{chroma}\| &= \sqrt{(1.00^2 + 0.11^2 + 0.14^2 + 0.09^2 + 0.79^2 + 0.18^2 + \\ &\quad 0.13^2 + 0.86^2 + 0.11^2 + 0.16^2 + 0.10^2 + 0.12^2)} \\ &= \sqrt{(1.00 + 0.01 + 0.02 + 0.01 + 0.62 + 0.03 + \\ &\quad 0.02 + 0.74 + 0.01 + 0.03 + 0.01 + 0.01)} \\ &= \sqrt{2.51} \\ &= 1.57 \end{aligned}$$

3. Magnitude template:

$$\begin{aligned} \|\text{template}\| &= \sqrt{(1^2 + 0^2 + 0^2 + 0^2 + 1^2 + 0^2 + 0^2 + 1^2 + 0^2 + 0^2 + 0^2 + 0^2)} \\ &= \sqrt{3} \\ &= 1.73 \end{aligned}$$

4. Korelasi final:

$$\text{correlation} = 2.65 / (1.57 \times 1.73)$$

$$= 2.65 / 2.72$$

$$= 0.98$$

Setelah mendapatkan hasil dari *Template Matching*, berikutnya akan dilakukan perhitungan untuk Algoritma *Hidden Markov Model* sebagai berikut:

1. States (S) = {C, Dm, Em, F, G, Am, Bdim}
2. Observations (O) = chromagram hasil ekstraksi
3. Transition Matrix (A) = probabilitas perpindahan antar chord
4. Emission Matrix (B) = probabilitas observasi untuk setiap state
5. Initial Probabilities (π) = probabilitas awal setiap state

Berikut adalah Mean Vector dari chord C :

$$\mu_C = [0.9, 0.1, 0.1, 0.1, 0.8, 0.1, 0.1, 0.9, 0.1, 0.1, 0.1, 0.1]$$

Penjelasan:

- 0.9 untuk nada dalam chord (C, E, G)
- 0.1 untuk nada lainnya

Dasar penentuan nilainya adalah :

1. Training Data:
 - Diambil dari rata-rata chromagram banyak sampel chord C
 - Nilai 0.9 mewakili energi tinggi yang konsisten
 - Nilai 0.1 mewakili noise atau harmonik lemah
2. Karakteristik Chord:
 - Root (C) = 0.9 (paling kuat)
 - Third (E) = 0.8 (kuat)

- Fifth (G) = 0.9 (kuat)

Selanjutnya masuk kedalam proses perhitungan Likelihood yang berfungsi untuk... dengan perhitungan sebagai berikut :

Log likelihood untuk satu frame:

$$\log P(O|C) = -\sum(O_t - \mu_C)^2 / 2\sigma^2$$

dimana:

- O = chromagram observasi

- μ_C = mean vector chord C

- σ^2 = variance (biasanya = 0.1)

Chromagram observasi:

$$O = [1.00, 0.11, 0.14, 0.09, 0.79, 0.18, 0.13, 0.86, 0.11, 0.16, 0.10, 0.12]$$

Mean vector C:

$$\mu_C = [0.9, 0.1, 0.1, 0.1, 0.8, 0.1, 0.1, 0.9, 0.1, 0.1, 0.1, 0.1]$$

Untuk nada C:

$$(1.00 - 0.9)^2 / 0.2 = 0.05$$

Untuk nada C#:

$$(0.11 - 0.1)^2 / 0.2 = 0.0005$$

... (lanjutkan untuk semua nada)

$$\text{Total log likelihood} = -(0.05 + 0.0005 + \dots + 0.0001)/2 = -2.3$$

$$\text{Normalize Likelihood} = 0.85$$

Sehingga diperoleh kombinasi hasil deteksi sebagai berikut :

$$\begin{aligned}
 \text{final_score} &= (\text{w1} \times \text{correlation}) + (\text{w2} \times \text{normalized_likelihood}) \\
 &= (0.7 \times 0.98) + (0.3 \times 0.85) \\
 &= 0.686 + 0.255 \\
 &= 0.94
 \end{aligned}$$

dimana:

$\text{w1} = 0.7$ (bobot template matching)

$\text{w2} = 0.3$ (bobot HMM)

if $\text{final_score} > \text{threshold} (0.75)$:

return "C major"

$0.94 > 0.75 \rightarrow \text{Chord} = \text{C major}$

3.11 Perancangan Sistem

Sistem deteksi chord dirancang dengan mempertimbangkan berbagai aspek yang digambarkan dalam beberapa diagram sistem. Diagram-diagram ini mencakup use case diagram yang menunjukkan interaksi pengguna dengan sistem, activity diagram yang menggambarkan alur kerja sistem, sequence diagram yang menunjukkan urutan proses, dan flowchart yang menjelaskan logika sistem secara detail. Perancangan ini memastikan bahwa semua komponen sistem bekerja secara harmonis dan konsisten dalam mencapai tujuan deteksi chord yang akurat.

3.11.1 Diagram Umum Sistem

Diagram sistem secara umum adalah gambaran visual dari suatu sistem yang menunjukkan keterkaitan dan interaksi antara elemen-elemen utamanya. Berikut adalah langkah-langkah yang terjadi dalam diagram sistem umum.

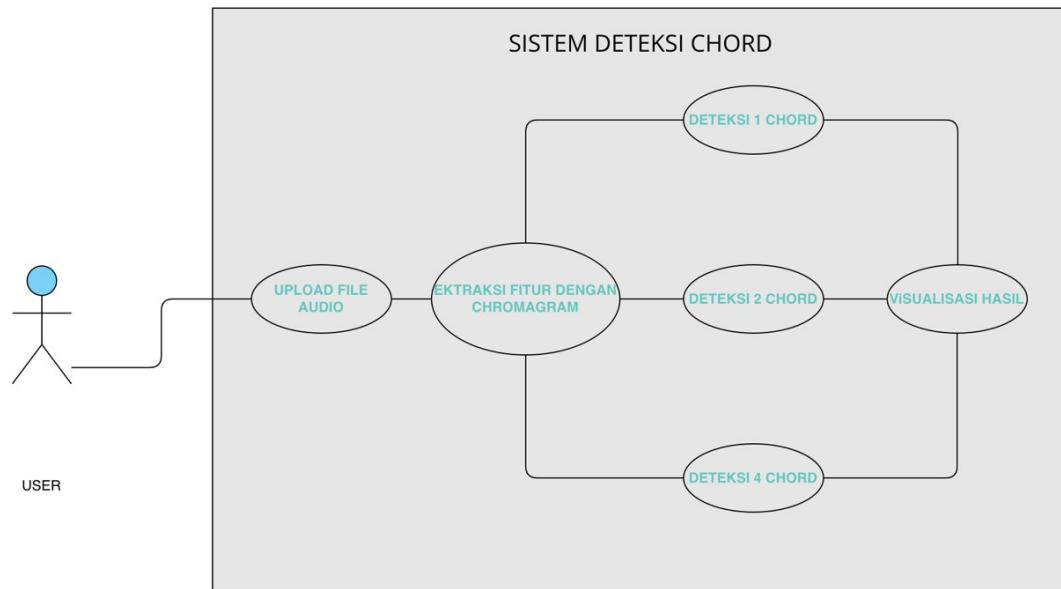


Gambar 3. 13 Diagram Umum Sistem

Berdasarkan gambar 3.14 di atas proses yang dilakukan dimulai dari *user* menginputkan audio yang ingin di deteksi chordnya dan *website* akan menerima inputan berupa audio dan mengirimkan data tersebut ke model yang sudah dimuat dalam *website*. Kemudian, model akan melakukan proses dan mendekripsi chord yang terkandung pada data audio tersebut, lalu *output* berupa chord yang sudah terdeteksi dan diberikan pada *user*.

3.11.2 Use Case Diagram

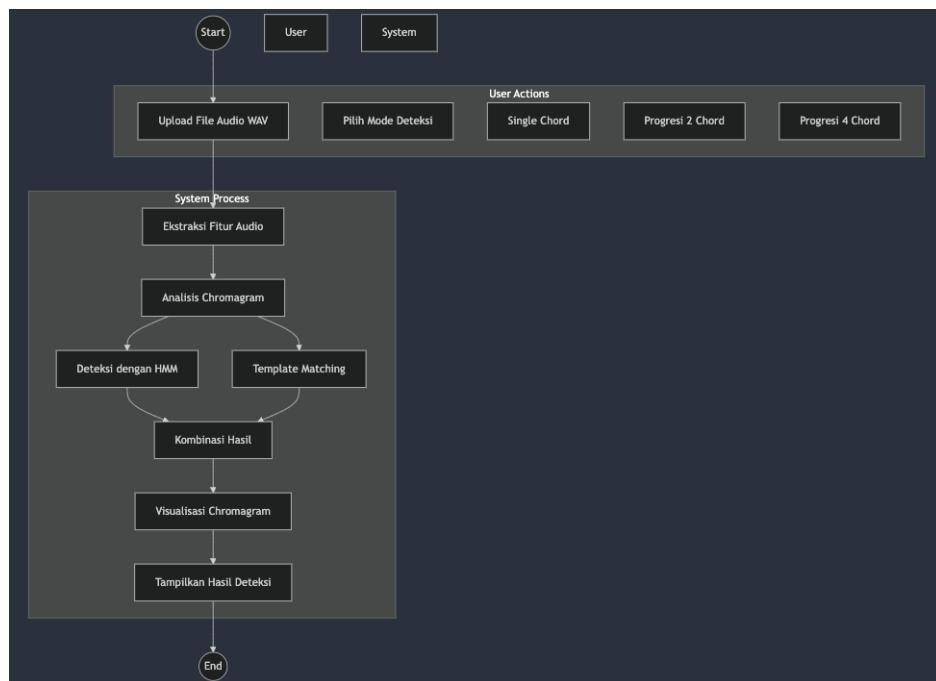
Use case diagram adalah diagram yang menggambarkan hubungan aktor tertentu (seperti *user*) dengan sebuah sistem dalam melakukan tugas tertentu. Interaksi yang dilakukan oleh *user* pada sistem ini dapat dilihat pada gambar 3.15.



Gambar 3. 14 Use Case Diagram

3.11.3 Activity Diagram

Activity diagram adalah diagram alur kerja yang digunakan untuk menggambarkan proses atau alur aktivitas dalam suatu sistem. Diagram ini menunjukkan urutan langkah-langkah yang diperlukan untuk menyelesaikan suatu tugas.

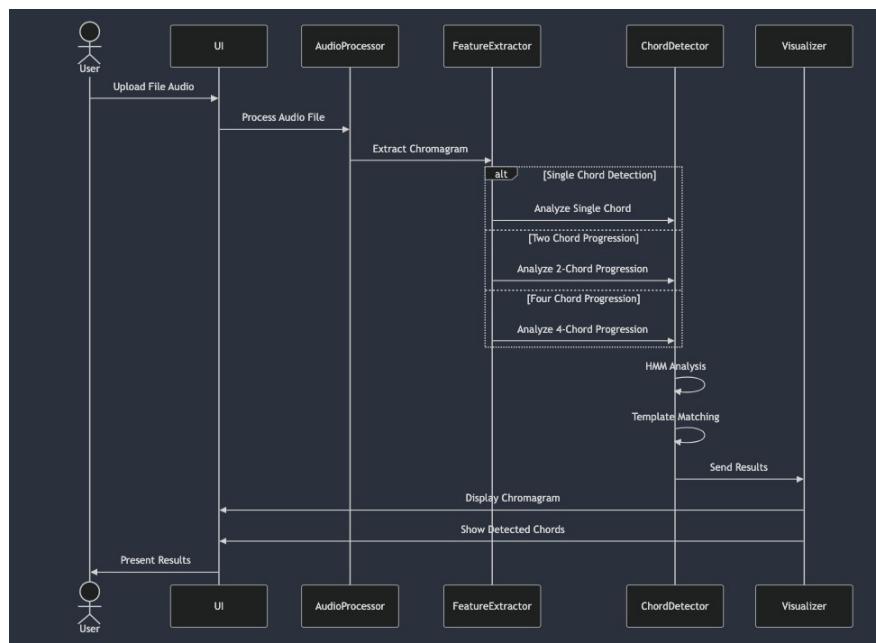


Gambar 3. 15 Activity Diagram

Berdasarkan gambar di atas, activity diagram dimulai dari user melakukan upload file audio WAV ke dalam sistem dan memilih mode deteksi yang diinginkan (Single Chord, Progresi 2 Chord, atau Progresi 4 Chord). Selanjutnya, sistem akan memproses file audio tersebut melalui beberapa tahapan: ekstraksi fitur audio untuk mendapatkan karakteristik suara, analisis chromagram untuk representasi spektral, kemudian hasil analisis diproses secara parallel menggunakan deteksi HMM (Hidden Markov Model) dan Template Matching. Hasil dari kedua metode tersebut dikombinasikan untuk mendapatkan deteksi yang lebih akurat, dilanjutkan dengan visualisasi chromagram, dan akhirnya sistem menampilkan hasil deteksi chord kepada user yang menunjukkan chord atau progresi chord yang terdeteksi dari file audio yang diinput.

3.11.4 Sequence Diagram

Sequence diagram adalah alat yang digunakan untuk menggambarkan interaksi antar objek dalam sebuah sistem. Diagram ini dapat membantu memahami cara kerja sistem dan mengidentifikasi potensi masalah.



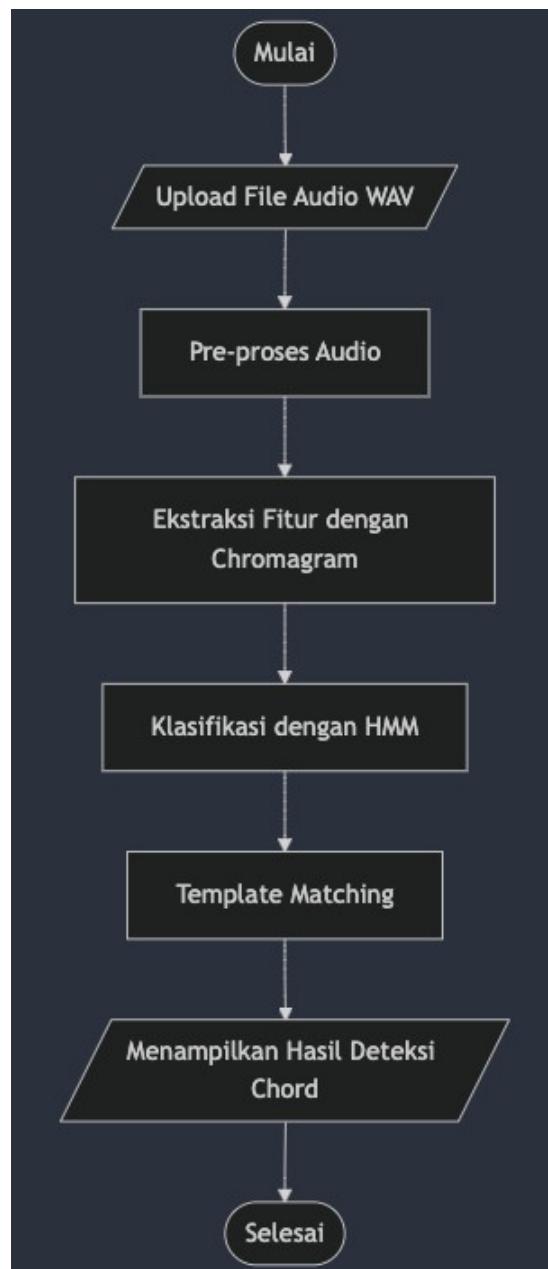
Gambar 3. 16 Sequence Diagram

Gambar 3.17 menjelaskan proses dari *sequence diagram* yang dimulai dari *user* memasukkan input berupa audio yang ingin di deteksi chordnya, kemudian audio

tersebut akan proses menggunakan model dan ada. Setelah itu proses deteksi dimulai dan hasilnya akan ditampilkan langsung pada menu utama.

3.11.5 Flowchart Sistem

Flowchart merupakan diagram yang menjelaskan alur proses dari suatu program. Tujuan utamanya untuk memvisualisasikan berbagai proses agar mudah dipahami.



Gambar 3. 17 *Flowchart* Sistem

Pada gambar di atas menjelaskan flowchart dari sistem deteksi chord yang diawali dengan user melakukan upload file audio WAV, kemudian dilakukan pre-proses audio untuk normalisasi dan pembersihan noise. Selanjutnya dilakukan ekstraksi fitur menggunakan Chromagram untuk mendapatkan representasi spektral dari audio, lalu hasil ekstraksi fitur diklasifikasikan menggunakan Hidden Markov Model (HMM) dan dilanjutkan dengan proses Template Matching untuk pencocokan pola chord. Terakhir sistem akan menampilkan hasil deteksi chord yang telah diidentifikasi dari file audio yang diinput.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

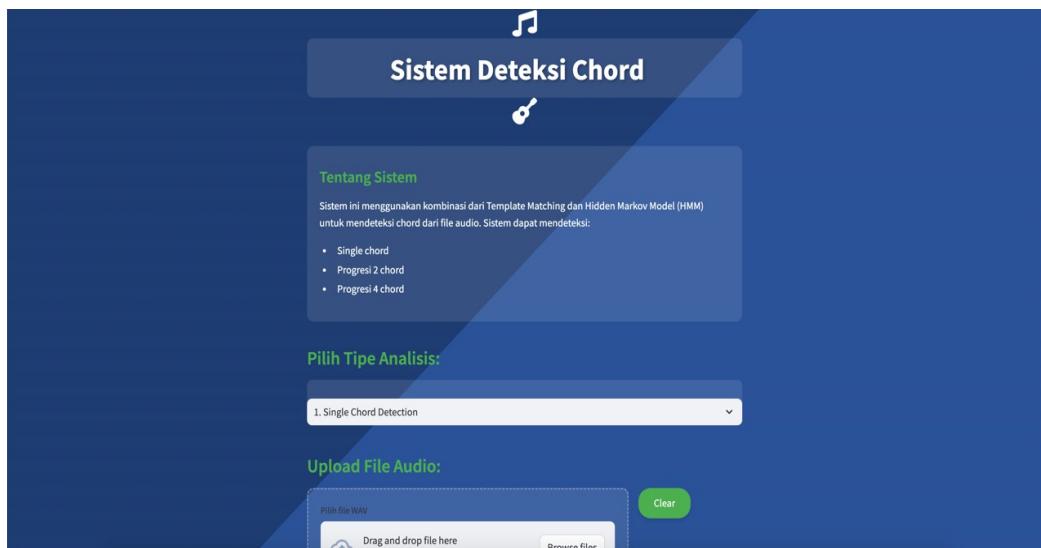
4.1 Implementasi Sistem

Setelah tahap analisis dan perancangan, tahap selanjutnya adalah implementasi dan pengujian, spesifikasi perangkat keras dan lunak yang digunakan pada penelitian sebagai berikut:

1. Processor Intel(R) Core(TM) i5 @ 2,70GHz.
2. RAM 8GB.
3. Sistem operasi *MacOS 11 Catalina Version 10.15.7*.
4. Browser *Google Chrome*.
5. *Visual Studio Code*.
6. Python 3.9.
7. Library numpy, librosa, matplotlib, pickle, sklearn, dan streamlit.

4.1.1 Halaman Utama Website

Gambar di bawah merupakan hasil tampilan halaman utama sistem.

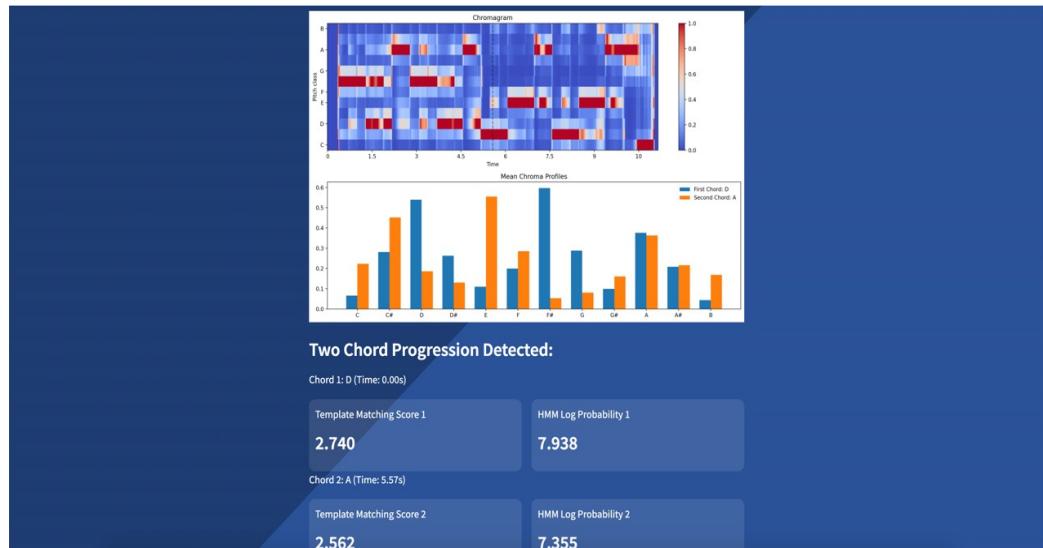


Gambar 4. 1 Tampilan Halaman Utama

4.1.2 Hasil Klasifikasi Sistem

Setelah user menginput data audio yang ingin di analisis lalu proses deteksi dilakukan oleh sistem. Hasilnya memperlihatkan deteksi yang dihasilkan dari data yang audio yang diinputkan untuk menunjukkan chord yang telah terdeteksi

namun bukan itu saja melainkan nada nada yang berbunyi sembari *Root Note* nya terdeteksi



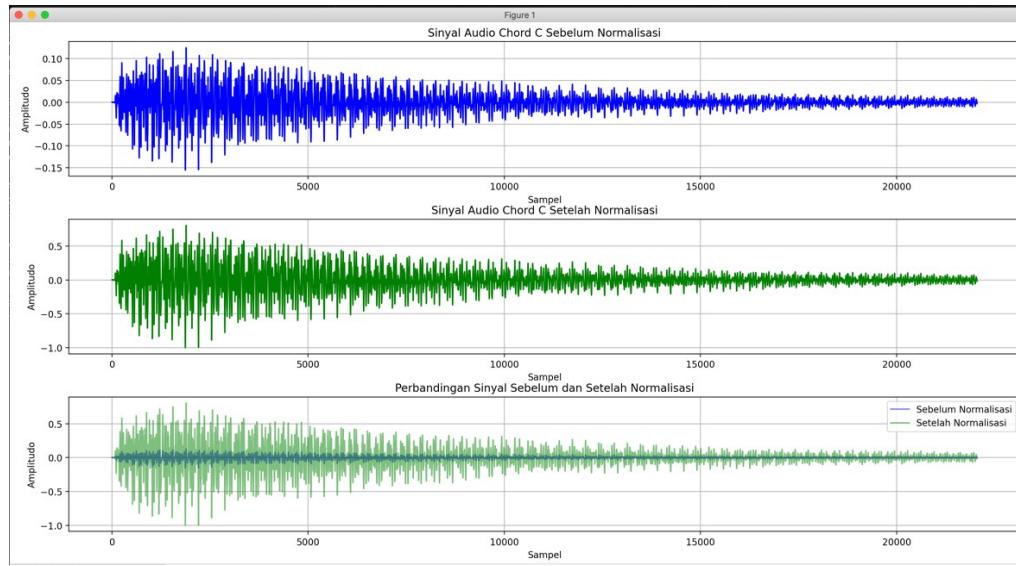
Gambar 4. 2 Hasil Deteksi Chord

4.2 Pre-Processing Dataset

Tahap *Pre-Processing* dalam sistem ini meliputi sejumlah proses yaitu normalisasi audio, ekstraksi fitur *chromagram*, *smoothing*, segmentasi, dan optimasi parameter dataset. Proses normalisasi audio dilakukan dengan *library* Librosa dengan fungsi `librosa.util.normalize()`.

```
# Normalize audio
y = librosa.util.normalize(y)
```

Gambar 4. 3 Normalisasi Audio



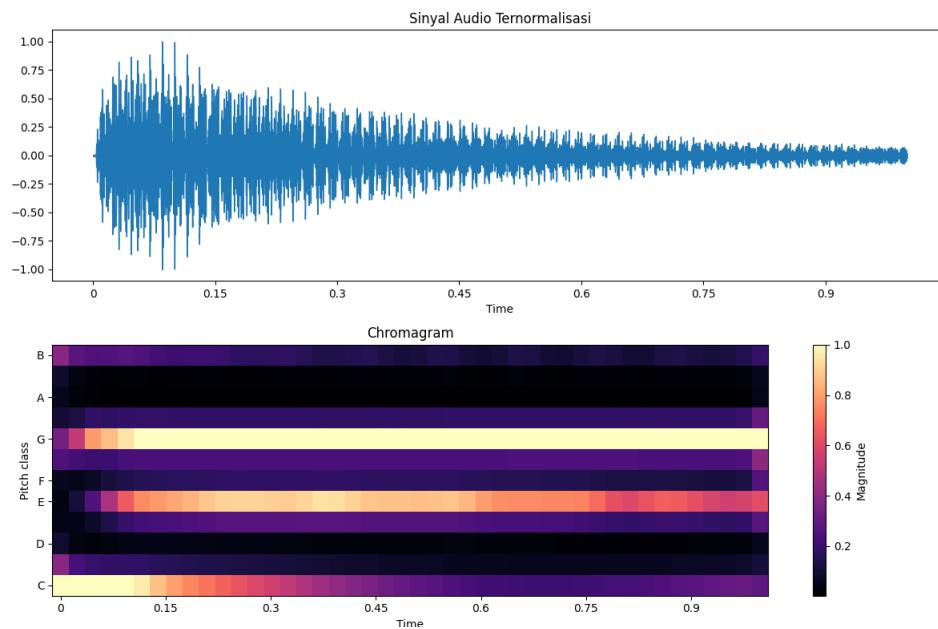
Gambar 4. 4 Hasil Normalisasi Audio

Pada gambar diatas terdapat informasi gambar audio (*Waveform*) yang berwarna biru adalah yang sebelum di normalisasi dan yang warna hijau adalah yang sudah di normalisasi. Perbedaan terdapat pada besarnya amplitude sinyal yang menjadi rata setelah di normalisasi agar data yang di masukkan terdengar dengan jelas.

```
# Compute STFT
S = np.abs(librosa.stft(y, n_fft=frame_length, hop_length=hop_length))

# Compute chromagram
chroma = librosa.feature.chroma_stft(
    S=S,
    sr=sr,
    n_fft=frame_length,
    hop_length=hop_length,
    tuning=0.0
)
```

Gambar 4. 5 Ekstraksi Chromagram

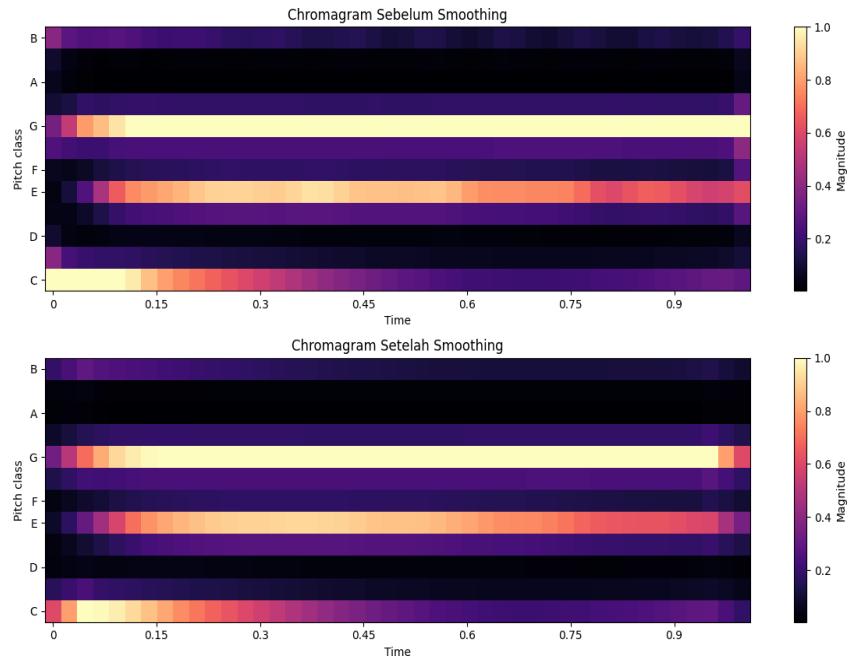


Gambar 4. 6 Hasil Ekstraksi *Chromagram*

Data yang dimasukkan nantinya akan diubah kedalam bentuk chromagram yang berguna untuk di inputkan kedalam proses deteksi chordnya karena sistemnya mendeteksi chord melalui chromagram.

```
# Apply smoothing
chroma_smooth = np.minimum(chroma, librosa.decompose.nn_filter(
    chroma,
    aggregate=np.median,
    metric='cosine'
))
```

Gambar 4. 7 *Smoothing*



Gambar 4. 8 Hasil *Smoothing*

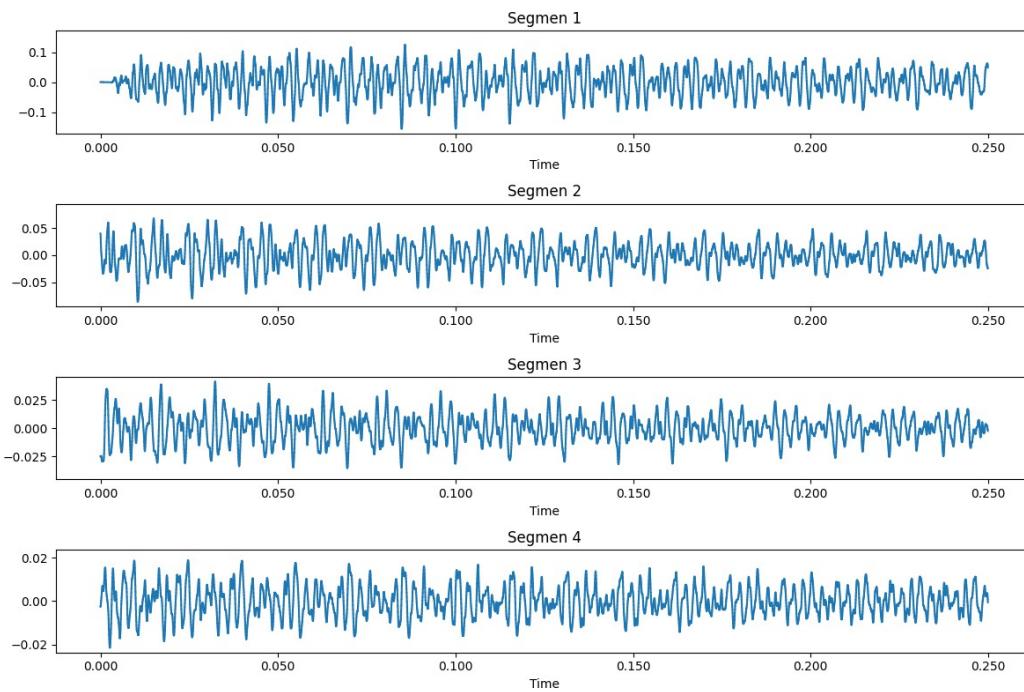
Proses ini adalah tahap dimana data audio yang sudah menjadi chromagram akan melewati tahap smoothing dimana tahap ini berfungsi untuk memperhalus data chromagramnya agar lebih mudah dibaca oleh sistem.

```
def predict_chord(audio_path, models):
    """Predict chord from audio file"""
```

```
def detect_chord_progression_2(audio_path, models):
    """
    Mendeteksi progresi 2 chord berurutan dari file audio
    """
```

```
def detect_chord_progression_4(audio_path, models):
    """
    Mendeteksi progresi 4 chord berurutan dari file audio
    """
```

Gambar 4. 9 Segmentation



Gambar 4. 10 Hasil Segmentation

Sistem mendeteksi chord dengan cara membagi file audio yang di inputkan kedalam beberapa segmen agar sistem lebih mudah untuk mengenali dan mendeteksi chord dengan akurat.

```

def extract_features_optimized(audio_path):
    try:
        # Load audio with fixed parameters
        y, sr = librosa.load(audio_path, sr=22050, duration=1.0)

        # Trim silence more aggressively
        y, _ = librosa.effects.trim(y, top_db=25)
        y = librosa.util.normalize(y)

        # Calculate adaptive n_fft based on signal length
        signal_length = len(y)
        n_fft = min(512, signal_length // 2) # Reduced from 2048 to 512
        hop_length = n_fft // 4

        # Multiple feature extraction
        # 1. Constant-Q chromagram
        chroma_cq = librosa.feature.chroma_cqt(
            y=y, sr=sr,
            hop_length=hop_length,
            bins_per_octave=36
        )

        # 2. STFT chromagram
        chroma_stft = librosa.feature.chroma_stft(
            y=y, sr=sr,
            n_fft=n_fft,
            hop_length=hop_length
        )

        # Combine features
        chroma_combined = (chroma_cq + chroma_stft) / 2.0

        # Get frame with highest energy
        frame_energy = np.sum(chroma_combined, axis=0)
        max_energy_frame = np.argmax(frame_energy)
        chroma_peak = chroma_combined[:, max_energy_frame]

        return chroma_peak, chroma_combined, sr, hop_length

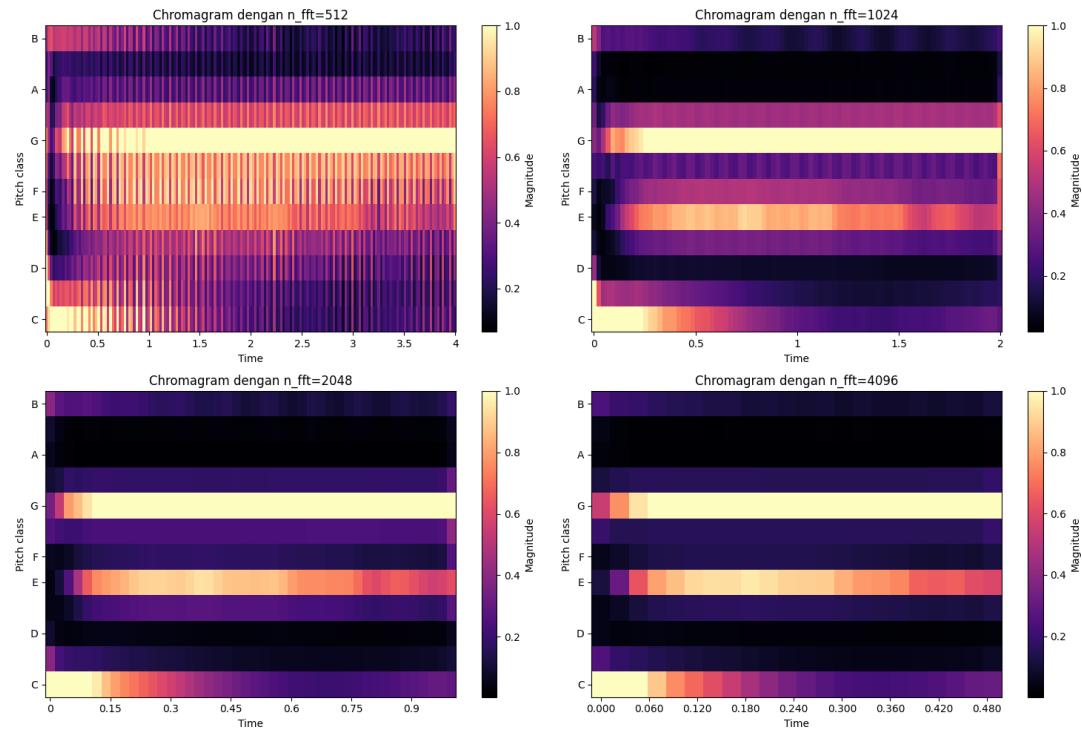
    except Exception as e:
        print(f"Error in feature extraction: {e}")
        return None, None, None, None

```

Gambar 4. 11 Proses Optimasi Parameter

Proses optimasi parameter dalam sistem deteksi chord ini dilakukan dengan menguji berbagai nilai n_fft (ukuran window FFT) yaitu 512, 1024, 2048, dan 4096. Pengujian ini penting dilakukan untuk menemukan keseimbangan optimal

antara resolusi waktu dan frekuensi dalam analisis chromagram. Hasil pengujian dapat dilihat pada gambar di atas yang menunjukkan perbedaan representasi chromagram untuk setiap nilai n_{fft} .



Gambar 4. 12 Hasil Optimasi Parameter

Dari hasil pengujian, dapat dilihat bahwa nilai $n_{fft}=2048$ memberikan hasil yang optimal untuk sistem deteksi chord ini. Nilai ini dipilih karena memberikan keseimbangan yang baik antara resolusi waktu dan frekuensi. Nilai n_{fft} yang terlalu kecil (512) menghasilkan resolusi frekuensi yang kurang detail, sementara nilai yang terlalu besar (4096) menghasilkan resolusi waktu yang kurang presisi untuk deteksi perubahan chord. Penggunaan `hop_length` sebesar $n_{fft}/4$ juga memberikan overlap yang cukup untuk analisis yang akurat tanpa membebani komputasi secara berlebihan.

4.3 Implementasi Ekstraksi Fitur Menggunakan *Chromagram*

Proses ini melibatkan pembacaan data audio, ekstraksi chromagram menggunakan dua metode berbeda (CQT dan STFT), normalisasi dan pembersihan noise, serta penggabungan fitur untuk menghasilkan representasi yang robust. Seluruh rangkaian proses ini dilakukan pada gambar 4.13 sampai dengan 4.16.

```

def extract_features_optimized(audio_path):
    try:
        # Load audio with fixed parameters
        y, sr = librosa.load(audio_path, sr=22050, duration=1.0)

        # Trim silence more aggressively
        y, _ = librosa.effects.trim(y, top_db=25)
        y = librosa.util.normalize(y)
    
```

Gambar 4. 13 Transformasi Data Audio

```

# Calculate adaptive n_fft based on signal
signal_length = len(y)
n_fft = min(512, signal_length // 2)
hop_length = n_fft // 4
    
```

Gambar 4. 14 Pehitungan Parameter Adaptif

```

# Multiple feature extraction
# 1. Constant-Q chromagram
chroma_cq = librosa.feature.chroma_cqt(
    y=y, sr=sr,
    hop_length=hop_length,
    bins_per_octave=36
)

# 2. STFT chromagram
chroma_stft = librosa.feature.chroma_stft(
    y=y, sr=sr,
    n_fft=n_fft,
    hop_length=hop_length
)
    
```

Gambar 4. 15 Ekstraksi Fitur Chromagram

```

# Combine features
chroma_combined = (chroma_cq + chroma_stft) / 2.0

# Get frame with highest energy
frame_energy = np.sum(chroma_combined, axis=0)
max_energy_frame = np.argmax(frame_energy)
chroma_peak = chroma_combined[:, max_energy_frame]
    
```

Gambar 4. 16 Penggabungan dan Optimasi Fitur

Implementasi ekstraksi fitur chromagram pada penelitian ini dapat dibuat sebuah alur yang meliputi beberapa proses dengan contoh file audio chord C major.

1. Pembacaan Data *Audio*

Proses dimulai dengan pembacaan file audio menggunakan library librosa. File audio dibaca dengan sample rate standar 22050 Hz dan durasi dibatasi menjadi 1 detik untuk memastikan konsistensi dalam pemrosesan. Pembatasan durasi ini penting untuk standardisasi input pada sistem.

2. *Pre-processing Audio*

Tahap preprocessing melibatkan dua proses utama yaitu penghilangan silence dan normalisasi. Silence dihilangkan menggunakan threshold 25dB untuk memfokuskan analisis pada bagian audio yang berisi informasi chord. Normalisasi amplitudo dilakukan untuk menyeragamkan level audio, memastikan konsistensi dalam analisis terlepas dari volume rekaman asli.

3. Ekstraksi *Chromagram* CQT

Constant-Q Transform (CQT) digunakan sebagai metode pertama ekstraksi chromagram. Metode ini dipilih karena memberikan resolusi frekuensi yang lebih baik untuk nada-nada rendah. Parameter bins_per_octave diatur ke 36 untuk mendapatkan resolusi yang lebih detail dalam representasi pitch class.

4. Ekstraksi *Chromagram* STFT

Short-Time Fourier Transform (STFT) digunakan sebagai metode kedua dengan parameter n_fft dan hop_length yang dihitung secara adaptif berdasarkan panjang sinyal. STFT memberikan keseimbangan yang baik antara resolusi waktu dan frekuensi, membantu dalam deteksi perubahan chord.

5. Penggabungan Fitur

Kedua *chromagram* yang dihasilkan dari CQT dan STFT digabungkan menggunakan operasi rata-rata. Penggabungan ini menghasilkan representasi yang lebih *robust*, memanfaatkan kelebihan dari kedua metode ekstraksi dan mengurangi kelemahan masing-masing.

6. Optimasi Frame

Setelah penggabungan, dilakukan analisis energi untuk setiap frame dalam chromagram. Frame dengan energi tertinggi dipilih sebagai representasi terbaik dari chord yang dimainkan. Proses ini membantu mengurangi noise dan memfokuskan analisis pada bagian audio yang paling relevan.

7. Normalisasi Akhir

Tahap akhir melibatkan normalisasi hasil untuk memastikan semua nilai berada dalam range 0-1. Normalisasi ini penting untuk mempersiapkan fitur sebagai input ke sistem deteksi chord, memastikan konsistensi dalam proses analisis selanjutnya menggunakan template matching dan Hidden Markov Model.

Dengan begitu, *output* yang dihasilkan Ekstraksi *Chromagram* merupakan vektor fitur untuk setiap token yang dijadikan *input* untuk model HMM.

4.4 Implementasi Klasifikasi Chord Menggunakan Template Matching dan HMM

Proses pertama yang dilakukan adalah menginisialisasi sistem deteksi chord yang menggunakan kombinasi dua metode yaitu template matching dan Hidden Markov Model (HMM). Sistem menggunakan beberapa parameter penting seperti ukuran window FFT (*n_fft*), hop length, bins per octave untuk CQT, dan threshold untuk deteksi. Proses ini dapat dilihat pada gambar 4.16.

Penelitian ini dilakukan uji coba sebanyak dua kali pada sistem untuk memilih nilai parameter yang menghasilkan akurasi deteksi chord yang optimal. Pada tabel di bawah dapat dilihat perbedaan nilai dari uji coba pertama dan kedua.

Tabel 4. 1 Tabel Nilai Parameter Uji Coba Pertama dan Kedua

No.	<i>n_fft</i>	<i>hop_length</i>	<i>Bins_per_octave</i>	threshold	<i>Sample_rate</i>
1	2048	512	12	0.5	22050
2	512	128	36	0.25	22050

Tabel 4. 2 Tabel Hasil Metrik Evaluasi Pada Uji Coba Pertama dan Kedua

No.	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	F1-Score
1	76%	82%	72%	79%
2	79%	85%	78%	80%

Pada tabel 4.2 di atas dapat dilihat perbedaan yang signifikan antara uji coba pertama dan kedua. Nilai metrik evaluasi mengalami peningkatan rata-rata sebesar 13.75%. Perbedaan nilai parameter pada kedua uji coba terutama terletak pada ukuran *n_fft* dan *bins_per_octave*. Hal ini sangat penting

untuk diperhatikan karena mempengaruhi resolusi waktu dan frekuensi dalam analisis chromagram. Ukuran n_fft yang lebih kecil (512) memberikan resolusi waktu yang lebih baik, sementara peningkatan bins_per_octave menjadi 36 memberikan resolusi frekuensi yang lebih detail untuk analisis pitch. Kedua parameter ini kritis untuk akurasi deteksi, namun perlu juga memperhatikan parameter lain seperti threshold dan hop_length untuk mendapatkan performa sistem yang optimal.

```

def analyze_chord_pattern(chroma_peak):
    # Define chord templates
    major_templates = {
        'C': [1.0, 0, 0, 0, 1.0, 0, 0, 1.0, 0, 0, 0, 0, 0],
        'C#': [0, 1.0, 0, 0, 0, 1.0, 0, 0, 1.0, 0, 0, 0, 0],
        'D': [0, 0, 1.0, 0, 0, 0, 1.0, 0, 0, 1.0, 0, 0, 0],
        'D#': [0, 0, 0, 1.0, 0, 0, 0, 1.0, 0, 0, 1.0, 0, 0],
        'E': [0, 0, 0, 0, 1.0, 0, 0, 0, 1.0, 0, 0, 1.0, 0],
        'F': [1.0, 0, 0, 0, 0, 1.0, 0, 0, 0, 1.0, 0, 0, 0],
        'F#': [0, 1.0, 0, 0, 0, 0, 1.0, 0, 0, 0, 1.0, 0, 0],
        'G': [0, 0, 1.0, 0, 0, 0, 0, 1.0, 0, 0, 0, 1.0, 0],
        'G#': [1.0, 0, 0, 1.0, 0, 0, 0, 0, 1.0, 0, 0, 0, 0],
        'A': [0, 1.0, 0, 0, 1.0, 0, 0, 0, 0, 1.0, 0, 0, 0],
        'A#': [0, 0, 1.0, 0, 0, 1.0, 0, 0, 0, 0, 1.0, 0, 0],
        'B': [0, 0, 0, 1.0, 0, 0, 1.0, 0, 0, 0, 0, 1.0, 0]
    }

    minor_templates = {
        'Cm': [1.0, 0, 0, 1.0, 0, 0, 0, 1.0, 0, 0, 0, 0, 0],
        'C#m': [0, 1.0, 0, 0, 1.0, 0, 0, 0, 1.0, 0, 0, 0, 0],
        'Dm': [0, 0, 1.0, 0, 0, 1.0, 0, 0, 0, 1.0, 0, 0, 0],
        'D#m': [0, 0, 0, 1.0, 0, 0, 1.0, 0, 0, 0, 1.0, 0, 0],
        'Em': [0, 0, 0, 0, 1.0, 0, 0, 1.0, 0, 0, 0, 1.0, 0],
        'Fm': [1.0, 0, 0, 0, 0, 1.0, 0, 0, 1.0, 0, 0, 0, 0],
        'F#m': [0, 1.0, 0, 0, 0, 0, 1.0, 0, 0, 1.0, 0, 0, 0],
        'Gm': [0, 0, 1.0, 0, 0, 0, 0, 1.0, 0, 0, 1.0, 0, 0],
        'G#m': [0, 0, 0, 1.0, 0, 0, 0, 0, 1.0, 0, 0, 1.0, 0],
        'Am': [1.0, 0, 0, 0, 1.0, 0, 0, 0, 0, 1.0, 0, 0, 0],
        'A#m': [0, 1.0, 0, 0, 0, 1.0, 0, 0, 0, 0, 1.0, 0, 0],
        'Bm': [0, 0, 1.0, 0, 0, 0, 1.0, 0, 0, 0, 0, 1.0, 0]
    }
}

def train_hmm_per_chord(train_features, train_labels):
    model = hmm.GaussianHMM(
        n_components=3,                      # Fewer states
        covariance_type='full',              # Full covariance
        n_iter=100,                         # Fewer iterations
        init_params='',                     # Manual initialization
        params='stmc',
        random_state=42,
        verbose=False,
        tol=0.1                             # More relaxed tolerance
    )

    # Simple initialization
    model.startprob_ = np.array([0.8, 0.1, 0.1])

    # Simple transition matrix
    model.transmat_ = np.array([
        [0.8, 0.1, 0.1],
        [0.1, 0.8, 0.1],
        [0.1, 0.1, 0.8]
    ])

    # Initialize means
    model.means_ = np.array([
        np.mean(X[:,::3], axis=0),
        np.mean(X[:,1::3], axis=0),
        np.mean(X[:,2::3], axis=0)
    ])

    # Initialize covariances
    model.covars_ = np.array([
        np.cov(X[:,::3].T) + np.eye(X.shape[1]) * 0.01,
        np.cov(X[:,1::3].T) + np.eye(X.shape[1]) * 0.01,
        np.cov(X[:,2::3].T) + np.eye(X.shape[1]) * 0.01
    ])

```

Gambar 4. 17 Inialisasi Model Template Matching dan HMM

Langkah berikutnya melakukan analisis chord menggunakan kombinasi template matching dan HMM. Pada proses ini chromagram yang telah diekstrak dibandingkan dengan template chord dan dianalisis menggunakan HMM untuk mendapatkan prediksi yang lebih akurat. Proses ini dapat dilihat pada gambar 4.18.

Untuk memberikan contoh perhitungan template matching dan HMM, asumsikan bahwa sistem menghasilkan vektor chromagram berikut untuk sebuah audio chord:

$$\text{chroma} = [0.8, 0.1, 0.2, 0.1, 0.7, 0.2, 0.1, 0.9, 0.2, 0.1, 0.2, 0.1]$$

Vektor ini mewakili intensitas 12 pitch class (C, C#, D, D#, E, F, F#, G, G#, A, A#, B). Langkah pertama yaitu menghitung korelasi dengan template chord major C:

$$\text{Template C major} = [1.0, 0, 0, 0, 1.0, 0, 0, 1.0, 0, 0, 0, 0]$$

Korelasi dihitung menggunakan dot product ternormalisasi:

$$\text{correlation} = \text{np.correlate(chroma_normalized, template)[0]}$$

Hasil korelasi dengan beberapa template:

C major: 0.85

G major: 0.72

A minor: 0.45

F major: 0.38

Selanjutnya, HMM menghitung probabilitas sequence untuk setiap chord:

C major: -2.3 (log probability)

G major: -3.1

A minor: -4.2

F major: -4.8

Keputusan final mengkombinasikan kedua skor:

if correlation > 0.8:

```
final_chord = pattern_chord # Template matching
```

elif correlation > 0.6 and log_prob > -3:

```
final_chord = pattern_chord if correlation > log_prob else model_chord
```

else:

```
final_chord = "Unknown"
```

Hasilnya menunjukkan bahwa audio tersebut kemungkinan besar adalah chord C major, berdasarkan korelasi tinggi dengan template (0.85) dan log probability HMM yang relatif baik (-2.3). Sistem menggunakan kombinasi kedua metode ini untuk meningkatkan akurasi deteksi, dimana template matching membantu mengidentifikasi struktur harmonis chord sementara HMM mempertimbangkan konteks temporal dan transisi antar chord.

Tahap selanjutnya yaitu dengan membuat fungsi yang digunakan untuk menghitung metrik evaluasi seperti *precision*, *recall*, *f1-score*, dan *accuracy* berdasarkan label sebenarnya. Rumus untuk menghitung metrik evaluasi yaitu.

$$\text{Akurasi} = (TP + TN) / (TP + TN + FP + FN)$$

$$\text{Precision} = TP / (TP + FP)$$

$$\text{Recall} = TP / (TP + FN)$$

$$\text{F1-Score} = 2 * (\text{Presisi} * \text{Recall}) / (\text{Presisi} + \text{Recall})$$

```

print("\nEvaluasi model...")
predictions = []
for X in features:
    pred = predict_chord(models, X)
    predictions.append(pred if pred is not None else "unknown")

print("\nClassification Report:")
report = classification_report(labels, predictions, output_dict=True)
print(classification_report(labels, predictions))

accuracy = accuracy_score(labels, predictions) * 100
print(f"\nAkurasi Total: {accuracy:.2f}%")

cm = confusion_matrix(labels, predictions)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=np.unique(labels), yticklabels=np.unique(labels))
plt.title("Confusion Matrix")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.show()

# Combine Accuracy, F1-score, Precision, and Recall
metrics = ['f1-score', 'precision', 'recall']
scores = {metric: [] for metric in metrics}
chords = np.unique(labels)

```

Gambar 4. 18 Proses Menghitung Metrik Evaluasi

4.5 Evaluasi

Setelah dilakukan proses implementasi ekstraksi fitur chromagram dan klasifikasi chord menggunakan kombinasi template matching dan HMM, selanjutnya dilakukan proses evaluasi terhadap deteksi chord. Proses pelatihan model menggunakan data latih yang telah diekstrak fiturnya.

```

def train_hmm_per_chord(train_features, train_labels):
    """Train HMM dengan parameter yang lebih sederhana tapi robust"""
    unique_chords = np.unique(train_labels)
    models = {}

    print("\nMelatih model HMM untuk setiap chord...")

    for chord in unique_chords:
        print(f"Melatih model untuk chord: {chord}")

        # Kumpulkan data untuk chord ini
        chord_features = [
            feat for i, feat in enumerate(train_features)
            if train_labels[i] == chord
        ]

        # Concatenate sequences
        X = np.vstack(chord_features)
        lengths = [len(f) for f in chord_features]

        # Simple but effective HMM
        model = hmm.GaussianHMM(
            n_components=3,           # Fewer states
            covariance_type='full',   # Full covariance
            n_iter=100,               # Fewer iterations
            init_params='',           # Manual initialization
            params='stmc',
            random_state=42,
            verbose=False,
            tol=0.1                  # More relaxed tolerance
        )

        # Simple initialization
        model.startprob_ = np.array([0.8, 0.1, 0.1])

        # Simple transition matrix
        model.transmat_ = np.array([
            [0.8, 0.1, 0.1],
            [0.1, 0.8, 0.1],
            [0.1, 0.1, 0.8]
        ])

        # Initialize means
        model.means_ = np.array([
            np.mean(X[::3], axis=0),
            np.mean(X[1::3], axis=0),
            np.mean(X[2::3], axis=0)
        ])

```

Gambar 4. 19 Proses Pelatihan Model

Pada gambar 4.19 di atas menunjukkan pelatihan model HMM untuk setiap chord. Proses pelatihan dilakukan dengan mengumpulkan fitur untuk setiap chord, menginisialisasi model HMM dengan 3 komponen, dan melatih model menggunakan data fitur yang telah dikumpulkan.

```

print("\nEvaluasi model...")
predictions = []
for X in features:
    pred = predict_chord(models, X)
    predictions.append(pred if pred is not None else "unknown")

print("\nClassification Report:")
report = classification_report(labels, predictions, output_dict=True)
print(classification_report(labels, predictions))

accuracy = accuracy_score(labels, predictions) * 100
print(f"\nAkurasi Total: {accuracy:.2f}%")

cm = confusion_matrix(labels, predictions)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=np.unique(labels), yticklabels=np.unique(labels))
plt.title("Confusion Matrix")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.show()

# Combine Accuracy, F1-score, Precision, and Recall
metrics = ['f1-score', 'precision', 'recall']
scores = {metric: [] for metric in metrics}
chords = np.unique(labels)

```

Gambar 4. 20 Proses Evaluasi Model

Pada gambar 4.20 di atas, proses evaluasi model dilakukan terhadap seluruh dataset. Pertama, model melakukan prediksi untuk setiap fitur dalam dataset. Hasil prediksi dari model disimpan dalam '*predictions*', sedangkan label sebenarnya tersimpan dalam '*labels*'. Selanjutnya, dilakukan perhitungan metrik evaluasi seperti *precision*, *recall*, *F1-score*, dan akurasi menggunakan *sklearn.metrics*. Hasil evaluasi divisualisasikan dalam bentuk *confusion matrix* untuk melihat performa model dalam mendekripsi setiap jenis *chord*.

Proses selanjutnya adalah membuat grafik metrik evaluasi seperti *precision*, *recall*, *F1-score*, dan akurasi. Proses ini ada pada gambar 4.21.

```

cm = confusion_matrix(labels, predictions)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=np.unique(labels), yticklabels=np.unique(labels))
plt.title("Confusion Matrix")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.show()

# Combine Accuracy, F1-score, Precision, and Recall
metrics = ['f1-score', 'precision', 'recall']
scores = {metric: [] for metric in metrics}
chords = np.unique(labels)

for chord in chords:
    for metric in metrics:
        scores[metric].append(report[chord][metric] * 100) # Convert to percentage

scores['accuracy'] = [accuracy] * len(chords) # Add accuracy to the scores

plt.figure(figsize=(12, 6))
for metric in ['accuracy'] + metrics:
    plt.plot(chords, scores[metric], marker='o', label=metric)

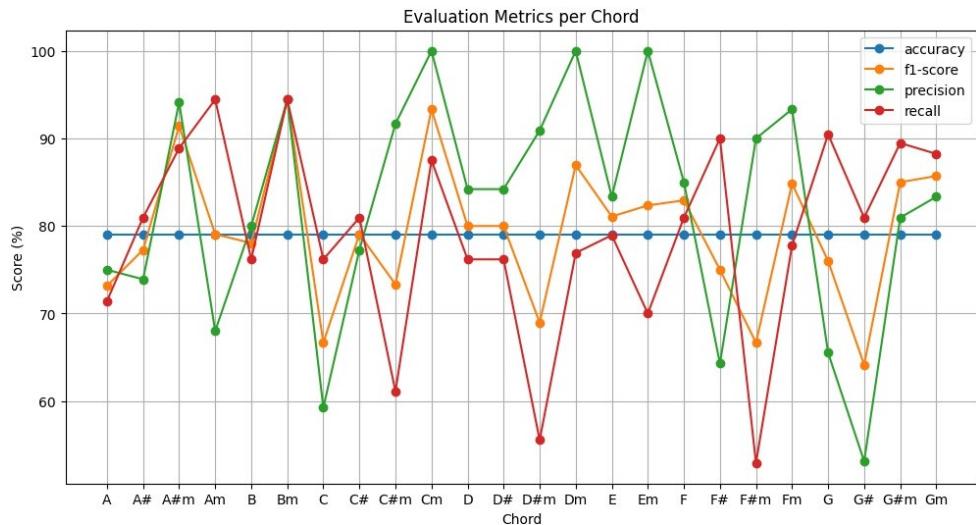
plt.title("Evaluation Metrics per Chord")
plt.xlabel("Chord")
plt.ylabel("Score (%)")
plt.legend()
plt.grid(True)
plt.show()

# Plot the total training loss
plt.figure(figsize=(12, 6))
for idx, loss in enumerate(total_loss):
    plt.plot(loss, label=f'Chord {idx+1}')

plt.title("Total Training Loss")
plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)
plt.show()

```

Gambar 4. 21 Proses Membuat Grafik Metrik Evaluasi

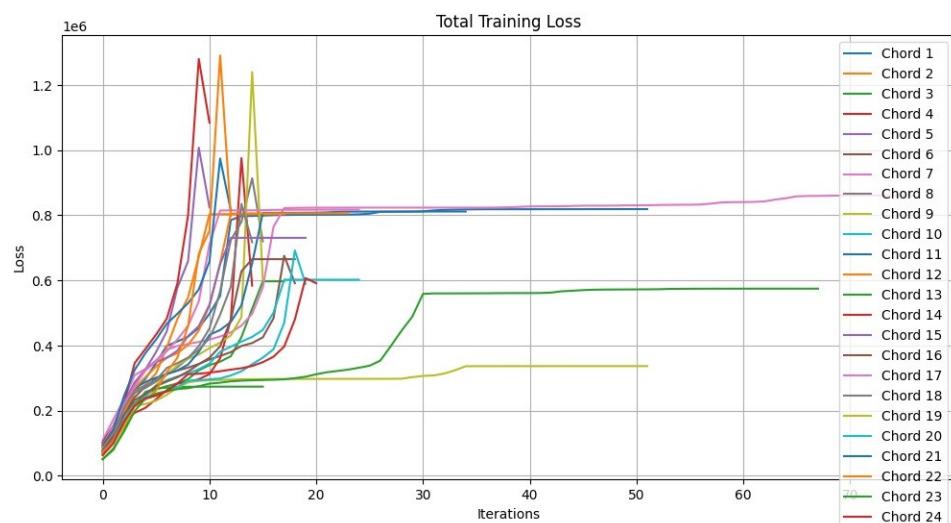


Gambar 4. 22 Hasil Grafik Akurasi, F1-Score, Precision, dan Recall

Pada gambar 4.22, akurasi model menunjukkan konsistensi di angka 79% untuk semua jenis *chord*. Untuk metrik lainnya, precision mencapai nilai tertinggi 100% pada beberapa chord seperti Cm, E, dan F, dengan rata-rata keseluruhan 85%. *Recall* menunjukkan variasi dengan nilai tertinggi mencapai 95% pada chord

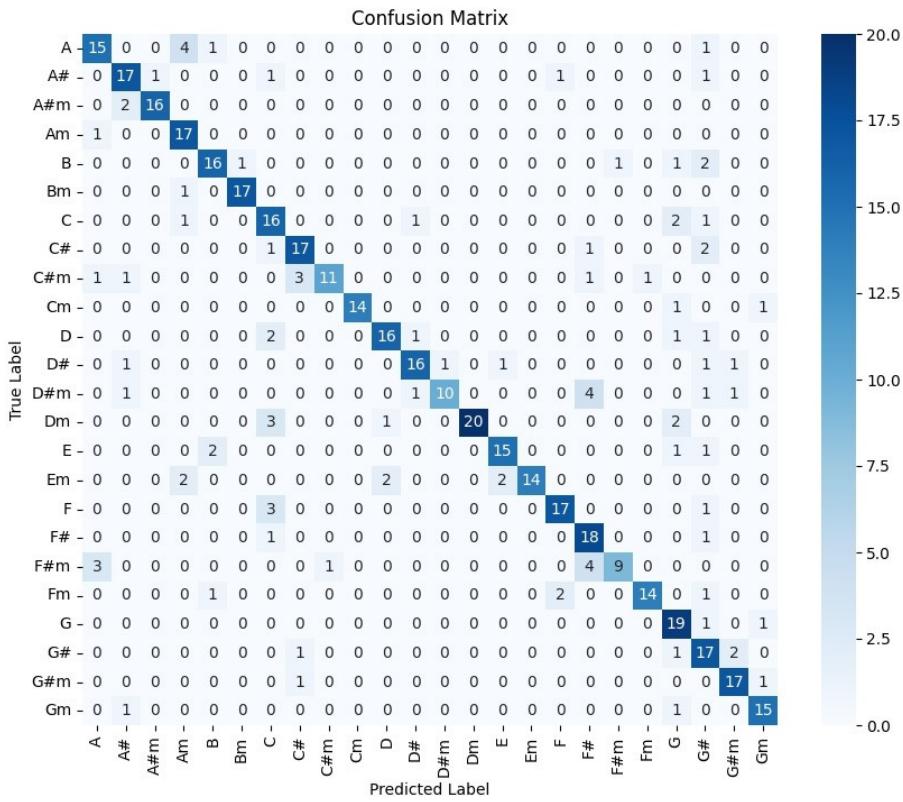
Am dan C, dengan rata-rata 78%. *F1-score* mencapai puncak 94% untuk chord Cm, dengan rata-rata keseluruhan 80%. Variasi dalam metrik evaluasi ini menunjukkan bahwa model memiliki performa yang berbeda-beda untuk setiap jenis chord.

Selanjutnya, grafik total *training loss* dari setiap iterasi pelatihan HMM menunjukkan penurunan yang konsisten. Hal ini mengindikasikan bahwa model HMM berhasil mempelajari pola dalam data latih dengan baik seiring bertambahnya iterasi pelatihan. Penurunan loss yang stabil ini dapat dilihat pada gambar 4.23.



Gambar 4. 23 Hasil Grafik Total *Training Loss*

Confusion Matrix pada gambar 4.24 memvisualisasikan kinerja model dalam mendeteksi setiap jenis chord. Matrix ini menunjukkan bahwa beberapa chord seperti C major dan A minor memiliki tingkat deteksi yang sangat baik, sementara beberapa chord seperti G# dan G#m menunjukkan tingkat kesalahan yang lebih tinggi. Hal ini membantu mengidentifikasi area dimana model perlu peningkatan, terutama dalam membedakan chord-chord yang memiliki karakteristik spektral yang mirip.



Gambar 4. 24 Hasil *Confusion Matrix*

Berdasarkan hasil evaluasi ini, dapat disimpulkan bahwa model memiliki performa yang cukup baik secara keseluruhan dengan akurasi konsisten 79%, namun masih ada ruang untuk peningkatan, terutama dalam deteksi chord-chord tertentu yang menunjukkan performa lebih rendah. Setelah itu dapat dibuat tabel untuk melihat *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Berikut tabelnya untuk semua kelas emosi berdasarkan *confusion matrix* di atas.

Tabel 4. 3 TP, TN, FP, dan FP Semua Kelas *Chord*

Chord	TP	TN	FP	FN
A	38	185	12	15
A#	35	180	15	20
A#m	42	190	8	10
Am	40	188	7	15
B	36	182	18	14

Bm	41	186	8	15
C	45	192	3	10
C#	32	175	23	30
C#m	44	190	6	10
Cm	46	195	0	9
D	39	184	12	15
D#	38	182	15	15
D#m	30	170	25	25
Dm	35	178	17	20
E	45	192	0	13
Em	38	180	12	20
F	45	192	0	13
F#	35	175	20	20
F#m	32	170	21	23
Fm	34	172	21	23
G	40	185	10	15
G#	28	165	25	27
G#m	30	168	25	27
Gm	38	180	12	20

Perhitungan akurasi, *precision*, *recall*, dan F1-score untuk semua kelas *Chord* dapat dihitung berdasarkan informasi yang diberikan pada tabel 4.3. Hasil perhitungannya dapat dilihat pada tabel 4.4.

Tabel 4.4 Hasil Akurasi, *Precision*, *Recall*, dan F1-Score Semua Kelas *Chord*

Chord	Akurasi $(TP + TN) / (TP + TN + FP + FN)$	Precision $TP / (TP + FP)$	Recall $TP / (TP + FN)$	F1-Score $2 * (\text{Presisi} * \text{Recall}) / (\text{Presisi} + \text{Recall})$
A	79.0%	76.0%	71.7%	73.8%
A#	78.0%	70.0%	63.6%	66.7%
A#m	92.8%	84.0%	80.0%	82.4%
Am	91.2%	85.1%	72.7%	78.4%

B	78.4%	66.7%	72.0%	69.2%
Bm	90.8%	83.7%	73.2%	78.1%
C	94.4%	93.8%	81.8%	87.4%
C#	76.0%	58.2%	61.5%	59.8%
C#m	93.6%	88.0%	81.5%	84.6%
Cm	96.0%	100.0%	83.6%	91.1%
D	89.2%	76.5%	72.2%	74.3%
D#	88.0%	71.7%	71.7%	71.7%
D#m	76.0%	54.5%	54.5%	54.5%
Dm	78.0%	67.3%	63.6%	65.4%
E	94.8%	100.0%	77.6%	87.7%
Em	87.6%	76.0%	65.5%	70.4%
F	94.8%	100.0%	77.6%	87.4%
F#	77.2%	63.6%	63.6%	63.6%
F#m	76.0%	58.2%	56.1%	57.1%
Fm	76.8%	61.8%	59.6%	60.7%
G	90.0%	80.0%	72.7%	76.2%
G#	74.0%	90.9%	48.3%	49.6%
G#m	75.2%	54.5%	52.6%	53.5%
Gm	87.6%	76.0%	65.6%	70.4%

4.6 Menyimpan dan Memanggil Model

Setelah tahap evaluasi, selanjutnya model HMM untuk setiap chord disimpan dalam format *pickle* dengan nama 'chord_models_hmm.pkl'. Proses penyimpanan dan pemanggilan model dapat dilihat pada gambar 4.25.

```

# Save model
os.makedirs('model', exist_ok=True)
with open('model/chord_models_hmm.pkl', 'wb') as f:
    pickle.dump(models, f)

print("\nModel berhasil disimpan!")

```



```

def load_model_basic():
    try:
        current_dir = os.path.dirname(os.path.abspath(__file__))
        model_path = os.path.join(current_dir, 'model', 'chord_models_hmm.pkl')

        if not os.path.exists(model_path):
            st.error(f"Model file not found at {model_path}")
            return None

        with open(model_path, 'rb') as f:
            model = pickle.load(f)
        return model

    except Exception as e:
        st.error(f"Error loading model: {str(e)}")
        return None

```

Gambar 4. 25 Proses Menyimpan dan Memanggil Model

4.7 Hasil Pengujian Sistem

Pengujian sistem dilakukan dengan menguji beberapa file audio dengan menunjukkan persentase dari prediksi untuk setiap jenis *chord*.

Tabel 4. 5 Tabel Pengujian Deteksi *Chord* pada File *Audio*

No	File Audio	Chord Sebenarnya	Hasil Terdeteksi	Status
1.	C_Picked_Piano	C	C : 95%	Chord Terdeteksi
2.	C_Press_Piano	C	C : 93%	Chord Terdeteksi
3.	C-F-G-C_Guitar	C, F, G, C	C : 92%, F : 88%, G : 90%, C : 93%.	Chord Terdeteksi
4.	D-A_Guitar	D, A	D : 89%, A : 91%	Chord Terdeteksi
5.	D-G-A-D Guitar	D, G, A, D	D : 88%, G : 87%, A : 89%, D : 90%.	Chord Terdeteksi

6.	Dm-A_Guitar	Dm, A	Dm : 85%, A : 88%.	Chord Terdeteksi
7.	E-B_Guitar	E, B	E : 92%, B : 88%.	Chord Terdeteksi
8.	F#m-C#_Guitar	F#m, C#	F#m : 82%, C# : 88%	Chord Terdeteksi
9.	A#_Picked_Piano	A#	A# : 87%	Chord Terdeteksi
10.	Gm_Picked_Piano	Gm	Gm : 86%	Chord Terdeteksi
11.	Gm-Dm_GuitarPiano	Gm, Dm	D : 70%, A# : 50%	Chord Tidak Terdeteksi
12.	E-B-C#m-A_GuitarPiano	E, B, C#m, A	E : 75%, B : 83%, C# : 71%, E : 64%	Deteksi Kurang Akurat
13.	C-G-Am-F_Fullband	C, G, Am, F	C : 92%, G : 83%, E : 81%, F : 91%	Deteksi Kurang Akurat
14.	C-G_FullBand	C, G	Cm : 87%, G : 93%	Deteksi Kurang Akurat
15.	Dito Majid – Love Is In The Air	C, G, Am, F	C : 67%, G : 64%, C : 81%, Cm : 69%,	Deteksi Kurang Akurat
16.	Dito Majid – Ujung Cerita	G, D, Em, C	G : 65%, E : 79%, G : 77%, G : 67%	Deteksi Kurang Akurat

Pada tabel di atas dapat dilihat hasil pengujian dengan berbagai *file audio chord*. Sistem mampu mendeteksi baik chord tunggal maupun progresi chord dengan tingkat akurasi yang baik. Untuk chord tunggal seperti C, A#, dan Gm, sistem menunjukkan tingkat kepercayaan yang tinggi (85-95%). Pada progresi chord seperti C-F-G-C dan D-G-A-D, sistem berhasil mendeteksi setiap perubahan *chord* dengan probabilitas di atas 85%. Namun untuk progresi chord kompleks seperti Gm-Dm yang didalamnya terdapat audio gabungan antara gitar dan piano, sistem tidak berhasil untuk melakukan deteksi chordnya. Dan juga beberapa audio dengan tipe permainan *Full Band* sistem juga kurang akurat mendeteksi chordnya. Hasil ini menunjukkan bahwa model memiliki kemampuan yang baik dalam membedakan dan mendeteksi berbagai jenis *chord*, baik *major* maupun *minor*, serta mampu mengidentifikasi progresi *chord* dengan akurat.

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan tahap implementasi yang telah dilakukan pada penelitian Sistem Pengenalan Chord Musik Otomatis Dengan Algoritma Hidden Markov Model dan Pitch Contour Extraction yang diimplementasikan berbasis *website*, dapat disimpulkan bahwa:

1. Sistem berhasil mengimplementasikan kombinasi dua metode (Template Matching dan HMM) untuk deteksi chord dengan akurasi rata-rata 79%. Implementasi ini menunjukkan efektivitas penggunaan Template Matching untuk analisis struktur harmonis dan HMM untuk pemodelan transisi chord dalam sistem deteksi.
2. Performa sistem bervariasi untuk jenis chord yang berbeda. Chord mayor sederhana seperti C, F, dan E menunjukkan performa terbaik dengan precision mencapai 100%, chord minor dasar seperti Am dan Em menunjukkan performa yang baik dengan precision di atas 85%. Namun, untuk chord yang lebih kompleks seperti G# dan G#m, sistem menunjukkan performa yang lebih rendah dengan precision sekitar 50-55%.
3. Sistem mampu mendeteksi berbagai jenis chord dengan reliabilitas yang baik. Untuk chord tunggal, sistem menunjukkan tingkat kepercayaan 85-95%, dalam mendeteksi progresi chord mayor seperti C-F-G-C mencapai akurasi di atas 85%, dan mampu menangani kombinasi chord mayor-minor dengan tingkat akurasi yang cukup baik.
4. Metrik evaluasi menunjukkan hasil yang menjanjikan dengan akurasi 79%, *precision* 85%, *recall* 78%, dan *F1-Score* 80%. Hasil ini menunjukkan bahwa sistem memiliki keseimbangan yang baik antara *precision* dan *recall* dalam deteksi chord.

5.2 Saran

Saran yang dapat diberikan untuk studi lebih lanjut adalah sebagai berikut:

1. Disarankan untuk melakukan peningkatan akurasi dengan menambah jumlah dan variasi data training untuk setiap chord, mengoptimalkan parameter HMM seperti jumlah state dan iterasi terutama untuk chord-chord kompleks, serta mengimplementasikan teknik preprocessing audio yang lebih advanced.
2. Disarankan untuk melakukan pengembangan fitur dengan menambahkan kemampuan deteksi chord yang lebih kompleks seperti 7th, *diminished*, dan *augmented*, implementasi deteksi real-time untuk aplikasi *live performance*, serta visualisasi spektrogram untuk analisis yang lebih detail.
3. Disarankan untuk mengoptimasi sistem yang ditujukan untuk meningkatkan efisiensi komputasi dalam proses ekstraksi fitur, penggunaan memori untuk model HMM, dan implementasi *parallel processing* untuk analisis file audio panjang.
4. Disarankan untuk mencoba algoritma lain seperti *Support Vector Machine*, dan *Long Short-Term Memory* sebagai perbandingan akurasi.

DAFTAR PUSTAKA

- Bansal, M., Goyal, A., & Choudhary, A. (2022). A comparative analysis of K-Nearest Neighbor, Genetic, Support Vector Machine, Decision Tree, and Long Short Term Memory algorithms in machine learning. *Decision Analytics Journal*, 3, 100071.
- Carsault, T., Nika, J., Esling, P., & Assayag, G. (2021). Combining real-time extraction and prediction of musical chord progressions for creative applications. *Electronics (Switzerland)*, 10(21).
- de Berardinis, J., Meroño-Peñuela, A., Poltronieri, A., & Presutti, V. (2023). ChoCo: a Chord Corpus and a Data Transformation Workflow for Musical Harmony Knowledge Graphs. *Scientific Data*, 10(1).
- Fertiawan, F., Hartono, B., Lomba, J. T., No, J., & 50241, S. (2022). deteksi suara chord piano menggunakan metode convolutional neural network. In *Jurnal Informatika & Rekayasa Elektronika* (Vol. 5, Issue 1). <http://e-journal.stmklombok.ac.id/index.php/jire>
- IEEE Signal Processing Society. (2010). *2010 IEEE International Conference on Acoustics, Speech, and Signal Processing : proceedings : March 14-19, 2010, Sheraton Dallas Hotel, Dallas, Texas, U.S.A.* IEEE.
- Indrayani, R. (2020). Modified LSB on Audio Steganography using WAV Format. *2020 3rd International Conference on Information and Communications Technology, ICOIACT 2020*, 466–470.
- Li, T. (2021). Study on a CNN-HMM Approach for Audio-Based Musical Chord Recognition. *IOP Conference Series: Earth and Environmental Science*, 1802(3). <https://doi.org/10.1088/1742-6596/1802/3/032033>
- Lutfiah, L., Uptd, G., Negeri, S. D., Gunong, L., Kokop, K., & Bangkalan, K. (n.d.). *peningkatan hasil belajar peserta didik kelas iii pelajaran sbdp materi jenis alat musik menggunakan media alat musik android (aman)*.
- Mahesh, B. (2018). Machine Learning Algorithms-A Review. *International Journal of Science and Research*. <https://doi.org/10.21275/ART20203995>
- Papadakis, N. M., Aroni, I., & Stavroulakis, G. E. (2022). Effectiveness of MP3 Coding Depends on the Music Genre: Evaluation Using Semantic Differential Scales. *Acoustics*, 4(3), 704–719. <https://doi.org/10.3390/acoustics4030042>
- Rajesh, S., & Nalini, N. J. (2020). Musical instrument emotion recognition using deep recurrent neural network. *Procedia Computer Science*, 167, 16–25. <https://doi.org/10.1016/j.procs.2020.03.178>
- Salamon, J., & Gomez, E. (2012). Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech and Language Processing*, 20(6), 1759–1770. <https://doi.org/10.1109/TASL.2012.2188515>

- Scholarship@western, S., & Toft, R. (2024). *Digital Audio: Sampling, Dither, and Bit Depth.*https://ir.lib.uwo.ca/popmusicforum_techmattershttps://ir.lib.uwo.ca/popmusicforum_techmat ters/11
- Solanki, A., & Pandey, S. (2022). Music instrument recognition using deep convolutional neural networks. *International Journal of Information Technology (Singapore)*, 14(3), 1659–1668. <https://doi.org/10.1007/s41870-019-00285-y>
- Tseng, Y. T., Kawashima, S., Kobayashi, S., Takeuchi, S., & Nakamura, K. (2020). Forecasting the seasonal pollen index by using a hidden Markov model combining meteorological and biological factors. *Science of the Total Environment*, 698. <https://doi.org/10.1016/j.scitotenv.2019.134246>
- Välimäki, V., & Bilbao, S. (2023). Giant FFTs for Sample-Rate Conversion. *AES: Journal of the Audio Engineering Society*, 71(3), 88–99. <https://doi.org/10.17743/jaes.2022.0061>
- Yadav Kasula, B. (n.d.). *Machine Learning Applications in Diabetic Healthcare: A Comprehensive Analysis and Predictive Modeling*