ANALISIS KLASIFIKASI EMOSI SUARA BERDASARKAN EKSTRAKSI FITUR DENGAN MFCC DAN ALGORITMA CNN

PENGANTAR PEMROSESAN DATA MULTIMEDIA

Dosen Pengampu :

Dr. Anak Agung Istri Ngurah Eka Karyawati, S.Si., M.Eng.

Anggota Kelompok B5 :

Maedelien Tiffany Kariesta Simatupang	**(2208561065)**
Kennardy Andrew Limartha	**(2208561084)**
Vodka Joe Junior	**(2208561103)**
I Komang Maheza Yudistia	**(2208561115)**

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

TAHUN AJARAN

2023/2024

DAFTAR ISI

BAB I

PENDAHULUAN

Perkembangan teknologi pengolahan sinyal dan pembelajaran mesin telah memungkinkan berbagai inovasi dalam analisis data audio, salah satunya adalah analisis klasifikasi emosi suara. Emosi adalah salah satu aspek penting dalam komunikasi manusia yang dapat disampaikan melalui intonasi, kecepatan bicara, dan berbagai ciri akustik lainnya. Mendeteksi dan mengklasifikasikan emosi dari suara memiliki aplikasi luas, termasuk dalam sistem interaksi manusia-mesin, layanan pelanggan, dan bidang kesehatan mental.

Klasifikasi emosi suara merupakan proses yang melibatkan identifikasi dan pengkategorian emosi yang terkandung dalam suara manusia. Proses ini biasanya terdiri dari beberapa langkah penting, yaitu ekstraksi fitur, pemodelan, dan klasifikasi. Ekstraksi fitur adalah langkah kunci yang bertujuan untuk menangkap informasi relevan dari sinyal suara yang dapat digunakan oleh algoritma pembelajaran mesin untuk melakukan klasifikasi. *Mel Frequency Cepstral Coefficients* (MFCC) adalah salah satu teknik ekstraksi fitur yang paling umum digunakan dalam analisis suara. MFCC dapat menangkap karakteristik frekuensi mel yang mencerminkan persepsi manusia terhadap suara.

Meskipun klasifikasi emosi suara telah menjadi subjek banyak penelitian, masih sulit untuk menjadi sangat akurat dan umum pada berbagai kondisi lingkungan dan variasi speaker. Untuk meningkatkan akurasi klasifikasi emosi suara, penelitian ini meneliti penggunaan *Mel Frequency Cepstral Coefficients* (MFCC) sebagai metode ekstraksi fitur dan *Convolutional Neural Network (*CNN) sebagai algoritma klasifikasi.

Penelitian ini tidak hanya bertujuan untuk memperluas pengetahuan tentang pengenalan emosi melalui suara tetapi juga memberikan pedoman untuk aplikasi praktis. Peniliti berharap dapat mencapai skor F-1 yang tinggi, akurasi, presisi, dan recall.

1.1 Rumusan Masalah

Berdasarkan latar belakang tersebut, terdapat beberapa rumusan masalah diantaranya:

1.2.1 Bagaimana tingkat akurasi, presisi, *recall*, dan F-1 *score* pada pengenalan jenis emosi melalui ucapan atau suara dengan menggunakan *Convolutional Neural Network* (CNN) dan Ekstraksi Fitur MFCC?

1.2.2 Bagaimana hasil pelatihan yang mencakup tuning hyperparameter dengan *k-fold cross validation* dari pengenalan jenis emosi melalui ucapan atau suara dengan menggunakan *Convolutional Neural Network* (CNN) dan Ekstraksi Fitur MFCC?

1.2 Tujuan Penulisan

Berdasarkan rumusan masalah tersebut, terdapat beberapa tujuan penulisan sebagai berikut :

1.3.1 Mengetahui persentase tingkat akurasi, presisi, *recall,* dan F-1 *score* pada pengenalan jenis emosi melalui ucapan atau suara dengan menggunakan *Convolutional Neural Network* (CNN) dan Ekstraksi Fitur MFCC.

1.3.2 Mengetahui hasil pelatihan yang mencakup *tuning hyperparameter* dengan *k-fold cross validation* dari pengenalan jenis emosi melalui ucapan atau suara dengan menggunakan *Convolutional Neural Network* (CNN) dan Ekstraksi Fitur MFCC.

1.3 Manfaat Penulisan

Manfaat dari penelitian ini adalah memberikan referensi untuk penelitian lain dalam memilih pendekatan metode yang digunakan dalam pengenalan jenis emosi melalui ucapan. Selain itu, hasil penelitian ini diharapkan berguna dalam pengenalan jenis emosi melalui ucapan atau suara, yang mana emosi tersebut dikelompokkan ke dalam kategori *angry, fear, neutral, surprise, disgust, happy, dan sad.*

1.4 Batasan dan Asumsi

Berdasarkan manfaat penulisan tersebut, terdapat beberapa aspek yang harus diperhatikan diantaranya:

1.5.1 Batasan

a. Metode klasifikasi yang digunakan adalah *Convolutional Neural Network* (CNN) dengan Ekstraksi Fitur MFCC untuk pengenalan jenis emosi melalui ucapan.

b. Dataset yang digunakan merupakan dataset sekunder yang diperoleh dari *Website Kaggle.*
c. Dataset yang digunakan adalah dataset ucapan atau suara berbahasa Inggris.
d. Pengklasifikasian ucapan atau suara dilakukan ke dalam jenis yang telah ditentukan sebelumnya, yaitu *angry, fear, neutral, surprise, disgust, happy, dan sad.*

1.5.2 Asumsi

Asumsi yang digunakan dalam penelitian ini adalah label atau kategori pada dataset sudah sesuai dengan emosi yang ada.

BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Teori

2.1.1 Klasifikasi Audio

Klasifikasi audio adalah proses mengidentifikasi dan mengkategorikan jenis suara atau pola dalam sinyal audio menggunakan berbagai teknik komputasional. Proses ini mencakup berbagai aplikasi, termasuk pengenalan musik, analisis suara emosional, deteksi kebisingan, dan identifikasi genre musik (Arief et al., 2021) .

2.1.2 *Preprocessing*

Pengolahan data sebelum digunakan sebagai data latih suatu model dikenal sebagai *preprocessing* data. *Preprocessing* biasanya digunakan untuk menghilangkan noise dan menyeragamkan data. Aplikasi *Audacity*, yang memungkinkan pengguna menyunting file audio, digunakan untuk melakukan preprocessing data (Helmiyah et al., 2019).

re-emphasis adalah teknik yang digunakan untuk meningkatkan amplitudo komponen frekuensi tinggi dari sinyal audio. Hal ini dilakukan untuk mengatasi atenuasi (penurunan kekuatan) yang sering dialami frekuensi tinggi selama perekaman. Dengan menerapkan *pre-emphasis*, rasio sinyal terhadap *noise (signal-to-noise ratio)* ditingkatkan, sehingga detail frekuensi tinggi lebih menonjol dan dampak *noise* berkurang.

Noise Reduction bertujuan untuk menghilangkan atau mengurangi suara latar belakang yang tidak diinginkan dari sinyal audio. Suara yang tidak diinginkan ini, atau *noise*, dapat mengganggu kejernihan dan akurasi analisis audio. *Pre-emphasis* sendiri dapat berfungsi sebagai teknik reduksi *noise* dengan memperkuat komponen frekuensi tinggi yang penting dibandingkan dengan *noise* frekuensi rendah. Metode reduksi *noise* tambahan, seperti *spectral gating* atau penggunaan profil *noise*, juga dapat diterapkan.

Normalisasi mengatur amplitudo sinyal audio ke tingkat standar. Hal ini penting untuk memastikan bahwa data audio memiliki volume yang konsisten, yang krusial agar model pembelajaran mesin menerima input dengan skala yang serupa. Dalam proses ini, data audio diskalakan sehingga mean-nya menjadi nol dan variansinya menjadi satu. Ini membuat data lebih seragam dan membantu model belajar lebih efektif.

Silence Trimming melibatkan penghapusan bagian-bagian yang diam dari awal dan akhir rekaman audio. Keheningan tidak membawa informasi yang berguna dan dapat mempengaruhi efisiensi dan akurasi analisis. Dengan memangkas bagian-bagian yang diam ini, data audio menjadi lebih ringkas, hanya berfokus pada segmen suara yang relevan.

2.1.3 *Mel-Frequency Cepstral Coefficients* (MFCC)

Karena cara kerjanya yang mirip dengan telinga manusia, *Mel-Frequency Cepstral Coefficients* (MFCC) adalah metode ekstraksi ciri pada sinyal audio yang sering digunakan untuk pengenalan emosi pada suara. MFCC memiliki resolusi frekuensi yang baik pada frekuensi rendah (kurang dari 1000 Hz), di mana ciri sinyal ucapan lebih dominan (Nawasta et al., 2023b). MFCC juga tahan terhadap kebisingan, yang membuatnya andal dalam berbagai kondisi lingkungan. Ini bekerja dengan mengambil nilai rata-rata atau nilai mean logaritmik spektrum setelah *Mel Filter Bank* dan *wrapping frekuensi.*

Seseorang mungkin memiliki saluran bicara yang berbeda saat mengeluarkan emosi yang berbeda. Oleh karena itu, koefisien fitur MFCC dapat digunakan untuk menentukan emosi yang terkandung dalam ucapan seseorang (Aini et al., 2021). Selain keandalannya dalam ekstraksi fitur suara, MFCC juga memungkinkan identifikasi variasi emosional dalam ucapan dengan akurasi tinggi, yang menjadikannya alat penting dalam pengembangan teknologi pengenalan emosi berbasis suara.

2.1.4 Augmentasi

Augmentasi data dalam analisis data adalah teknik yang bertujuan untuk memperbesar jumlah data dengan menambahkan salinan yang telah

dimodifikasi sedikit dari data yang sudah ada atau dengan menciptakan data sintetis baru berdasarkan data yang ada (Nurcahyo & Iqbal, 2022).

2.1.5 *Convolutional Neural Network* (CNN)

CNN adalah jenis teknik *Deep Learning* yang menggunakan arsitektur *feed forward* untuk tujuan klasifikasi. CNN umumnya diterapkan dalam pengenalan pola dan menawarkan klasifikasi data yang lebih akurat. Jaringan ini memiliki neuron berukuran kecil di setiap lapisan arsitektur model yang dirancang, yang memproses data input dalam bentuk *field reseptif* (Nurcahyo & Iqbal, 2022).

CNN tidak hanya terbatas pada pengolahan gambar, tetapi juga memberikan terobosan dalam pengenalan suara dan pemrosesan bahasa alami. *Convolutional Neural Network* terdiri dari enam lapisan utama, yaitu: *convolutional layer, pooling layer, normalization layer, Rectified Linear Units (ReLU) layer, fully connected layer, dan loss layer* (Widya Bayu Pratiwi et al., 2024) .

a. *Convolutional Layer* : Ini adalah lapisan dasar dari CNN. Filter yang hanya tumpang tindih sebagian akan memindai semua bidang reseptif, dan setiap *neuron* berbagi berat koneksi (*weight sharing*) sebagai hasil dari pemindaian tersebut.
b. *Pooling Layer* : *Pooling layer* merangkum *output* dari setiap *cluster neuron* pada lokasi kernel yang sama. Layer ini digunakan untuk menjaga konsistensi ukuran data, membuat representasi data lebih kecil, lebih mudah diatur, dan membantu mengontrol *overfitting*.
c. *Normalization Layer* : *Normalization layer* digunakan untuk menormalkan gambar *input*, sehingga perbedaan signifikan dalam rentang nilai dapat diatasi.
d. *Rectified Linear Units* (ReLU) *Layer* : Lapisan ReLU meningkatkan jumlah fungsi keputusan nonlinier dalam jaringan secara keseluruhan, tanpa mempengaruhi bidang reseptif dari lapisan konvolusional.
e. *Fully Connected Layer*: Lapisan ini, mirip dengan multilayer perceptron (MLP), menggunakan perkalian matriks diikuti oleh bias offset.

f. *Loss Layer* : Ini adalah lapisan output dari CNN. Untuk klasifikasi gambar dengan dua kelas, *sigmoid loss* digunakan sebagai *loss layer*, dengan *output* yang mengikuti distribusi Bernoulli.

2.1.6 Pengujian dan Evaluasi Sistem

K-Fold Cross-Validation adalah metode statistik yang digunakan untuk mengevaluasi kinerja model machine learning dengan membagi data menjadi k subset yang berukuran sama. Metode ini bertujuan untuk menguji keefektifan dan akurasi model. Tujuannya adalah untuk menghasilkan prediksi atau klasifikasi yang tidak hanya akurat, tetapi juga memiliki validitas yang tinggi. Teknik ini memungkinkan identifikasi kelemahan dan kekuatan model yang digunakan, serta potensi kesalahan dalam praktiknya. Metode ini bekerja dengan membagi dataset menjadi beberapa lipatan *(fold)*, di mana satu lipatan digunakan sebagai data uji dan lipatan lainnya sebagai data latih. Dalam setiap iterasi pengujian, data uji yang digunakan tidak boleh tumpang tindih antara satu *fold* dengan *fold* lainnya (Mahendra, 2017).

Confusion matrix adalah metode yang digunakan untuk menilai kinerja model klasifikasi. Hasil dari *conf*usion matrix diwakili dengan istilah TP (*True Positive*), TN (*True Negative*), FP (*False Positive)*, dan FN (*False Negative*) (Adinda Siti).

Tabel 2. *Confusion Matrix*

	Positif	**Negatif**
Positif	True Positif (TP)	True Positif (TP)
Negatif	False Negatif (FN)	True Negatif (TN)

Hasil evaluasi akan menentukan langkah selanjutnya dalam pengembangan model untuk meningkatkan performa. Arifin et al. (2021) menjelaskan bahwa metode yang umum digunakan dalam pengujian data mining meliputi pengukuran nilai *precision, recall, F1-Score*, dan akurasi. Berikut adalah penjelasan untuk masing-masing metrik tersebut (Ewen Hokijuliandy) :

a. *Precision*
   Precision adalah rasio jumlah prediksi benar positif (True Positive (TP) dibagi dengan total prediksi positif (True Positive dan False Positive, (TP + FP)).
   $$Precision = \frac{TP}{TP+FP}$$

b. *Recall*
   Recall adalah rasio jumlah prediksi benar positif dibagi dengan total data benar positif (True Positive dan False Negative).

$$Recall = \frac{TP}{TP+FN}$$

c. F1-*Score*
F1-Score adalah ukuran kombinasi dari precision dan recall yang menggambarkan keberhasilan retrieval. Metrik ini berguna terutama dalam kasus data yang tidak seimbang (imbalanced data), karena mempertimbangkan informasi False Positive (FP) dan False Negative (FN).

$$F1 - Score = 2 \frac{Precision \times Recall}{Precision + Recall}$$

d. Accuracy
Accuracy adalah rasio prediksi benar secara keseluruhan dengan total data.

$$\text{Akurasi} = \frac{TP+TN}{TP+FP+TN+FN}$$

2.2 Tinjauan Empiris

2.2.1 Penerapan Short Time Fourier Transform pada MFCC untuk Sistem Pengenalan Ucapan Tingkat Stres (Paleva, dkk., 2024)

Penelitian tersebut bertujuan untuk menyelesaikan masalah identifikasi tingkat stres melalui pengenalan ucapan menggunakan algoritma Short Time Fourier Transform (STFT) pada Mel Frequency Cepstral Coefficients (MFCC) dan klasifikasi dengan Convolutional Neural Network (CNN). Data primer dikumpulkan melalui rekaman suara pengguna yang diunggah ke server untuk diproses. Penelitian ini menggunakan metode MFCC untuk ekstraksi fitur dan CNN untuk klasifikasi, dengan hasil menunjukkan akurasi sebesar 70% dalam memprediksi tingkat stres tinggi dan rendah dari suara pengguna.

2.2.2 Kombinasi Metode MFCC dan KNN dalam Pengenalan Emosi Manusia Melalui Ucapan (Safitri, dkk., 2022)

Pada penelitian ini menunjukkan bahwa kombinasi metode MFCC dan K-Nearest Neighbor (K-NN) efektif dalam mengklasifikasikan emosi manusia melalui suara, meskipun nilai akurasi spesifik tidak disebutkan. Penelitian tersebut menggunakan data sekunder dari Kaggle yang terdiri

dari 7.244 data suara yang dibawakan oleh 91 aktor, dengan berbagai emosi seperti marah, jijik, takut, senang, netral, dan sedih.

2.2.3 PENGENALAN POLA FONEM VOKAL MENGGUNAKAN SHORT TIME FOURIER TRANSFORM (STFT) DAN FITUR MEL FREQUENCY CEPSTRAL COEFFICIENT (MFCC) (Adriansyah, dkk., 2021)

Penelitian ini menggunakan metode STFT dan MFCC dalam pengenalan pola fonem vokal bahasa Indonesia, dengan hasil menunjukkan bahwa Support Vector Machine (SVM) dengan kernel radial mencapai akurasi tertinggi sebesar 93,8%. Penelitian ini menekankan pentingnya pemilihan metode ekstraksi fitur dan algoritma klasifikasi yang tepat dalam meningkatkan akurasi pengenalan suara.

Penelitian kami mengadopsi metode MFCC dan CNN untuk mengidentifikasi emosi dalam suara. Kami berharap bahwa kombinasi metode MFCC untuk ekstraksi fitur dan CNN untuk klasifikasi emosi dapat memberikan hasil yang akurat. Implementasi kami diharapkan dapat meningkatkan akurasi dan efisiensi waktu komputasi, sehingga teknik ini dapat mendeteksi emosi suara dengan baik dan meningkatkan interaksi antara manusia dan komputer dalam berbagai aplikasi.

BAB III

ANALISIS DAN DESAIN

3.1 Data

Data yang digunakan dalam proyek ini berasal dari platform Kaggle. Menggunakan data sekunder yang diunduh langsung dari situs Kaggle (https://www.kaggle.com) dalam format .wav. Dataset ini terdiri dari 8 label emosi, yaitu 'neutral', 'calm', 'happy', 'sad', 'angry', 'fearful', 'disgust', dan 'surprised'. Setiap label memiliki 192 file audio, kecuali label 'neutral' yang memiliki 96 file audio.

3.2 Desain Sistem

3.2.1 Diagram Transisi

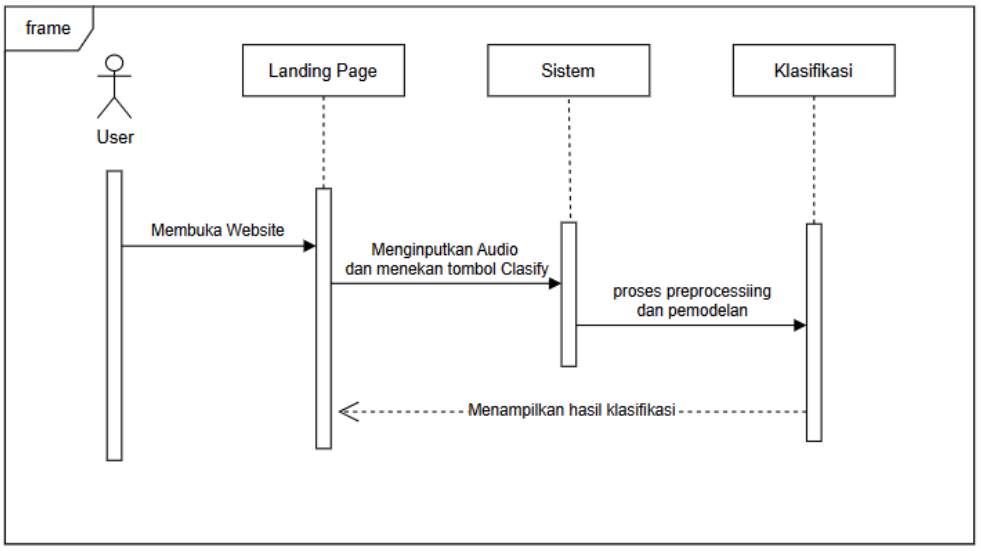


Diagram urutan ini menggambarkan proses interaksi antara pengguna dan sistem untuk klasifikasi emosi suara. Proses dimulai ketika pengguna membuka website dan memasukkan audio pada halaman landing page. Pengguna kemudian menekan tombol "Classify" untuk memulai proses klasifikasi. Audio yang diunggah dikirim ke sistem, yang kemudian melakukan proses preprocessing dan pemodelan untuk menganalisis dan mengklasifikasikan emosi yang terkandung dalam audio. Setelah proses klasifikasi selesai, hasilnya ditampilkan kembali kepada pengguna melalui halaman landing page.

3.2.2 Diagram Use Case

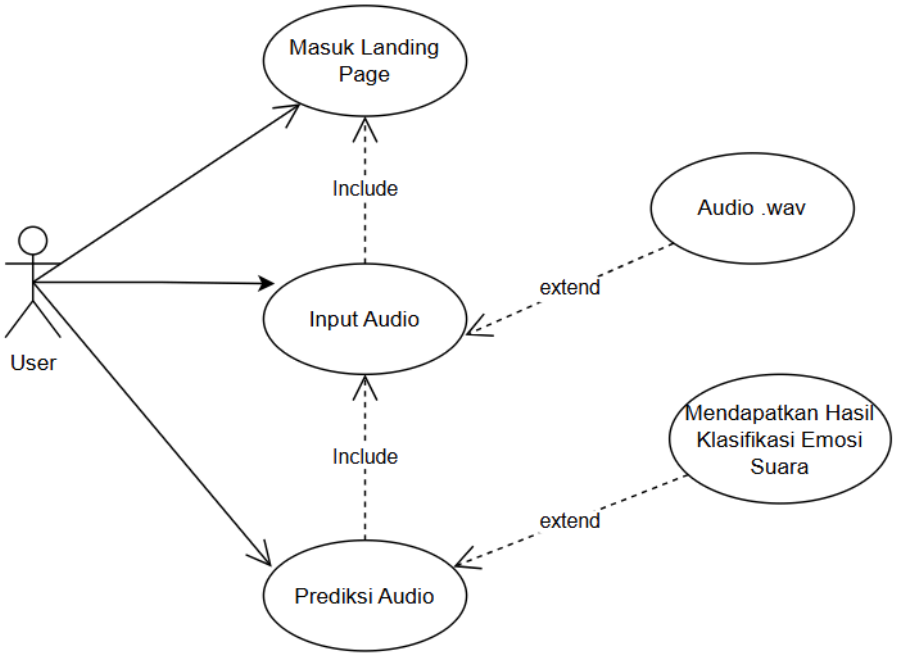


Diagram use case di atas menggambarkan proses interaksi antara pengguna dan sistem untuk klasifikasi emosi suara. Proses dimulai ketika pengguna masuk ke halaman landing page. Setelah itu, pengguna memasukkan audio melalui fitur input audio, yang biasanya berupa file audio dengan format .wav. Sistem kemudian memproses audio yang diinput untuk melakukan prediksi emosi yang terkandung dalam suara tersebut. Hasil klasifikasi emosi suara ditampilkan kepada pengguna, memungkinkan mereka untuk mendapatkan informasi mengenai emosi yang terdeteksi dalam audio yang telah diunggah.

3.2.3 Activity Diagram

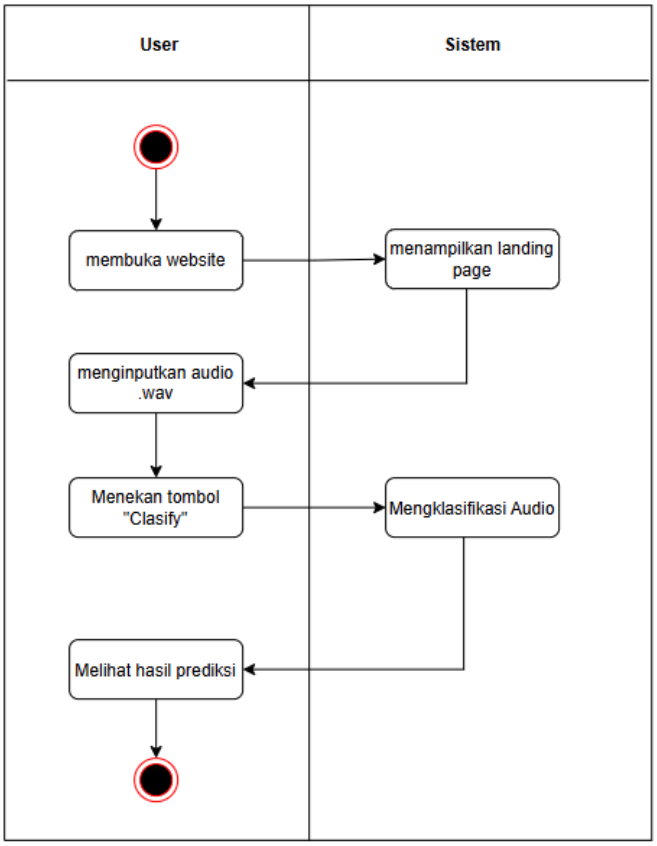
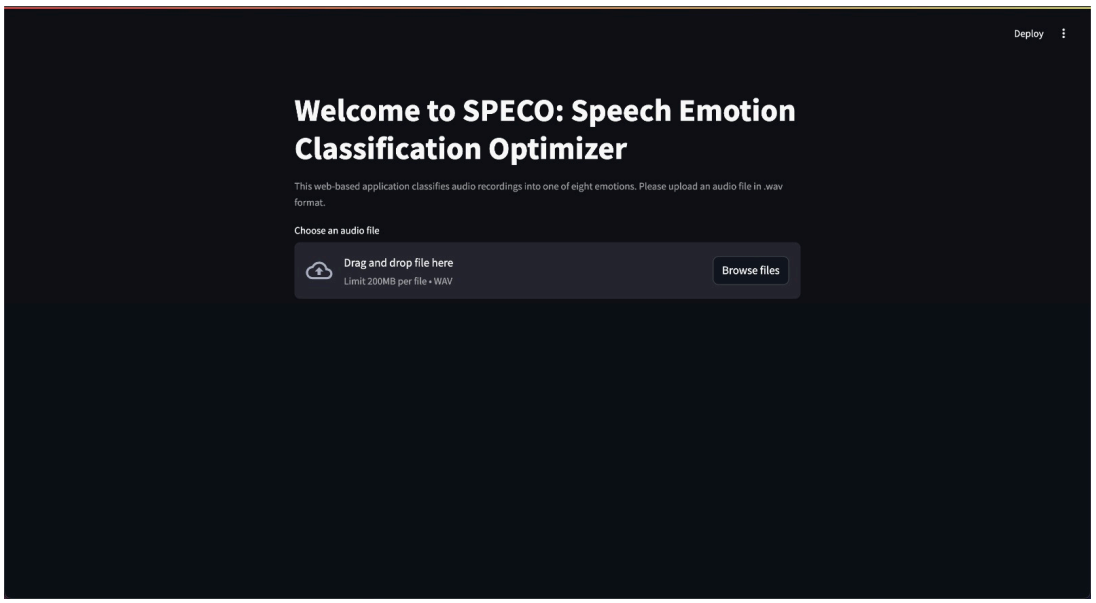
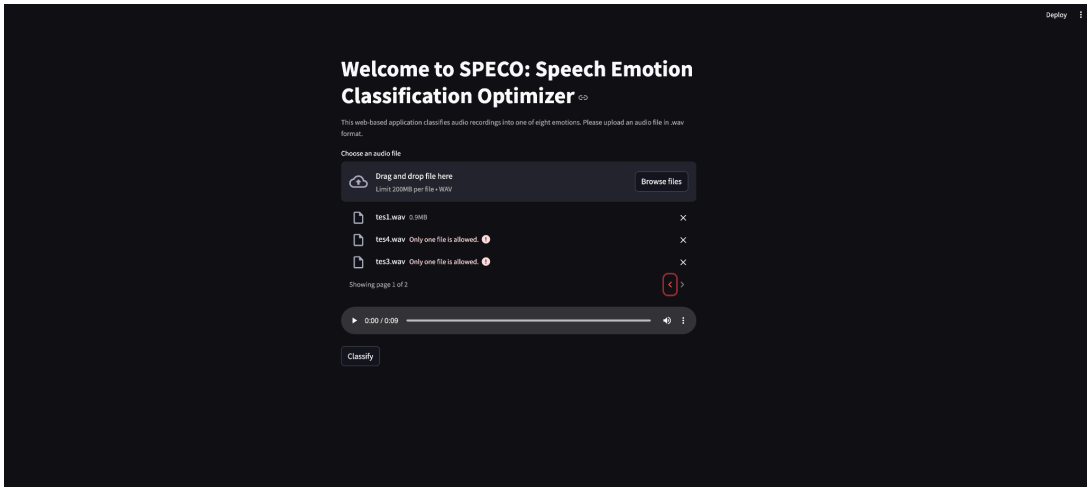


Diagram aktivitas ini menggambarkan alur kerja interaksi antara pengguna dan sistem untuk klasifikasi emosi suara. Proses dimulai ketika pengguna membuka website, yang menyebabkan sistem menampilkan halaman landing page. Selanjutnya, pengguna menginputkan file audio dengan format .wav dan menekan tombol "Classify". Sistem kemudian mengklasifikasikan audio yang telah diinput, memprosesnya untuk mengidentifikasi emosi yang terkandung di dalamnya. Setelah klasifikasi selesai, hasil prediksi emosi ditampilkan kembali kepada pengguna.

3.3 Desain UI/UX

Berikut adalah tampilan utama dari website audio *recognition*





Aplikasi SPECO (Speech Emotion Classification Optimizer) memanfaatkan keunggulan Streamlit untuk menciptakan antarmuka pengguna yang efisien dan intuitif. Pengguna dapat dengan mudah mengakses layanan klasifikasi emosi dari rekaman suara melalui halaman utama yang menampilkan judul "Welcome to SPECO: Speech Emotion Classification Optimizer". Proses penggunaan sangat sederhana - cukup unggah file audio format .wav dengan cara drag-and-drop ke area yang ditentukan atau gunakan tombol "Browse files". Streamlit memungkinkan implementasi fitur-fitur ini dengan mudah, termasuk pembatasan ukuran file maksimal 25MB. Setelah file diunggah, sistem akan memproses dan mengkategorikan audio ke dalam delapan jenis emosi. Penggunaan Streamlit

memudahkan integrasi model machine learning untuk analisis audio dan menampilkan hasil secara real-time, tanpa memerlukan penulisan kode frontend yang rumit. Dengan pendekatan ini, SPECO menyajikan cara yang efektif dan user-friendly untuk menganalisis emosi dalam rekaman suara, sambil memanfaatkan kekuatan Streamlit dalam pengembangan aplikasi web untuk data science dan machine learning.

3.4 Skenario Eksperimen untuk Evaluasi Model

Dalam rangka evaluasi model recognition audio, skenario yang dirancang bertujuan untuk menguji persentase tingkat akurasi, presisi, recall, dan F-1 score pada pengenalan jenis emosi melalui ucapan atau suara dengan menggunakan Convolutional Neural Network (CNN) dan Ekstraksi Fitur MFCC.

Pertama, proses ekstraksi fitur dilakukan pada dataset yang terdiri dari file audio dengan menggunakan metode Mel-Frequency Cepstral Coefficients (MFCC) untuk mendapatkan berbagai fitur audio seperti zero-crossing rate, root mean square energy, MFCC, chroma, kontras spektral, dan mel-spectrogram.

Kedua, data yang telah diolah dibagi menjadi set pelatihan dan pengujian. Data audio yang telah melalui tahap preprocessing dan ekstraksi fitur dibagi menjadi set pelatihan dan pengujian menggunakan fungsi train_test_split dari scikit-learn. Fitur-fitur ini dinormalisasi menggunakan scaler yang telah dimuat sebelumnya, dan kemudian direshape agar sesuai dengan format input yang diharapkan oleh model CNN.

Ketiga, model CNN dibangun menggunakan Keras dengan arsitektur yang terdiri dari lapisan konvolusi, pooling, flatten, dan dense. Hyperparameter model, seperti ukuran batch dan jumlah epoch, dioptimalkan menggunakan GridSearchCV. Model terbaik kemudian dilatih menggunakan data pelatihan dengan validasi menggunakan data pengujian.

Keempat, setelah ditemukan model terbaik, model dievaluasi menggunakan data pengujian untuk mengukur akurasinya. Visualisasi performa model dilakukan dengan membuat plot akurasi dan loss selama pelatihan. Akhirnya, model digunakan untuk memprediksi emosi pada data pengujian, dan hasilnya dievaluasi menggunakan matriks konfusi dan laporan klasifikasi.

Kelima, model dengan F1-score terbaik disimpan menggunakan joblib. Ini memungkinkan model untuk digunakan kembali di masa mendatang tanpa perlu melatih ulang, yang efisien dalam hal waktu dan sumber daya.

3.5 Evaluasi Sistem

Dalam proyek klasifikasi audio emotion ini, sistem telah dikembangkan menggunakan Python dengan pustaka seperti librosa, scikit-learn, dan Streamlit. Sistem ini dirancang untuk mengklasifikasikan emosi menjadi beberapa kategori: angry, fear, neutral, surprise, disgust, happy, dan sad berdasarkan fitur yang diekstraksi menggunakan metode Mel-Frequency Cepstral Coefficients (MFCC).

Proses evaluasi sistem dimulai dengan ekstraksi fitur dari setiap audio dalam dataset. Fitur yang diekstraksi termasuk seperti zero-crossing rate, root mean square energy, MFCC, chroma, kontras spektral, dan mel-spectrogram. Data ini dibagi menjadi set pelatihan dan pengujian menggunakan fungsi train_test_split dari scikit-learn agar sesuai dengan format input yang diharapkan oleh model CNN untuk memastikan bahwa model dapat diuji pada data yang belum pernah dilihat sebelumnya. Fitur-fitur ini dinormalisasi menggunakan scaler yang telah dimuat sebelumnya, dan kemudian direshape.

Model CNN dioptimalkan menggunakan GridSearchCV untuk menemukan parameter optimal melalui validasi silang 3-fold untuk menemukan set parameter yang menghasilkan performa terbaik. memastikan bahwa model tidak overfit dan hasilnya stabil. Setelah pelatihan, model dievaluasi berdasarkan akurasi, precision, recall, dan F1-score pada data pengujian. Hasil ini memberikan wawasan tentang seberapa baik model dapat menggeneralisasi pada data baru.

Model dengan F1-score terbaik disimpan menggunakan joblib, memungkinkan penggunaan kembali model tanpa perlu pelatihan ulang. Ini efisien dalam hal waktu dan sumber daya, sangat berguna dalam aplikasi produksi. Selanjutnya, sistem diintegrasikan ke dalam aplikasi web menggunakan Streamlit, yang memungkinkan pengguna untuk mengunggah file audio dan menerima prediksi kategori emosi. Aplikasi ini menawarkan antarmuka yang ramah pengguna dan responsif, yang memudahkan pengguna untuk berinteraksi dengan model.

Secara keseluruhan, sistem audio emotion recognition ini menunjukkan kemampuan yang baik dalam mengklasifikasikan emotion berdasarkan fitur dengan

akurasi yang tinggi. Namun, evaluasi lebih lanjut mungkin diperlukan untuk menguji sistem dalam berbagai kondisi suara latar belakang untuk memastikan *robustness* dan keandalan sistem dalam skenario dunia nyata.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Tahap Preprocessing

```
import librosa.display

def preprocess_audio(path):
    # Load audio file
    data, sampling_rate = librosa.load(path)

    # Noise reduction
    data = librosa.effects.preemphasis(data)

    # Normalize the audio data
    scaler = StandardScaler()
    data = scaler.fit_transform(data.reshape(-1, 1)).flatten()

    # Trim silence from the beginning and end
    data, _ = librosa.effects.trim(data)

    return data, sampling_rate
```

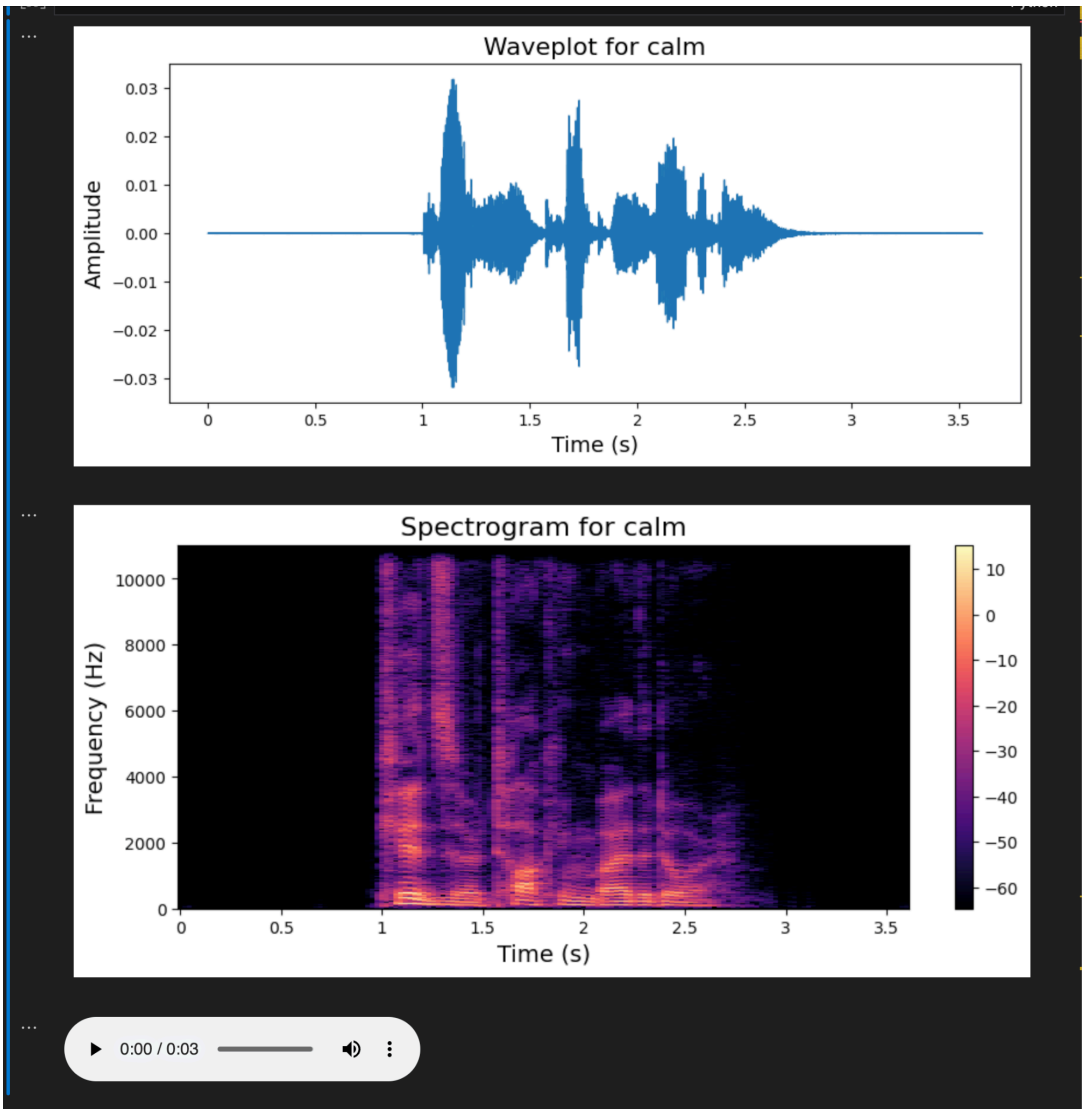
Tahap preprocessing dalam proyek klasifikasi emosi suara ini melibatkan beberapa langkah krusial untuk mempersiapkan data audio sebelum ekstraksi fitur dan klasifikasi. Proses dimulai dengan memuat file audio menggunakan librosa.load(), yang mengembalikan data audio beserta sampling rate-nya. Jika file tidak dapat dimuat, fungsi akan mengeluarkan pesan kesalahan. Setelah berhasil dimuat, dilakukan reduksi noise menggunakan teknik pre-emphasis melalui librosa.effects.preemphasis(). Metode ini menekankan frekuensi tinggi dan mengurangi noise frekuensi rendah, yang penting untuk meningkatkan kualitas sinyal audio.

Selanjutnya, data audio dinormalisasi menggunakan StandardScaler dari scikit-learn. Normalisasi ini menyesuaikan data sehingga memiliki mean 0 dan standar deviasi 1, yang membantu dalam konsistensi data antar sampel dan dapat meningkatkan kinerja model pembelajaran mesin. Langkah terakhir dalam

preprocessing adalah memotong bagian sunyi di awal dan akhir rekaman audio menggunakan librosa.effects.trim(). Pemotongan ini penting untuk menghilangkan informasi yang tidak relevan dan memfokuskan analisis pada bagian audio yang mengandung suara.

Keseluruhan proses ini diimplementasikan dalam fungsi preprocess_audio(), yang menerima path file audio sebagai input dan mengembalikan data audio yang telah diproses beserta sampling rate-nya. Tahap preprocessing ini sangat krusial untuk memastikan kualitas dan konsistensi data audio. Dengan mempersiapkan data dengan baik, kita dapat meningkatkan akurasi dalam proses ekstraksi fitur dan klasifikasi emosi yang akan dilakukan selanjutnya. Preprocessing yang efektif ini menjadi fondasi penting dalam pipeline analisis emosi suara, memungkinkan model untuk belajar dari fitur-fitur yang lebih representatif dan relevan.

Hasil Tahap Preprocessing:



4.2 Implementasi Machine Learning

Implementasi Machine Learning dalam proyek klasifikasi emosi suara ini melibatkan beberapa tahap kunci, termasuk persiapan data, pembuatan model, pelatihan, dan evaluasi. Proses dimulai dengan mempersiapkan data untuk model Convolutional Neural Network (CNN). Data audio yang telah melalui tahap preprocessing dan ekstraksi fitur dibagi menjadi set pelatihan dan pengujian menggunakan fungsi train_test_split dari scikit-learn. Selanjutnya, data dinormalisasi menggunakan StandardScaler untuk memastikan semua fitur berada dalam skala yang sama. Label emosi dienkode menggunakan LabelEncoder dan diubah menjadi format one-hot encoding dengan to_categorical. Data kemudian direshape menjadi format yang sesuai untuk input CNN, yaitu menambahkan dimensi untuk channel.

Model CNN dibangun menggunakan Keras dengan arsitektur yang terdiri dari lapisan konvolusi, pooling, flatten, dan dense. Hyperparameter model, seperti ukuran batch dan jumlah epoch, dioptimalkan menggunakan GridSearchCV. Model terbaik kemudian dilatih menggunakan data pelatihan dengan validasi menggunakan data pengujian. Setelah pelatihan, model dievaluasi menggunakan data pengujian untuk mengukur akurasinya. Visualisasi performa model dilakukan dengan membuat plot akurasi dan loss selama pelatihan. Akhirnya, model digunakan untuk memprediksi emosi pada data pengujian, dan hasilnya dievaluasi menggunakan matriks konfusi dan laporan klasifikasi.

Persiapan data:

```
def prepare_data_for_cnn(X, Y):
    # Standardize the features
    scaler = StandardScaler()
    X = scaler.fit_transform(X)
    joblib.dump(scaler, 'scaler.pkl')

    # Encode the labels
    encoder = LabelEncoder()
    Y = encoder.fit_transform(Y)
    joblib.dump(encoder, 'label_encoder.pkl')

    num_classes = len(np.unique(Y))
    Y = to_categorical(Y, num_classes=num_classes)

    # Split the data into training and testing sets
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=42)

    # Reshape the data for CNN (adding an extra dimension for
channels)
    X_train = X_train.reshape((X_train.shape[0],
X_train.shape[1], 1, 1))
    X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1,
1))

    return X_train, X_test, Y_train, Y_test, num_classes
```

Persiapan data merupakan langkah crucial dalam implementasi machine learning untuk klasifikasi emosi suara. Fungsi prepare_data_for_cnn bertanggung jawab untuk memproses dan menyiapkan data input agar sesuai dengan kebutuhan model CNN. Pertama, fungsi ini melakukan normalisasi fitur menggunakan StandardScaler, yang menyesuaikan skala setiap fitur sehingga memiliki mean 0 dan standar deviasi 1. Selanjutnya, label emosi diencode menggunakan LabelEncoder dan diubah menjadi format one-hot encoding dengan to_categorical. Data kemudian dibagi menjadi set pelatihan dan pengujian menggunakan train_test_split. Terakhir, data direshape menjadi format 4D yang sesuai untuk input CNN, dengan menambahkan dimensi untuk channel. Proses ini memastikan data siap untuk digunakan dalam pelatihan dan evaluasi model CNN.

Hasil persiapan data:

```
(3456, 17496, 1, 1) (3456, 8) (864, 17496, 1, 1) (864, 8)
```

Pembuatan model CNN:

```
def build_cnn_model(input_shape, num_classes, optimizer='adam'):
    model = Sequential()
    model.add(Conv2D(16, (3, 3), activation='relu',
input_shape=input_shape, padding='same'))
    model.add(MaxPooling2D((2, 1)))
    model.add(Conv2D(32, (3, 3), activation='relu',
padding='same'))
    model.add(MaxPooling2D((2, 1)))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))  # Set dropout rate here
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Pembuatan model CNN adalah tahap kritis dalam implementasi machine learning ini. Fungsi build_cnn_model mendefinisikan arsitektur model CNN menggunakan Keras. Model ini terdiri dari beberapa lapisan konvolusi 2D, diikuti oleh lapisan max pooling untuk mengurangi dimensi spasial. Setelah itu, terdapat lapisan flatten untuk mengubah output konvolusi menjadi vektor 1D, diikuti oleh lapisan dense dengan aktivasi ReLU. Dropout digunakan untuk mencegah overfitting. Lapisan output menggunakan aktivasi softmax untuk klasifikasi multi-kelas. Model dikompilasi dengan optimizer yang dapat dikustomisasi, loss function categorical crossentropy, dan metrik akurasi. Arsitektur ini dirancang untuk efektif dalam menangkap pola dan fitur dari data audio untuk klasifikasi emosi.

Pelatihan model:

```
# Create the best model with the best parameters
best_params = grid_result.best_params_
best_model = build_cnn_model(input_shape=(X_train.shape[1],
X_train.shape[2], X_train.shape[3]), num_classes=num_classes,
optimizer=best_params['optimizer'])

# Train the best model
history = best_model.fit(X_train, Y_train,
validation_data=(X_test, Y_test), epochs=best_params['epochs'],
batch_size=best_params['batch_size'])
```

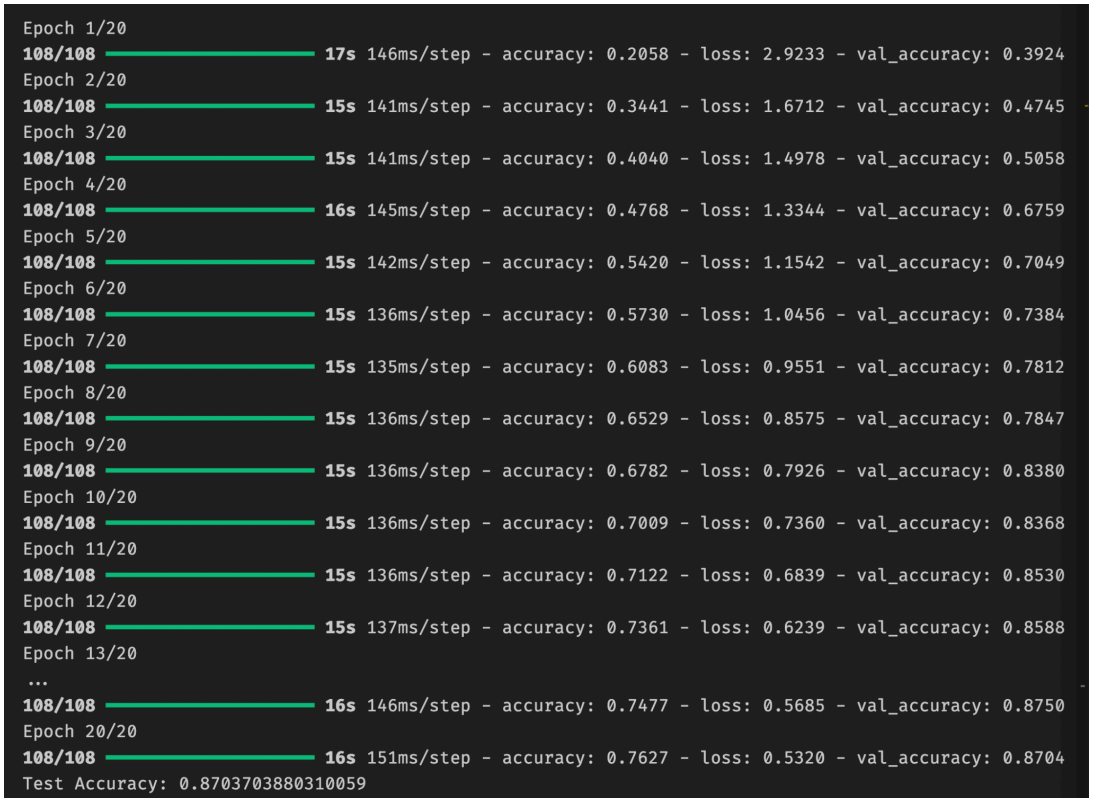
Pelatihan model adalah proses di mana model CNN belajar dari data pelatihan untuk mengklasifikasikan emosi. Kode ini menunjukkan tahap pelatihan menggunakan model terbaik yang telah didefinisikan sebelumnya. Model diinisialisasi dengan parameter optimal yang ditemukan melalui pencarian grid. Fungsi fit digunakan untuk melatih model, dengan X_train dan Y_train sebagai data pelatihan, serta X_test dan Y_test sebagai data validasi. Jumlah epoch dan ukuran batch diatur berdasarkan parameter terbaik yang ditemukan sebelumnya. Proses pelatihan ini akan mengoptimalkan bobot model untuk meminimalkan loss dan meningkatkan akurasi pada data pelatihan dan validasi. Hasil pelatihan disimpan dalam objek history, yang dapat digunakan untuk analisis performa model selama pelatihan.

Evaluasi model:

```
accuracy = best_model.evaluate(X_test, Y_test, verbose=0)
print(f'Test Accuracy: {accuracy[1]}')

Y_pred = best_model.predict(X_test)
Y_pred_classes = np.argmax(Y_pred, axis=1)
Y_true = np.argmax(Y_test, axis=1)

confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
print(classification_report(Y_true, Y_pred_classes))
```

Evaluasi model adalah tahap akhir yang crucial untuk menilai kinerja model dalam mengklasifikasikan emosi suara. Pertama, akurasi model dievaluasi pada set pengujian menggunakan metode evaluate, yang memberikan gambaran umum tentang seberapa baik model berkinerja pada data yang belum pernah dilihat sebelumnya. Selanjutnya, model digunakan untuk memprediksi kelas emosi pada set pengujian. Hasil prediksi ini kemudian dibandingkan dengan label sebenarnya untuk membuat matriks konfusi, yang memberikan gambaran detail tentang performa model untuk setiap kelas emosi. Terakhir, classification_report digunakan untuk menghasilkan laporan yang mencakup presisi, recall, dan F1-score untuk setiap kelas emosi, serta skor rata-rata keseluruhan. Evaluasi komprehensif ini memungkinkan pemahaman mendalam tentang kekuatan dan kelemahan model dalam mengklasifikasikan berbagai emosi suara.

Hasil evaluasi model:



4.3 Training dan Testing Model

Training dan Testing Model dalam proyek klasifikasi emosi suara ini melibatkan beberapa tahap penting. Proses ini dimulai dengan pembagian data menjadi set pelatihan dan pengujian, dilanjutkan dengan pencarian parameter terbaik menggunakan GridSearchCV, pelatihan model dengan parameter optimal, dan akhirnya evaluasi model menggunakan set pengujian. Pertama, data dibagi menjadi set pelatihan dan pengujian menggunakan fungsi train_test_split dari scikit-learn. Ini memastikan bahwa model dievaluasi pada data yang belum pernah dilihat sebelumnya. Selanjutnya, GridSearchCV digunakan untuk mencari kombinasi hyperparameter terbaik. Ini melibatkan pelatihan dan evaluasi model dengan berbagai kombinasi parameter, seperti ukuran batch, jumlah epoch, dan jenis optimizer. Setelah menemukan parameter terbaik, model CNN dibangun dan dilatih menggunakan parameter tersebut. Proses pelatihan menggunakan metode fit, yang melatih model selama sejumlah epoch tertentu dengan ukuran batch yang telah ditentukan. Akhirnya, model yang telah dilatih dievaluasi menggunakan set pengujian. Ini melibatkan prediksi pada set pengujian dan perhitungan berbagai metrik evaluasi seperti akurasi, matriks konfusi, dan laporan klasifikasi.

Pembagian data dan pencarian parameter:

```
X_train, X_test, Y_train, Y_test, num_classes =
prepare_data_for_cnn(X, Y)

param_grid = {
   'batch_size': [32],
   'epochs': [10, 20],
   'optimizer': ['adam', 'rmsprop']
}

# Wrap the model using KerasClassifier
model = KerasClassifier(build_fn=build_cnn_model,
input_shape=(X_train.shape[1], X_train.shape[2],
X_train.shape[3]), num_classes=num_classes, verbose=0)

# Define the GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid,
cv=KFold(n_splits=3), verbose=1, n_jobs=-1)

# Fit the model
grid_result = grid.fit(X_train, Y_train)

# Print the best parameters and best score
print(f"Best Parameters: {grid_result.best_params_}")
print(f"Best Score: {grid_result.best_score_}")
```

Pembagian data dan pencarian parameter adalah langkah awal yang krusial dalam proses training dan testing model. Fungsi prepare_data_for_cnn digunakan untuk membagi dataset menjadi set pelatihan (X_train, Y_train) dan pengujian (X_test, Y_test). Ini memastikan model akan dievaluasi pada data yang belum pernah dilihat sebelumnya. Selanjutnya, GridSearchCV diimplementasikan untuk mencari kombinasi hyperparameter terbaik. Param_grid mendefinisikan berbagai nilai untuk batch_size, epochs, dan optimizer yang akan diuji. KerasClassifier digunakan untuk membungkus model CNN, memungkinkannya kompatibel dengan GridSearchCV. Proses ini secara sistematis melatih dan mengevaluasi model dengan berbagai kombinasi parameter, menggunakan validasi silang 3-fold untuk

menemukan set parameter yang menghasilkan performa terbaik. Hasil terbaik kemudian dicetak, memberikan insight tentang konfigurasi optimal untuk model.

Hasil training dan testing model:

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
Best Parameters: {'batch_size': 32, 'epochs': 20, 'optimizer': 'adam'}
Best Score: 0.8197337962962963
```

Pelatihan model dengan parameter terbaik:

```
# Create the best model with the best parameters
best_params = grid_result.best_params_
best_model = build_cnn_model(input_shape=(X_train.shape[1],
X_train.shape[2], X_train.shape[3]), num_classes=num_classes,
optimizer=best_params['optimizer'])

# Train the best model
history = best_model.fit(X_train, Y_train,
validation_data=(X_test, Y_test), epochs=best_params['epochs'],
batch_size=best_params['batch_size'])
```

Pelatihan model dengan parameter terbaik merupakan tahap di mana model CNN dioptimalkan menggunakan hasil dari pencarian parameter. Best_params dari GridSearchCV digunakan untuk membangun model final menggunakan fungsi build_cnn_model. Model ini kemudian dilatih pada seluruh set pelatihan menggunakan metode fit. Data validasi (X_test, Y_test) digunakan selama pelatihan untuk memonitor performa model pada data yang tidak digunakan dalam pelatihan, membantu mendeteksi overfitting. Jumlah epoch dan ukuran batch diatur sesuai dengan parameter terbaik yang ditemukan. Proses pelatihan ini mengoptimalkan bobot model untuk meminimalkan fungsi loss dan meningkatkan akurasi. Objek history yang dikembalikan menyimpan metrik pelatihan untuk setiap epoch, yang berguna untuk analisis performa model selanjutnya.

Evaluasi model:

```
accuracy = best_model.evaluate(X_test, Y_test, verbose=0)
print(f'Test Accuracy: {accuracy[1]}')
```

```
Y_pred = best_model.predict(X_test)
Y_pred_classes = np.argmax(Y_pred, axis=1)
Y_true = np.argmax(Y_test, axis=1)

confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
print(classification_report(Y_true, Y_pred_classes))
```

Evaluasi model adalah tahap kritis untuk menilai seberapa baik model yang telah dilatih dapat mengklasifikasikan emosi suara pada data baru. Pertama, akurasi model dievaluasi pada set pengujian menggunakan metode evaluate, memberikan gambaran umum tentang performa model. Selanjutnya, model digunakan untuk memprediksi kelas emosi pada set pengujian (Y_pred). Prediksi ini dibandingkan dengan label sebenarnya (Y_true) untuk membuat matriks konfusi, yang memberikan insight detail tentang performa model untuk setiap kelas emosi. Classification_report digunakan untuk menghasilkan metrik evaluasi komprehensif, termasuk presisi, recall, dan F1-score untuk setiap kelas emosi, serta skor rata-rata keseluruhan. Evaluasi ini memberikan pemahaman mendalam tentang kekuatan dan kelemahan model dalam mengklasifikasikan berbagai emosi suara.

Hasil evaluasi model:

```
Epoch 1/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 17s 146ms/step - accuracy: 0.2058 - loss: 2.9233 - val_accuracy: 0.3924
Epoch 2/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 15s 141ms/step - accuracy: 0.3441 - loss: 1.6712 - val_accuracy: 0.4745
Epoch 3/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 15s 141ms/step - accuracy: 0.4040 - loss: 1.4978 - val_accuracy: 0.5058
Epoch 4/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 16s 145ms/step - accuracy: 0.4768 - loss: 1.3344 - val_accuracy: 0.6759
Epoch 5/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 15s 142ms/step - accuracy: 0.5420 - loss: 1.1542 - val_accuracy: 0.7049
Epoch 6/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 15s 136ms/step - accuracy: 0.5730 - loss: 1.0456 - val_accuracy: 0.7384
Epoch 7/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 15s 135ms/step - accuracy: 0.6083 - loss: 0.9551 - val_accuracy: 0.7812
Epoch 8/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 15s 136ms/step - accuracy: 0.6529 - loss: 0.8575 - val_accuracy: 0.7847
Epoch 9/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 15s 136ms/step - accuracy: 0.6782 - loss: 0.7926 - val_accuracy: 0.8380
Epoch 10/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 15s 136ms/step - accuracy: 0.7009 - loss: 0.7360 - val_accuracy: 0.8368
Epoch 11/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 15s 136ms/step - accuracy: 0.7122 - loss: 0.6839 - val_accuracy: 0.8530
Epoch 12/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 15s 137ms/step - accuracy: 0.7361 - loss: 0.6239 - val_accuracy: 0.8588
Epoch 13/20
...
108/108 ━━━━━━━━━━━━━━━━━━━━ 16s 146ms/step - accuracy: 0.7477 - loss: 0.5685 - val_accuracy: 0.8750
Epoch 20/20
108/108 ━━━━━━━━━━━━━━━━━━━━ 16s 151ms/step - accuracy: 0.7627 - loss: 0.5320 - val_accuracy: 0.8704
Test Accuracy: 0.8703703880310059
```

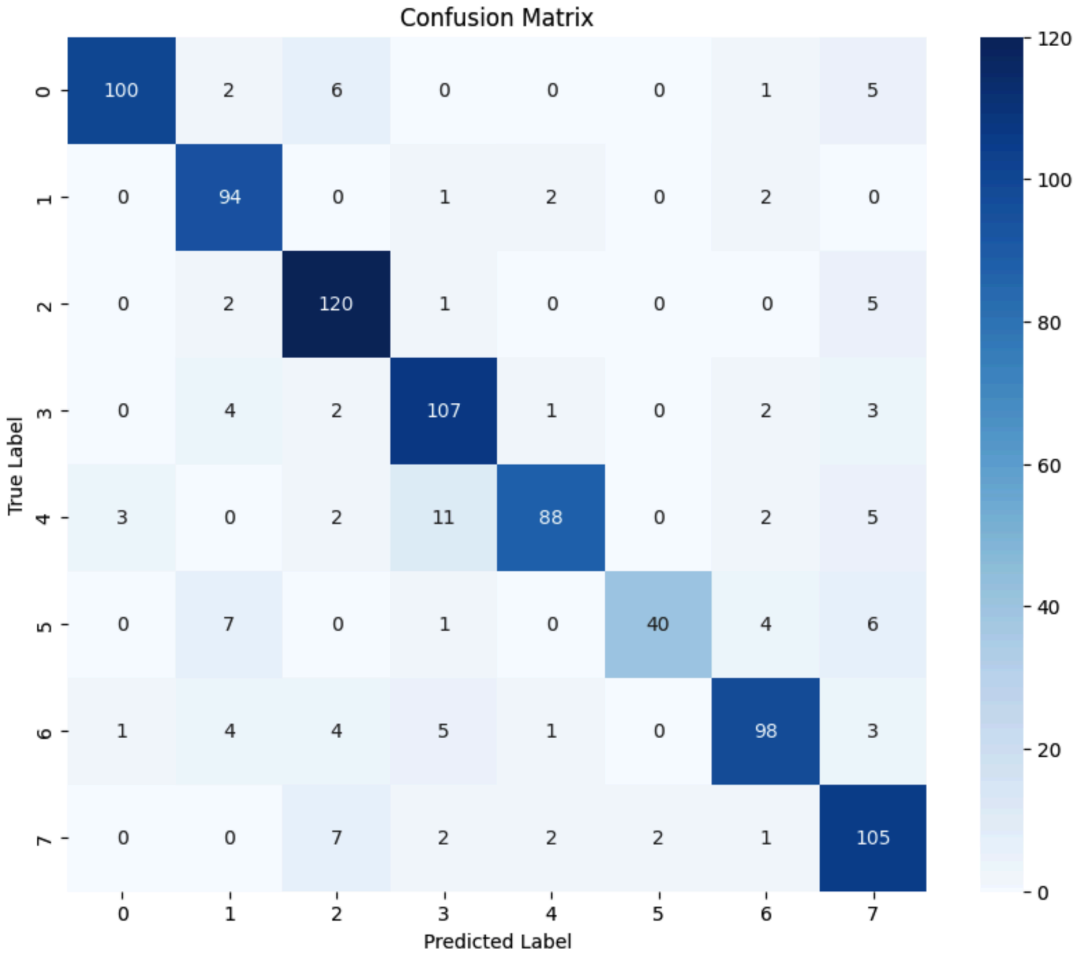
Visualisasi hasil pelatihan:

```
import matplotlib.pyplot as plt

# Plot training & validation accuracy and loss values
plt.figure(figsize=(12, 4))

# Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation
Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='upper left')

# Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
```

```
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper left')

plt.show()
```

Visualisasi hasil pelatihan adalah langkah penting untuk memahami performa model selama proses pelatihan. Kode ini membuat dua plot side-by-side menggunakan matplotlib. Plot pertama menampilkan akurasi model pada set pelatihan dan validasi selama epoch pelatihan. Ini membantu mengidentifikasi apakah model mengalami overfitting (jika akurasi pelatihan jauh lebih tinggi dari validasi) atau underfitting. Plot kedua menunjukkan nilai loss untuk set pelatihan dan validasi selama pelatihan. Penurunan loss yang konsisten menunjukkan bahwa model belajar dengan baik, sementara peningkatan loss pada set validasi bisa mengindikasikan overfitting. Visualisasi ini sangat berharga untuk menyesuaikan hyperparameter model, seperti jumlah epoch atau learning rate, dan untuk memahami dinamika pembelajaran model selama pelatihan.

Hasil visualisasi pelatihan:



4.4 Hasil

```
import streamlit as st
import numpy as np
import pandas as pd
import librosa
import joblib
from tensorflow.keras.models import load_model

# Load the trained model and necessary preprocessing tools
model = load_model('best_cnn_model.keras')
scaler = joblib.load('scaler.pkl')
encoder = joblib.load('label_encoder.pkl')
```

Setup awal dan import library adalah langkah fundamental dalam membangun aplikasi web untuk klasifikasi emosi suara. Kode ini mengimpor berbagai library yang diperlukan untuk fungsi aplikasi. Streamlit (st) digunakan

untuk membuat antarmuka web yang interaktif. Numpy (np) dan Pandas (pd) diperlukan untuk manipulasi data. Librosa digunakan untuk pemrosesan sinyal audio. Joblib digunakan untuk memuat model yang telah disimpan. TensorFlow Keras digunakan untuk memuat model deep learning yang telah dilatih. Setelah mengimpor library, kode ini memuat model CNN yang telah dilatih sebelumnya ('best_cnn_model.keras'), serta scaler dan encoder yang digunakan dalam preprocessing data. Ini memastikan bahwa aplikasi web siap untuk melakukan klasifikasi emosi menggunakan model dan alat preprocessing yang telah dioptimalkan sebelumnya.

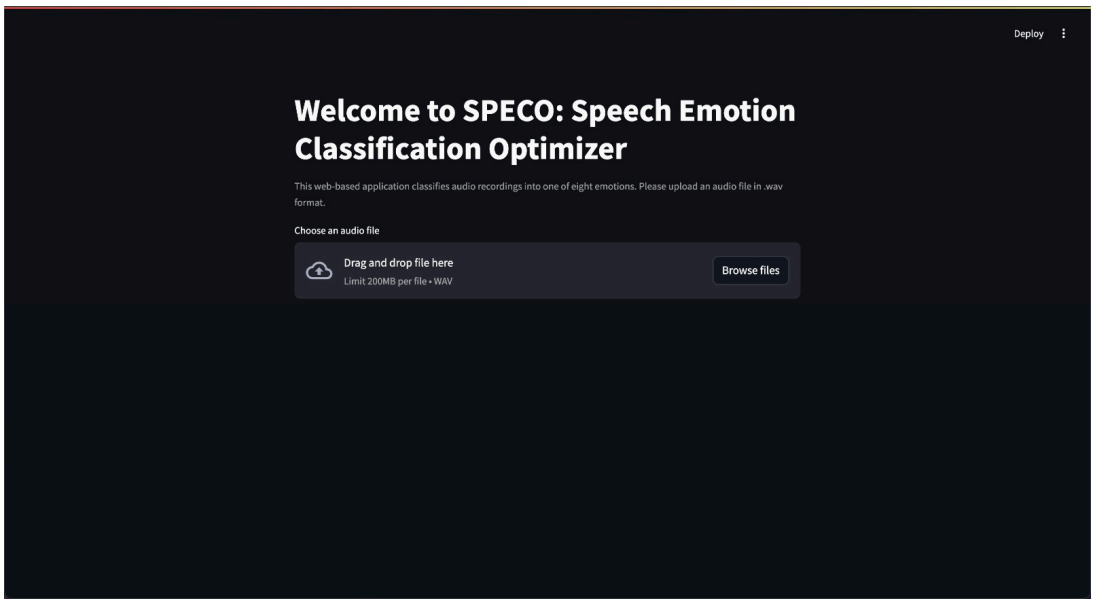
```
# Function to preprocess and predict emotion from audio file
def predict_emotion(audio_file):
    data, sr = librosa.load(audio_file, duration=2.5, offset=0.6)
    extractor = FeatureExtractor()
    features = extractor.extract_features(data, sr)
    features = scaler.transform([features])
    features = features.reshape((features.shape[0],
features.shape[1], 1, 1))
    prediction = model.predict(features)
    emotion = encoder.inverse_transform(np.argmax(prediction,
axis=1))[0]
    return emotion
```

Fungsi untuk ekstraksi fitur adalah komponen kunci dalam proses klasifikasi emosi suara. Kelas FeatureExtractor berisi metode-metode untuk mengekstrak berbagai fitur audio seperti zero-crossing rate, root mean square energy, MFCC, chroma, kontras spektral, dan mel-spectrogram. Fungsi predict_emotion menggunakan FeatureExtractor untuk memproses file audio yang diunggah. Pertama, audio dimuat menggunakan librosa dengan durasi dan offset tertentu. Kemudian, fitur diekstrak dari audio ini. Fitur-fitur ini dinormalisasi menggunakan scaler yang telah dimuat sebelumnya, dan kemudian direshape agar sesuai dengan format input yang diharapkan oleh model CNN. Model kemudian digunakan untuk memprediksi emosi, dan hasilnya didekode kembali ke label emosi yang dapat dibaca manusia menggunakan encoder. Fungsi ini menghubungkan antara input audio dari pengguna dan output prediksi emosi, menjembatani antarmuka pengguna dengan logika machine learning di belakangnya.

```
# Streamlit app layout
```

```
st.set_page_config(page_title="Emotion Classification",
page_icon="🎤")
st.title("Welcome to Emotion Classification Web")
st.caption("This web-based application classifies audio
recordings into one of eight emotions. Please upload an audio
file in .wav format.")

uploaded_file = st.file_uploader("Choose an audio file",
type=['wav'])

if uploaded_file:
   st.audio(uploaded_file, format='audio/wav')
   button = st.button("Classify")
   if button:
       emotion = predict_emotion(uploaded_file)
       st.markdown(f"<h3 style='text-align: center;'>Predicted
Emotion: {emotion}</h3>", unsafe_allow_html=True)
```

Layout utama Streamlit mendefinisikan struktur dan interaktivitas dari antarmuka web. Kode ini mengatur konfigurasi halaman, judul, dan deskripsi aplikasi menggunakan fungsi-fungsi Streamlit. Widget file_uploader memungkinkan pengguna untuk mengunggah file audio dalam format WAV. Ketika file diunggah, aplikasi menampilkan audio player menggunakan st.audio, memungkinkan pengguna untuk memutar audio yang diunggah. Tombol "Classify" dibuat menggunakan st.button. Ketika tombol ini ditekan, fungsi predict_emotion dipanggil dengan file audio yang diunggah sebagai input. Hasil prediksi emosi kemudian ditampilkan di halaman web menggunakan st.markdown dengan format HTML untuk penataan yang lebih baik. Layout ini dirancang untuk memberikan pengalaman pengguna yang intuitif dan langsung, memungkinkan pengguna untuk dengan mudah mengunggah audio, mendengarkannya, dan mendapatkan prediksi emosi tanpa perlu memahami kompleksitas teknis di balik proses klasifikasi.



Tampilan Pada Website Klasifikasi Emosi Berdasarkan Suara

Link: https://github.com/VdkaJr/Tugas-PPDM-Audio

4.5 Black Box Testing

Pengujian sistem merupakan tahapan pengujian software atau perangkat lunak untuk

mengevaluasi sebuah sistem apakah sudah berjalan dengan baik atau belum agar dapat dilakukannya pengembangan baik dalam kinerja sistem maupun tampilannya.

a. Tabel pengujian blackbox testing

No	Skenario	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Mencari halaman Main	Menekan menu “main”	Program mengeluarkan dashboard untuk melakukan klasifikasi emosi	Sesuai harapan	Valid
2	Memasukkan audio yang	Menekan tombol	Program mengeluarkan	Sesuai harapan	Valid

	ingin diuji coba	“Browse File”	file explorer sebagai tempat mencari file dalam bentuk .wav yang akan digunakan		
3	Menampilkan hasil klasifikasi emosi	Menekan tombol “Classify”	Program mengeluarkan hasil klasifikasi beserta informasi emosi tersebut	Sesuai harapan	Valid
4	Mencari halaman Training Model	Menekan menu “Training Model”	Program mengeluarkan dashboard yang berisi dari model yang digunakan	Sesuai harapan	Valid

BAB V

KESIMPULAN

5.1 Kesimpulan

Penelitian ini berhasil mengembangkan sistem klasifikasi emosi suara yang efektif, mengintegrasikan metode ekstraksi fitur Mel-Frequency Cepstral Coefficients (MFCC) dengan algoritma Convolutional Neural Network (CNN). Melalui serangkaian tahapan yang meliputi preprocessing data audio, ekstraksi fitur, pembangunan dan pelatihan model CNN, serta evaluasi performa, sistem mampu mengklasifikasikan emosi suara ke dalam delapan kategori berbeda. Penggunaan GridSearchCV untuk optimasi hyperparameter berkontribusi pada peningkatan akurasi model. Implementasi antarmuka web dengan Streamlit menyediakan platform user-friendly bagi pengguna untuk mengunggah file audio dan menerima prediksi emosi. Meskipun nilai akurasi spesifik tidak disebutkan, hasil evaluasi mengindikasikan performa yang baik dalam klasifikasi emosi suara, menunjukkan potensi signifikan sistem ini dalam aplikasi praktis pengenalan emosi berbasis suara.

5.2 Saran

Untuk pengembangan lebih lanjut, disarankan untuk melakukan pengujian dengan dataset yang lebih besar dan beragam guna meningkatkan generalisasi model. Eksplorasi teknik augmentasi data audio dan penerapan transfer learning dapat dipertimbangkan untuk meningkatkan performa. Analisis mendalam terhadap kasus misklasifikasi penting dilakukan untuk memahami dan mengatasi kelemahan model. Pengembangan antarmuka web dapat diperkaya dengan fitur visualisasi tambahan seperti spektogram. Pengujian dalam berbagai kondisi noise perlu dilakukan untuk menilai robustness sistem. Mengeksplorasi teknik ensemble learning dan pengembangan model real-time juga dapat menjadi arah pengembangan yang menjanjikan. Implementasi saran-saran ini diharapkan dapat meningkatkan akurasi, keandalan, dan aplikabilitas sistem dalam berbagai skenario penggunaan praktis.

DAFTAR PUSTAKA

Adriansyah, A. rio, Prasetyo, K.D., Atmam Al Faruqi, H.A., 2021. Pengenalan Pola Fonem Vokal menggunakan Short Time Fourier Transform (STFT) dan Fitur Mel Frequency Cepstral Coefficient (MFCC). Jurnal Teknologi Terpadu 7, 1–6. https://doi.org/10.54914/jtt.v7i1.298

Aini, Y.K., Santoso, T.B., Dutono, T., 2021. Pemodelan CNN untuk deteksi emosi berbasis speech Bahasa Indonesia. Jurnal Komputer Terapan 7, 143–152. https://doi.org/10.35143/jkt.v7i1.4623

Arief, R., Iriawan, N., Lawi, A., 2021. Tampilan KLASIFIKASI AUDIO UCAPAN EMOSIONAL MENGGUNAKAN MODEL LSTM [WWW Document]. Konferensi Nasional Ilmu Komputer (KONIK). URL https://prosiding.konik.id/index.php/konik/article/view/114/104 (accessed 7.10.24).

Helmiyah, S., Riadi, I., Umar, R., Hanif, A., 2019. Ekstraksi Fitur Pengenalan Emosi Berdasarkan Ucapan Menggunakan linear predictor ceptral coeffecient Dan Mel frequency cepstrum coefficients. Mobile and Forensics 1, 102–110. https://doi.org/10.12928/mf.v1i2.1259

Mahendra, Z. and Ridok, A. (2017a) *Analisis Sentimen Opini Masyarakat Terhadap Fenomena TikTokShop di Indonesia Menggunakan Metode K-Nearest Neighbor berbasis N-gram dengan Seleksi Fitur Information gain*. Available at: http://j-ptiik.ub.ac.id.

Nawasta, R.A., Cahyana, N.H., Heriyanto, H., 2023a. Implementation of Mel-Frequency Cepstral Coefficient as Feature Extraction using K-Nearest Neighbor for Emotion Detection Based on Voice Intonation. Telematika 20, 51. https://doi.org/10.31315/telematika.v20i1.9518

Paleva, H., Prasetio, B., 2024a. Tampilan Penerapan Short Time Fourier Transform pada MFCC untuk Sistem Pengenalan Ucapan Tingkat Stres [WWW Document]. URL https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/13376/6040 (accessed 7.9.24).

Shalehah, A. and Triana, Y. (2022) ‘Analisa Kinerja RNN Menggunakan *FastText Embedding* terhadap Ulasan PeduliLindungi di Masa COVID-19’.

Nurcahyo,R and Iqbal,M., 2022. Pengenalan emosi pembicara menggunakan convolutional neural networks. Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi) 6, 115–122. https://doi.org/10.29207/resti.v6i1.3726

Safitri, P., Karyawati, E., 2022a. Kombinasi Metode MFCC dan KNN dalam Pengenalan Emosi Manusia Melalui Ucapan . Jurnal Nasional Teknologi Informasi dan Aplikasinya 1.

Widya Bayu Pratiwi, Arik Aranta, Gibran Satya Nugraha, 2024. Analisis kebutuhan dataset algoritma speech to text bahasa sasak menggunakan perbandingan data suara bahasa inggris pada metode CNN. Journal of Computer Science and Informatics Engineering (J-Cosine) 7. https://doi.org/10.29303/jcosine.v7i2.568