FINAL PROJECT
MATA KULIAH PENGANTAR PEMROSESAN DATA MULTIMEDIA
KLASIFIKASI GENRE MUSIK

Disusun Oleh Kelompok B1:

Shelomita Putrinda Culio	(2208561002)
Pande Nyoman Weda Wesnawa	(2208561021)
Syelvia Julianti	(2208561034)
I Kadek Angga Kusuma Diatmika	(2208561109)
Ida Ayu Tri Sabina Putri	(2208561139)

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

TAHUN 2024

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Musik memiliki berbagai genre yang memiliki karakteristik unik, membuat klasifikasi genre musik menjadi topik menarik dalam bidang analisis data dan pengenalan pola. Proses klasifikasi ini menggunakan metode analisis yang bertujuan untuk memprediksi atau mengelompokkan data ke dalam beberapa kategori berdasarkan atribut atau fitur yang dimiliki oleh data tersebut (Sinaga *et al*, 2024). Dalam konteks ini, genre musik seperti classical, jazz, dan pop dapat diklasifikasikan menggunakan teknik-teknik tertentu untuk memahami perbedaan karakteristik masing-masing genre.

Salah satu metode yang sering digunakan untuk klasifikasi genre musik adalah *Convolutional Neural Network* (CNN) yang dikenal karena kemampuannya dalam mengenali pola dari data berbasis gambar atau spektrogram audio. CNN merupakan jaringan saraf tiruan yang terinspirasi dari struktur visual cortex pada otak manusia, yang sangat efektif dalam menangkap fitur-fitur lokal dari data visual seperti gambar atau spektrogram musik. Dalam klasifikasi genre musik, CNN dapat dilatih untuk mengenali pola dari data audio yang telah diubah menjadi spektrogram.

Ekstraksi fitur merupakan tahap krusial dalam proses klasifikasi genre musik. *Mel Frequency Cepstral Coefficient* (MFCC) adalah salah satu teknik ekstraksi fitur yang populer digunakan dalam pengolahan sinyal audio. MFCC bekerja dengan cara mengubah sinyal audio menjadi representasi yang lebih mudah dianalisis oleh model pembelajaran mesin. Proses ini melibatkan beberapa tahap seperti *Frame Blocking, Windowing, Fast Fourier Transform* (FFT), *Mel Frequency Wrapping*, dan *Cepstrum*. MFCC dikenal efektif dalam menangkap karakteristik temporal dan spektral dari sinyal audio, menjadikannya pilihan ideal untuk klasifikasi genre musik.

Penelitian terdahulu telah menunjukkan efektivitas kombinasi MFCC dan berbagai metode klasifikasi untuk tugas klasifikasi genre musik. Contohnya, penelitian oleh Ramadhana dan Widiartha (2023) menunjukkan bahwa kombinasi MFCC dan K-NN dapat mencapai akurasi optimal 77,5% dalam klasifikasi musik pop dan RnB. Selain itu, penelitian oleh Soerkarta *et al* (2023) menunjukkan bahwa optimasi *hyperparameter* pada

classifier CNN untuk klasifikasi genre musik menggunakan MFCC dapat mencapai akurasi tertinggi 72%.

Penelitian ini bertujuan untuk menyelidiki akurasi penggunaan metode ekstraksi fitur MFCC dan klasifikasi CNN dalam klasifikasi genre musik jazz, classical, dan pop. Selain itu, penelitian ini juga akan mencari parameter terbaik yang dapat digunakan dalam ekstraksi fitur MFCC dan CNN untuk mendapatkan hasil akurasi terbaik pada klasifikasi genre musik. Penelitian ini diharapkan dapat memberikan kontribusi signifikan dalam bidang klasifikasi musik dan membantu dalam pengembangan sistem rekomendasi musik yang lebih akurat dan efisien.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam makalah ini dapat dirumuskan sebagai berikut:

1.2.1 Bagaimanakah akurasi penggunaan metode ekstraksi fitur MFCC dan klasifikasi CNN dalam klasifikasi genre musik jazz, classical, dan pop?

1.2.2 Bagaimanakah parameter terbaik yang dapat digunakan dalam klasifikasi CNN untuk mendapatkan hasil akurasi terbaik pada klasifikasi genre musik?

1.3 Tujuan

Berdasarkan rumusan masalah tersebut, tujuan penulisan dalam makalah ini sebagai berikut:

1.3.1 Untuk mengetahui akurasi penggunaan metode ekstraksi fitur MFCC dan klasifikasi CNN dalam klasifikasi genre musik jazz, classical, dan pop.

1.3.2 Untuk mengetahui parameter terbaik yang dapat digunakan dalam klasifikasi CNN untuk mendapatkan hasil akurasi terbaik pada klasifikasi genre musik.

1.4 Manfaat

Berdasarkan rumusan masalah dan tujuan yang telah dirumuskan tersebut, penelitian ini memiliki beberapa manfaat, baik secara teoritis maupun praktis. Secara teoritis, penelitian ini dapat memberikan kontribusi pada pengembangan ilmu pengetahuan di bidang pengolah audio dan *machine learning*, khususnya dalam konteks klasifikasi genre musik menggunakan metode ekstraksi fitur MFCC dan algoritma klasifikasi CNN. Sedangkan, secara praktis, hasil penelitian ini dapat diterapkan oleh

pengembang dalam sistem pengenalan genre musik yang dapat digunakan dalam berbagai aplikasi, seperti sistem pengenalan genre musik.

1.5 Batasan

Terdapat beberapa batasan yang digunakan dalam penelitian ini sebagai berikut:

1.5.1 Metode klasifikasi yang digunakan adalah CNN dengan ekstraksi fitur MFCC.

1.5.2 Penelitian ini dibatasi pada klasifikasi tiga jenis genre musik, yaitu jazz, pop, dan classical. Genre musik lain seperti rock, hiphop, atau yang lainnya tidak termasuk dalam cakupan penelitian ini.

1.5.3 Penelitian ini dibatasi pada klasifikasi *file* audio dengan ekstensi .wav.

1.5.4 Parameter dibatasi pada beberapa parameter tertentu saja yang diujicobakan untuk menemukan parameter terbaik. Pada CNN hanya mengujicobakan parameter *dropout rate, learning rate, convolutional filters* berupa *convolution* filter 1 dan 2.

1.6 Asumsi

Terdapat beberapa asumsi yang digunakan dalam penelitian ini sebagai berikut:

1.6.1 Semua audio dalam dataset diasumsikan memiliki kualitas yang memadai untuk ekstraksi fitur dan klasifikasi, tanpa terdistorsi oleh faktor-faktor lain, seperti pencahayaan yang buruk, resolusi rendah, atau gangguan lainnya yang dapat mengganggu proses klasifikasi.

1.6.2 Label genre musik yang terdapat di dalam dataset diasumsikan telah ditetapkan dengan benar dan sesuai.

BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Teori

2.1.1 Klasifikasi

Klasifikasi adalah metode analisis yang bertujuan untuk memprediksi atau mengelompokkan data ke dalam beberapa kategori atau kelas yang dilakukan berdasarkan atribut atau fitur yang dimiliki oleh data tersebut (Sinaga *et al*, 2024). Proses klasifikasi melibatkan pelatihan model pada dataset yang telah diberi label, di mana model tersebut belajar untuk mengenali pola atau karakteristik dari data yang sesuai dengan kategori tertentu. Misalnya, dalam sistem pengenalan wajah, model klasifikasi akan dilatih untuk mengenali berbagai wajah dan mengklasifikasikannya ke dalam kategori berdasarkan identitas individu. Klasifikasi menjadi alat yang sangat penting dalam berbagai aplikasi mulai dari deteksi spam email hingga diagnosis penyakit (Li *et al*, 2020).

Model klasifikasi dapat dibangun menggunakan berbagai algoritma, seperti K-Nearest Neighbors (KNN), Support Vector Machines (SVM), dan jaringan saraf tiruan (neural networks). Setiap algoritma memiliki pendekatan yang berbeda dalam mengenali pola dan memprediksi kategori untuk data baru. Misalnya, KNN mengklasifikasikan data berdasarkan kedekatannya dengan data yang telah diberi label dalam ruang fitur, sementara SVM mencari *hyperplane* yang memisahkan kelas-kelas dalam data dengan margin terbesar. Dalam konteks modern, metode deep learning seperti Convolutional Neural Networks (CNN) telah menunjukkan kemampuan luar biasa dalam klasifikasi data yang kompleks, terutama dalam analisis gambar dan pengenalan suara.

2.1.2 *Machine Learning*

Machine Learning adalah teknologi yang dikembangkan untuk dapat belajar dengan sendirinya tanpa arahan dari penggunanya yang sangat membantu dalam menyelesaikan masalah dan membuat mudah dalam mengerjakan sesuatu. Pembelajaran mesin dikembangkan berdasarkan disiplin ilmu lainnya seperti statistika, matematika, dan *data mining*, sehingga mesin dapat belajar dengan menganalisa data tanpa perlu di program ulang atau diubah. *Machine Learning*

memiliki kemampuan untuk memperoleh data yang ada dengan perintah sendiri dan mempelajari data yang ada serta data yang diperoleh, sehingga dapat melakukan tugas tertentu yang sangat beragam (Diana, dkk., 2017).

Machine Learning sangat berguna untuk menyelesaikan kasus yang berubah–ubah dalam waktu tertentu. Misalnya algoritma untuk mengenali wajah. Dalam deteksi ekspresi citra wajah, berikut beberapa algoritma *machine learning* yang digunakan adalah (Clara, 2024) :

a. *Random Forest*: Algoritma yang digunakan untuk mendapatkan prediksi akhir dari beberapa pohon keputusan yang dibangun secara acak
b. *Decision Tree*: Algoritma yang digunakan untuk memilih dan memisahkan data menjadi subset yang homogen dari striktur pohon yang telah diklasifikasi
c. *Naive Bayes*: Algoritma yang digunakan dengan asumsi bahwa atribut yang independen memberikan kontribusi secara langsung terhadap probabilitas dari kelas yang diprediksi
d. *Support Vector Machine*: Algoritma yang digunakan dengan klasifikasi data yang kompleks dengan menggunakan hyperplane yang terbaik untuk memisahkan kelas-kelas yang berbeda.

Namun, perlu diingat bahwa penggunaan algoritma *machine learning* dalam deteksi ekspresi citra wajah dapat berbeda-beda tergantung pada tujuan dan kebutuhan spesifik dari aplikasi yang dibuat.

2.1.3 Audio *Preprocessing*

Audio *preprocessing* adalah langkah penting dalam analisis dan pemrosesan sinyal suara yang bertujuan untuk meningkatkan kualitas data dan mempersiapkannya untuk tahap pemrosesan lebih lanjut atau analisis. Pemrosesan awal ini mencakup berbagai teknik dan prosedur yang dirancang untuk membersihkan dan mengubah data audio mentah agar lebih berguna dalam aplikasi seperti pengenalan ucapan, pengenalan pola, atau analisis suara. Sebuah penelitian oleh (Chiu et al., 2022) menunjukkan bahwa kualitas dari proses pemrosesan awal dapat sangat mempengaruhi kinerja model pembelajaran mesin dalam tugas pengenalan ucapan.

Tahap-tahap utama dalam pemrosesan awal audio meliputi normalisasi, penghapusan kebisingan, framing, dan ekstraksi fitur. Normalisasi bertujuan untuk menyamakan amplitudo sinyal audio sehingga memiliki tingkat suara yang konsisten, yang membantu dalam mengurangi variabilitas yang tidak diinginkan. Penghapusan kebisingan adalah proses yang digunakan untuk mengurangi atau menghilangkan komponen sinyal yang tidak diinginkan yang dapat mengganggu analisis lebih lanjut. Misalnya, filter low-pass dan teknik pengurangan kebisingan adaptif sering digunakan untuk menghilangkan komponen frekuensi tinggi dan kebisingan lingkungan. Penelitian oleh (Lee et al., 2023) menyoroti pentingnya penghapusan kebisingan dalam meningkatkan kualitas sinyal untuk pengenalan ucapan otomatis.

Selain itu, framing dan ekstraksi fitur adalah langkah kritis dalam pemrosesan awal audio. Ekstraksi fitur melibatkan pemecahan sinyal audio menjadi segmen-segmen kecil yang disebut frame, yang memungkinkan analisis rinci dalam domain waktu atau frekuensi. Setiap frame kemudian dapat dianalisis secara terpisah untuk menangkap perubahan dinamis dalam sinyal. Ekstraksi fitur bertujuan untuk mengidentifikasi dan mengambil informasi yang paling relevan dari sinyal audio, seperti koefisien Mel-Frequency Cepstral Coefficients (MFCC), yang digunakan untuk menggambarkan spektrum frekuensi suara dengan cara yang sesuai dengan persepsi manusia. Penelitian terbaru oleh Patel dan (Gupta, 2021) menunjukkan bahwa framing dan ekstraksi fitur yang efektif adalah hal utama untuk menghasilkan representasi data yang lebih informatif dan kompak, yang sangat penting untuk keberhasilan aplikasi berbasis suara.

2.1.4 *Mel-Frequency Cepstral Coefficients* (MFCC)

Mel Frequency Cepstral Coefficients (MFCC) adalah metode umum dalam pemrosesan sinyal suara yang digunakan untuk mengekstraksi fitur dari sinyal akustik. MFCC berperan penting dalam pengenalan ucapan dan analisis suara dengan memetakan spektrum frekuensi suara ke skala mel, yang lebih sesuai dengan persepsi frekuensi oleh telinga manusia. Sebuah studi baru-baru ini oleh Sharma et al. (2022) menyoroti bahwa MFCC memberikan representasi

kompak yang dapat menangkap komponen penting dari spektrum suara yang relevan untuk analisis dan pengenalan suara.

Proses perhitungan MFCC melibatkan beberapa tahap kunci. Pertama, sinyal suara dipecah menjadi frame-frame kecil untuk menangkap variasi temporal. Setiap frame kemudian diubah ke domain frekuensi menggunakan Transformasi Fourier Cepat (FFT). Setelah itu, hasil dari FFT dipetakan ke skala mel melalui penggunaan bank filter mel. Nilai-nilai yang diperoleh ini kemudian diambil logaritmanya untuk menormalkan perubahan energi, dan Transformasi Cosinus Diskrit (DCT) diterapkan untuk menghasilkan koefisien MFCC. Koefisien ini memberikan representasi yang lebih kompak dan tetap mempertahankan informasi penting mengenai karakteristik suara. (Gupta et al., 2021).

Dalam berbagai aplikasi, seperti pengenalan ucapan otomatis dan analisis emosi dalam suara, MFCC telah terbukti sangat efektif. Ini karena kemampuannya untuk menangkap informasi frekuensi penting yang relevan dengan persepsi pendengaran manusia. Menurut penelitian terbaru (Li dan Huang, 2023), MFCC digunakan secara luas karena keefektifannya dalam merepresentasikan fitur-fitur suara yang penting untuk analisis suara dan pengenalan pola.

2.1.5 *Convolution Neural Network* (CNN)

Convolutional Neural Networks (CNN) adalah jenis arsitektur jaringan saraf tiruan yang dirancang khusus untuk pemrosesan data grid seperti gambar. CNN memiliki arsitektur yang terdiri dari beberapa lapisan utama: lapisan konvolusi, lapisan pooling, dan lapisan sepenuhnya terhubung. Lapisan konvolusi bertugas mengekstraksi fitur-fitur dari input menggunakan filter atau kernel yang melakukan operasi konvolusi. Filter ini bergerak melintasi input untuk mendeteksi fitur seperti tepi, tekstur, atau pola tertentu dalam data. Setiap filter menghasilkan peta fitur (feature map) yang menyoroti keberadaan fitur tertentu di lokasi yang berbeda dalam input (Allugunti, 2022).

Lapisan pooling, yang biasanya mengikuti lapisan konvolusi, berfungsi untuk mengurangi dimensi peta fitur sambil mempertahankan informasi penting.

Operasi pooling, seperti max pooling atau average pooling, mengambil nilai maksimum atau rata-rata dari area kecil dalam peta fitur, yang membantu mengurangi kompleksitas komputasi dan meningkatkan ketahanan terhadap perubahan kecil pada input (Kiranyaz et al., 2021). Kombinasi lapisan konvolusi dan pooling memungkinkan CNN untuk secara efektif mengurangi ukuran data input secara bertahap sambil mempertahankan fitur-fitur yang relevan.

Lapisan sepenuhnya terhubung, yang biasanya terletak di bagian akhir arsitektur CNN, bertugas untuk melakukan klasifikasi berdasarkan fitur-fitur yang telah diekstraksi oleh lapisan sebelumnya. Setiap neuron di lapisan ini terhubung ke semua neuron di lapisan sebelumnya, memungkinkan jaringan untuk menggabungkan berbagai fitur yang telah diekstraksi untuk membuat keputusan akhir (Zhou, 2020). CNN telah menunjukkan kinerja luar biasa dalam berbagai tugas, termasuk pengenalan gambar, klasifikasi objek, dan analisis video, karena kemampuannya untuk belajar dan mengenali pola kompleks dalam data dengan efisiensi tinggi.

2.1.6 *Accuracy, Precision, Recall,* F1-*Score*

Evaluasi model dapat dihitung dengan menggunakan beberapa metrik, seperti *accuracy, precision, recall,* dan f1-*score* sebagai berikut (Arthana, 2019) :

a. *Accuracy* merupakan metrik yang mengukur sejauh mana model dapat melakukan klasifikasi data dengan benar dari seluruh data yang diuji. Metrik ini dihitung sebagai jumlah prediksi yang benar dibagi dengan total jumlah prediksi.
b. *Precision* merupakan metrik yang mengukur seberapa tepat model dalam mengidentifikasi kelas tertentu, dalam konteks ini yaitu berapa persen dari prediksi yang benar-benar merupakan ekspresi wajah yang dimaksud.
c. *Recall* merupakan metrik yang mengukur seberapa baik model dalam menemukan kembali semua contoh dari suatu label klasifikasi tertentu dalam dataset, dalam konteks ini yaitu seberapa banyak ekspresi wajah yang benar-benar terdeteksi oleh model.

d. F1-*Score* merupakan metrik yang memberikan rata-rata harmonis antara *precision* dan *recall*, sehingga memberikan keseimbangan antara kedua metrik tersebut.

2.2 Tinjauan Empiris

2.2.1 Classification of Pop And RnB (Rhythm And Blues) Songs With MFCC Feature Extraction And K-NN Classifier (Zhaqy Hikkammi Gullam Ramadhana & I Made Widiartha, 2023).

Penelitian ini membahas klasifikasi genre musik untuk menentukan apakah musik tersebut termasuk genre pop atau RnB menggunakan metode MFCC untuk ekstraksi fitur dan K-NN sebagai metode klasifikasi. Data yang digunakan adalah data sekunder dari dataset GTZAN, dengan dua genre yaitu RnB dan pop, masing-masing terdiri dari 100 lagu. Proses ekstraksi fitur menggunakan MFCC melibatkan beberapa tahap seperti Frame Blocking, Windowing, FFT, Mel Frequency Wrapping, dan Cepstrum. Proses klasifikasi menggunakan K-NN melibatkan tahap input dataset, pre-processing, perhitungan jarak Euclidean, penyortiran jarak terdekat, dan penentuan kelas. Pengujian dilakukan dengan beberapa nilai K yang berbeda (3, 7, 11, 21, 31, 41, 51, dan 61), dengan proporsi data latih dan data uji adalah 80:20. Hasil pengujian menunjukkan bahwa nilai optimal k adalah 31 dengan akurasi 77,5%.

2.2.2 Hyperparameter Optimization of CNN Classifier for Music Genre Classification (Soerkarta, R., Aras, S., & Aswad, A. N., 2023).

Penelitian ini menyelidiki optimalisasi hyperparameter pada classifier CNN untuk klasifikasi genre musik menggunakan dataset GTZAN. Dataset ini terdiri dari klip audio berdurasi 30 detik yang dioptimalkan menggunakan Mel-Frequency Cepstral Coefficients (MFCC) untuk ekstraksi fitur. Penelitian ini bertujuan untuk mengklasifikasikan genre musik dalam 3 detik pertama dari setiap klip, meskipun metode ini memiliki potensi kesalahan yang tinggi karena variabilitas pada segmen awal musik. Penelitian ini mengeksplorasi berbagai skenario yang melibatkan hyperparameter seperti ukuran batch, jumlah epoch, dan rasio pembagian data. Akurasi tertinggi yang dicapai adalah 72% dengan rasio pembagian data 85:15, ukuran batch 32, dan 500 epoch. Penelitian ini menyoroti

pentingnya penyetelan hyperparameter dalam meningkatkan kinerja model CNN untuk klasifikasi genre musik, menunjukkan bahwa konfigurasi spesifik dapat secara signifikan meningkatkan akurasi.

2.2.3 A comparative analysis of CNN and LSTM for music genre classification (Gessle, G., & Åkesson, S. 2019).

Penelitian ini membandingkan kinerja Convolutional Neural Networks (CNN) dan Long Short-Term Memory (LSTM) dalam tugas klasifikasi genre musik. Model dilatih menggunakan Mel-Frequency Cepstral Coefficients (MFCCs) pada dua dataset yang berbeda: GTZAN dan FMA. Hasil penelitian menunjukkan bahwa CNN memiliki kinerja yang lebih tinggi dibandingkan dengan LSTM pada kedua dataset. CNN mencapai akurasi sebesar 56,0% pada dataset GTZAN dan 50,5% pada dataset FMA, sedangkan LSTM hanya mencapai akurasi 42,0% pada dataset GTZAN dan 33,5% pada dataset FMA. Performa CNN yang lebih baik ini menunjukkan bahwa CNN efektif dalam menangkap fitur dari data audio yang direpresentasikan sebagai spektrogram atau MFCCs.

Penelitian ini menyimpulkan bahwa CNN lebih cocok untuk tugas klasifikasi genre musik dibandingkan dengan LSTM ketika menggunakan MFCCs sebagai fitur input. Ini penting bagi peneliti dan pengembang yang bekerja pada proyek klasifikasi audio, menunjukkan bahwa CNN mungkin merupakan titik awal yang lebih baik untuk tugas-tugas tersebut. Dengan demikian, studi ini menyarankan untuk memilih CNN daripada LSTM karena kinerja yang lebih unggul dalam konteks klasifikasi genre musik.

2.2.4 Klasifikasi Genre Musik Dengan Mel Frequency Cepstral Coefficient Dan Spektogram Menggunakan Convolutional Neural Network (Fardhani, Wihardi, dan Piantri, 2021)

Penelitian ini membahas klasifikasi genre musik menggunakan teknik Mel Frequency Cepstral Coefficient (MFCC) dan Spektogram yang diekstraksi menggunakan Convolutional Neural Network (CNN). Penelitian bertujuan untuk membandingkan efektivitas dua metode ekstraksi fitur dalam mengklasifikasikan genre musik. Data yang digunakan terdiri dari 500 lagu dari dataset GTZAN dan 500 lagu berbahasa Indonesia, masing-masing terdiri dari 5 genre yaitu Country,

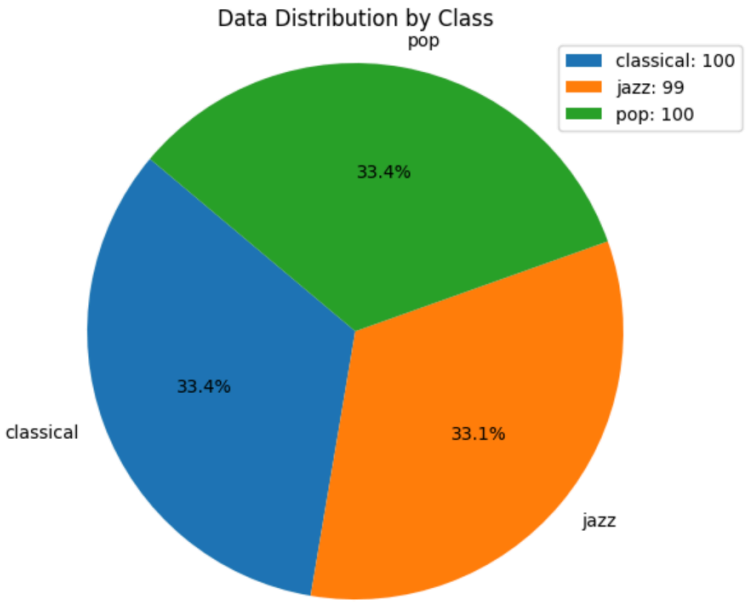
Jazz, Pop, Reggae, dan Rock. Hasil penelitian menunjukkan bahwa metode ekstraksi Spektogram menghasilkan akurasi yang lebih tinggi dibandingkan MFCC. Pada eksperimen menggunakan dataset GTZAN, klasifikasi dengan Spektogram mencapai akurasi tertinggi 76%, sedangkan dengan MFCC hanya mencapai rata-rata 54.2%. Khususnya, genre Jazz dan Pop menunjukkan performa klasifikasi yang lebih baik dibandingkan genre lainnya, dengan akurasi masing-masing mencapai 90% dan 75% untuk Jazz dan Pop pada eksperimen spektogram. Validasi pada dataset lagu Indonesia menunjukkan hasil yang lebih rendah dibandingkan GTZAN, dengan akurasi tertinggi mencapai 66% untuk spektogram. Penelitian ini menyarankan peningkatan jumlah dataset dan eksplorasi fitur musik lainnya seperti pitch, timbral, dan harmoni untuk penelitian mendatang. Temuan ini dapat digunakan untuk meningkatkan sistem rekomendasi musik dan fitur pencarian musik berdasarkan genre.

BAB III
ANALISIS DAN DESAIN

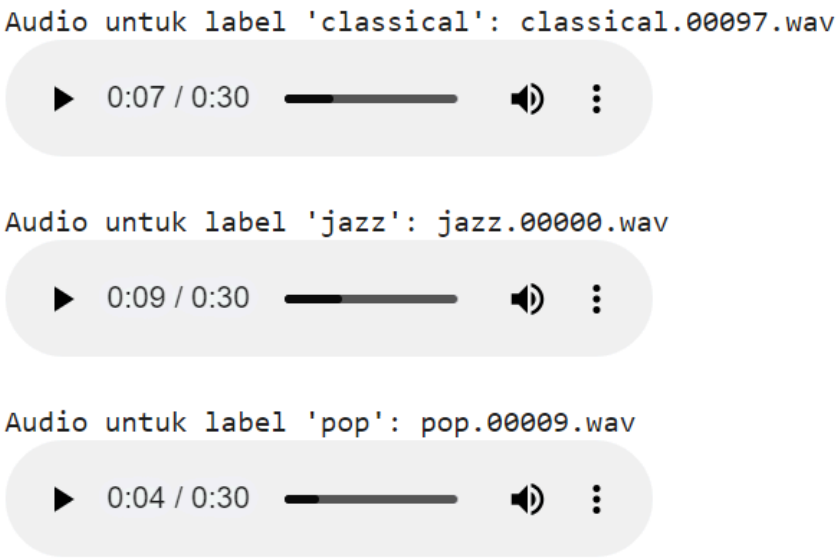
3.1 Data dan Pengumpulan Data

Dataset yang digunakan dalam penelitian ini adalah data sekunder berupa GTZAN Dataset (*music audio dataset*) yang diambil dari situs Kaggle dengan link https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification. Dataset tersebut berisi audio musik yang diklasifikasikan dalam 10 label genre, yaitu *blues, classical, country, disco, hiphop, jazz, metal, pop, reggae,* dan *rock*. Akan tetapi, pada penelitian ini hanya akan menggunakan 3 label genre saja, yaitu *pop, jazz,* dan *classical.*

Dari keseluruhan jumlah data citra pada dataset tersebut, jumlah data audio yang akan digunakan untuk penelitian ini adalah sejumlah 299 data audio, di mana pada label *pop* sejumlah 100, label *jazz* sejumlah 99, dan label *classical* sejumlah 100, distribusi data dapat dilihat pada gambar 3.1. Data audio berdurasi 30 detik dengan format ekstensi .wav. Contoh data tiap label yang digunakan dapat dilihat pada Gambar 3.2.



Gambar 3.1 Distribusi Dataset

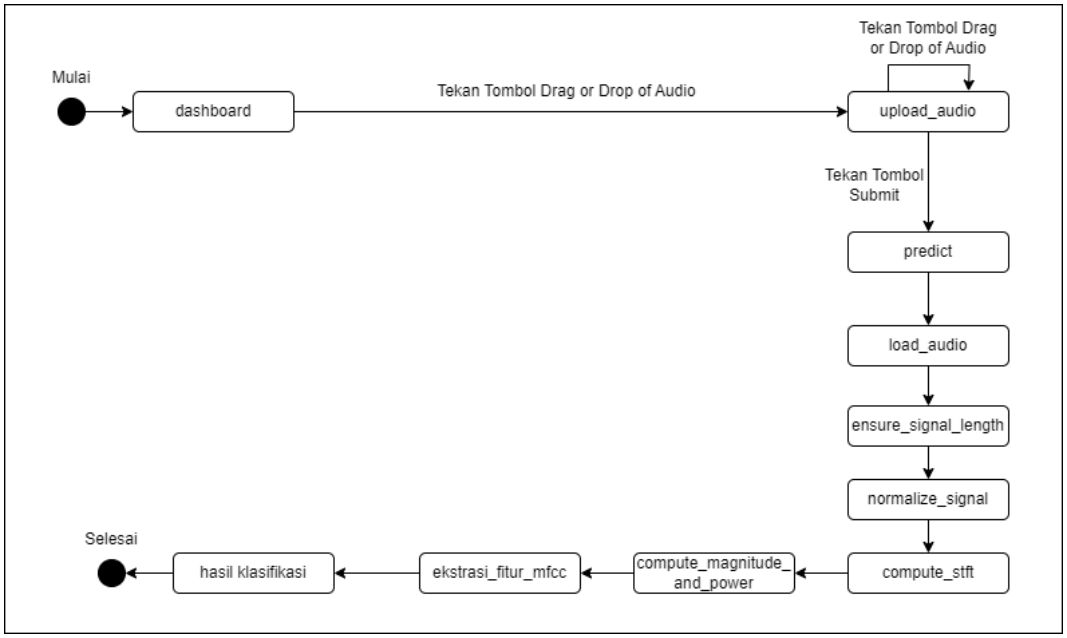


Gambar 3.2 Contoh Data Label

3.2 Desain Sistem

3.2.1 Diagram Transisi

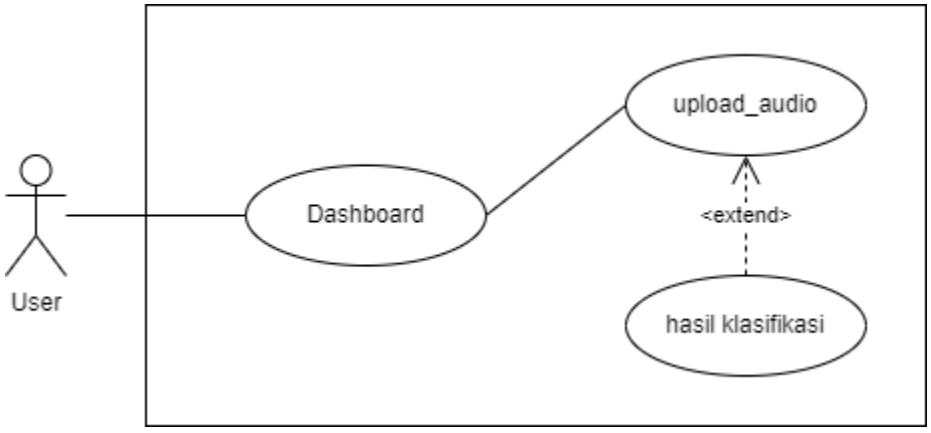
Diagram transisi menggambarkan transisi suatu sistem melalui berbagai keadaan, transisi yang menghubungkannya, dan tindakan yang memicu transisi. Dalam penelitian ini, diagram transisi dapat dilihat pada Gambar 3.3 dimana proses awal *user* akan masuk pada sistem yang langsung terarah pada *dashboard*, lalu untuk melakukan klasifikasi genre musik, *user* perlu menekan tombol *Drag or Drop of Audio* untuk *upload* audio. Selanjutnya, gambar audio akan masuk ke dalam sistem setelah *user* menekan tombol *submit* dan diproses mulai dari memuat file audio untuk diproses, memastikan panjang sinyal audio sesuai, normalisasi sinyal audio, menghitung Transformasi Fourier Waktu-Pendek (STFT) dari sinyal audio, menghitung magnitudo dan daya dari sinyal, melakukan proses ekstraksi fitur hingga akhirnya sistem menjalankan proses klasifikasi dengan hasil klasifikasi yang akan muncul pada *dashboard user*.



Gambar 3.3 Diagram Transisi

3.2.2 *Use Case Diagram*

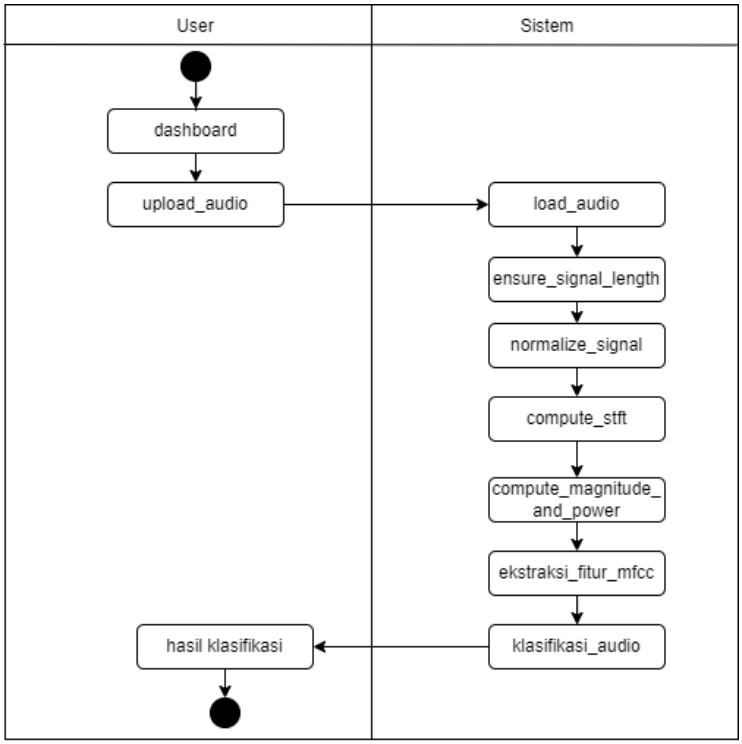
Use case diagram menjelaskan proses yang berlangsung pada sistem dengan mendeskripsikan interaksi *user* dengan sistem. Sistem klasifikasi ekspresi dalam penelitian ini dapat dilihat pada Gambar 3.4, hanya terdapat satu *user* dengan interaksi pada sistem berupa *upload* audio pada *dashboard*, lalu sistem akan memberikan *output* berupa hasil klasifikasi genre musik berupa genre pop, jazz, dan classical.



Gambar 3.4 *Use Case Diagram*

3.2.3 *Activity Diagram*

Activity diagram berfungsi untuk memperlihatkan urutan aktifitas pada sistem. Dalam penelitian ini, *activity diagram* yang dimiliki dapat dilihat pada Gambar 3.5 dimana proses awal *user* akan masuk pada sistem yang langsung terarah pada *dashboard*, lalu untuk melakukan klasifikasi genre musik *user* akan *upload* satu audio. Selanjutnya, gambar tersebut akan masuk ke dalam sistem dan diproses, mulai dari memuat file audio untuk diproses, memastikan panjang sinyal audio sesuai, normalisasi sinyal audio, menghitung *Transformasi Fourier* Waktu-Pendek (STFT) dari sinyal audio, menghitung magnitudo dan daya dari sinyal melakukan proses ekstraksi fitur hingga akhirnya sistem menjalankan proses klasifikasi dengan hasil klasifikasi yang akan muncul pada *dashboard user*.

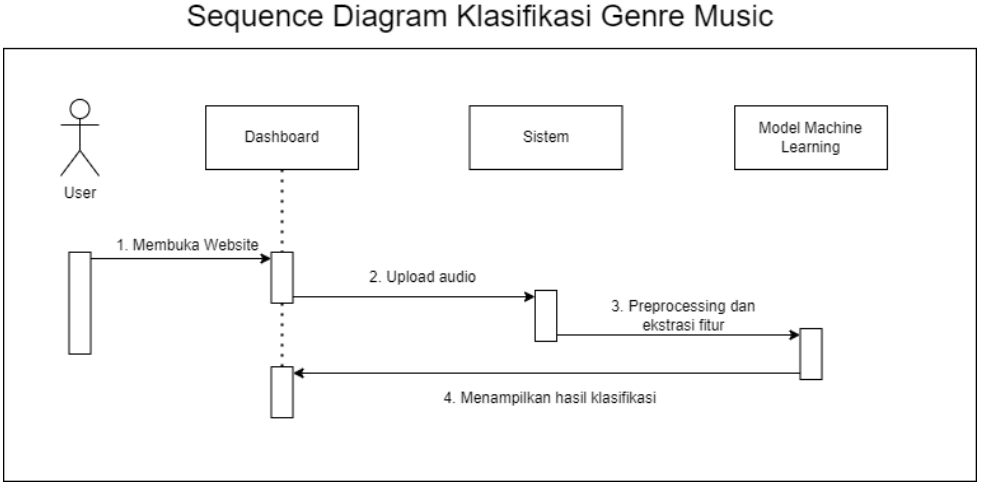


Gambar 3.5 *Activity Diagram*

3.2.4 *Sequence Diagram*

Activity diagram berfungsi untuk menggambarkan bagaimana objek dalam sistem berkomunikasi dan bekerja bersama dalam sebuah skenario sehingga bisa menghasilkan *output*. Dalam penelitian ini *sequence diagram* dapat dilihat pada

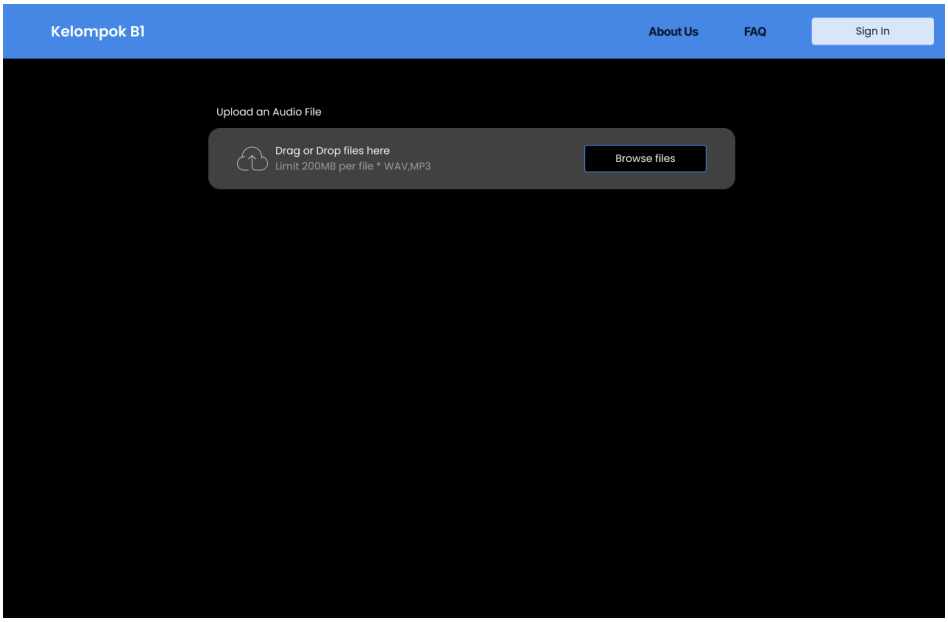
Gambar 3.6 dimana terdapat tiga objek, yaitu *dashboard*, sistem, dan model *machine learning* dengan tahapan *user* akan masuk ke halaman *dashboard* untuk *upload* audio lalu sistem akan menerima audio tersebut dan melakukan *preprocessing* dan ekstraksi fitur dengan hasil yang dikirimkan ke dalam model *machine learning* yang dilatih sehingga dapat memberikan hasil klasifikasi genre musik yang dikirimkan ke *dashboard* sebagai *output* yang diterima oleh *user*.



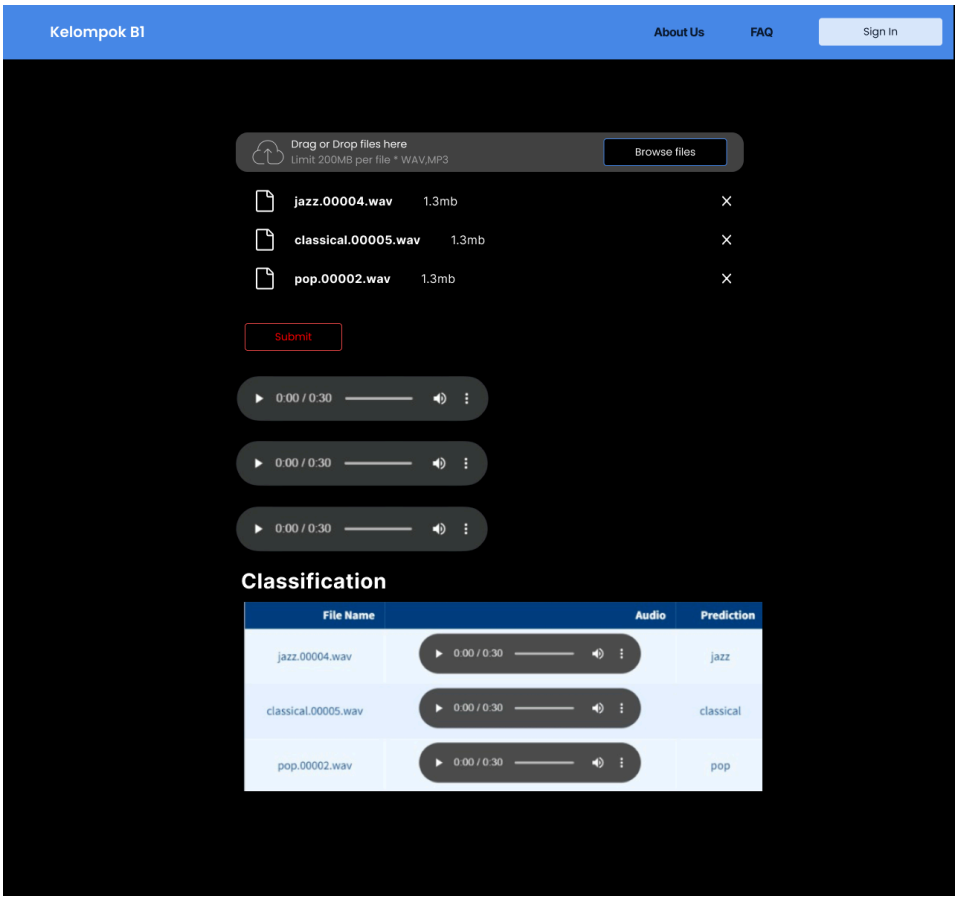
Gambar 3.6 *Sequence Diagram*

3.3 Desain UI/UX Web

Pada halaman *dashboard* klasifikasi genre musik, *user* dapat menginput beberapa *audio* musik yang diinginkan dengan format *.wav* pada bagian *upload audio*. Setelah selesai menginput *audio* yang ingin diklasifikasi, akan muncul tombol Submit. Desain halaman *dashboard* dapat dilihat pada Gambar 3.7. *User* dapat menekan tombol *Submit* untuk melihat hasil klasifikasi *genre* musik dari audio yang telah diinputkan sebelumnya. Hasil yang akan ditampilkan oleh sistem berupa prediksi apakah audio tersebut merupakan musik *jazz, pop*, atau *classical*. Desain halaman hasil klasifikasi dapat dilihat pada Gambar 3.8.



Gambar 3.7 Tampilan *Dashboard*



Gambar 3.8 Tampilan *Output* Klasifikasi

3.4 Skenario Eksperimen Evaluasi Model dan Sistem

Penelitian pada makalah ini bertujuan untuk mengembangkan model yang dapat melakukan klasifikasi genre musik, yaitu genre classical, pop, dan jazz. Alur pengembangan model ini terdiri dari beberapa tahap, yaitu pengumpulan data, *preprocessing* data, ekstraksi fitur, klasifikasi menggunakan *machine learning*, hingga evaluasi model dan sistem. Berikut merupakan skenario setiap tahap penelitian termasuk evaluasi model dan evaluasi sistem :

3.4.1 Pengumpulan Data

Pengumpulan data dilakukan menggunakan dataset audio genre musik GTZAN dari situs Kaggle seperti yang sudah dijelaskan pada sub bab 3.1.

3.4.2 *Preprocessing* Data

Terdapat enam tahap dalam *preprocessing* data. Tahap pertama memuat file audio dengan parameter *sampling rate* yang telah ditentukan sebelumnya. Langkah ini dilakukan untuk memastikan bahwa file audio dimuat dengan benar dan sesuai dengan spesifikasi yang dibutuhkan untuk analisis selanjutnya dan mempertahankan kualitas audio yang optimal.

Tahap kedua dalam *preprocessing* data audio adalah menyesuaikan panjang sinyal audio. Setelah file audio dimuat, panjang sinyal dievaluasi untuk memastikan konsistensi durasi antar-sinyal. Jika sinyal lebih pendek dari yang diharapkan, dilakukan proses *padding* dengan nilai konstan untuk mencapai panjang yang diinginkan. Sebaliknya, jika sinyal terlalu panjang, sinyal dipotong agar sesuai dengan panjang maksimum yang ditentukan. Langkah dilakukan supaya semua data audio memiliki panjang yang seragam sebelum dilakukan analisis lanjutan.

Tahap ketiga adalah normalisasi sinyal. Normalisasi mengubah rentang nilai sinyal menjadi antara -1 hingga 1. Hal ini membantu dalam menjaga kualitas sinyal dan meminimalkan distorsi yang mungkin terjadi selama proses analisis atau klasifikasi berikutnya. Normalisasi dilakukan untuk memastikan rentang nilai sinyal konsisten dan optimal untuk pemrosesan berikutnya. Dengan normalisasi ini, sinyal audio siap untuk tahap pengolahan berikutnya, seperti transformasi *Fourier* atau ekstraksi fitur spektral.

Tahap keempat adalah melakukan *windowing* dan *framing* menggunakan *Short-Time Fourier Transform* (STFT). Langkah ini membagi sinyal menjadi segmen-segmen kecil menggunakan STFT dengan *parameter-frame size* (ukuran *frame*) dan *hop length* (langkah hop) yang telah ditentukan sebelumnya. STFT ini memungkinkan representasi sinyal audio dalam domain frekuensi terhadap waktu, yang berguna untuk analisis lebih lanjut terhadap karakteristik spektral sinyal.

Tahap kelima dalam *preprocessing* data audio adalah menghitung magnitudo dari *Short-Time Fourier Transform* (STFT) berupa pengambilan nilai absolut dari STFT untuk setiap frame yang menghasilkan magnitudo dari sinyal dalam domain frekuensi terhadap waktu. Magnitudo ini merepresentasikan amplitudo dari komponen frekuensi yang ada dalam setiap *frame* sinyal audio. Tahap ini bertujuan untuk memahami distribusi energi frekuensi dalam sinyal audio.

Tahap keenam adalah menghitung *power* dari *Short-Time Fourier Transform* (STFT). Langkah ini mengkuadratkan nilai magnitudo tersebut untuk mendapatkan nilai power dari sinyal audio. Ini dilakukan dengan menghitung power = magnitude^2, di mana power menggambarkan distribusi energi sinyal audio dalam domain frekuensi terhadap waktu. Tahapan ini bertujuan untuk memperoleh representasi yang lebih kuat tentang distribusi energi frekuensi dalam sinyal audio.

3.4.3 *Splitting* Data dan Ekstraksi Fitur

Pada penelitian ini, dataset berjumlah 299 data audio, dengan masing-masing jumlah per label adalah label *classical* sejumlah 100, label *pop* sejumlah 100, dan label *jazz* sejumlah 99. Dataset akan dibagi menjadi dua bagian, yaitu data 80% *training* dan 20% data *testing* dimana jumlah data *training* adalah 239 dan data *testing* adalah 60. Model akan dilatih dengan data *training* dan diuji dengan data *testing* yang tidak pernah dilihat sebelumnya oleh model.

Kemudian, pada tahap ekstraksi fitur, metode yang digunakan adalah *Mel Frequency Cepstral Coefficients* (MFCC). Proses ini dimulai dengan mengubah representasi daya dari spektrum suara ke dalam unit desibel, yang membuat data lebih sesuai dengan persepsi manusia terhadap suara. Terdapat tiga parameter utama, yaitu data, sr, dan n_mfcc. Data merupakan daftar atau array yang berisi spektrum daya dari audio, di mana setiap elemen mewakili kekuatan frekuensi

dalam suatu segmen suara. Kedua, sr adalah sampling rate yang digunakan, yaitu jumlah sampel per detik dalam audio. Nilai default untuk sr adalah 22050 Hz, yang merupakan nilai umum digunakan karena mencakup frekuensi yang bisa didengar oleh manusia, sehingga seringkali cukup untuk berbagai aplikasi pemrosesan audio. Ketiga, n_mfcc menentukan jumlah koefisien *Mel-Frequency Cepstral Coefficients* (MFCC) yang akan diekstraksi dari setiap bingkai audio. Nilai defaultnya adalah 13, yang merupakan jumlah yang biasanya cukup untuk menangkap informasi penting dari spektrum frekuensi dalam bentuk yang lebih ringkas, cocok untuk tugas seperti pengenalan suara atau klasifikasi musik. Dengan ketiga parameter ini, ekstraksi fitur dapat menghasilkan serangkaian fitur yang menggambarkan karakteristik frekuensi dari audio secara detail dan terstruktur.

3.4.4 Klasifikasi dengan *Machine Learning*

Metode *machine learning* yang akan digunakan untuk klasifikasi dalam penelitian ini adalah CNN. Pada tahap ini, akan dilakukan pencarian parameter CNN terbaik, yaitu parameter dropout_rate, learning_rate, conv_filters_1, conv_filters_2. Kombinasi nilai yang akan digunakan adalah dropout_rate sebanyak 0.3, 0.5, 0.7, learning_rate sebanyak 0.01, 0.001, 0.0001, conv_filters_1 sebanyak 16, 32, 64, dan conv_filters_2 sebanyak 32, 64, 128. Pencarian kombinasi parameter terbaik tersebut akan dilakukan secara manual untuk seluruh pasangan kombinasi, sehingga akan dilakukan 12 percobaan. Dropout rate digunakan untuk mencegah model terlalu bergantung pada node-node tertentu dalam jaringan, sehingga meningkatkan kemampuan model untuk beradaptasi dengan data baru. Learning rate digunakan mengontrol seberapa cepat atau lambat model belajar dari data selama proses optimisasi. Convolutional filters (`conv_filters_1` dan `conv_filters_2`) untuk menentukan jumlah filter yang diterapkan pada lapisan konvolusi pertama dan kedua, masing-masing, untuk mengekstraksi fitur-fitur yang relevan dari representasi spasial data masukan. Hasil dari pencarian ini akan menghasilkan *best model* yang akan digunakan dalam tahap *testing*.

3.4.5 Evaluasi Data *Training*

Evaluasi data training pada klasifikasi CNN dilakukan dengan melatih model menggunakan data *training* selama 250 epochs dengan *batch size* 32 dan *validation*

split 1/12. Proses *training* menghasilkan objek yang akan menyimpan performa model setiap epoch. Setelah *training* tersebut, rata-rata akurasi pada data *training* dan data validasi dihitung. Evaluasi ini dilakukan untuk mengukur seberapa baik model belajar dari data *training,* yang kemudian diuji pada data *testing* untuk memastikan performa sebenarnya.

3.4.6 Evaluasi Data *Testing*

Model terbaik (*best model)* yang telah dihasilkan sebelumnya akan dievaluasi pada data *testing*. Proses evaluasi ini bertujuan untuk memastikan bahwa model tidak hanya memiliki performa yang baik pada data *training*, tetapi juga memiliki performa yang baik pada data yang benar-benar baru. Metrik evaluasi yang akan digunakan dalam proses evaluasi ini adalah *akurasi*, *precision*, *recall*, dan f1-*score*.

3.4.7 Evaluasi Sistem

Dalam penelitian ini, evaluasi sistem perangkat lunak (web) dilakukan menggunakan metode *Black Box Testing*, yang berfokus pada pengujian fungsionalitas tanpa memperhatikan struktur internal kode. Tiga teknik utama digunakan dalam pengujian ini :

a. Pertama, metode *Graph Based Testing*, yang memverifikasi hubungan antar objek dalam sistem dan memastikan setiap komponen berfungsi sesuai dengan relasinya, menunjukkan semua objek seperti *dashboard*, *upload* audio, dan hasil klasifikasi bekerja dengan baik tanpa masalah.
b. Kedua, metode *Equivalence Partitioning,* yang mengidentifikasi kelas kesalahan dan mengurangi jumlah *test case* dengan membagi domain input menjadi kelas-kelas data.
c. Ketiga, metode *Boundary Value Analysis*, yang fokus pada nilai batas input dan output untuk mengungkap cacat tersembunyi, memastikan bahwa sistem menangani nilai batas dengan benar.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi Tahap *Preprocessing* dan Ekstraksi Fitur

Implementasi tahap *preprocessing* dalam penelitian ini menggunakan bahasa pemrograman Python dan dijalankan pada platform Google Collab. Tahap *preprocessing* menggunakan library numpy untuk manipulasi data seperti padding dan konversi ke bentuk array dan library librosa untuk untuk memuat file audio, normalisasi, dan transformasi STFT. Tahapan *preprocessing* disimpan dalam dalam fungsi "preprocess_audio" dimana terdapat lima parameter yang didefinisikan, yaitu file_path untuk jalur file dari file audio yang akan diproses, sample_rate untuk frekuensi pengambilan sampel, frame_size untuk ukuran jendela Fourier, hop_length untuk langkah antar frame dalam STFT, dan max_length untuk durasi maksimum sinyal audio yang diproses untuk konsistensi panjang dengan nilai kembalian berupa 'powe', 'signal', dan 'sr'. Adapun nilai yang diatur untuk masing-masing parameter adalah sample_rate=22050, frame_size=2048, hop_length=512, dan max_length=5. Fungsi "preprocess_audio" terdiri dari lima langkah, yaitu:

4.1.1 Memuat File Audio

Perintah "librosa.load(file_path, sr=sample_rate)" digunakan untuk memuat file audio yang terletak di "file_path" menggunakan library librosa. Parameter "sr=sample_rate" mengatur sample rate (tingkat sampel) audio sesuai dengan nilai yang ditetapkan. Variabel "signal" akan menyimpan data audio yang dimuat dalam bentuk array numpy, yang mewakili amplitudo dari sinyal audio. Variabel "sr" akan menyimpan nilai sample rate yang digunakan untuk memuat audio tersebut, yaitu 22050 sampel per detik dalam penelitian ini. Adapun program untuk memuat file audio dalam penelitian ini terdapat pada gambar 4.1.

```
signal, sr = librosa.load(file_path, sr=sample_rate)
```

Gambar 4.1 *Preprocessing* Untuk Memuat File Audio

4.1.2 Menyesuaikan Panjang Sinyal

Tahapan ini bertujuan untuk memastikan konsistensi panjang sinyal audio sebelum dilakukan analisis lebih lanjut. Jika panjang sinyal audio yang dimuat kurang dari durasi maksimum atau 'max_length', proses akan menambahkan nilai nol di ujung

sinyal agar mencapai panjang yang diinginkan. Sebaliknya, jika panjang sinyal melebihi nilai 'max_length', sinyal akan dipotong agar hanya bagian awalnya yang digunakan sesuai dengan durasi maksimum yang ditentukan. Dalam penelitian ini 'max_length' diatur 5 detik yang berarti sinyal audio akan diproses atau dipotong sehingga tidak lebih dari 5 detik untuk memastikan konsistensi data sebelum dilakukan langkah-langkah analisis lebih lanjut. Adapun program untuk menyesuaikan panjang signal dalam penelitian ini terdapat pada gambar 4.2.

```
if len(signal) < sample_rate * max_length:
    # Pad the signal
    pad_width = sample_rate * max_length - len(signal)
    signal = np.pad(signal, (0, pad_width), mode='constant')
else:
    # Truncate the signal
    signal = signal[:sample_rate * max_length]
```

Gambar 4.2 *Preprocessing* Untuk Menyesuaikan Panjang Sinyal

4.1.3 Normalisasi Sinyal

Pada tahap ini, sinyal audio dinormalisasi menggunakan fungsi 'librosa.util.normalize', yang mengubah rentang nilai sinyal menjadi antara -1 dan 1. Normalisasi ini membantu menjaga kualitas sinyal dan meminimalkan distorsi yang mungkin terjadi selama proses analisis atau klasifikasi berikutnya, sehingga sinyal siap untuk tahap pemrosesan lebih lanjut. Adapun program untuk normalisasi sinyal dalam penelitian ini terdapat pada gambar 4.3.

```
signal = librosa.util.normalize(signal)
```

Gambar 4.3 *Preprocessing* Untuk Normalisasi Sinyal

4.1.4 Windowing dan Framing (STFT)

Pada tahap ini melakukan windowing dan framing pada sinyal audio menggunakan Short-Time Fourier Transform (STFT). STFT membagi sinyal menjadi segmen-segmen kecil yang memungkinkan representasi sinyal dalam domain frekuensi terhadap waktu. Dalam tahap ini terdapat dua parameter yang digunakan, yaitu frame_size dan hop_length. Nilai parameter frame_size yang digunakan adalah 2048 yang merupakan nilai *default* frame_size dan nilai parameter hop_length adalah 512. Dengan nilai 2048, masing-masing frame akan terdiri dari 2048 sampel audio dan dengan nilai hop_length 512 maka setiap frame akan digeser sebesar 512 sampel audio dari

frame sebelumnya. Adapun program untuk windowing dan framing dalam penelitian ini terdapat pada gambar 4.4.

```
stft = librosa.stft(signal, n_fft=frame_size, hop_length=hop_length)
```

Gambar 4.4 *Preprocessing* Untuk Windowing dan Framing (STFT)

4.1.5 Menghitung Magnitudo dan Power STFT

Pada tahap ini, magnitudo dari STFT dihitung dengan mengambil nilai absolut dari hasil STFT untuk setiap frame, menghasilkan magnitudo dari sinyal dalam domain frekuensi terhadap waktu. Selanjutnya, nilai power dari STFT dihitung dengan mengkuadratkan nilai magnitudo tersebut. Power menggambarkan distribusi energi sinyal audio dalam domain frekuensi terhadap waktu, memberikan representasi yang lebih kuat tentang distribusi energi frekuensi dalam sinyal audio. Adapun program untuk windowing dan framing dalam penelitian ini terdapat pada gambar 4.5.

```
magnitude = np.abs(stft)
power = magnitude**2
```

Gambar 4.5 *Preprocessing* Untuk Menghitung Magnitudo dan Power STFT

Setelah melakukan *preprocessing* pada data audio musik, langkah berikutnya adalah menjalankan proses ekstraksi fitur menggunakan MFCC (Mel Frequency Cepstral Coefficients). Tahap ekstraksi fitur disimpan dalam fungsi ‘extract_features’ dengan nilai parameter n_mfcc=13 dan sr=22050 fungsi ini mengambil ‘data’ yang telah melalui tahap *preprocessing* dan kemudian menghitung MFCC dari setiap frame menggunakan ‘librosa.feature.mfcc’. Dalam penelitian ini, `librosa.power_to_db(power)` digunakan untuk mengkonversi nilai power dari STFT ke skala desibel sebelum menghitung MFCC. Parameter ‘sr’ menentukan sample rate yang digunakan, dan ‘n_mfcc’ mengatur jumlah koefisien MFCC yang diekstraksi dari setiap frame audio. Hasil ekstraksi fitur berupa array MFCC untuk setiap frame yang digunakan sebagai input untuk tahap berikutnya dalam analisis atau pengklasifikasian genre musik. Adapun program ekstraksi fitur dengan MFCC dalam penelitian ini terdapat pada gambar 4.6.

```python
def extract_features(data, sr=22050, n_mfcc=13):
    features = []
    for power in data:
        # Compute MFCCs from the power of the STFT
        mfccs = librosa.feature.mfcc(S=librosa.power_to_db(power), sr=sr, n_mfcc=n_mfcc)
        features.append(np.expand_dims(mfccs, axis=-1))
    return np.array(features)

# Extract MFCC features
features = extract_features(data)
```

Gambar 4.6 Ekstraksi Fitur dengan MFCC

Keseluruhan program pada tahap *preprocessing* dan ekstraksi fitur sebagai berikut :

```python
def preprocess_audio(file_path, sample_rate=22050,
frame_size=2048, hop_length=512, max_length=5):
    try:
        # Load the audio file
        signal, sr = librosa.load(file_path, sr=sample_rate)

        # Ensure the signal length is consistent
        if len(signal) < sample_rate * max_length:
            # Pad the signal
            pad_width = sample_rate * max_length - len(signal)
            signal = np.pad(signal, (0, pad_width),
mode='constant')
        else:
            # Truncate the signal
            signal = signal[:sample_rate * max_length]

        # Normalize the signal
        signal = librosa.util.normalize(signal)

        # Windowing and framing (STFT)
        stft = librosa.stft(signal, n_fft=frame_size,
hop_length=hop_length)

        # Compute the magnitude of the STFT
        magnitude = np.abs(stft)
        power = magnitude**2
        return power, sr, signal
    except Exception as e:
        print(f"Error processing {file_path}: {e}")
```

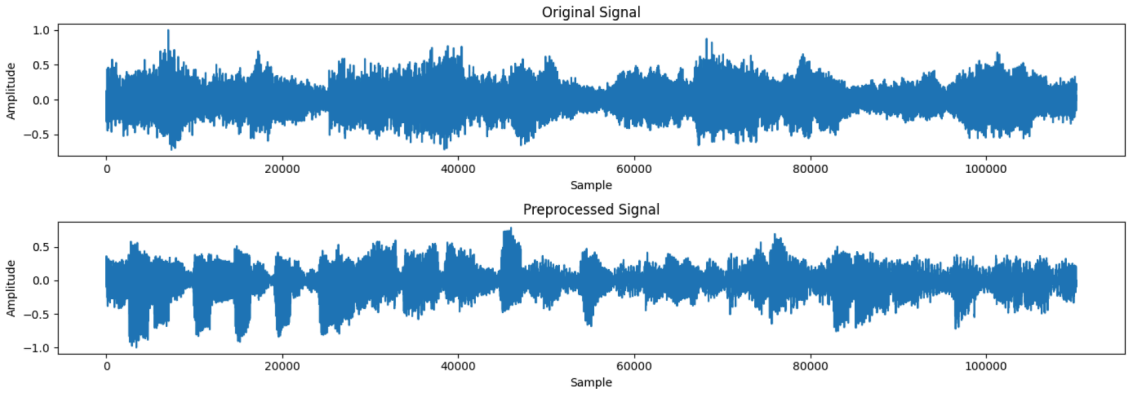
```
        return None, None, None

def extract_features(data, sr=22050, n_mfcc=13):
    features = []
    for power in data:
        # Compute MFCCs from the power of the STFT
        mfccs = librosa.feature.mfcc(S=librosa.power_to_db(power),
sr=sr, n_mfcc=n_mfcc)
        features.append(np.expand_dims(mfccs, axis=-1))
    return np.array(features)

# Extract MFCC features
features = extract_features(data)
```

4.2 Hasil dan Pembahasan Tahap Preprocessing dan Ekstraksi Fitur

Tahap *preprocessing* sebelumnya akan memberikan hasil pada salah satu data audio musik seperti pada Gambar 4.6.



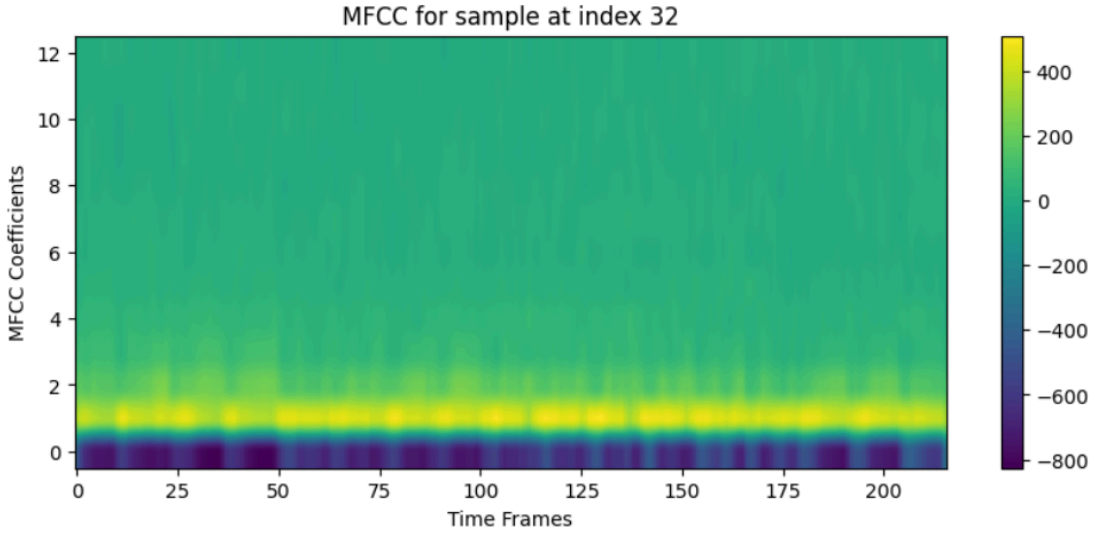
Gambar 4.7 Hasil Tahap *Preprocessing* pada Salah Satu Data

Selanjutnya, tahap ekstraksi fitur dengan metode MFCC akan menghasilkan output fitur MFCC sebanyak 2808 fitur untuk satu dokumen seperti pada gambar 4.8.

```
MFCC dimensions for data at index 32: (13, 216, 1)
- 13 MFCC coefficients
- 216 time frames
- 1 channel
Total features for this document: 2808
```

Gambar 4.8 Jumlah Fitur MFCC pada Salah Satu Data

Dokumen audio yang diproses adalah 5 detik pertama dari audio dengan durasi 30 detik (max_length = 5). Dalam periode 5 detik tersebut, audio diproses menjadi 216 *frame* waktu, dan setiap frame direpresentasikan oleh 13 koefisien MFCC, yang merupakan fitur umum digunakan untuk analisis audio. Jumlah *frame* tersebut bergantung pada durasi video, frame_size, dan hop length. 1 channel menunjukkan bahwa audio tersebut adalah mono (satu saluran). Hasil akhirnya adalah 2808 fitur yang mewakili dokumen audio tersebut. Total fitur didapatkan dari hasil perkalian jumlah koefisien MFCC dengan jumlah frame waktu, yaitu 13 x 216 = 2808 fitur. Contoh bentuk representasi hasil MFCC pada salah satu data dapat dilihat pada Gambar 4.9.



Gambar 4.9 Representasi MFCC pada Salah Satu Data

4.3 *Splitting* Data

Splitting data adalah tahapan membagi data menjadi dua atau lebih bagian untuk tujuan yang berbeda selama pelatihan dan evaluasi model *machine learning*. Dalam penelitian data dibagi menjadi dua bagian, yaitu data latih dan data uji dengan rasio 80% data latih dan 20% data uji. *Splitting* data dalam penelitian ini menggunakan program pada gambar 4.10. Dengan langkah pertama, label kelas yang berupa kategori diubah menjadi bentuk numerik menggunakan `LabelEncoder`, kemudian label yang telah di-encode diubah menjadi representasi one-hot encoding menggunakan `to_categorical`. Setelah itu, data fitur ’features’ dan label yang telah diubah ‘labels_categorical’ dibagi

menjadi data latih dan data uji menggunakan `train_test_split`, dengan proporsi 80% untuk pelatihan dan 20% untuk pengujian serta 'random_state' diatur ke 42 untuk memastikan hasil pembagian yang konsisten. Dalam pembagian ini, terdapat 239 sampel dalam set pelatihan (`X_train`) dan 60 sampel dalam set pengujian (`X_test`).

```
# Encode labels
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(labels)
labels_categorical = to_categorical(labels_encoded)

# Split data
X_train, X_test, y_train, y_test = train_test_split(features, labels_categorical, test_size=0.2, random_state=42)
```

Gambar 4.10 *Splitting* Data

4.4 Implementasi *Machine Learning*

Machine learning yang digunakan untuk klasifikasi genre musik dalam penelitian ini adalah Convolutional Neural Network (CNN). Dalam tahap implementasi yang dilakukan pertama adalah membuat model dengan program yang ada pada gambar 4.11.

```
def create_model(dropout_rate=0.5, learning_rate=0.001, conv_filters_1=32, conv_filters_2=64):
    model = Sequential()
    model.add(Conv2D(conv_filters_1, kernel_size=(3, 3), activation='relu', input_shape=(X_train.shape[1], X_train.shape[2], 1)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(dropout_rate))
    model.add(Conv2D(conv_filters_2, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(dropout_rate))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(len(class_labels), activation='softmax'))

    # Compile the model
    optimizer = Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

    return model
```

Gambar 4.11 Pembuatan Model CNN

Pembuatan model disimpan dalam fungsi 'create_model' dimana terdapat beberapa layer di dalamnya. Layer pertama menggunakan `Conv2D` untuk menerapkan konvolusi pada data MFCC yang direpresentasikan sebagai vektor 2D. Dengan `conv_filters_1` filter berukuran 3x3, layer ini mengekstrak fitur-fitur penting dari setiap frame MFCC dalam format satu channel (grayscale). Activation function ReLU (Rectified Linear Unit) digunakan untuk memperkenalkan non-linearitas, memungkinkan jaringan untuk belajar representasi yang lebih kompleks dari data input. Parameter 'input_shape' menyesuaikan dimensi data masukan dengan '(X_train.shape[1], X_train.shape[2], 1)', di mana 'X_train.shape[1]' adalah jumlah koefisien MFCC per

frame, ‘X_train.shape[2]’ adalah jumlah frame waktu, dan ‘1’ menunjukkan satu channel untuk setiap frame MFCC.

Proses berlanjut dengan penambahan layer `MaxPooling2D`. Layer ini memiliki parameter ‘pool_size=(2, 2)’ yang bertujuan untuk melakukan pengurangan dimensi dari output yang dihasilkan oleh layer konvolusi sebelumnya. Ini dilakukan dengan mengambil nilai maksimum dari setiap window 2x2 dalam grid input. Tujuan dari MaxPooling adalah untuk mengurangi jumlah parameter model dan meningkatkan invariansi terhadap translasi dalam data.

Setelah MaxPooling, layer ‘Dropout’ ditambahkan dengan tingkat dropout yang diberikan (`dropout_rate`). Dropout berfungsi untuk secara acak menonaktifkan sebagian unit selama pelatihan, yang membantu mencegah model menjadi terlalu tergantung pada data pelatihan, sehingga memungkinkan model untuk lebih umum dan lebih baik dalam melakukan klasifikasi pada data yang tidak terlihat.

Setelah itu, dua layer ‘Conv2D’ lagi diterapkan dengan konfigurasi yang serupa untuk terus mengekstrak fitur-fitur yang lebih kompleks dari data yang makin terurut. Kemudian, setelah layer konvolusi kedua, proses dilanjutkan dengan MaxPooling dan Dropout kembali untuk mempertahankan dan meningkatkan kemampuan generalisasi model. Setelah semua operasi konvolusi dan pooling, output dari layer konvolusi dan pooling akan di-flatten menggunakan layer ‘Flatten()’, yang mengubah data dari format matriks ke vektor satu dimensi. Ini diperlukan karena layer-layer Dense yang akan datang membutuhkan input berupa vektor satu dimensi.

Kemudian, dua layer ‘Dense’ ditambahkan, layer pertama dengan 128 neuron dan aktivasi ReLU, yang bertujuan untuk memproses hasil dari fitur-fitur yang telah diekstraksi sebelumnya. Layer terakhir menggunakan jumlah neuron yang sesuai dengan jumlah kelas (`len(class_labels)`) dengan aktivasi softmax, yang cocok untuk kasus klasifikasi multi-kelas, seperti pengenalan genre musik dalam kasus Anda.

Terakhir, model dikompilasi dengan optimizer Adam yang disesuaikan dengan learning rate yang ditentukan (`learning_rate`). Loss function yang dipilih adalah `categorical_crossentropy`, yang sesuai untuk masalah klasifikasi multi-kelas, sementara metrik yang digunakan untuk evaluasi adalah akurasi (`accuracy`).

Nilai dari parameter yang digunakan adalah hasil *tunning hyper-parameter* pada subbab 4.5 dimana didapatkan nilai parameter terbaik yaitu, 'dropout_rate' sebesar 0.5, 'learning_rate' sebesar 0.001, 'conv_filters_1' sebesar 32, dan 'conv_filters_2' sebesar 64. Dari nilai dari masing-masing parameter tersebut dan fungsi 'create_model' yang telah dibuat sebelumnya model dapat dibangun dengan menggabungkan kedua hal tersebut sehingga didapatkan model terbaik dalam penelitian ini untuk klasifikasi genre musik *pop*, *jazz*, dan *classical*. Adapun penerapannya program pembangunan model tersebut dalam penelitian ini terdapat pada gambar 4.12.

```
model = create_model(dropout_rate=dropout_rate, learning_rate=learning_rate,
                     conv_filters_1=conv_filters_1, conv_filters_2=conv_filters_2)
```

Gambar 4.12 Implementasi Model CNN

4.5 Implementasi Evaluasi *Training* dan *Testing*

Implementasi evaluasi training dan testing dilakukan untuk menilai kinerja model CNN dalam klasifikasi genre musik. Pertama, data *training* (X_train) dan data *testing* (X_test) diperluas dimensinya menjadi 3D untuk sesuai dengan format input model CNN. Model kemudian dibuat dan dilatih menggunakan data *training* yang telah disiapkan dengan 250 epoch dan batch size 32. Selama *training*, sebagian kecil data disisihkan untuk validasi guna memantau kinerja model secara *real-time*. Setelah training selesai, rata-rata akurasi pada data *training* dihitung dan ditampilkan seperti pada Gambar 4.13.

```
train_acc = np.mean(history.history['accuracy'])
print(f"Average training accuracy: {train_acc}")
```

Gambar 4.13 Evaluasi pada Data *Training*

Model yang telah dilatih kemudian disimpan ke file 'model_trained.h5' untuk digunakan kembali di masa depan dalam format .h5 seperti kode pada Gambar 4.14.

```
model.save('model_trained.h5')
print("Model berhasil disimpan.")
```

Gambar 4.14 Penyimpanan Model

Langkah berikutnya adalah menggunakan model yang telah dilatih untuk memprediksi hasil pada data *testing* dengan kode seperti pada Gambar 4.15. Prediksi ini dibandingkan dengan label sebenarnya untuk menghitung metrik evaluasi seperti *precision, recall, f1-score*, dan akurasi secara keseluruhan melalui laporan klasifikasi.

```
# Predict the test set results
y_pred = model.predict(X_test_cnn)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

# Print classification report
print(classification_report(y_true, y_pred_classes, target_names=class_labels))
```

Gambar 4.15 Evaluasi pada Data *Testing*

4.6 Hasil dan Pembahasan Tahap *Training* (Eksperimen *Tuning Hyper-Parameter)*

Pada tahap *training*, dilakukan eksperimen untuk mencari parameter terbaik dari metode CNN sehingga dapat menghasilkan *best model.* Epoch yang digunakan pada tahap *training* adalah 250 dengan *batch size* 32 dan *validation split* 1/12. Pada eksperimen tersebut menggunakan 4 parameter, yaitu dropout_rate, learning_rate, conv_filters_1, conv_filters_2. Kombinasi nilai yang akan digunakan adalah dropout_rate sebanyak 0.3, 0.5, 0.7, learning_rate sebanyak 0.01, 0.001, 0.0001, conv_filters_1 sebanyak 16, 32, 64, dan conv_filters_2 sebanyak 32, 64, 128. Pencarian kombinasi parameter terbaik ini dilakukan secara manual dan *random*, yaitu hanya melakukan sekitar 12 percobaan. Hasil akurasi untuk kombinasi parameter CNN pada data *training* dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil Akurasi Kombinasi Parameter CNN

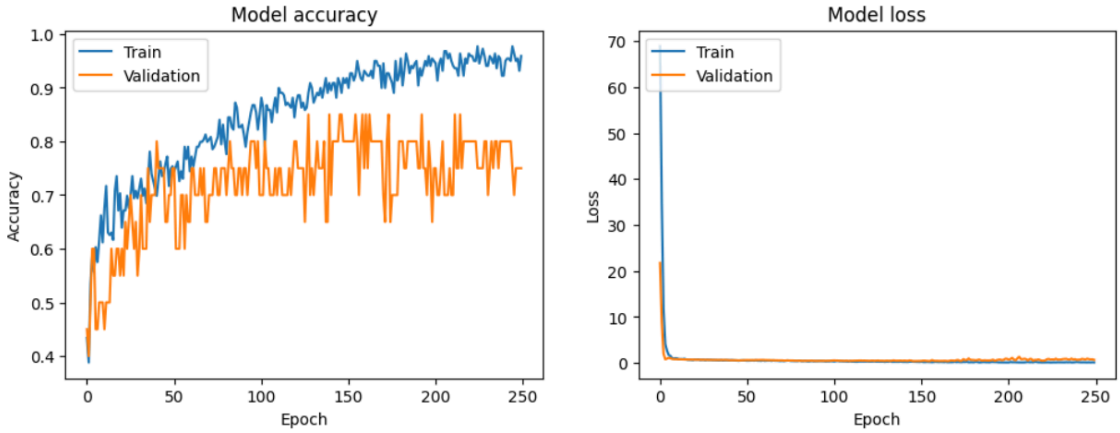
No	dropout_rate	learning_rate	conv_filters_1	conv_filters_2	Akurasi Rata-Rata
1	0.3	0.01	16	32	70.5%
2	0.3	0.001	32	64	72.34%
3	0.3	0.0001	64	128	73%
4	0.3	0.001	32	64	77.65%
5	0.5	0.01	16	32	60.2%
6	**0.5**	**0.001**	**32**	**64**	**84.85%**
7	0.5	0.0001	64	128	79.2%
8	0.5	0.001	32	32	74%

9	0.7	0.01	16	64	65.40%
10	0.7	0.001	32	126	60.78%
11	0.7	0.0001	64	32	66.98%
12	0.7	0.001	32	64	76.87%

Dari Tabel 4.1 tersebut, ditunjukkan bahwa kombinasi parameter terbaik untuk CNN adalah dropout_rate sebesar 0.5, learning_rate sebesar 0.001, conv_filters_1 sebesar 32, dan conv_filters_2 sebesar 64. Maka, didapatkan *best model* yang akan digunakan selanjutnya pada tahap *testing*. Hasil evaluasi untuk *best model* tersebut pada data *training* untuk 10 epoch terakhir dapat dilihat pada Gambar 4.16. Kemudian, grafik plot *training* dan *validation* dapat dilihat pada Gambar 4.17. Dari grafik tersebut terlihat bahwa akurasi model semakin meningkat untuk setiap peningkatan epochnya, begitu pula dengan *model loss* yang semakin menurun untuk setiap peningkatan epochnya.

```
Epoch 240/250
7/7 [==============================] - 1s 132ms/step - loss: 0.1931 - accuracy: 0.9224 - val_loss: 0.6674 - val_accuracy: 0.8000
Epoch 241/250
7/7 [==============================] - 1s 125ms/step - loss: 0.1492 - accuracy: 0.9498 - val_loss: 0.6800 - val_accuracy: 0.8000
Epoch 242/250
7/7 [==============================] - 1s 79ms/step - loss: 0.1313 - accuracy: 0.9543 - val_loss: 0.7121 - val_accuracy: 0.8000
Epoch 243/250
7/7 [==============================] - 1s 81ms/step - loss: 0.1168 - accuracy: 0.9543 - val_loss: 0.8732 - val_accuracy: 0.8000
Epoch 244/250
7/7 [==============================] - 1s 80ms/step - loss: 0.1321 - accuracy: 0.9498 - val_loss: 0.8188 - val_accuracy: 0.8000
Epoch 245/250
7/7 [==============================] - 1s 73ms/step - loss: 0.1171 - accuracy: 0.9772 - val_loss: 0.7300 - val_accuracy: 0.7500
Epoch 246/250
7/7 [==============================] - 1s 81ms/step - loss: 0.1028 - accuracy: 0.9635 - val_loss: 0.9231 - val_accuracy: 0.7000
Epoch 247/250
7/7 [==============================] - 1s 81ms/step - loss: 0.1313 - accuracy: 0.9498 - val_loss: 0.8103 - val_accuracy: 0.7500
Epoch 248/250
7/7 [==============================] - 1s 82ms/step - loss: 0.1006 - accuracy: 0.9543 - val_loss: 0.7616 - val_accuracy: 0.7500
Epoch 249/250
7/7 [==============================] - 1s 79ms/step - loss: 0.1320 - accuracy: 0.9315 - val_loss: 0.7894 - val_accuracy: 0.7500
Epoch 250/250
7/7 [==============================] - 1s 83ms/step - loss: 0.1102 - accuracy: 0.9589 - val_loss: 0.6575 - val_accuracy: 0.7500
Average training accuracy: 0.8485296807289123
```

Gambar 4.16 Hasil Evaluasi Data *Training*



Gambar 4.17 Grafik *Model Accuracy* dan *Model Loss*

4.7 Hasil dan Pembahasan Tahap *Testing* Model

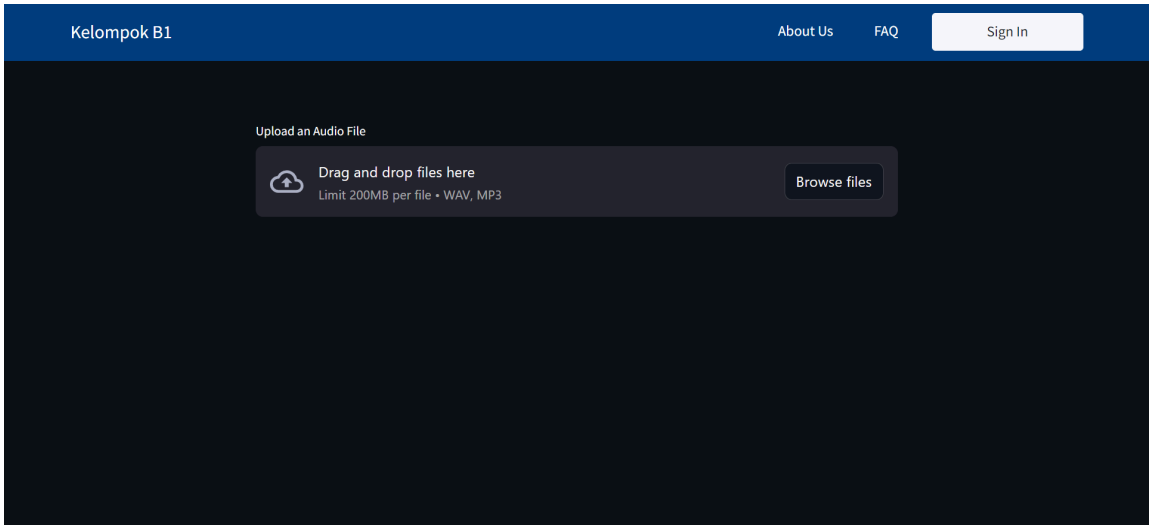
Dengan menggunakan *best model* yang telah didapatkan pada tahap sebelumnya, dilakukan evaluasi pada data *testing*. Parameter yang digunakan adalah dropout_rate sebesar 0.5, learning_rate sebesar 0.001, conv_filters_1 sebesar 32, dan conv_filters_2 sebesar 64. Hasil evaluasi untuk *best model* tersebut pada data *testing* dapat dilihat pada Gambar 4.18.

	precision	recall	f1-score	support
classical	0.92	0.96	0.94	23
jazz	0.95	0.86	0.90	21
pop	0.94	1.00	0.97	16
accuracy			0.93	60
macro avg	0.94	0.94	0.94	60
weighted avg	0.93	0.93	0.93	60

Gambar 4.18 Hasil *Classification Report* Data *Testing* dengan *Best Model*

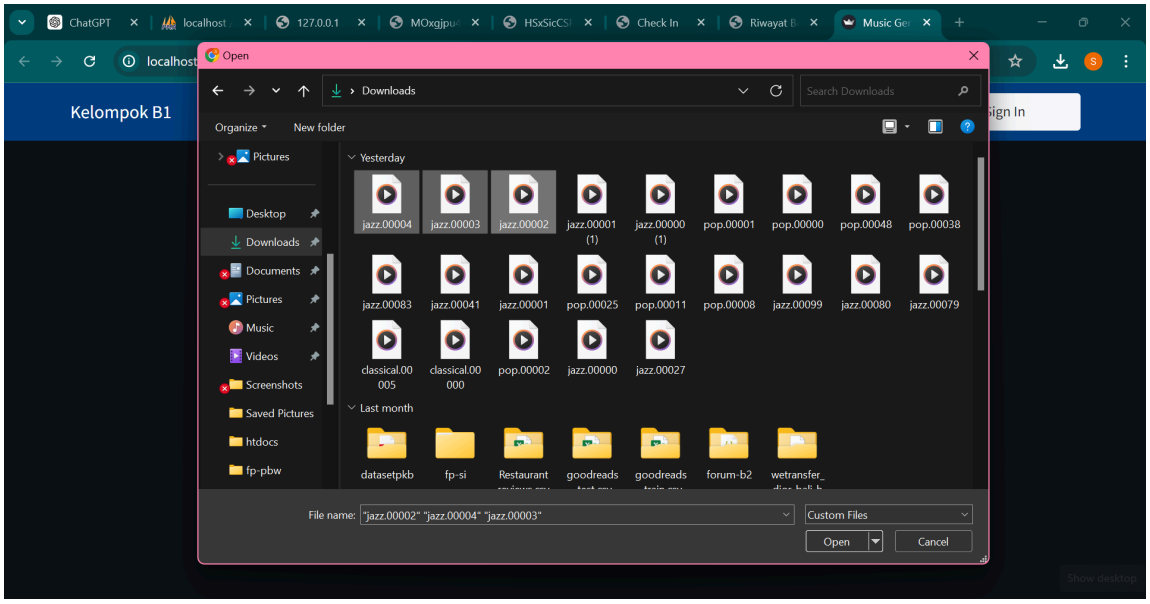
Berdasarkan gambar tersebut, ditunjukkan bahwa hasil akurasi rata-rata pada data *testing* sebesar 93%, dengan rata-rata *precision* sebesar 93%, *recall* sebesar 93%, dan F1-*score* sebesar 93%. Di mana genre pop mendapatkan hasil akurasi tertinggi sebesar 97%, diikuti dengan classical sebesar 94% dan jazz sebesar 90%. Hasil akurasi yang tersebut menunjukkan bahwa metode ekstraksi fitur MFCC dan klasifikasi CNN dapat melakukan klasifikasi ekspresi wajah.

4.8 Hasil Antarmuka Fitur Aplikasi

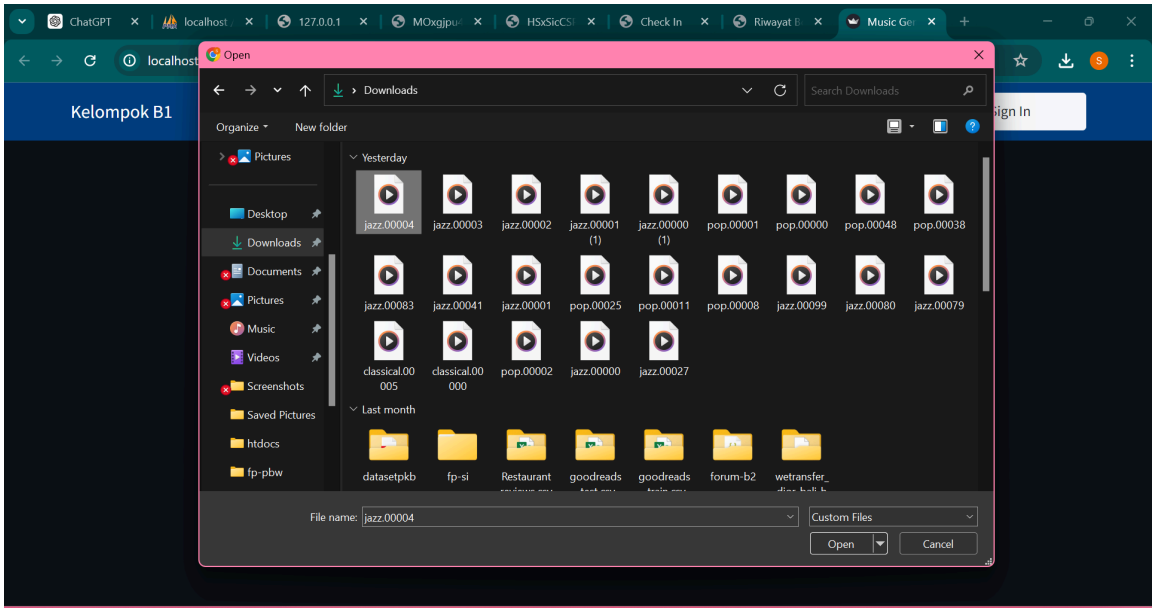


Gambar 4.19 Tampilan Awal Website

Tampilan awal *website* dapat dilihat pada Gambar 4.19, terdapat header yang berisikan nama website dan menu-menu lain seperti about us, FAQ dan tombol sign in, untuk sementara ini semua fitur ini belum tersedia. *Website* ini dapat digunakan secara gratis tanpa perlu bayar atau login terlebih dahulu.

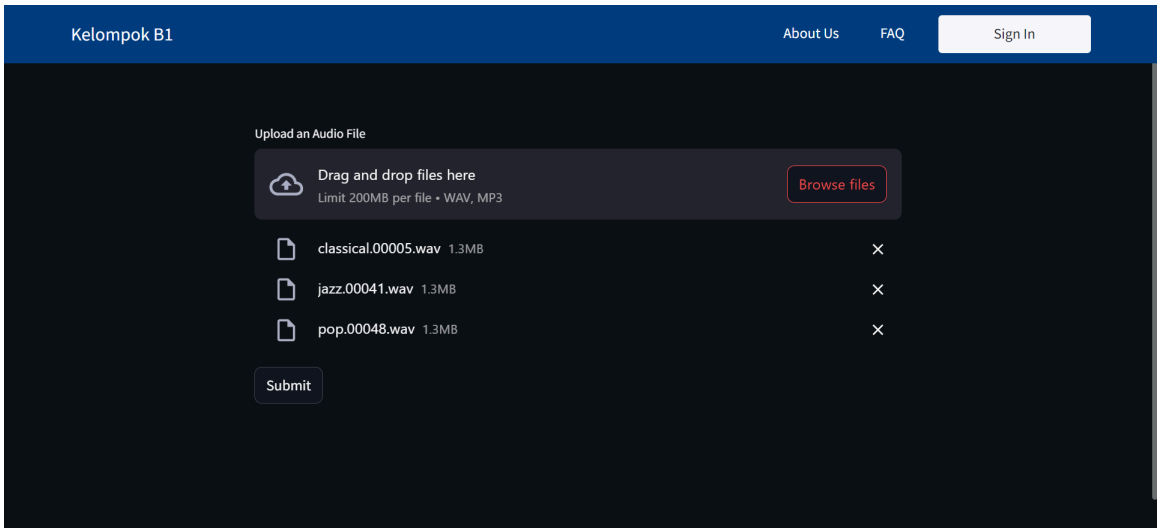


Gambar 4.20 Tampilan Pemilihan Audio *Multiple*



Gambar 4.21 Tampilan Pemilihan Audio *single*

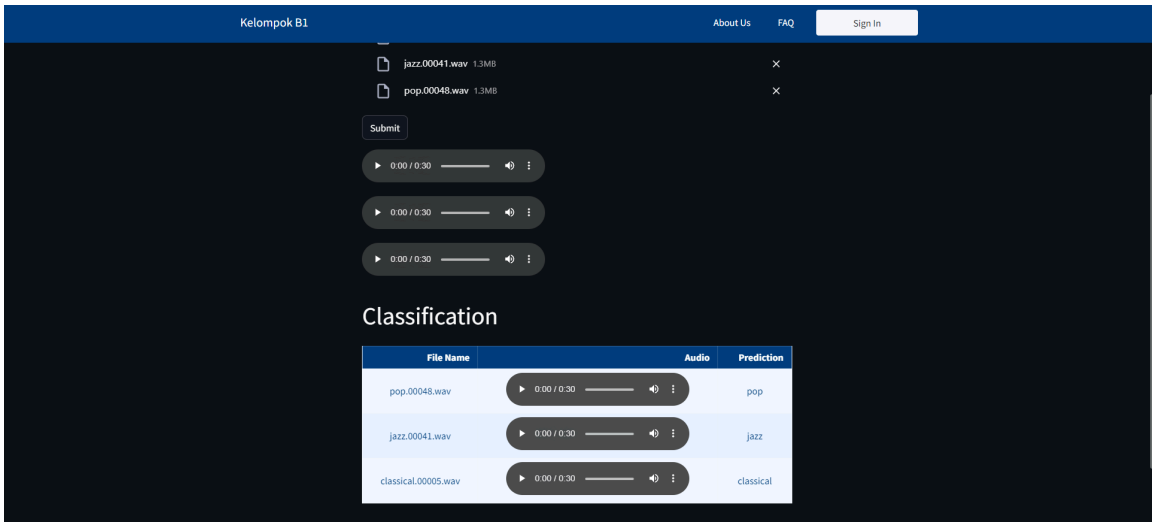
Pengguna dapat memasukan/menginputkan banyak audio (Gambar 4.20) atau satu audio (Gambar 4.21) dengan menekan tombol *browse file* atau melakukan *drag and drop* pada tempat yang sudah disediakan. Format audio yang akan diterima adalah .WAV dan .MP3 dengan *limit* per *file* adalah 200MB



Gambar 4.22 Tampilan Audio Sudah di Upload

Audio yang sudah berhasil di *upload* akan muncul pada bawah area *drag dan drop*. Audio yang sudah di *upload* dapat dihapuskan dengan menekan tombol *close* atau x, dengan begitu file audio akan otomatis dihapuskan. Tombol Submit hanya akan muncul jika pengguna sudah berhasil mengupload audio, dengan menekan tombol *submit*

audio yang sudah diupload akan langsung diklasifikasikan. Audio yang sudah di klasifikasikan akan ditampilkan pada tabel ini serta prediksi genre dari audio yang sudah di-*upload* seperti pada Gambar 4.23.



Gambar 4.23 Tampilan Hasil Klasifikasi

Pengguna dapat meng-*upload* audio dengan langkah-langkah yang sama tanpa perlu khawatir audio yang pertama kali di upload akan hilang, karena audio akan tetap berada pada tabel terkecuali pengguna melakukan *reload page*.

4.9 Hasil Evaluasi Sistem Perangkat Lunak (*Black Box Testing*)

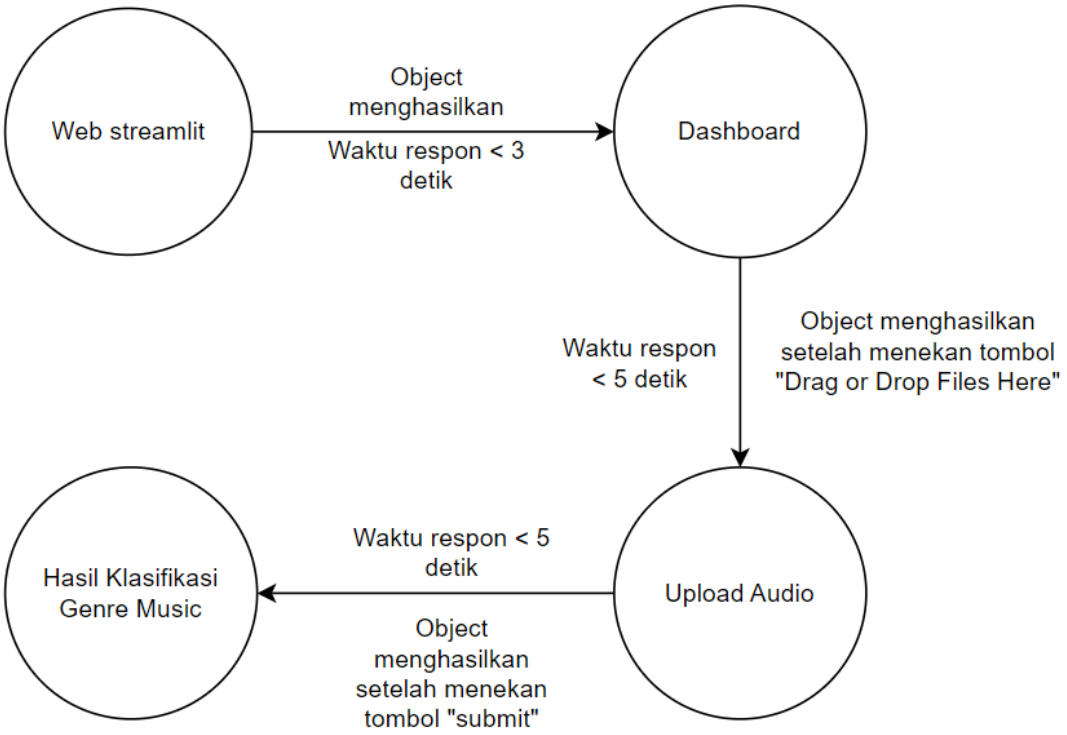
Hasil evaluasi sistem perangkat lunak yang digunakan dalam penelitian ini yaitu dengan *Black Box Testing* yang merupakan metode pengujian perangkat lunak di mana tester menguji fungsionalitas sistem tanpa mengetahui rincian internal atau struktur kode sumbernya. Dalam *black box testing*, fokusnya adalah pada input yang diberikan dan output yang dihasilkan tanpa memperhatikan bagaimana program mencapai hasil tersebut. Adapun beberapa metode atau teknik pengujian *black box* sebagai berikut :

4.8.1 Metode *Graph Based*

Pada teknik/metode ini langkah yang dilakukan adalah memahami objek (data dan program) yang dimodelkan di dalam perangkat lunak. Langkah selanjutnya menentukan sederetan pengujian yang membuktikan bahwa semua objek memiliki hubungan antara satu dengan lainnya. Tabel perancangan *test case* dan hasil pengujian *graph based testing* pada web dapat dilihat pada Tabel 4.1.

Tabel 4.2 Tabel Rancangan *Test Case* dan Hasil Pengujian *Graph Based* Testing

Nomor Pengujian	Deskripsi Pengujian	Masukan	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
A1	Menguji *link streamlit*	Mengetik alamat "..." pada *web browser*	Muncul halaman *dashboard* untuk *user*	Muncul halaman *dashboard* untuk *user*	Sesuai
A2	*User* menekan tombol "Drag or Drop File"	Menekan tombol "Drag or Drop File"	Sistem menerima dan mengupload audio pada sistem	Sistem menerima dan mengupload audio pada sistem	Sesuai
A3	User menekan tombol "submit"	Menekan tombol "submit"	Sistem menerima dan menampilkan hasil klasifikasi genre audio	Sistem menerima dan menampilkan hasil klasifikasi genre audio	Sesuai



Gambar 4.24 *Black Box Testing Web* Dengan Metode *Graph Based*

Berdasarkan gambar 4.24 di atas, terdapat beberapa *object* yang ada di web streamlit yaitu *dashboard*, *upload audio*, dan hasil klasifikasi genre musik yang tiap *object*-nya dihubungkan oleh relasi dapat membuktikan bahwa semua *objects* memiliki hubungan antara satu dengan lainnya.

Berdasarkan hasil pengujian *Black Box Testing* dengan menggunakan metode *Graph Based Testing* terhadap web penelitian ini, dapat disimpulkan bahwa di dalam pengujian ini tidak ditemukannya masalah atau bisa dikatakan berjalan dengan baik dan sesuai.

4.8.2 Metode *Equivalence Partitioning*

Metode *Equivalence Partitioning* merupakan *test case* yang mengungkapkan kelas kesalahan sehingga mengurangi jumlah total *test case* yang harus dikembangkan. Metode ini membagi domain input dari suatu program kedalam kelas - kelas data sehingga *test case* dapat diperoleh. Kelas data yang terbentuk disajikan sebagai kondisi input dalam kasus uji. Desain *test case* partisi ekivalensi didasarkan pada evaluasi terhadap kelas ekivalensi untuk suatu kondisi

input. Rancangan *test case* dengan metode *Equivalence Partitioning* pada web penelitian ini dapat dilihat pada Tabel 4.2.

Tabel 4.3 Tabel Rancangan *Test Case* dan Hasil Pengujian *Equivalence Partitioning Testing* Pada Web

Nomor Pengujian	*Test Case*	Hasil yang Diharapkan	Hasil Pengujian
B1	Menekan tombol "Drag or Drop File Here" untuk memilih *file* audio yang akan di-*upload*	Sistem dapat menerima audio	Sistem dapat menerima audio
B2	Memilih *file* audio untuk di-*upload*	Sistem dapat menerima *file* audio	Sistem dapat menerima *file* audio
B3	Memilih lebih dari 1 *file* audio untuk di-*upload*	Sistem dapat menerima lebih dari satu *file*	Sistem dapat menerima lebih dari satu *file*
B4	Memilih 1 *file* audio lebih dari 200 MB untuk di-*upload*	*File* pada sistem tidak dapat diupload	*File* pada sistem tidak dapat diupload
B5	Memilih *file* audio yang bukan musik untuk di-*upload*	Sistem akan mengeluarkan output "invalid label"	Sistem mengeluarkan output berupa label genre musik "jazz", "classical", atau "pop"
B6	Memilih *file* audio dengan format .mp3	Sistem dapat menerima *file* audio	Sistem dapat menerima *file* audio

	dan WAV		
B7	Meng-*upload audio* dengan menekan tombol “submit”	Sistem dapat menampilkan hasil klasifikasi	Sistem dapat menampilkan hasil klasifikasi

Tabel 4.4 Tabel Kesimpulan Hasil Pengujian *Equivalence Partitioning*

ID Testing	Kesimpulan
B1	Berhasil
B2	Berhasil
B3	Berhasil
B4	Berhasil
B5	Tidak Berhasil
B6	Berhasil
B7	Berhasil

Total jumlah pengujian pada kajian ini adalah 7 pengujian pada 2 fungsi, yaitu fungsi *upload* audio dan menampilkan hasil klasifikasi. Fungsi *upload* audio diuji sebanyak 6 kali yang menghasilkan gagal 1 kali pada *test case* melakukan *upload* audio dengan mengosongkan seluruh field. Sedangkan, fungsi menampilkan hasil klasifikasi diuji sebanyak 2 kali yang menghasilkan gagal 1 kali pada *test case*.

Hasil pengujian menunjukkan kualitas web pada penelitian ini masih perlu ditingkatkan lagi, khususnya pada fungsionalitas menampilkan hasil klasifikasi agar dapat memberikan pesan error jika gambar yang dimasukkan tidak sesuai. Harapannya adanya kekurangan tersebut dapat diperbaiki supaya kualitas web yang dibuat menjadi lebih baik.

4.8.3 *Boundary Value Analysis*

Boundary Value fokus pada suatu batasan nilai dimana kemungkinan terdapat cacat yang tersembunyi. Teknik ini mengarahkan pada pemilihan kasus uji yang melatih nilai-nilai batas. Boundary Value merupakan desain teknik kasus uji yang melengkapi *Equivalence Class Testing* yang daripada memfokuskan hanya pada kondisi input, BVA juga menghasilkan kasus uji dari domain output serta menguji untuk input di sekitar batas atas maupun bawah sebuah range nilai yang valid. Pengujian metode *boundary value analysis* pada web penelitian ini dapat dilihat pada tabel 4.7.

Tabel 4.5 Pengujian Field *Upload* Audio Metode *Boundary Value Analysis*

ID Testing	Sample Data	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
C1	.mp3	T	T	Berhasil
C2	.WAV	T	T	Berhasil
C3	.FLAC	F	F	Berhasil
C4	.mp4	F	F	Berhasil

Link Github (Data, Coding, dan link Video Presentasi Proyek dan Demo Aplikasi) :
https://github.com/shelomitaa20/PPDM-Klasifikasi-Genre-Musik

BAB V
KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan sebelumnya, maka dapat ditarik kesimpulan sebagai berikut :

1. Akurasi rata-rata yang dihasilkan dari penggunaan metode ekstraksi fitur MFCC dan metode klasifikasi CNN dalam klasifikasi genre musik jazz, classical, dan pop dengan menggunakan *best model* adalah sebesar 93%, dengan rata-rata *precision* sebesar 93%, *recall* sebesar 93%, dan F1-*score* sebesar 93%.
2. Parameter terbaik pada metode CNN yang dihasilkan dalam eksperimen *Tuning Hyper-Parameter* untuk mendapatkan hasil akurasi terbaik pada klasifikasi genre musik adalah dropout_rate sebesar 0.5, learning_rate sebesar 0.001, conv_filters_1 sebesar 32, dan conv_filters_2 sebesar 64.

DAFTAR PUSTAKA

Allugunti, V.R. (2022). A machine learning model for skin disease classification using convolution neural network. Journal of Computing, Programming and Database.

Arthana, R. "Mengenal Accuracy, Precision, Recall dan Specificity serta yang Diprioritaskan dalam Machine Learning," Medium, 5 April 2019. [Online]. Available: https://rey1024.medium.com/mengenal-accuracy-precission-recall-dan-specificity-serta-yang-diprioritaskan-b79ff4d77de8. [Accessed 5 May 2024].

Chiu, C., Sainath, T. N., Wu, Y., Prabhavalkar, R., & Nguyen, P. (2022). State-of-the-art speech recognition with sequence-to-sequence models. *IEEE Transactions on Audio, Speech, and Language Processing*, 30, 571-584.

Clara, D.G.A. Inilah Beberapa Machine Learning Algorithm yang Wajib Diketahui, 3 Januari 2024. [Online]. Available: https://bce.telkomuniversity.ac.id/inilah-beberapa-machine-learning-algorithm-yang-wajib-diketahui/. [Accessed 14 Mei 2024]

Gupta, R., Khatri, A., & Singh, S. (2021). Enhanced feature extraction using MFCC for robust speech recognition. *Journal of Computer Science and Technology*, 36(3), 445-457.

Gessle, G., & Åkesson, S. (2019). A comparative analysis of CNN and LSTM for music genre classification.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T. (2021). 1D convolutional neural networks and applications: A survey. Mechanical Systems and Signal Processing.

Li, Z., & Huang, Y. (2023). Advances in MFCC-based speech processing techniques for automated speech recognition. *Journal of Electrical and Computer Engineering*, 2023, 1-10.

Lee, H., Lee, C., & Kim, S. (2023). Noise reduction in speech signals using adaptive filtering techniques. *Journal of Acoustical Engineering and Technology*, 62(2), 235-244.

Li, J., Wang, X., & Zhang, H. (2020). A review of classification methods in machine learning. *Journal of Data Science and Engineering*, 1(1), 25-35.

Patel, R., & Gupta, R. (2021). Efficient feature extraction for audio signal processing. *Journal of Signal Processing Systems*, 93(4), 591-602.

Ramadhana, Z. H. G., & Widiartha, I. M. (2021). Classification of Pop And RnB (Rhythm And Blues) Songs With MFCC Feature Extraction And K-NN Classifier. *Jurnal Elektronik Ilmu Komputer Udayana p-ISSN*, *2301*, 5373.

Sinaga, M. H. V., Albirra, M., & Sidiq, M. F. (2024). Klasifikasi Gambar Pemandangan dengan Kecerdasan Buatan Berbasis CNN. *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, *8*(2), 412-417.

Soekarta, R., Aras, S., & Aswad, A. N. (2023). Hyperparameter Optimization of CNN Classifier for Music Genre Classification. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *7*(5), 1205-1210.

Sharma, A., Rana, M. K., & Kaushik, H. (2022). A comprehensive review on MFCC feature extraction for speech recognition systems. *International Journal of Advanced Computer Science and Applications*, 13(7), 384-391.

Zhou, D.X. (2020). Universality of deep convolutional neural networks. Applied and Computational Harmonic Analysis.