FINAL PROJECT

DESAIN MODEL KLASIFIKASI AUDIO CNN UNTUK IDENTIFIKASI SPESIES HEWAN DI LEMBAH MAGDALENA TENGAH, KOLOMBIA

Mata Kuliah IS411 – Data Modelling

Dosen Pengampu: Irmawati

Disusun oleh Kelompok 8:

Bonardo Gregorius B.S	-	(NIM: 00000097365)
Ricardo Cipta	-	(NIM: 00000098947)
Jonathan Chandra	-	(NIM: 00000094067)
Samuel Christophorus.W	-	(NIM: 00000100797)
Gabriel Yohanes.A	-	(NIM: 00000082326)

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS TEKNIK DAN INFORMATIKA

UNIVERSITAS MULTIMEDIA NUSANTARA

TANGERANG

2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pengidentifikasian spesies hewan memainkan peran penting dalam konservasi keanekaragaman hayati, pengelolaan ekosistem, dan penelitian ekologi dan biologi. Pada umumnya, proses ini dilakukan melalui pengamatan visual langsung dan analisis morfologi, yang membutuhkan keahlian khusus dan sering kali menghabiskan banyak waktu dan sumber daya, terutama di daerah yang sulit dijangkau seperti Lembah Magdalena Tengah di Kolombia yang terkenal dengan keanekaragaman hayatinya yang kaya. Dalam beberapa tahun terakhir, teknologi berbasis kecerdasan buatan, khususnya deep learning, telah memberikan pendekatan baru yang lebih efisien untuk identifikasi spesies melalui sinyal audio, khususnya vokalisasi hewan [1] [2].

Pemanfaatan teknik pemrosesan sinyal audio untuk mengidentifikasi suara spesies telah menjadi hal yang menarik di bidang bioakustik. Teknik-teknik ini melibatkan pengubahan data audio menjadi representasi visual seperti spektogram, yang kemudian dapat dianalisis dengan menggunakan model klasifikasi seperti Convolutional Neural Network (CNN) [3] [4]. CNN telah terbukti sangat efektif dalam mengenali pola visual yang kompleks, termasuk pola frekuensi suara yang unik untuk setiap spesies [5].

Beberapa penelitian sebelumnya telah menunjukkan bahwa pengenalan spesies berbasis suara menggunakan deep learning memiliki akurasi yang tinggi dan dapat diaplikasikan di lingkungan nyata [6] [7]. Namun, sebagian besar penelitian tersebut masih berfokus pada spesies burung di Amerika Utara atau Eropa, sehingga implementasi di daerah tropis seperti Kolombia belum banyak dieksplorasi [8]. Maka dari itu, penelitian ini berusaha merancang model klasifikasi audio berbasis CNN yang dioptimasi untuk mengenali spesies hewan di Lembah Magdalena Tengah, Kolombia.

1.2 Tujuan Penelitian

Penelitian ini bertujuan untuk merancang dan melakukan implementasi model klasifikasi audio dengan menggunakan metode Convolutional Neural Network (CNN) untuk mengidentifikasi spesies hewan yang berdasarkan pada rekaman suaranya. Model ini diharapkan dapat membedakan spesies berdasarkan karakteristik suara yang unik dengan tingkat akurasi yang tinggi. Penelitian ini juga bertujuan untuk mengembangkan pipeline pemrosesan audio yang efisien yang dapat diintegrasikan ke dalam aplikasi praktis untuk pemantauan keanekaragaman hayati.

1.3 Manfaat

1.3.1 Manfaat Teoritis

Penelitian ini berkontribusi pada pengembangan teori di bidang bioakustik dan pengenalan pola menggunakan deep learning. Dengan menggabungkan pemrosesan sinyal audio dan CNN, penelitian ini memperkaya literatur tentang penerapan pembelajaran mesin dalam identifikasi spesies berbasis suara. Hasil penelitian juga memberikan wawasan tentang tingkat keefektifan CNNs dalam menganalisis data audio yang memiliki kompleksitas tinggi, terutama dalam konteks lingkungan tropis yang berisik dan heterogen [9] [10].

1.3.2 Manfaat Praktisi

Penelitian ini secara praktis dapat digunakan sebagai dasar untuk mengembangkan sistem identifikasi spesies secara otomatis yang berguna untuk konservasi satwa liar, pemantauan lingkungan, dan penelitian ekologi. Selanjutnya, sistem yang dihasilkan dapat diimplementasikan dalam bentuk aplikasi berbasis web maupun mobile yang dapat digunakan oleh para peneliti lapangan, konservasionis, dan pengambil kebijakan. Dengan demikian, efisiensi pengumpulan data keanekaragaman hayati dapat ditingkatkan, dan deteksi dini terhadap perubahan populasi spesies dapat dilakukan dengan lebih baik.

BAB II

METHODOLOGY

Metodologi yang digunakan dalam penelitian ini mengacu pada pendekatan CRISP-DM (Cross Industry Standard Process for Data Mining), yang merupakan kerangka kerja umum dan sistematis untuk proses data mining. CRISP-DM terdiri dari enam fase utama: Pemahaman Bisnis, Pemahaman Data, Persiapan Data, Pemodelan, Evaluasi, dan Penerapan. Setiap fase saling terkait dan dapat dilalui secara iteratif tergantung pada hasil dan kebutuhan penelitian.

2.1 Business Understanding

Tahap ini merupakan pondasi utama dalam memahami tujuan bisnis atau penelitian dari proyek ini. Dalam konteks penelitian ini, tujuan utamanya adalah mengembangkan sistem klasifikasi berbasis audio untuk mengidentifikasi spesies hewan di Lembah Magdalena Tengah, Kolombia. Wilayah ini merupakan wilayah tropis dengan tingkat keanekaragaman hayati yang sangat tinggi, namun juga menghadapi ancaman dari aktivitas manusia dan perubahan iklim.

Masalah yang dihadapi adalah kebutuhan akan sistem pemantauan keanekaragaman hayati yang efisien, non-invasif, dan dapat diterapkan secara luas. Oleh karena itu, identifikasi spesies melalui suara menjadi solusi yang menjanjikan karena sebagian besar hewan, terutama burung, mamalia kecil dan amfibi, menghasilkan suara khas yang dapat dideteksi melalui teknologi perekaman audio. Dengan memahami masalah ini, penelitian ini diarahkan untuk merancang solusi berbasis machine learning, khususnya deep learning.

2.2 Data Understanding

Tahap Pemahaman Data bertujuan untuk memahami struktur, konten, kualitas, dan potensi data yang tersedia sebelum melakukan proses pemodelan. Dalam konteks penelitian ini, data yang digunakan berasal dari berbagai sumber audio yang telah dikurasi dan disediakan secara resmi oleh penyelenggara kompetisi, dengan fokus utama pada rekaman suara spesies dari kawasan Cagar Alam El Silencio di Lembah Magdalena Tengah, Kolombia.

Dataset ini mencerminkan kompleksitas ekosistem tropis dan mencakup suara dari berbagai kelompok taksonomi seperti burung (Aves), amfibi (Amphibia), mamalia (Mamalia), dan serangga (Insecta). Dataset ini terdiri dari beberapa komponen utama:

A. Train Audio (*train_audio/*)

Dalam folder ini terdapat rekaman audio pendek dari setiap spesies, yang diunggah oleh pengguna dari tiga koleksi utama: xeno-canto (XC), iNaturalist (iNat), dan Colombian Sound Archive (CSA). Semua file telah dikonversi ke format .ogg dan frekuensinya disamakan ke 32 kHz untuk mencocokkan format dengan data pengujian. Setiap file audio diberi nama sesuai dengan format [collection][id_file].ogg. Rekaman ini memiliki kualitas yang bervariasi, tergantung pada peringkat yang diberikan oleh pengguna (1 = sangat rendah, 5 = sangat tinggi).

B. Data Train (*train.csv*)

Data ini merupakan file metadata utama yang menjelaskan karakteristik setiap file audio di train_audio/. Beberapa bidang yang penting termasuk:

- **primary_label:** Label utama spesies (kode eBird untuk burung atau ID takson iNaturalist untuk non-burung).
- **secondary_label:** Spesies lain yang mungkin juga tercatat di dalam file, tapi bukan merupakan label utama.
- **latitude & longitude:** Lokasi geografis rekaman, berguna untuk memahami variasi lokal (misalnya perbedaan dialek suara antar lokasi).
- **author:** Nama pengguna yang merekam audio.
- **rating:** Kualitas rekaman menurut pengguna.
- **collection:** Sumber rekaman (XC, iNat, CSA).
- **filename:** Nama file audio yang sesuai.

C. Test Soundscape (*test_soundscape/*)

Folder yang merupakan kumpulan data pengujian yang akan digunakan untuk penilaian akhir model. Folder ini berisi sekitar 700 rekaman satu menit, masing-masing dalam format .ogg dan frekuensi 32 kHz. Nama file disamarkan

dengan format umum soundscape_xxxxxx.ogg. Data ini hanya tersedia ketika notebook dikirim dan dijalankan pada platform penilaian. Tidak semua spesies dari data pelatihan muncul di dataset pengujian.

D. Train Soundscape (*train_soundscape/*)

Berisi rekaman tanpa label dari lokasi yang sama dengan data uji, tetapi direkam pada titik lokasi yang berbeda. File ini dapat digunakan untuk strategi pembelajaran semi-pengawasan, seperti menggunakan *pseudo-labeling* atau teknik prapelatihan. Nama file memiliki pola [site]_[date]_[local_time].ogg.

E. Taxonomy (*taxonomy.csv*)

File ini menyediakan informasi taksonomi dari setiap spesies, termasuk ID takson iNaturalist dan kelas biologisnya (Aves, Amphibia, Mamalia, Insecta). File ini berguna untuk memahami hubungan antar spesies dan melakukan analisis berbasis taksonomi.

F. Lokasi Rekaman (*recoding_location.txt*)

Menyediakan informasi umum tentang lokasi perekaman, Cagar Alam El Silencio, yang merupakan pusat konservasi dan restorasi ekosistem yang penting di Kolombia. Informasi lokasi ini berguna untuk memberikan konteks ekologis pada data dan dapat digunakan untuk menganalisis pengaruh kondisi lingkungan pada suara yang direkam.

Melalui pemahaman yang mendalam mengenai struktur dan konten data ini, strategi pemrosesan dan pelatihan model dapat dirancang dengan lebih akurat. Tantangan utama meliputi:

- Ketidakseimbangan kelas (beberapa spesies memiliki lebih banyak contoh dibandingkan spesies lainnya),
- Variasi dalam kualitas rekaman,
- Kebisingan lingkungan alami
- Terbatasnya jumlah data berlabel untuk spesies langka.

Pemahaman awal ini merupakan pondasi penting sebelum melangkah ke langkah selanjutnya, yaitu persiapan data dan pemodelan.

2.3 Data Preparation

Tahap Data Preparation merupakan salah satu komponen kunci dalam pipeline data mining karena kualitas dan representasi data secara langsung mempengaruhi kinerja model. Pada penelitian ini, beberapa langkah penting dilakukan untuk mempersiapkan data audio dan metadata sebelum digunakan dalam proses pelatihan model klasifikasi berbasis CNN. Pada penelitian ini, proses data preparation mencakup beberapa langkah berikut:

1. **Eksplorasi Data Awal (EDA)**

   Langkah pertama dimulai dengan membaca file train.csv menggunakan pustaka Pandas untuk memuat metadata yang berisi informasi mengenai primary_labels, secondary_labels, nama file, lokasi geografis, kualitas audio (rating), dan asal pengambilan data (XC, CSA, iNat). Data ini dieksplorasi untuk memahami distribusi spesies, kualitas rekaman, dan distribusi geografis.

2. **Visualisasi Data dan Deteksi Outlier**

   Visualisasi awal dilakukan untuk melihat sebaran jumlah rekaman per spesies, histogram kualitas (rating), dan sebaran geografis berdasarkan koordinat lintang dan bujur. Analisis korelasi antar fitur numerik dilakukan dengan menggunakan *heatmap*, sementara deteksi outlier diterapkan pada rating, lintang, dan bujur dengan menggunakan metode IQR (Interquartile Range).

3. **Korelasi Antar Variabel Kategorikal**

   Dalam memahami hubungan antara atribut kategorikal, analisis Cramer's V yang merupakan kekuatan hubungan atau asosiasi antara dua variabel kategorikal (nominal) pada fitur seperti *primary_label*, *collection*, dan *author.* Hasilnya divisualisasikan dalam bentuk *heatmap* matriks korelasi, yang membantu mengidentifikasi fitur mana yang paling relevan dengan model atau yang berpotensi berlebihan.

4. **Pembersihan Data**

   Berdasarkan hasil eksplorasi awal, pembersihan data dilakukan untuk menghapus atau memperbaiki entri yang tidak valid, seperti file yang tidak ditemukan atau file dengan kualitas yang sangat rendah (rating < 2). Kumpulan data yang telah dibersihkan disimpan sebagai df_clean.

5. **Ekstraksi Fitur Audio (MFCC)**

   Seluruh file audio diubah menjadi representasi numerik berupa Mel-Frequency Cepstral Coefficients (MFCC), yang dianggap paling efektif untuk analisis sinyal audio karena merepresentasikan karakteristik frekuensi suara.

6. **Preprocessing Audio**
   Melakukan trimming untuk menghilangkan silent segment, normalisasi amplitude, serta bandpass filter pada rentang 300 Hz hingga 8000 Hz untuk menjaga suara relevan. Selanjutnya hasil MFCC dipotong atau dipadatkan menjadi dimensi seragam (400 frame × 40 koefisien).
7. **Encoding Label**
   Label primary_label yang bersifat kategorikal diubah menjadi numerik menggunakan LabelEncoder, yang kemudian disimpan dalam format .pkl untuk keperluan prediksi.
8. **Train-Test Split**
   Dataset MFCC yang sudah diproses dibagi menjadi data latih dan data validasi menggunakan metode stratified split untuk menjaga proporsi kelas yang seimbang.

2.4 Modeling

Tahap Modeling bertujuan untuk membangun model klasifikasi berbasis Convolutional Neural Network (CNN) yang mampu memprediksi jenis spesies berdasarkan fitur audio MFCC.

Langkah-langkah yang dilakukan meliputi:

1. **Pembangunan Arsitektur CNN:**
   Model CNN terdiri dari dua blok Conv2D + BatchNormalization + MaxPooling2D, dilanjutkan dengan Dropout, GlobalAveragePooling2D, dan Dense Layer dengan aktivasi softmax pada output layer.
2. **Training Model:**
   Model dilatih menggunakan optimizer Adam dan loss function sparse categorical crossentropy selama maksimal 50 epoch, dengan EarlyStopping berdasarkan nilai validation loss untuk menghindari overfitting.
3. **Menyimpan Model dan LabelEncoder:**
   Model yang telah terlatih disimpan dalam format .keras, dan LabelEncoder dalam format .pkl untuk digunakan pada tahap prediksi.

2.5 Evaluation

Tahap Evaluasi bertujuan untuk menguji kinerja model pada hidden test set yang disediakan oleh penyelenggara kompetisi. Langkah yang dilakukan:

1. **Inference pada Test Soundscape**
   File audio pada direktori test_soundscapes dipotong menjadi segment 5 detik, kemudian diekstrak fitur MFCC-nya. Model CNN yang telah dilatih digunakan untuk memprediksi probabilitas kemunculan 206 spesies untuk setiap bagian segmentnya.

2. **Pembuatan Submission CSV**
   Hasil prediksi disimpan dalam file submission.csv sesuai format sample_submission.csv.
3. **Proses submission** dilakukan via notebook runtime Kaggle tanpa akses internet, dimana test_soundscapes baru muncul saat proses submission berjalan.
4. **Scoring Kaggle Leaderboard**
   Hasil prediksi diuji melalui submission ke leaderboard dan memperoleh skor publik sebesar 0.581.

BAB III

EKSEKUSI DAN HASIL

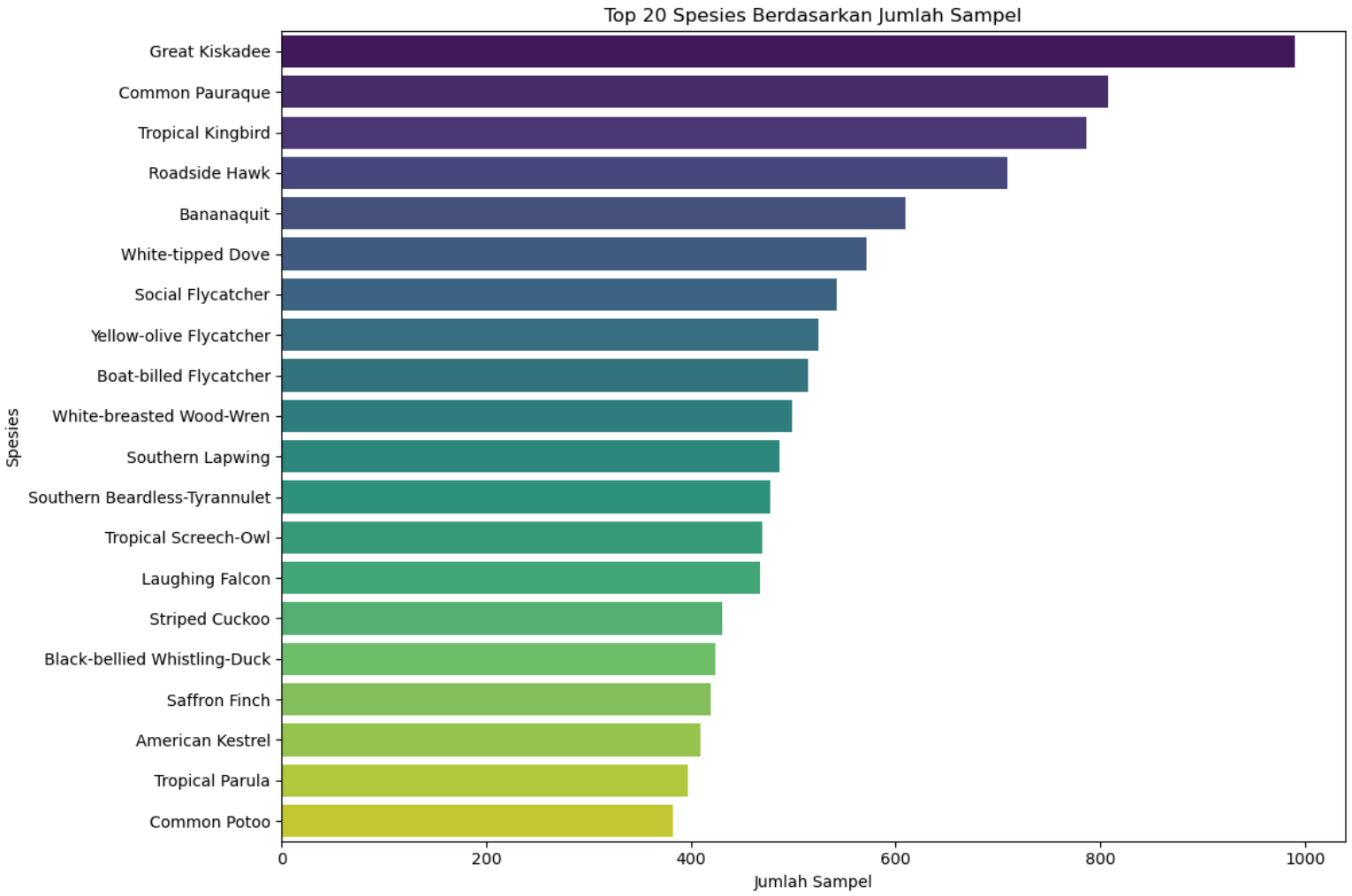
3.1 Evaluasi Hasil Visualisasi EDA

Analisis data eksplorasi (EDA) dilakukan untuk memahami secara menyeluruh karakteristik dataset sebelum memasuki tahap pemodelan. Visualisasi pertama yang dilakukan adalah distribusi jumlah rekaman audio per spesies. Hasilnya menunjukkan adanya ketidakseimbangan kelas di mana beberapa spesies memiliki jumlah data yang sangat tinggi, sementara spesies lainnya memiliki sampel yang sangat sedikit. Hal ini penting untuk diantisipasi karena dapat mempengaruhi kinerja model dalam mendeteksi spesies langka.

Selanjutnya, sebaran kualitas audio (rating) divisualisasikan dalam bentuk histogram yang menunjukkan bahwa mayoritas rekaman memiliki kualitas yang baik (rating ≥ 3), meskipun masih ada beberapa rekaman yang memiliki kualitas rendah. Visualisasi sebaran geografis dengan menggunakan scatter plot berdasarkan garis lintang dan garis bujur menunjukkan bahwa rekaman terdistribusi cukup merata di wilayah El Silencio, namun terdapat konsentrasi di wilayah-wilayah tertentu yang menjadi lokasi pengambilan data utama.

Selain itu, *heatmap* korelasi antara fitur numerik dan visualisasi korelasi antara fitur kategorikal menggunakan metrik Cramér's V juga dilakukan. Visualisasi ini membantu mengidentifikasi fitur mana yang memiliki hubungan yang kuat dan relevan untuk proses pelatihan model.

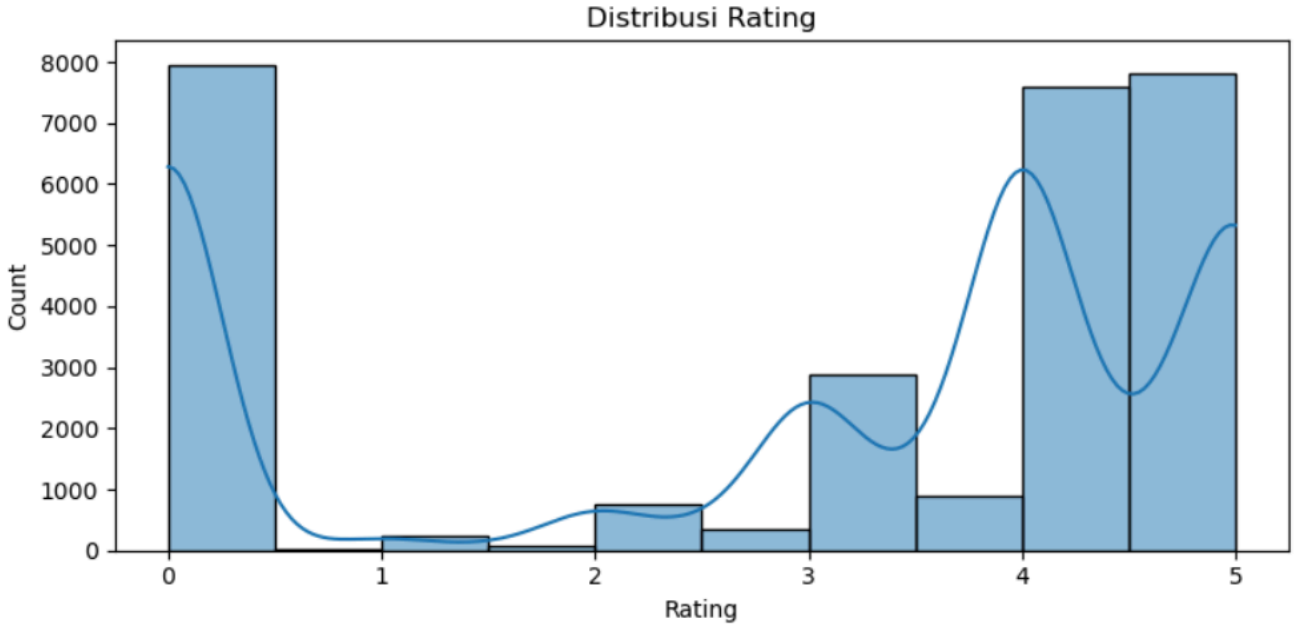
3.1.1 Visualisasi Top 20 Spesies Berdasarkan Jumlah Sampel



Gambar 3.1 Top 20 Spesies Teratas Berdasarkan Jumlah Sampel

Grafik 20 Spesies Teratas Berdasarkan Jumlah Sampel adalah grafik batang horizontal yang menampilkan 20 spesies dengan jumlah sampel terbanyak. Sumbu Y menunjukkan nama spesies, diurutkan dari sampel terendah (bawah) hingga tertinggi (atas), sedangkan sumbu X menunjukkan "Jumlah Sampel". Batang-batang tersebut memiliki panjang dan warna yang berbeda-beda, yang menunjukkan jumlah sampel dari setiap spesies. "Kiskadee Besar" memiliki sampel paling banyak (mendekati 950), sementara spesies seperti 'Potoo Biasa' memiliki lebih sedikit sampel (sekitar 350-400). Visualisasi ini secara efektif membandingkan kelimpahan sampel antar spesies.

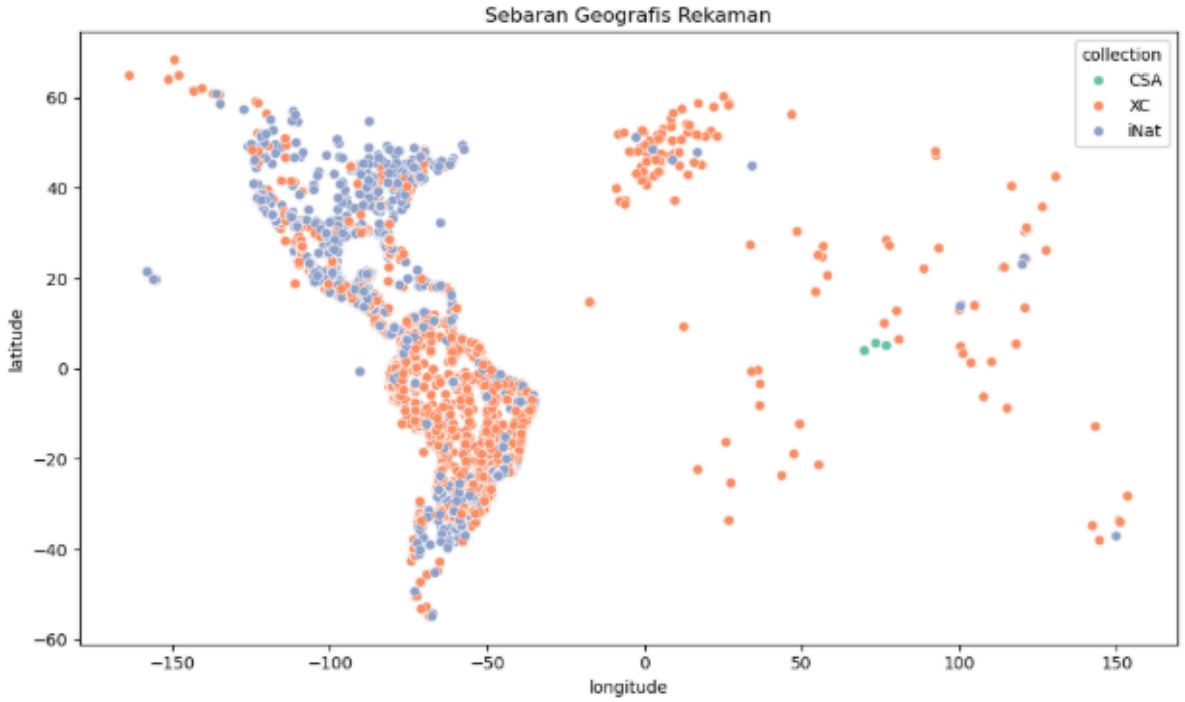
3.1.2 Visualisasi Distribution Rating



Gambar 3.2 Visualisasi Distribution Rating

Gambar 3.1.2 Visualisasi Distribusi diatas menampilkan histogram dan estimasi kepadatan kernel (KDE). Pada sumbu horizontal, terdapat “Rating” yang dapat berkisar dari 0 hingga 5, sedangkan sumbu vertikal menunjukkan “Count” atau jumlah frekuensi. Dari grafik tersebut, dapat dilihat bahwa distribusi rating sangat terpusat pada dua puncak utama: satu puncak yang sangat tinggi berada di rating 0 (sekitar 7900), dan puncak tinggi lainnya berada di rating 4 dan 5 (masing-masing sekitar 7500 dan 7800). Terdapat “lembah” atau frekuensi yang sangat rendah antara peringkat 0 dan sekitar 1,5, dan peningkatan bertahap dari peringkat 1,5 hingga mencapai puncaknya pada peringkat 4 dan 5. Estimasi KDE yang digambarkan oleh garis lengkung juga mengkonfirmasi pola distribusi bimodal (dua puncak) ini, yang menyoroti konsentrasi data pada nilai peringkat ekstrem (0 dan 4-5) dan frekuensi yang jarang dari nilai peringkat di antaranya.

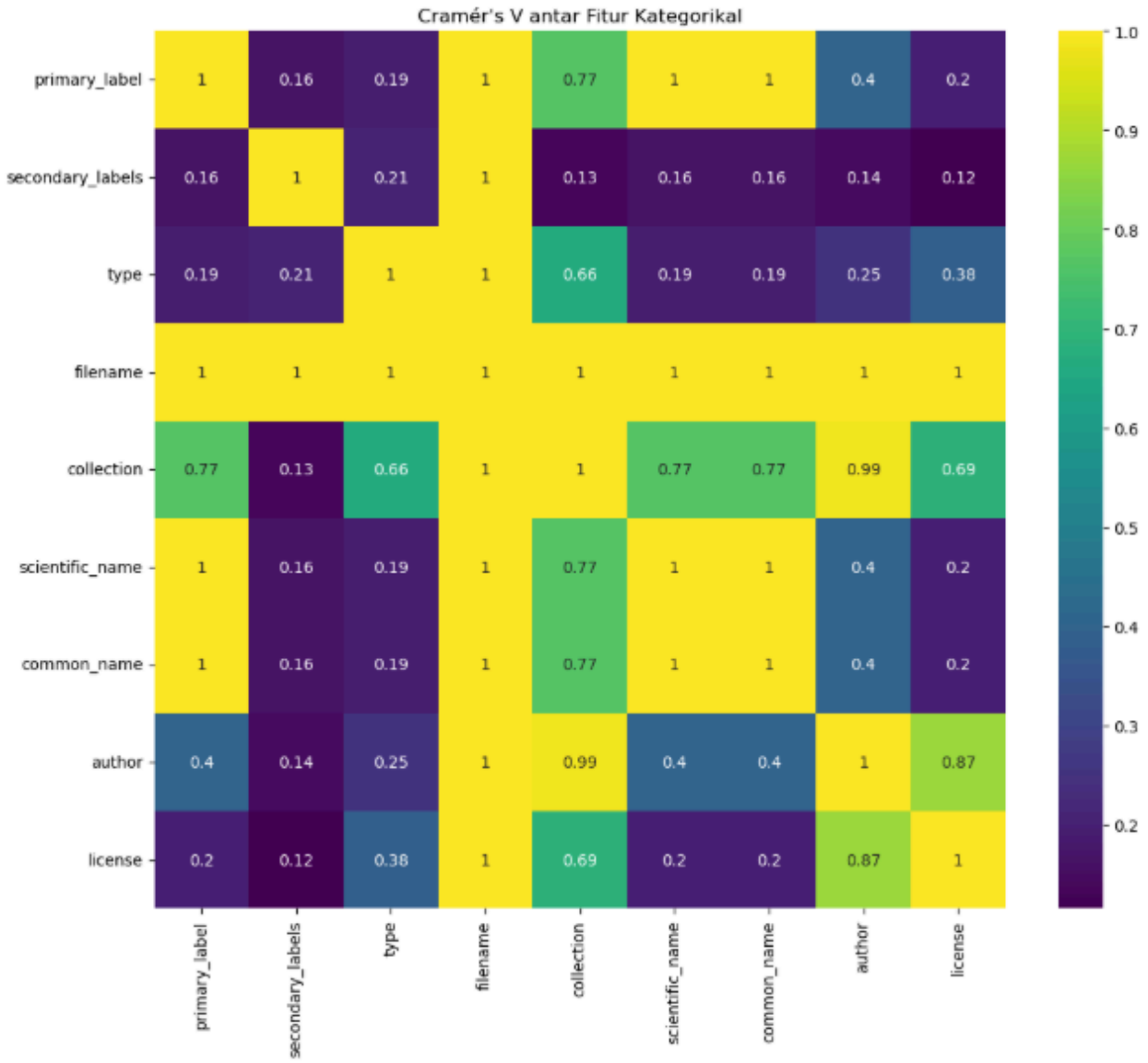
3.1.3 Visualisasi Persebaran Geografis



Gambar 3.3 Visualisasi Persebaran Geografis

Pada peta distribusi geografis rekaman ini, sumbu X menunjukkan garis bujur dan sumbu Y menunjukkan garis lintang, yang menampilkan lokasi geografis titik-titik rekaman di seluruh dunia. Ada tiga kategori koleksi rekaman yang dibedakan berdasarkan warna: “CSA” (hijau kebiruan), 'XC' (oranye), dan 'iNat' (biru keunguan). Dominasi warna “XC” dan “iNat” terlihat jelas, khususnya di benua Amerika Utara dan Selatan, serta sebagian besar Eropa, Asia dan Australia. Koleksi “XC” menunjukkan distribusi yang luas di seluruh benua, sementara “iNat” juga tersebar luas, tetapi tampak lebih padat di Amerika Utara. Menariknya, koleksi “CSA” hanya muncul sebagai beberapa titik kecil yang terkonsentrasi di wilayah Asia Selatan atau Timur Tengah, yang menunjukkan cakupan geografis yang sangat terbatas dibandingkan dengan dua koleksi lainnya. Secara keseluruhan, peta ini memberikan gambaran visual yang komprehensif tentang di mana sebagian besar data yang direkam dikumpulkan, dengan dominasi koleksi “XC” dan “iNat” yang terdistribusi secara global.

3.1.4 Visualisasi Cramer’s V antar Fitur Kategorikal



Gambar 3.4 Cramér's V antar Fitur Kategorikal

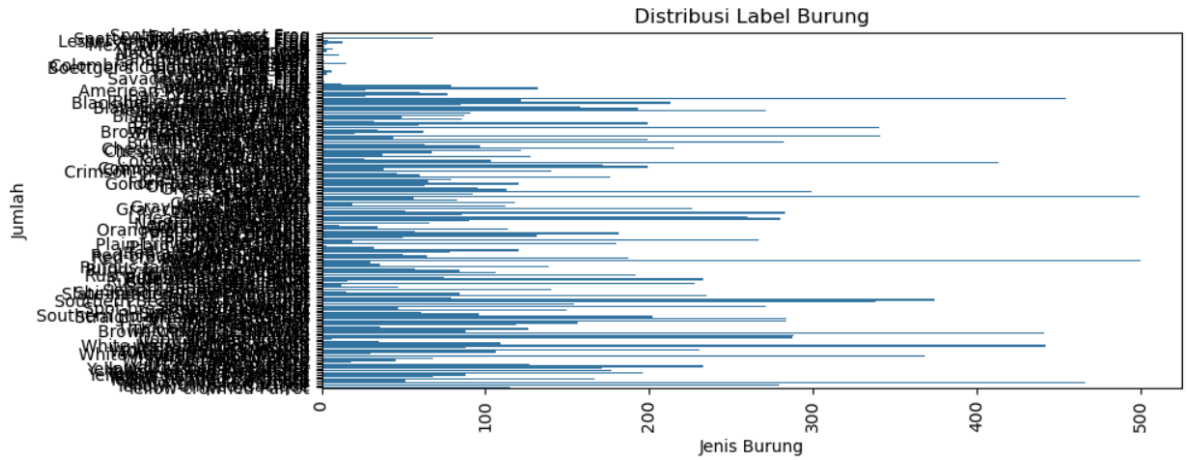
Gambar yang disajikan adalah *heatmap* yang menampilkan Cramér's V di antara Fitur Kategorikal. *heatmap* ini mengukur kekuatan asosiasi antara fitur kategorikal yang berbeda, dengan nilai mulai dari 0 (tidak ada asosiasi) hingga 1 (asosiasi sempurna). Warna *heatmap* bervariasi dari ungu tua (asosiasi lemah) hingga kuning terang (asosiasi kuat).

Dari *heatmap* ini, kita dapat mengamati beberapa hal:

- Diagonal utama memiliki nilai 1, yang mengindikasikan bahwa setiap fitur memiliki asosiasi yang sempurna dengan dirinya sendiri.

- Asosiasi yang kuat (nilai yang mendekati 1) terlihat pada banyak pasangan fitur, terutama yang melibatkan *filename, collection, scientific_name, and common_name*. Sebagai contoh, nama_file memiliki asosiasi sempurna (1) dengan hampir semua fitur lainnya, yang menunjukkan kemungkinan hubungan langsung atau duplikasi informasi. *collection* dan *scientific_name*, serta *common_name*, juga menunjukkan asosiasi yang sangat kuat (0.77 hingga 1) satu sama lain dan dengan *filename*. Hal ini mengindikasikan bahwa fitur-fitur ini mungkin sangat erat kaitannya, atau bahkan mungkin salah satu dari fitur tersebut dapat memprediksi fitur lainnya dengan baik.
- Asosiasi sedang hingga lemah terlihat untuk fitur *primary_label, secondary_label, type, author*, dan *license* dengan fitur lainnya. Sebagai contoh, *primary_label* memiliki asosiasi yang relatif lemah (0,16 hingga 0,19) dengan *secondary_label* dan type, tetapi asosiasi yang lebih kuat (0,77) dengan *collection, scientific_name*, dan *common_name*. pengarang dan lisensi secara umum menunjukkan asosiasi yang bervariasi dari lemah hingga sedang dengan fitur lainnya, meskipun pengarang memiliki asosiasi yang kuat (0,87) dengan lisensi.

3.1.5 Visualisasi Distribusi Label Burung

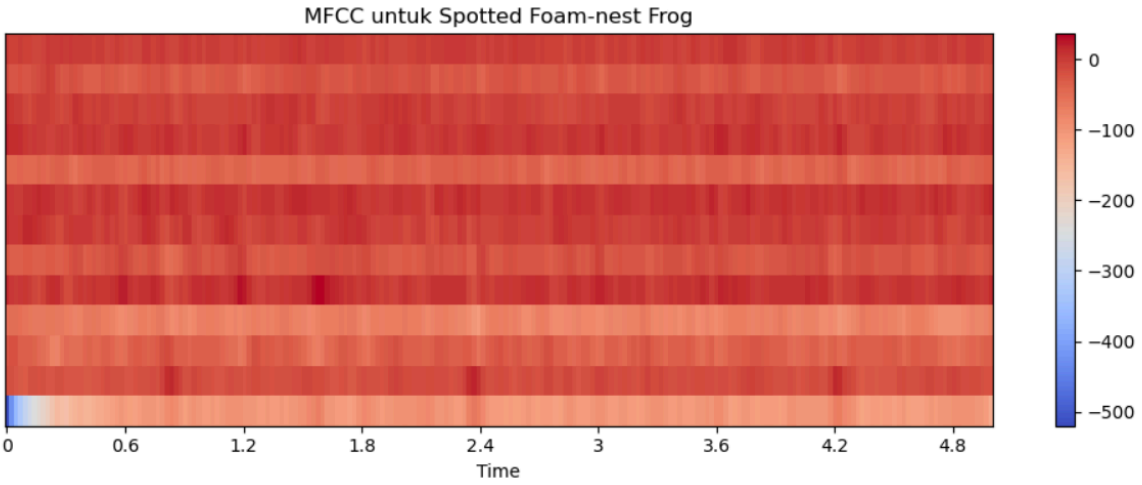


Gambar 3.6 Distribusi Label Burung

Gambar ini menunjukkan grafik batang horizontal dengan sumbu Y (vertikal) menunjukkan berbagai "Jenis Burung" yang, meskipun padat, mewakili spesies yang diamati. Sumbu X (horizontal) menunjukkan "Jumlah" atau frekuensi kemunculan setiap jenis burung, yang diskalakan hingga 500. Panjang setiap batang menunjukkan jumlah pengamatan untuk jenis burung tersebut, yang menunjukkan distribusi yang tidak merata; beberapa jenis burung memiliki jumlah pengamatan yang sangat tinggi (mendekati atau melebihi 500), sementara

yang lainnya hanya memiliki sedikit pengamatan (kurang dari 50). Penting untuk dicatat bahwa visualisasi ini merupakan hasil ekstraksi spesies burung dari data suara (file audio). Dengan demikian, "Angka" pada sumbu X menunjukkan seberapa sering suatu spesies burung teridentifikasi atau terdeteksi dalam koleksi rekaman suara yang dianalisis.

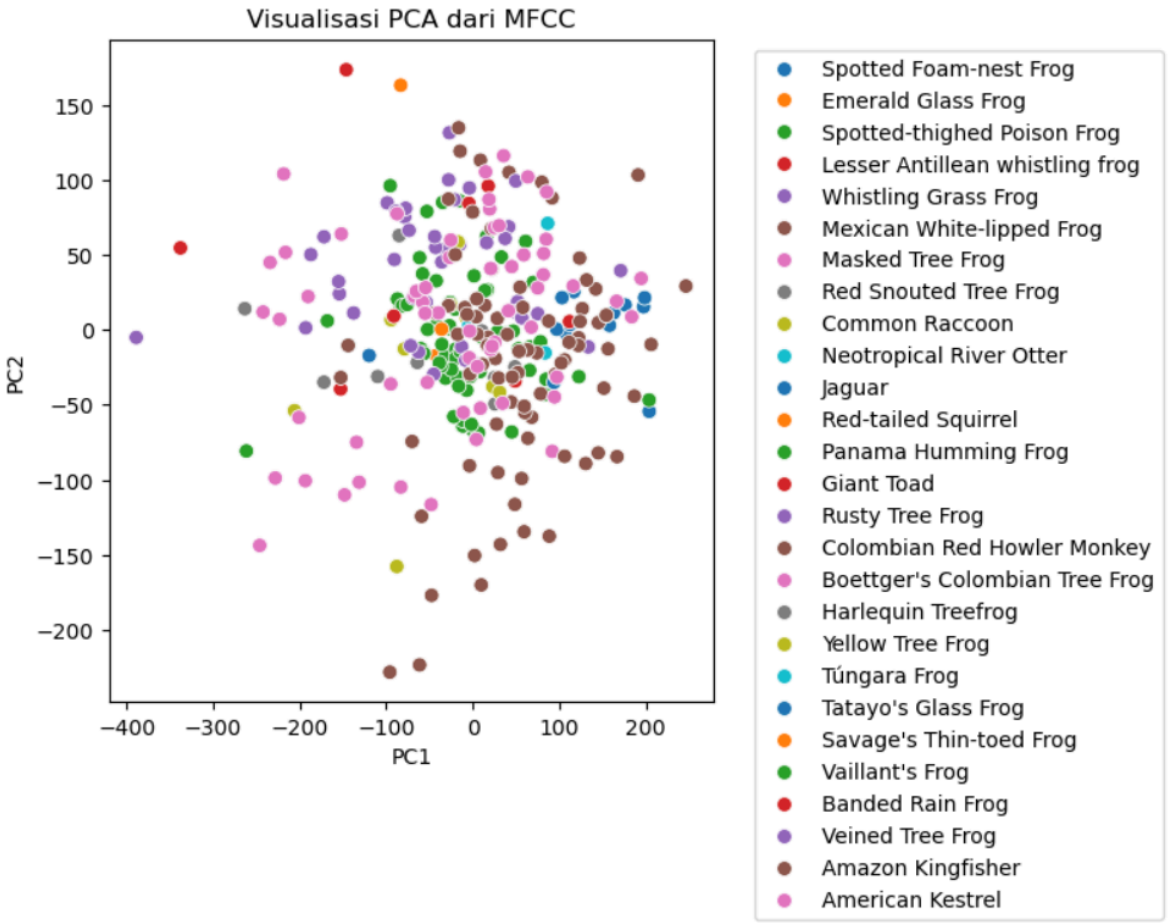
3.1.6 Visualisasi MFCC (Mel-frequency Cepstral Coefficients)



Gambar 3.7 MFCC (Mel-frequency Cepstral Coefficients)

Visualisasi yang ditampilkan adalah *heatmap* yang mewakili MFCC (Mel-frequency Cepstral Coefficients) untuk suara "Spotted Foam-nest Frog". Pada sumbu horizontal (X) adalah "Time" dalam detik, yang menunjukkan durasi rekaman suara yang dianalisis, sekitar 5 detik. Sumbu vertikal (Y) mewakili indeks koefisien MFCC, yang merupakan fitur yang menangkap karakteristik spektral suara, terutama yang relevan dengan persepsi pendengaran manusia. Warna pada *heatmap* menunjukkan nilai amplitudo MFCC, dengan skala warna di sebelah kanan mulai dari biru tua (nilai negatif yang rendah/kuat, sekitar -500) hingga merah terang (nilai positif yang tinggi, sekitar 0). Secara umum, warna yang dominan adalah merah dan jingga, mengindikasikan nilai MFCC yang relatif tinggi, atau mendekati nol untuk sebagian besar durasi perekaman. Namun demikian, terdapat perbedaan nilai yang bervariasi dari waktu ke waktu dan di antara koefisien, membentuk pola horizontal dan vertikal. Pola-pola ini mencerminkan perubahan karakteristik spektral suara katak dari waktu ke waktu, yang sangat berguna dalam aplikasi pengenalan suara dan klasifikasi spesies.

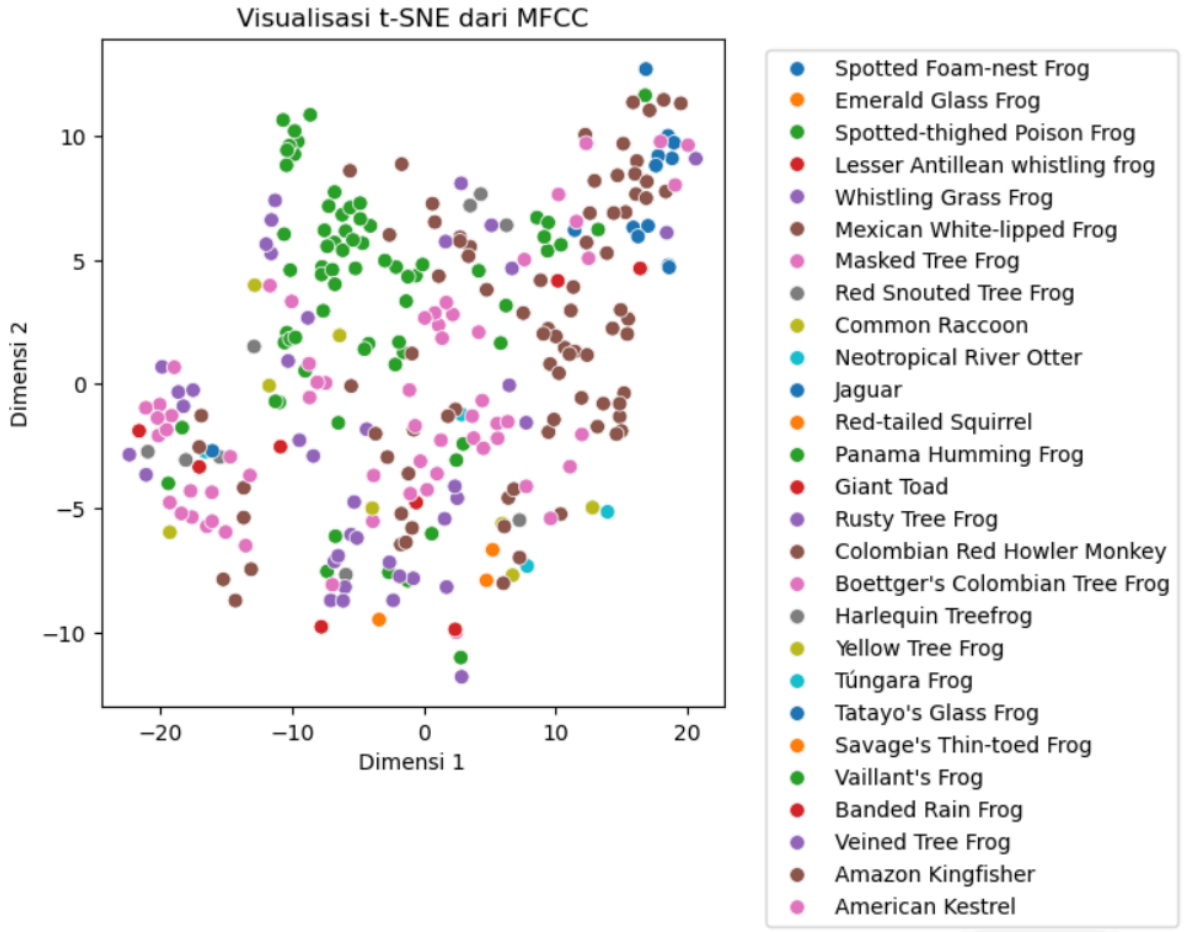
3.1.7 Visualisasi PCA (Principal Component Analysis)



Gambar 3.8 PCA (Principal Component Analysis) dari fitur MFCC

Gambar ini menampilkan visualisasi PCA (Principal Component Analysis) dari fitur MFCC (Mel-frequency Cepstral Coefficients) dari rekaman suara hewan, di mana setiap titik adalah sampel suara yang diproyeksikan ke dalam dua komponen utama (PC1 dan PC2) dan diwarnai berdasarkan spesies. Meskipun ada sedikit pengelompokan, banyak spesies yang tumpang tindih, menunjukkan bahwa kedua komponen ini saja tidak cukup untuk pemisahan yang jelas. Visualisasi PCA ini digunakan untuk mengurangi dimensi data yang kompleks, mengidentifikasi pola awal atau cluster, dan memberikan pemahaman tentang seberapa efektif fitur MFCC dapat membedakan spesies sebelum proses klasifikasi lebih lanjut.

3.1.8 Visualisasi t-SNE (t-Distributed Stochastic Neighbor Embedding)



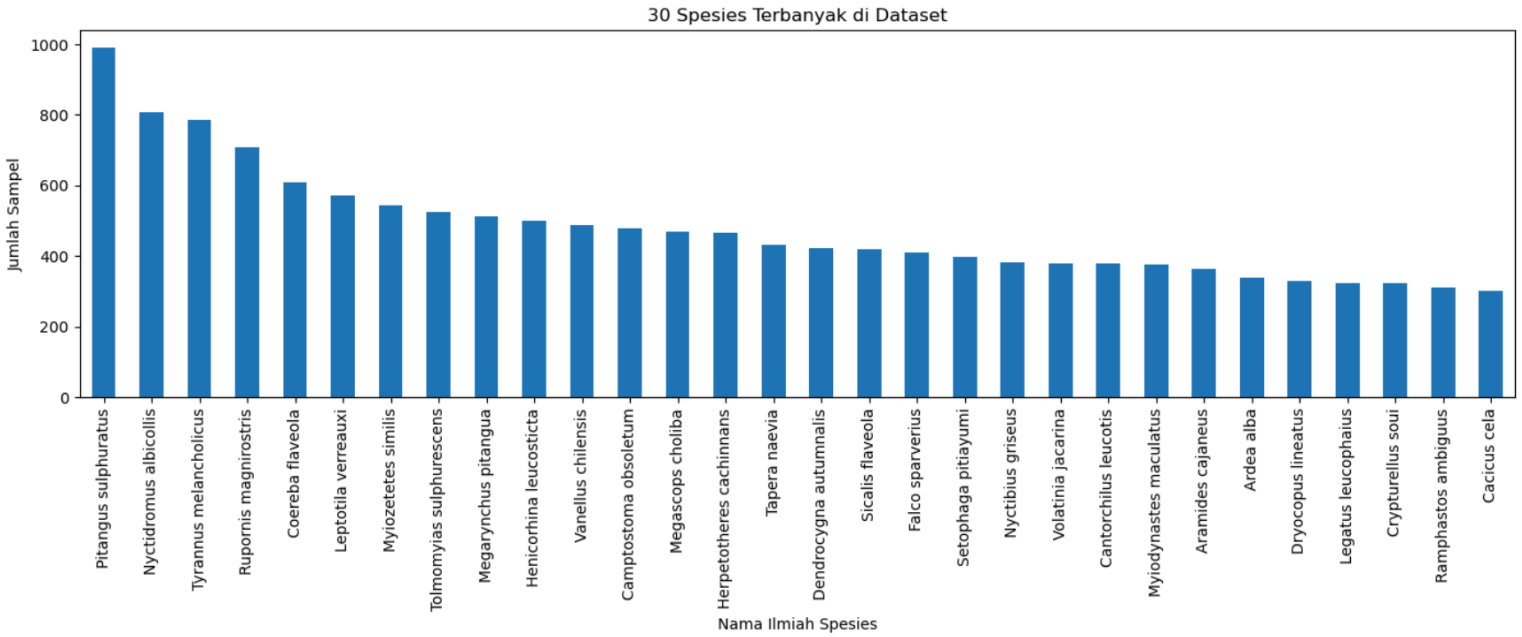
Gambar 3.9 t-SNE (t-Distributed Stochastic Neighbor Embedding) dari fitur MFCC

Gambar ini menampilkan visualisasi t-SNE (t-Distributed Stochastic Neighbor Embedding) dari fitur MFCC dari rekaman suara hewan, memproyeksikan data ke dalam dua dimensi baru di mana titik-titiknya diwarnai berdasarkan spesies asalnya. T-SNE digunakan untuk mengurangi dimensi data dan mengungkapkan struktur klaster non-linear, dan dalam plot ini, meskipun ada beberapa pengelompokan yang lebih ketat dibandingkan dengan PCA, masih ada tumpang tindih yang signifikan antara berbagai spesies, yang menunjukkan kerumitan dalam membedakan mereka hanya dari fitur suara.

Dibandingkan dengan visualisasi PCA sebelumnya, yang merupakan metode linier yang mempertahankan varians global, plot t-SNE ini berusaha mempertahankan jarak yang dekat antara titik-titik dalam ruang dimensi tinggi, yang sering kali menghasilkan kelompok yang terlihat lebih padat. Namun, tumpang tindih yang signifikan tetap ada di kedua visualisasi tersebut, yang menunjukkan bahwa fitur MFCC, baik yang direduksi secara linier (PCA) atau

non-linier (t-SNE), memiliki keterbatasan dalam memisahkan secara sempurna semua spesies hewan berdasarkan karakteristik suaranya.

3.1.9 Visualisasi 30 Spesies Terbanyak dalam Dataset dengan Nama Ilmiahnya



Gambar 3.10 Visualisasi 30 Spesies Terbanyak dalam Dataset dengan Nama Ilmiahnya

Grafik batang vertikal berjudul “30 Spesies Teratas dalam Dataset” ini menyajikan frekuensi kemunculan 30 spesies teratas berdasarkan jumlah sampel. Sumbu horizontal menampilkan “Nama Ilmiah Spesies” yang dirotasi agar mudah dibaca, sedangkan sumbu vertikal menunjukkan “Jumlah Sampel” hingga 1000. Berdasarkan visualisasi ini, terlihat jelas bahwa “Pitangus sulphuratus” merupakan spesies dengan jumlah sampel terbanyak, mendekati 1000, diikuti oleh spesies lain seperti “Nyctidromus albicollis” dan “Tyrannus melancholicus”, yang menunjukkan penyebaran yang tidak merata di mana beberapa spesies lebih sering muncul di dalam set data dibandingkan spesies lainnya.

3.2 Hasil Analisis Modelling dan Akurasi EDA

1. **Arsitektur Model**

   Model klasifikasi ucapan pada penelitian ini dirancang menggunakan Keras (TensorFlow backend) dengan pendekatan bertahap menggunakan arsitektur Convolutional Neural Network (CNN). CNN dipilih karena kemampuannya dalam mengenali pola spasial dan temporal dari sebuah citra dua dimensi yang dihasilkan oleh fitur MFCC (Mel Frequency Cepstral Coefficients) yang merupakan representasi visual dari sinyal audio.

Struktur arsitektur CNN yang digunakan adalah sebagai berikut:

- **2 Conv2D layers,** masing-masing menggunakan filter 3x3 dan fungsi aktivasi ReLU, efektif dalam menangkap pola lokal dari data MFCC.
- **MaxPooling2D layer** diterapkan setelah setiap lapisan konvolusi untuk mengurangi dimensi fitur dan mencegah overfitting dengan mengurangi kompleksitas spasial.
- **Dropout layer** digunakan untuk mengurangi kemungkinan overfitting dengan menonaktifkan sejumlah neuron secara acak selama pelatihan.
- **Flatten layer** untuk mengubah output 2D menjadi 1D sebelum memasuki dense layer.
- The first dense layer dengan 64 neuron dan aktivasi ReLU.
- **The last dense layer** menggunakan fungsi aktivasi Softmax untuk melakukan klasifikasi ke dalam sejumlah kelas spesies yang telah ditentukan.

Gambar Code Arsitektur CNN:

```
model = models.Sequential()
model.add(layers.Conv2D(16, (3,3), activation='relu', input_shape=(400, 40, 1)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(32, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.3))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(len(le.classes_), activation='softmax'))
```

Gambar 3.2.1 Code Arsitektur CNN

Model ini kemudian dikompilasi menggunakan:

- **Loss function:** sparse_categorical_crossentropy karena label masih berupa bilangan bulat (tidak dikodekan satu-panas).
- **Optimizer:** Adam, yang sangat populer untuk melatih model jaringan syaraf karena adaptif terhadap laju pembelajaran.
- **Evaluation metric:** akurasi untuk memantau kinerja selama pelatihan dan validasi.

Gambar Code Kompilasi:

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Gambar 3.2.2 Code Kompilasi

2. **Proses Pelatihan dan Validasi**

   Sebelum pelatihan dilakukan, dataset dibagi menjadi dua bagian:

   - 80% untuk training data (X_train, y_train)
   - 20% validation data (X_val, y_val).

   Proses pelatihan dilakukan selama maksimal 50 epoch dengan ukuran batch 16. Untuk menghindari overfitting, digunakan strategi EarlyStopping, yang secara otomatis menghentikan pelatihan ketika tidak ada peningkatan val_loss selama beberapa epoch berturut-turut (dalam hal ini ditentukan oleh sabar = 5).

Gambar Code Pelatihan (training):

```
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(X_train, y_train,
                    epochs=50,
                    batch_size=16,
                    validation_data=(X_val, y_val),
                    callbacks=[early_stop])
```

Gambar 3.2.3 Code Pelatihan (training)

3. **Visualisasi Kurva Akurasi**

   Untuk mengevaluasi stabilitas dan proses pembelajaran model selama pelatihan, grafik akurasi pelatihan dan validasi per epoch dibuat. Visualisasi ini sangat penting untuk memahami apakah model mengalami overfitting (akurasi meningkat pada saat pelatihan namun menurun pada saat validasi) atau underfitting (akurasi rendah pada keduanya).

   Grafik yang dihasilkan menunjukkan tren akurasi pelatihan dan validasi yang meningkat dan konvergen, yang mengindikasikan bahwa model belajar dengan mantap dan tidak mengalami overfitting secara signifikan.

Gambar Code Visualisasi:

```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.legend()
plt.title('Training & Validation Accuracy (Subset Class)')
plt.show()
```

Gambar 3.2.4 Code Visualisasi Akurasi Kurva

Visualisasi ini memberikan gambaran penting bahwa model dapat belajar dari data training dengan baik dan tetap mempertahankan akurasi yang tinggi pada data yang belum pernah dilihat sebelumnya (data validasi). Pada tahap akhir, model CNN diuji melalui leaderboard kompetisi dengan hidden test set, menghasilkan public score sebesar 0.581, yang tergolong kompetitif di leaderboard awal.

3.3 Lampiran Code Project

Project Datmod

[4]:
```
from pathlib import Path

dataset_path = Path("C:/Users/ricar/Downloads/Dataset Datmod")

for item in dataset_path.iterdir():
    print(item)
```

```
C:\Users\ricar\Downloads\Dataset Datmod\recording_location.txt
C:\Users\ricar\Downloads\Dataset Datmod\sample_submission.csv
C:\Users\ricar\Downloads\Dataset Datmod\taxonomy.csv
C:\Users\ricar\Downloads\Dataset Datmod\test_soundscapes
C:\Users\ricar\Downloads\Dataset Datmod\train.csv
C:\Users\ricar\Downloads\Dataset Datmod\train_audio
C:\Users\ricar\Downloads\Dataset Datmod\train_soundscapes
```

[ ]:

EDA

[6]:
```
import pandas as pd
df = pd.read_csv("C:/Users/ricar/Downloads/Dataset Datmod/train.csv")
print(df.head(10))
print("\nKolom yang tersedia:")
print(df.columns)
```

	primary_label	secondary_labels	type
0	1139490	['']	['']
1	1139490	['']	['']
2	1192948	['']	['']
3	1192948	['']	['']
4	1192948	['']	['']
5	1192948	['']	['']
6	1194042	['']	['']
7	1194042	['']	['']
8	1194042	['']	['']
9	126247	['65448', '22976', '476538']	['advertisement call']

	filename	collection	rating
0	1139490/CSA36385.ogg	CSA	0.0
1	1139490/CSA36389.ogg	CSA	0.0
2	1192948/CSA36358.ogg	CSA	0.0
3	1192948/CSA36366.ogg	CSA	0.0
4	1192948/CSA36373.ogg	CSA	0.0
5	1192948/CSA36388.ogg	CSA	0.0
6	1194042/CSA18783.ogg	CSA	0.0
7	1194042/CSA18794.ogg	CSA	0.0
8	1194042/CSA18802.ogg	CSA	0.0
9	126247/XC941297.ogg	XC	3.5

	url	latitude	longitude
0	http://colecciones.humboldt.org.co/rec/sonidos...	7.3206	-73.7128
1	http://colecciones.humboldt.org.co/rec/sonidos...	7.3206	-73.7128
2	http://colecciones.humboldt.org.co/rec/sonidos...	7.3791	-73.7313
3	http://colecciones.humboldt.org.co/rec/sonidos...	7.2800	-73.8582
4	http://colecciones.humboldt.org.co/rec/sonidos...	7.3791	-73.7313
5	http://colecciones.humboldt.org.co/rec/sonidos...	7.3791	-73.7313
6	http://colecciones.humboldt.org.co/rec/sonidos...	5.7892	-73.5504
7	http://colecciones.humboldt.org.co/rec/sonidos...	5.7892	-73.5504
8	http://colecciones.humboldt.org.co/rec/sonidos...	5.7892	-73.5504
9	https://xeno-canto.org/941297	9.0465	-79.3024

```
           scientific_name            common_name                  author  \
0     Ragoniella pulchella    Ragoniella pulchella      Fabio A. Sarria-S
1     Ragoniella pulchella    Ragoniella pulchella      Fabio A. Sarria-S
2    Oxyprora surinamensis   Oxyprora surinamensis      Fabio A. Sarria-S
3    Oxyprora surinamensis   Oxyprora surinamensis      Fabio A. Sarria-S
4    Oxyprora surinamensis   Oxyprora surinamensis      Fabio A. Sarria-S
5    Oxyprora surinamensis   Oxyprora surinamensis      Fabio A. Sarria-S
6      Copiphora colombiae     Copiphora colombiae  Diego A Gómez-Morales
7      Copiphora colombiae     Copiphora colombiae  Orlando Acevedo-Charry
8      Copiphora colombiae     Copiphora colombiae  Orlando Acevedo-Charry
9  Leptodactylus insularum  Spotted Foam-nest Frog          Chris Harrison

           license
0  cc-by-nc-sa 4.0
1  cc-by-nc-sa 4.0
2  cc-by-nc-sa 4.0
3  cc-by-nc-sa 4.0
4  cc-by-nc-sa 4.0
5  cc-by-nc-sa 4.0
6  cc-by-nc-sa 4.0
7  cc-by-nc-sa 4.0
8  cc-by-nc-sa 4.0
9  cc-by-nc-sa 4.0

Kolom yang tersedia:
Index(['primary_label', 'secondary_labels', 'type', 'filename', 'collection',
       'rating', 'url', 'latitude', 'longitude', 'scientific_name',
       'common_name', 'author', 'license'],
      dtype='object')
```

[11]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

print("Statistik Deskriptif:")
print(df[["rating", "latitude", "longitude"]].describe())

plt.figure(figsize=(12, 8))
top_labels = df["common_name"].value_counts().head(20)
sns.barplot(y=top_labels.index, x=top_labels.values, palette="viridis")
plt.title("Top 20 Spesies Berdasarkan Jumlah Sampel")
plt.xlabel("Jumlah Sampel")
plt.ylabel("Spesies")
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 4))
sns.histplot(df["rating"], bins=10, kde=True)
plt.title("Distribusi Rating")
plt.xlabel("Rating")
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x="longitude", y="latitude", hue="collection", palette="Set2")
plt.title("Sebaran Geografis Rekaman")
plt.tight_layout()
plt.show()

plt.figure(figsize=(6, 5))
corr = df[["rating", "latitude", "longitude"]].corr()
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Korelasi Antar Fitur Numerik")
plt.tight_layout()
plt.show()

def detect_outliers(col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    outliers = ((df[col] < (Q1 - 1.5 * IQR)) | (df[col] > (Q3 + 1.5 * IQR))).sum()
    return outliers

print("\nJumlah Outliers:")
for col in ["rating", "latitude", "longitude"]:
    print(f"{col}: {detect_outliers(col)}")
```

```
Statistik Deskriptif:
             rating      latitude     longitude
count  28564.000000  27755.000000  27755.000000
mean       2.917063     -0.533480    -68.524380
std        1.964896     17.609276     21.614566
min        0.000000    -54.857400   -163.680000
25%        0.000000    -15.084600    -79.649200
50%        4.000000      1.131600    -73.495400
75%        4.500000      9.511000    -53.801800
max        5.000000     68.374800    153.551400
```

```
C:\Users\ricar\AppData\Local\Temp\ipykernel_26076\3419720181.py:10: FutureWarning:
```

```python
from scipy.stats import chi2_contingency
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

def cramers_v(confusion_matrix):
    chi2 = chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    r, k = confusion_matrix.shape
    return np.sqrt(chi2 / (n * (min(k-1, r-1))))


cat_cols = ['primary_label', 'secondary_labels', 'type', 'filename', 'collection',
            'scientific_name', 'common_name', 'author', 'license']


v_matrix = pd.DataFrame(index=cat_cols, columns=cat_cols)


for col1 in cat_cols:
    for col2 in cat_cols:
        if col1 == col2:
            v_matrix.loc[col1, col2] = 1.0
        else:
            ct = pd.crosstab(df[col1], df[col2])
            v = cramers_v(ct)
            v_matrix.loc[col1, col2] = v


v_matrix = v_matrix.astype(float)
plt.figure(figsize=(12,10))
sns.heatmap(v_matrix, annot=True, cmap='viridis')
plt.title("Cramér's V antar Fitur Kategorikal")
plt.show()
```

Cleaning

```
df_clean = df[df["rating"] > 0].copy()
df_clean = df_clean.drop_duplicates(subset=["filename"])

print(f"Jumlah data setelah dibersihkan: {len(df_clean)}")
```

```
Jumlah data setelah dibersihkan: 20616
```

Install librosa untuk extract audio

```
pip install librosa
```

```
import librosa
import numpy as np
from tqdm import tqdm
import os
audio_base_path = "C:/Users/ricar/Downloads/Dataset Datmod/train_audio"
def extract_mfcc(file_path, n_mfcc=13, duration=5, sr=22050):
    try:
        audio, _ = librosa.load(file_path, sr=sr, duration=duration)
        mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=n_mfcc)
        return mfcc.mean(axis=1)
    except Exception as e:
        print(f"Error loading {file_path}: {e}")
        return np.zeros(n_mfcc)
features = []
labels = []

for i, row in tqdm(df_clean.iterrows(), total=len(df_clean)):
    path = os.path.join(audio_base_path, row["filename"])
    mfcc = extract_mfcc(path)
    features.append(mfcc)
    labels.append(row["common_name"])
X = np.array(features)
y = np.array(labels)

print("Shape fitur MFCC:", X.shape)
```

```
100%|██████████| 20616/20616 [14:58<00:00, 22.95it/s]
Shape fitur MFCC: (20616, 13)
```

```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import pandas as pd


plt.figure(figsize=(10, 4))
sns.countplot(y)
plt.xticks(rotation=90)
plt.title("Distribusi Label Burung")
plt.xlabel("Jenis Burung")
plt.ylabel("Jumlah")
plt.tight_layout()
plt.show()


example_file = os.path.join(audio_base_path, df_clean["filename"].iloc[0])
audio, sr = librosa.load(example_file, sr=22050, duration=5)
mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=13)

plt.figure(figsize=(10, 4))
librosa.display.specshow(mfcc, x_axis='time', sr=sr)
plt.colorbar()
plt.title(f"MFCC untuk {df_clean['common_name'].iloc[0]}")
plt.tight_layout()
plt.show()


sample_size = min(300, len(X))
X_sample = X[:sample_size]
y_sample = y[:sample_size]


pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_sample)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=y_sample, palette='tab10', s=50)
plt.title("Visualisasi PCA dari MFCC")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()


tsne = TSNE(n_components=2, perplexity=30, n_iter=1000, random_state=42)
X_tsne = tsne.fit_transform(X_sample)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_tsne[:, 0], y=X_tsne[:, 1], hue=y_sample, palette='tab10', s=50)
plt.title("Visualisasi t-SNE dari MFCC")
plt.xlabel("Dimensi 1")
plt.ylabel("Dimensi 2")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

```python
import pandas as pd

label_counts = pd.Series(labels).value_counts()
valid_labels = label_counts[label_counts >= 2].index
filtered_indices = [i for i, label in enumerate(labels) if label in valid_labels]
X_filtered = X[filtered_indices]
y_filtered = np.array([labels[i] for i in filtered_indices])
```

```python
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import pandas as pd



filtered_mask = pd.Series(y_filtered).map(pd.Series(y_filtered).value_counts()) >= 2
X_filtered = X_filtered[filtered_mask]
y_filtered = y_filtered[filtered_mask]


le = LabelEncoder()
y_encoded = le.fit_transform(y_filtered)


X_train, X_val, y_train, y_val = train_test_split(
    X_filtered, y_encoded, test_size=0.2, stratify=y_encoded, random_state=42
)

print(f"Train shape: {X_train.shape}, Validation shape: {X_val.shape}")
```

```
Train shape: (16485, 13), Validation shape: (4122, 13)
```

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_val)

print("Accuracy:", accuracy_score(y_val, y_pred))



unique_labels = np.unique(y_val)
```

Classification Report:

	precision	recall	f1-score	support
Amazon Kingfisher	0.00	0.00	0.00	16
American Kestrel	0.21	0.15	0.18	26
American Pygmy Kingfisher	0.00	0.00	0.00	5
Anhinga	0.33	0.17	0.22	12
Bananaquit	0.15	0.45	0.23	91
Bare-faced Ibis	0.25	0.20	0.22	5
Bay-breasted Warbler	0.00	0.00	0.00	15
Bicolored Wren	0.36	0.24	0.29	17
Black Vulture	0.12	0.08	0.10	12
Black-bellied Whistling-Duck	0.23	0.19	0.21	43
Black-bellied Wren	0.25	0.06	0.09	18
Black-capped Donacobius	0.22	0.12	0.16	32
Black-chested Jay	0.10	0.06	0.07	17
Black-collared Hawk	0.00	0.00	0.00	10
Black-crested Antshrike	0.31	0.29	0.30	17
Black-crowned Tityra	0.00	0.00	0.00	6
Blue-and-yellow Macaw	0.12	0.08	0.10	24
Blue-billed Curassow	0.00	0.00	0.00	4
Blue-black Grassquit	0.11	0.13	0.12	54
Blue-gray Tanager	0.07	0.07	0.07	40
Blue-headed Parrot	0.05	0.07	0.06	40
Boat-billed Flycatcher	0.06	0.07	0.07	68
Boat-billed Heron	0.00	0.00	0.00	7
Brown-throated Parakeet	0.00	0.00	0.00	12
Brown-winged Schiffornis	0.25	0.11	0.15	18
Buff-breasted Wren	0.15	0.28	0.20	68
Buff-throated Saltator	0.10	0.14	0.11	57
Carib Grackle	0.23	0.23	0.23	13
Cattle Tyrant	0.09	0.05	0.07	19
Chestnut-backed Antbird	0.12	0.14	0.13	43
Chestnut-fronted Macaw	0.11	0.12	0.12	24
Cinnamon Becard	0.67	0.15	0.25	13
Cocoa Woodcreeper	0.12	0.12	0.12	26
Cocoi Heron	0.50	0.14	0.22	7
Collared Aracari	0.00	0.00	0.00	5
Colombian Chachalaca	0.20	0.19	0.20	21
Common Pauraque	0.14	0.27	0.18	83
Common Potoo	0.19	0.21	0.20	34
Common Raccoon	0.00	0.00	0.00	1
Common Tody-Flycatcher	0.05	0.05	0.05	40
Crested Bobwhite	0.00	0.00	0.00	9
Crested Caracara	0.08	0.04	0.05	25
Crested Guan	0.00	0.00	0.00	12
Crested Oropendola	0.14	0.09	0.11	35
Crimson-backed Tanager	0.00	0.00	0.00	8
Crimson-crested Woodpecker	0.10	0.04	0.05	28
Eared Dove	0.00	0.00	0.00	16
Fork-tailed Flycatcher	1.00	0.23	0.38	13
Golden-headed Manakin	0.20	0.12	0.15	24
Gray Seedeater	0.33	0.25	0.29	4

Smooth-billed Ani	0.12	0.11	0.11	47
Snowy Egret	0.33	0.06	0.11	16
Social Flycatcher	0.10	0.10	0.10	68
Solitary Sandpiper	0.11	0.03	0.05	31
Southern Beardless-Tyrannulet	0.06	0.07	0.06	75
Southern Lapwing	0.27	0.37	0.31	54
Southern Rough-winged Swallow	0.12	0.05	0.07	19
Spectacled Owl	0.35	0.23	0.28	30
Spectacled Parrotlet	0.00	0.00	0.00	12
Spot-breasted Woodpecker	0.50	0.11	0.18	9
Spotted-thighed Poison Frog	0.35	0.50	0.41	14
Straight-billed Woodcreeper	0.09	0.10	0.09	40
Streaked Flycatcher	0.07	0.07	0.07	57
Striated Heron	0.29	0.13	0.18	31
Striped Cuckoo	0.18	0.14	0.16	57
Striped Owl	0.00	0.00	0.00	24
Thick-billed Euphonia	0.00	0.00	0.00	25
Thick-billed Seed-Finch	0.00	0.00	0.00	7
Tropical Kingbird	0.15	0.32	0.20	88
Tropical Parula	0.13	0.21	0.16	58
Tropical Screech-Owl	0.16	0.14	0.15	58
Turkey Vulture	0.00	0.00	0.00	1
Túngara Frog	0.00	0.00	0.00	1
Veined Tree Frog	0.00	0.00	0.00	2
Vermilion Flycatcher	0.00	0.00	0.00	7
Wattled Jacana	0.00	0.00	0.00	22
Whistling Grass Frog	0.00	0.00	0.00	3
White-bearded Manakin	0.00	0.00	0.00	46
White-bellied Antbird	0.50	0.28	0.36	18
White-breasted Wood-Wren	0.11	0.15	0.13	89
White-fringed Antwren	0.00	0.00	0.00	21
White-headed Marsh Tyrant	0.00	0.00	0.00	6
White-tailed Trogon	0.25	0.07	0.11	14
White-tipped Dove	0.20	0.30	0.24	74
White-winged Swallow	0.00	0.00	0.00	9
Whooping Motmot	0.00	0.00	0.00	9
Wood Stork	0.00	0.00	0.00	4
Yellow Oriole	0.00	0.00	0.00	10
Yellow Tree Frog	0.00	0.00	0.00	1
Yellow-bellied Elaenia	0.26	0.30	0.28	47
Yellow-bellied Seedeater	0.19	0.09	0.12	35
Yellow-breasted Flycatcher	0.00	0.00	0.00	34
Yellow-chinned Spinetail	0.24	0.21	0.22	39
Yellow-crowned Parrot	0.22	0.09	0.12	23
Yellow-crowned Tyrannulet	0.00	0.00	0.00	18
Yellow-headed Caracara	0.17	0.06	0.09	33
Yellow-hooded Blackbird	0.17	0.07	0.10	14
Yellow-olive Flycatcher	0.08	0.13	0.10	93
Yellow-rumped Cacique	0.05	0.04	0.04	56
Yellow-throated Toucan	0.45	0.36	0.40	39
micro avg	0.14	0.14	0.14	4122
macro avg	0.14	0.10	0.11	4122
weighted avg	0.14	0.14	0.13	4122

```
import matplotlib.pyplot as plt

species_counts = df['scientific_name'].value_counts()

plt.figure(figsize=(14, 6))
species_counts.head(30).plot(kind='bar')
plt.title('30 Spesies Terbanyak di Dataset')
plt.xlabel('Nama Ilmiah Spesies')
plt.ylabel('Jumlah Sampel')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

```
import os
import numpy as np
import pandas as pd
import librosa
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tqdm.notebook import tqdm
from pathlib import Path
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt


train_audio_path = Path("C:/Users/ricar/Downloads/Dataset Datmod/train_audio")
df = pd.read_csv("C:/Users/ricar/Downloads/Dataset Datmod/train.csv")
```

```
le = LabelEncoder()
df = df[df['primary_label'].notnull()]
y = le.fit_transform(df['primary_label'])
```

```
def extract_mfcc(file_path, max_len=400, n_mfcc=40):
    try:
        y, sr = librosa.load(file_path, sr=32000)
        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
        mfcc = mfcc.T
        if mfcc.shape[0] < max_len:
            pad_width = max_len - mfcc.shape[0]
            mfcc = np.pad(mfcc, ((0, pad_width), (0,0)), mode='constant')
        else:
            mfcc = mfcc[:max_len, :]
        return mfcc
    except:
        return np.zeros((max_len, n_mfcc))


X = []
for filename in tqdm(df['filename']):
    file_path = train_audio_path / filename
    X.append(extract_mfcc(file_path))

X = np.array(X)
y = np.array(y)
```

```
  0%|          | 0/28564 [00:00<?, ?it/s]
```

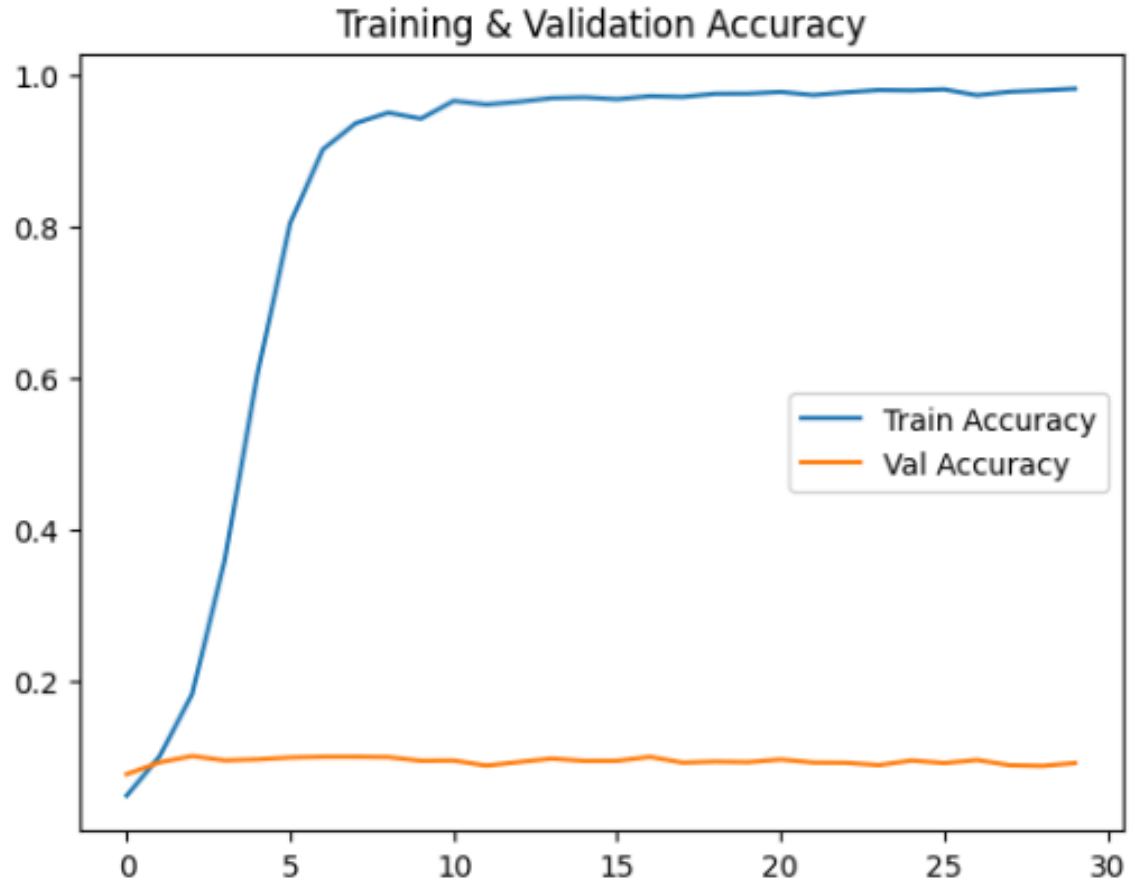
```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)


X_train = X_train[..., np.newaxis]
X_val = X_val[..., np.newaxis]


model = models.Sequential()
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(400, 40, 1)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(len(le.classes_), activation='softmax'))


model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=30, batch_size=16, validation_data=(X_val, y_val))
```

```
Epoch 1/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 228s 157ms/step - accuracy: 0.0353 - loss: 6.3722 - val_accuracy: 0.0775 - val_loss: 4.5248
Epoch 2/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 223s 156ms/step - accuracy: 0.0964 - loss: 4.3175 - val_accuracy: 0.0929 - val_loss: 4.3459
Epoch 3/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 230s 161ms/step - accuracy: 0.1826 - loss: 3.7270 - val_accuracy: 0.1013 - val_loss: 4.4106
Epoch 4/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 256s 157ms/step - accuracy: 0.3830 - loss: 2.6747 - val_accuracy: 0.0956 - val_loss: 5.1328
Epoch 5/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 260s 155ms/step - accuracy: 0.6241 - loss: 1.5597 - val_accuracy: 0.0970 - val_loss: 7.1111
Epoch 6/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 221s 155ms/step - accuracy: 0.8230 - loss: 0.7086 - val_accuracy: 0.0994 - val_loss: 9.1263
Epoch 7/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 222s 156ms/step - accuracy: 0.9202 - loss: 0.3235 - val_accuracy: 0.1005 - val_loss: 11.6019
Epoch 8/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 262s 155ms/step - accuracy: 0.9440 - loss: 0.2432 - val_accuracy: 0.1005 - val_loss: 13.2339
Epoch 9/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 222s 155ms/step - accuracy: 0.9615 - loss: 0.1611 - val_accuracy: 0.0999 - val_loss: 14.1303
Epoch 10/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 222s 155ms/step - accuracy: 0.9484 - loss: 0.2255 - val_accuracy: 0.0950 - val_loss: 15.6284
Epoch 11/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 262s 155ms/step - accuracy: 0.9746 - loss: 0.1048 - val_accuracy: 0.0954 - val_loss: 15.7254
```

```
Epoch 12/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 263s 156ms/step - accuracy: 0.9710 - loss: 0.1181 - val_accuracy: 0.0887 - val_loss: 17.0113
Epoch 13/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 231s 162ms/step - accuracy: 0.9708 - loss: 0.1279 - val_accuracy: 0.0935 - val_loss: 17.3865
Epoch 14/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 223s 156ms/step - accuracy: 0.9748 - loss: 0.1129 - val_accuracy: 0.0980 - val_loss: 19.1130
Epoch 15/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 222s 155ms/step - accuracy: 0.9765 - loss: 0.0962 - val_accuracy: 0.0949 - val_loss: 19.2373
Epoch 16/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 222s 156ms/step - accuracy: 0.9755 - loss: 0.1034 - val_accuracy: 0.0950 - val_loss: 20.3971
Epoch 17/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 222s 155ms/step - accuracy: 0.9764 - loss: 0.0966 - val_accuracy: 0.1003 - val_loss: 23.3367
Epoch 18/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 223s 156ms/step - accuracy: 0.9778 - loss: 0.1115 - val_accuracy: 0.0926 - val_loss: 23.0229
Epoch 19/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 223s 156ms/step - accuracy: 0.9793 - loss: 0.0977 - val_accuracy: 0.0940 - val_loss: 22.8667
Epoch 20/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 263s 156ms/step - accuracy: 0.9798 - loss: 0.0848 - val_accuracy: 0.0933 - val_loss: 24.5049
Epoch 21/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 223s 156ms/step - accuracy: 0.9841 - loss: 0.0698 - val_accuracy: 0.0968 - val_loss: 26.0032
Epoch 22/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 261s 155ms/step - accuracy: 0.9752 - loss: 0.1223 - val_accuracy: 0.0928 - val_loss: 24.2805
Epoch 23/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 223s 156ms/step - accuracy: 0.9811 - loss: 0.0977 - val_accuracy: 0.0924 - val_loss: 26.2088
Epoch 24/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 224s 157ms/step - accuracy: 0.9862 - loss: 0.0667 - val_accuracy: 0.0894 - val_loss: 27.4121
Epoch 25/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 222s 155ms/step - accuracy: 0.9849 - loss: 0.0802 - val_accuracy: 0.0954 - val_loss: 29.5772
Epoch 26/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 222s 155ms/step - accuracy: 0.9816 - loss: 0.0848 - val_accuracy: 0.0921 - val_loss: 29.0969
Epoch 27/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 262s 155ms/step - accuracy: 0.9776 - loss: 0.1253 - val_accuracy: 0.0961 - val_loss: 30.6832
Epoch 28/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 264s 157ms/step - accuracy: 0.9835 - loss: 0.0917 - val_accuracy: 0.0893 - val_loss: 28.3088
Epoch 29/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 224s 157ms/step - accuracy: 0.9799 - loss: 0.1104 - val_accuracy: 0.0884 - val_loss: 29.9295
Epoch 30/30
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 268s 161ms/step - accuracy: 0.9873 - loss: 0.0694 - val_accuracy: 0.0921 - val_loss: 30.1474
```

```
val_loss, val_acc = model.evaluate(X_val, y_val)
print(f'Validation Accuracy: {val_acc:.4f}')


plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.legend()
plt.title('Training & Validation Accuracy')
plt.show()
```

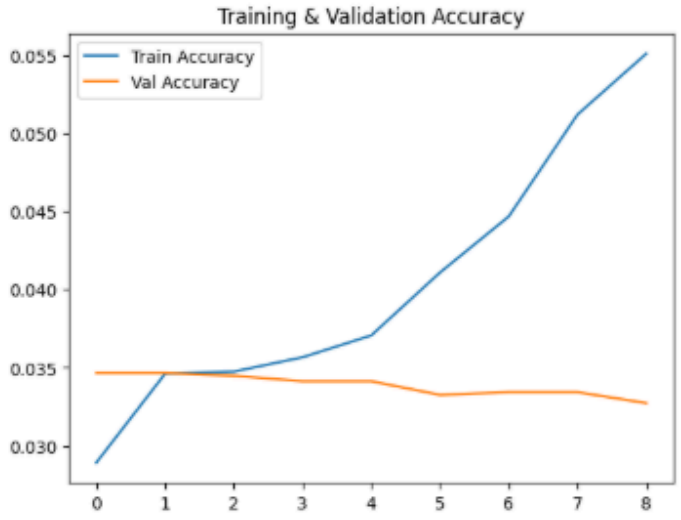
```
179/179 ━━━━━━━━━━━━━━━━━━━━ 12s 65ms/step - accuracy: 0.0885 - loss: 29.9701
Validation Accuracy: 0.0921
```



```python
import os
import numpy as np
import pandas as pd
import librosa
import scipy.signal
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tqdm.notebook import tqdm
from pathlib import Path
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import random


train_audio_path = Path("C:/Users/ricar/Downloads/Dataset Datmod/train_audio")
df = pd.read_csv("C:/Users/ricar/Downloads/Dataset Datmod/train.csv")


le = LabelEncoder()
df = df[df['primary_label'].notnull()]
y = le.fit_transform(df['primary_label'])


def extract_mfcc(file_path, max_len=400, n_mfcc=40, augment=False):
    try:
        y, sr = librosa.load(file_path, sr=32000)


        y = librosa.util.normalize(y)


        y, _ = librosa.effects.trim(y, top_db=20)


        b, a = scipy.signal.butter(6, [300/(sr/2), 8000/(sr/2)], btype='band')
        y = scipy.signal.filtfilt(b, a, y)

        if augment:

            if random.random() < 0.5:
                y = librosa.effects.pitch_shift(y, sr, n_steps=random.uniform(-2, 2))

            if random.random() < 0.5:
                rate = random.uniform(0.8, 1.2)
                y = librosa.effects.time_stretch(y, rate)
                y = librosa.util.fix_length(y, int(len(y)/rate))


        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
        mfcc = mfcc.T

        if mfcc.shape[0] < max_len:
            pad_width = max_len - mfcc.shape[0]
            mfcc = np.pad(mfcc, ((0, pad_width), (0,0)), mode='constant')
        else:
            mfcc = mfcc[:max_len, :]

        return mfcc
    except:
        return np.zeros((max_len, n_mfcc))
```

```python
X = []
for filename in tqdm(df['filename']):
    file_path = train_audio_path / filename
    X.append(extract_mfcc(file_path, augment=True))

X = np.array(X)
y = np.array(y)


X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)


X_train = X_train[..., np.newaxis]
X_val = X_val[..., np.newaxis]


model = models.Sequential()
model.add(layers.Conv2D(16, (3,3), activation='relu', input_shape=(400, 40, 1)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(32, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.3))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(len(le.classes_), activation='softmax'))


model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=50, batch_size=16, validation_data=(X_val, y_val), callbacks=[early_stop])


val_loss, val_acc = model.evaluate(X_val, y_val)
print(f'Validation Accuracy: {val_acc:.4f}')


plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.legend()
plt.title('Training & Validation Accuracy')
plt.show()
```

```
  0%|          | 0/28564 [00:00<?, ?it/s]
C:\Users\ricar\anaconda3\envs\tf_env\lib\site-packages\keras\src\layers\convolutional\base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/50
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 85s 57ms/step - accuracy: 0.0251 - loss: 5.5888 - val_accuracy: 0.0347 - val_loss: 4.8135
Epoch 2/50
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 80s 56ms/step - accuracy: 0.0321 - loss: 4.8002 - val_accuracy: 0.0347 - val_loss: 4.7190
Epoch 3/50
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 77s 54ms/step - accuracy: 0.0358 - loss: 4.7439 - val_accuracy: 0.0345 - val_loss: 4.7237
Epoch 4/50
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 75s 53ms/step - accuracy: 0.0346 - loss: 4.7293 - val_accuracy: 0.0341 - val_loss: 4.7180
Epoch 5/50
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 87s 56ms/step - accuracy: 0.0364 - loss: 4.7091 - val_accuracy: 0.0341 - val_loss: 4.7314
Epoch 6/50
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 82s 57ms/step - accuracy: 0.0440 - loss: 4.6696 - val_accuracy: 0.0333 - val_loss: 4.7453
Epoch 7/50
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 82s 57ms/step - accuracy: 0.0482 - loss: 4.6544 - val_accuracy: 0.0334 - val_loss: 4.7809
Epoch 8/50
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 82s 57ms/step - accuracy: 0.0551 - loss: 4.6132 - val_accuracy: 0.0334 - val_loss: 4.8566
Epoch 9/50
1429/1429 ━━━━━━━━━━━━━━━━━━━━ 82s 57ms/step - accuracy: 0.0544 - loss: 4.6051 - val_accuracy: 0.0327 - val_loss: 4.9096
179/179 ━━━━━━━━━━━━━━━━━━━━ 5s 27ms/step - accuracy: 0.0363 - loss: 4.6963
Validation Accuracy: 0.0341
```



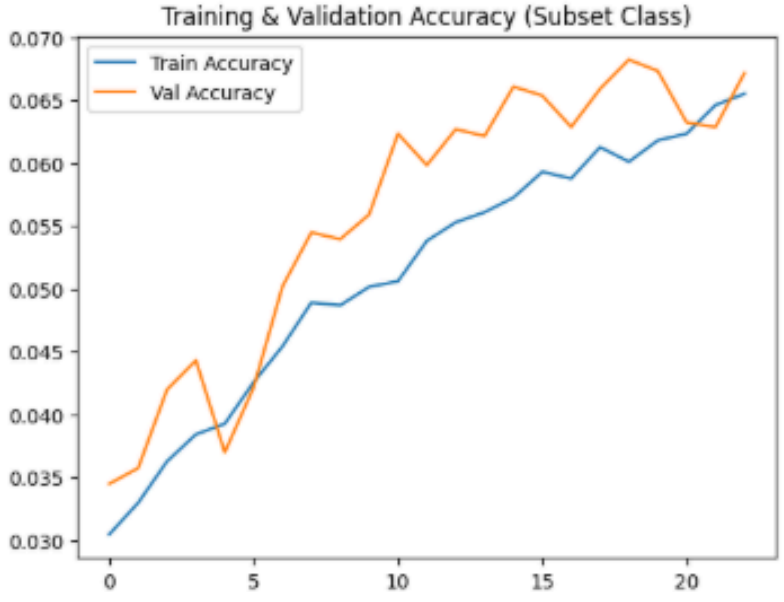

```
df['primary_label'].value_counts()
```

```
primary_label
grekis     990
compau     808
trokin     787
roahaw     709
banana     610
          ...
42113        2
21116        2
1139490      2
21038        2
64862        2
Name: count, Length: 206, dtype: int64
```

```python
import os
import numpy as np
import pandas as pd
import librosa
import scipy.signal
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tqdm.notebook import tqdm
from pathlib import Path
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import random


train_audio_path = Path("C:/Users/ricar/Downloads/Dataset Datmod/train_audio")
df = pd.read_csv("C:/Users/ricar/Downloads/Dataset Datmod/train.csv")


df_counts = df['primary_label'].value_counts()
valid_classes = df_counts[df_counts >= 30].index.tolist()
df_filtered = df[df['primary_label'].isin(valid_classes)]


le = LabelEncoder()
df_filtered = df_filtered[df_filtered['primary_label'].notnull()]
y = le.fit_transform(df_filtered['primary_label'])


def extract_mfcc(file_path, max_len=400, n_mfcc=40, augment=False):
    try:
        y, sr = librosa.load(file_path, sr=32000)
        y = librosa.util.normalize(y)
        y, _ = librosa.effects.trim(y, top_db=20)
        b, a = scipy.signal.butter(6, [300/(sr/2), 8000/(sr/2)], btype='band')
        y = scipy.signal.filtfilt(b, a, y)
        if augment:
            if random.random() < 0.5:
                y = librosa.effects.pitch_shift(y, sr, n_steps=random.uniform(-2, 2))
            if random.random() < 0.5:
                rate = random.uniform(0.8, 1.2)
                y = librosa.effects.time_stretch(y, rate)
                y = librosa.util.fix_length(y, int(len(y)/rate))
        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
        mfcc = mfcc.T
        if mfcc.shape[0] < max_len:
            pad_width = max_len - mfcc.shape[0]
            mfcc = np.pad(mfcc, ((0, pad_width), (0,0)), mode='constant')
        else:
            mfcc = mfcc[:max_len, :]
        return mfcc
    except:
        return np.zeros((max_len, n_mfcc))
```

```
X = []
for filename in tqdm(df_filtered['filename']):
    file_path = train_audio_path / filename
    X.append(extract_mfcc(file_path, augment=True))

X = np.array(X)
y = np.array(y)


X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)


X_train = X_train[..., np.newaxis]
X_val = X_val[..., np.newaxis]


model = models.Sequential()
model.add(layers.Input(shape=(400, 40, 1)))
model.add(layers.Conv2D(16, (3,3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(32, (3,3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.3))
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(len(le.classes_), activation='softmax'))


model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=50, batch_size=16, validation_data=(X_val, y_val), callbacks=[early_stop])


val_loss, val_acc = model.evaluate(X_val, y_val)
print(f'Validation Accuracy: {val_acc:.4f}')


plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.legend()
plt.title('Training & Validation Accuracy (Subset Class)')
plt.show()
```

```
  0%|          | 0/27988 [00:00<?, ?it/s]
Epoch 1/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 136s 94ms/step - accuracy: 0.0294 - loss: 4.7651 - val_accuracy: 0.0345 - val_loss: 4.6894
Epoch 2/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 131s 94ms/step - accuracy: 0.0327 - loss: 4.6548 - val_accuracy: 0.0357 - val_loss: 4.6430
Epoch 3/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 144s 95ms/step - accuracy: 0.0341 - loss: 4.6459 - val_accuracy: 0.0420 - val_loss: 4.6251
Epoch 4/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 142s 94ms/step - accuracy: 0.0376 - loss: 4.6211 - val_accuracy: 0.0443 - val_loss: 4.5996
Epoch 5/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 130s 93ms/step - accuracy: 0.0388 - loss: 4.6082 - val_accuracy: 0.0370 - val_loss: 4.6099
Epoch 6/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 144s 94ms/step - accuracy: 0.0419 - loss: 4.5906 - val_accuracy: 0.0422 - val_loss: 4.5991
Epoch 7/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 133s 95ms/step - accuracy: 0.0423 - loss: 4.5825 - val_accuracy: 0.0502 - val_loss: 4.5757
Epoch 8/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 143s 96ms/step - accuracy: 0.0490 - loss: 4.5520 - val_accuracy: 0.0545 - val_loss: 4.5394
Epoch 9/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 134s 95ms/step - accuracy: 0.0496 - loss: 4.5191 - val_accuracy: 0.0539 - val_loss: 4.5205
Epoch 10/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 144s 96ms/step - accuracy: 0.0497 - loss: 4.5164 - val_accuracy: 0.0559 - val_loss: 4.5327
Epoch 11/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 132s 95ms/step - accuracy: 0.0521 - loss: 4.5065 - val_accuracy: 0.0623 - val_loss: 4.5212
Epoch 12/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 131s 94ms/step - accuracy: 0.0523 - loss: 4.4966 - val_accuracy: 0.0598 - val_loss: 4.5063
Epoch 13/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 145s 96ms/step - accuracy: 0.0564 - loss: 4.4771 - val_accuracy: 0.0627 - val_loss: 4.4908
Epoch 14/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 144s 97ms/step - accuracy: 0.0559 - loss: 4.4664 - val_accuracy: 0.0622 - val_loss: 4.4925
Epoch 15/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 140s 95ms/step - accuracy: 0.0586 - loss: 4.4602 - val_accuracy: 0.0661 - val_loss: 4.4861
Epoch 16/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 133s 95ms/step - accuracy: 0.0600 - loss: 4.4542 - val_accuracy: 0.0654 - val_loss: 4.4723
Epoch 17/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 143s 96ms/step - accuracy: 0.0598 - loss: 4.4534 - val_accuracy: 0.0629 - val_loss: 4.4750
Epoch 18/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 133s 95ms/step - accuracy: 0.0610 - loss: 4.4612 - val_accuracy: 0.0659 - val_loss: 4.4663
Epoch 19/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 141s 95ms/step - accuracy: 0.0590 - loss: 4.4315 - val_accuracy: 0.0682 - val_loss: 4.4696
Epoch 20/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 136s 97ms/step - accuracy: 0.0626 - loss: 4.4421 - val_accuracy: 0.0673 - val_loss: 4.5034
Epoch 21/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 134s 96ms/step - accuracy: 0.0621 - loss: 4.4175 - val_accuracy: 0.0632 - val_loss: 4.5291
Epoch 22/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 134s 96ms/step - accuracy: 0.0645 - loss: 4.4137 - val_accuracy: 0.0629 - val_loss: 4.5163
Epoch 23/50
1400/1400 ━━━━━━━━━━━━━━━━━━━━ 133s 95ms/step - accuracy: 0.0603 - loss: 4.4228 - val_accuracy: 0.0672 - val_loss: 4.4853
175/175 ━━━━━━━━━━━━━━━━━━━━ 6s 33ms/step - accuracy: 0.0650 - loss: 4.4662
Validation Accuracy: 0.0659
```



```
model.save("trained_cnn_balanced.h5")

import pickle

with open("label_encoder.pkl", "wb") as f:
    pickle.dump(le, f)
```

```
## Pseudo Label
```

```
import os
import numpy as np
import pandas as pd
import librosa
import scipy.signal
from tqdm.notebook import tqdm
from pathlib import Path
from tensorflow.keras.models import load_model


soundscape_path = Path("C:/Users/ricar/Downloads/Dataset Datmod/train_soundscapes")
soundscape_files = sorted([f for f in soundscape_path.glob("*.ogg")])


model = load_model("trained_cnn_balanced.h5")


import pickle
with open("label_encoder.pkl", "rb") as f:
    le = pickle.load(f)


def extract_mfcc(file_path, max_len=400, n_mfcc=40):
    try:
        y, sr = librosa.load(file_path, sr=32000)
        y = librosa.util.normalize(y)
        y, _ = librosa.effects.trim(y, top_db=20)
        b, a = scipy.signal.butter(6, [300/(sr/2), 8000/(sr/2)], btype='band')
        y = scipy.signal.filtfilt(b, a, y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
        mfcc = mfcc.T
        if mfcc.shape[0] < max_len:
            pad_width = max_len - mfcc.shape[0]
            mfcc = np.pad(mfcc, ((0, pad_width), (0,0)), mode='constant')
        else:
            mfcc = mfcc[:max_len, :]
        return mfcc
    except:
        return np.zeros((max_len, n_mfcc))


predictions = []
for file_path in tqdm(soundscape_files):
    mfcc = extract_mfcc(file_path)
    mfcc = mfcc[np.newaxis, ..., np.newaxis]
    prob = model.predict(mfcc)[0]
    pred_class = le.inverse_transform([np.argmax(prob)])[0]
    predictions.append({"file": file_path.name, "predicted_label": pred_class, "confidence": np.max(prob)})


pseudo_df = pd.DataFrame(predictions)
pseudo_df.to_csv("pseudo_labels_soundscapes.csv", index=False)


pseudo_df.head()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
```

[14]:

	file	predicted_label	confidence
0	H02_20230420_074000.ogg	banana	0.058236
1	H02_20230420_112000.ogg	yeafly1	0.043585
2	H02_20230420_154500.ogg	yeafly1	0.103635
3	H02_20230420_164000.ogg	yeafly1	0.083115
4	H02_20230420_223500.ogg	compau	0.115261

```
pseudo_df
```

	file	predicted_label	confidence
0	H02_20230420_074000.ogg	banana	0.058236
1	H02_20230420_112000.ogg	yeafly1	0.043585
2	H02_20230420_154500.ogg	yeafly1	0.103635
3	H02_20230420_164000.ogg	yeafly1	0.083115
4	H02_20230420_223500.ogg	compau	0.115261
...	...	...	...
9721	O203_20230524_202500.ogg	compot1	0.531562
9722	O203_20230525_023500.ogg	compot1	0.134023
9723	O203_20230525_081500.ogg	yeafly1	0.107704
9724	O203_20230525_234000.ogg	compot1	0.510114
9725	O203_20230526_023000.ogg	compot1	0.223507

9726 rows × 3 columns

```
pseudo_df_filtered_30 = pseudo_df[pseudo_df['confidence'] > 0.3]
pseudo_df_filtered_30
```

	file	predicted_label	confidence
36	H02_20230425_185500.ogg	compau	0.323190
37	H02_20230426_004500.ogg	compau	0.476329
46	H02_20230427_040500.ogg	compau	0.314420
49	H02_20230428_034500.ogg	compau	0.320178
68	H02_20230502_045500.ogg	compot1	0.402418
...	...	...	...
9665	O203_20230515_204500.ogg	compot1	0.484580
9688	O203_20230518_220000.ogg	compot1	0.333890
9715	O203_20230523_213000.ogg	compot1	0.562364
9721	O203_20230524_202500.ogg	compot1	0.531562
9724	O203_20230525_234000.ogg	compot1	0.510114

1070 rows × 3 columns

```
## Merge Train Audio + Pseudo-Labels
```

```
train_df = pd.read_csv("C:/Users/ricar/Downloads/Dataset Datmod/train.csv")
train_df = train_df[['filename', 'primary_label']]
train_df.rename(columns={'filename': 'file', 'primary_label': 'label'}, inplace=True)


pseudo_df = pd.read_csv("pseudo_labels_soundscapes.csv")


threshold = 0.3
pseudo_df_filtered = pseudo_df[pseudo_df['confidence'] > threshold]


full_df = pd.concat([train_df, pseudo_df_filtered[['file', 'predicted_label']]], axis=0).reset_index(drop=True)


print(full_df['predicted_label'].value_counts())


full_df.to_csv("merged_train_pseudo.csv", index=False)


full_df.head()
```

```
predicted_label
compot1    796
compau     255
whtdov       9
strcuc1      4
yeofly1      2
trokin       1
laufal1      1
littin1      1
speowl1      1
Name: count, dtype: int64
```

	file	label	predicted_label
0	1139490/CSA36385.ogg	1139490	NaN
1	1139490/CSA36389.ogg	1139490	NaN
2	1192948/CSA36358.ogg	1192948	NaN
3	1192948/CSA36366.ogg	1192948	NaN
4	1192948/CSA36373.ogg	1192948	NaN

```
full_df
```

	file	label	predicted_label
0	1139490/CSA36385.ogg	1139490	NaN
1	1139490/CSA36389.ogg	1139490	NaN
2	1192948/CSA36358.ogg	1192948	NaN
3	1192948/CSA36366.ogg	1192948	NaN
4	1192948/CSA36373.ogg	1192948	NaN
...	...	...	...
29629	0203_20230515_204500.ogg	NaN	compot1
29630	0203_20230518_220000.ogg	NaN	compot1
29631	0203_20230523_213000.ogg	NaN	compot1
29632	0203_20230524_202500.ogg	NaN	compot1
29633	0203_20230525_234000.ogg	NaN	compot1

29634 rows × 3 columns

Train Merged

```
import os
import numpy as np
import pandas as pd
import librosa
import scipy.signal
from tqdm.notebook import tqdm
from pathlib import Path
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.preprocessing import LabelEncoder
import pickle


merged_df = pd.read_csv("merged_train_pseudo.csv")


le = LabelEncoder()
le.fit(merged_df['label'])

with open("final_label_encoder.pkl", "wb") as f:
    pickle.dump(le, f)

y = le.transform(merged_df['label'])


def extract_mfcc(file_path, max_len=400, n_mfcc=40):
    try:
        y, sr = librosa.load(file_path, sr=32000)
        y = librosa.util.normalize(y)
        y, _ = librosa.effects.trim(y, top_db=20)
        b, a = scipy.signal.butter(6, [300/(sr/2), 8000/(sr/2)], btype='band')
        y = scipy.signal.filtfilt(b, a, y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
        mfcc = mfcc.T
        if mfcc.shape[0] < max_len:
            pad_width = max_len - mfcc.shape[0]
            mfcc = np.pad(mfcc, ((0, pad_width), (0,0)), mode='constant')
        else:
            mfcc = mfcc[:max_len, :]
        return mfcc
    except:
        return np.zeros((max_len, n_mfcc))


train_audio_path = Path("C:/Users/ricar/Downloads/Dataset Datmod/train_audio")
soundscape_path = Path("C:/Users/ricar/Downloads/Dataset Datmod/train_soundscapes")

X = []
for filename in tqdm(merged_df['file']):
    file_path = train_audio_path / filename if (train_audio_path / filename).exists() else soundscape_path / filename
    X.append(extract_mfcc(file_path))

X = np.array(X)
y = np.array(y)


from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
X_train = X_train[..., np.newaxis]
X_val = X_val[..., np.newaxis]


model = models.Sequential()
model.add(layers.Input(shape=(400, 40, 1)))
model.add(layers.Conv2D(32, (3,3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.3))
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(len(le.classes_), activation='softmax'))
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
model.fit(X_train, y_train, epochs=50, batch_size=16, validation_data=(X_val, y_val), callbacks=[early_stop])


model.save("final_cnn_model.keras")


submission_df = pd.read_csv("C:/Users/ricar/Downloads/Dataset Datmod/sample_submission.csv")
test_soundscape_path = Path("C:/Users/ricar/Downloads/Dataset Datmod/test_soundscapes")

from tensorflow.keras.models import load_model
model = load_model("final_cnn_model.keras")

with open("final_label_encoder.pkl", "rb") as f:
    le = pickle.load(f)


species_cols = submission_df.columns[1:]


pred_rows = []
for file_path in tqdm(sorted(test_soundscape_path.glob("*.ogg"))):
    y, sr = librosa.load(file_path, sr=32000)
    for i in range(0, len(y), sr*5):
        chunk = y[i:i+sr*5]
        if len(chunk) < sr*5:
            pad_width = sr*5 - len(chunk)
            chunk = np.pad(chunk, (0, pad_width))


        if len(chunk) >= 2048:
            mfcc = librosa.feature.mfcc(y=chunk, sr=sr, n_mfcc=40)
        else:
            mfcc = librosa.feature.mfcc(y=chunk, sr=sr, n_mfcc=40, n_fft=512)
        mfcc = mfcc.T

        if mfcc.shape[0] < 400:
            mfcc = np.pad(mfcc, ((0, 400-mfcc.shape[0]), (0,0)), mode='constant')
        else:
            mfcc = mfcc[:400, :]
        mfcc = mfcc[np.newaxis, ..., np.newaxis]
        prob = model.predict(mfcc)[0]
        row_id = f"{file_path.stem}_{i//sr+5}"
        prob_full = pd.Series(0, index=species_cols)
        for sp, p in zip(le.classes_, prob):
            if sp in prob_full.index:
                prob_full[sp] = p
        pred_rows.append([row_id] + prob_full.tolist())



submission_result = pd.DataFrame(pred_rows, columns=["row_id"] + list(species_cols))
submission_result.to_csv("final_submission.csv", index=False)


submission_result.head()
```

```
  0%|          | 0/29634 [00:00<?, ?it/s]
C:\Users\ricar\anaconda3\envs\tf_env\lib\site-packages\librosa\core\spectrum.py:266: UserWarning: n_fft=2048 is too large for input signal of length
1236
  warnings.warn(
C:\Users\ricar\anaconda3\envs\tf_env\lib\site-packages\librosa\core\spectrum.py:266: UserWarning: n_fft=2048 is too large for input signal of length
648
  warnings.warn(
Epoch 1/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 344s 229ms/step - accuracy: 0.0770 - loss: 4.6102 - val_accuracy: 0.1426 - val_loss: 4.0361
Epoch 2/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 382s 229ms/step - accuracy: 0.1465 - loss: 3.9824 - val_accuracy: 0.1893 - val_loss: 3.7373
Epoch 3/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 339s 229ms/step - accuracy: 0.1940 - loss: 3.6750 - val_accuracy: 0.2254 - val_loss: 3.5256
Epoch 4/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 383s 229ms/step - accuracy: 0.2264 - loss: 3.4693 - val_accuracy: 0.2652 - val_loss: 3.3364
Epoch 5/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 382s 229ms/step - accuracy: 0.2543 - loss: 3.3079 - val_accuracy: 0.2851 - val_loss: 3.2308
Epoch 6/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 340s 230ms/step - accuracy: 0.2782 - loss: 3.1683 - val_accuracy: 0.3059 - val_loss: 3.0820
Epoch 7/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 380s 228ms/step - accuracy: 0.3030 - loss: 3.0553 - val_accuracy: 0.3282 - val_loss: 3.0304
Epoch 8/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 338s 228ms/step - accuracy: 0.3139 - loss: 2.9808 - val_accuracy: 0.3261 - val_loss: 3.0212
Epoch 9/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 339s 229ms/step - accuracy: 0.3349 - loss: 2.9053 - val_accuracy: 0.3315 - val_loss: 3.0129
Epoch 10/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 341s 230ms/step - accuracy: 0.3303 - loss: 2.8760 - val_accuracy: 0.3565 - val_loss: 2.8464
Epoch 11/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 338s 228ms/step - accuracy: 0.3308 - loss: 2.8545 - val_accuracy: 0.3658 - val_loss: 2.8206
Epoch 12/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 381s 227ms/step - accuracy: 0.3512 - loss: 2.7921 - val_accuracy: 0.3820 - val_loss: 2.7698
Epoch 13/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 386s 230ms/step - accuracy: 0.3546 - loss: 2.7565 - val_accuracy: 0.3894 - val_loss: 2.7262
Epoch 14/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 380s 228ms/step - accuracy: 0.3521 - loss: 2.7421 - val_accuracy: 0.3810 - val_loss: 2.7914
Epoch 15/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 337s 228ms/step - accuracy: 0.3627 - loss: 2.7014 - val_accuracy: 0.3737 - val_loss: 2.8072
Epoch 16/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 385s 229ms/step - accuracy: 0.3672 - loss: 2.6614 - val_accuracy: 0.4002 - val_loss: 2.7058
Epoch 17/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 382s 229ms/step - accuracy: 0.3776 - loss: 2.6362 - val_accuracy: 0.3992 - val_loss: 2.7160
Epoch 18/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 339s 229ms/step - accuracy: 0.3763 - loss: 2.6438 - val_accuracy: 0.3968 - val_loss: 2.6857
Epoch 19/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 384s 230ms/step - accuracy: 0.3780 - loss: 2.6104 - val_accuracy: 0.3968 - val_loss: 2.7273
Epoch 20/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 340s 229ms/step - accuracy: 0.3929 - loss: 2.5573 - val_accuracy: 0.4196 - val_loss: 2.6233
Epoch 21/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 345s 232ms/step - accuracy: 0.3869 - loss: 2.5645 - val_accuracy: 0.3926 - val_loss: 2.7354
Epoch 22/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 340s 229ms/step - accuracy: 0.3968 - loss: 2.5554 - val_accuracy: 0.4081 - val_loss: 2.6325
Epoch 23/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 383s 230ms/step - accuracy: 0.3938 - loss: 2.5444 - val_accuracy: 0.4152 - val_loss: 2.6254
Epoch 24/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 341s 230ms/step - accuracy: 0.3965 - loss: 2.5107 - val_accuracy: 0.4237 - val_loss: 2.5931
Epoch 25/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 382s 230ms/step - accuracy: 0.4000 - loss: 2.5025 - val_accuracy: 0.4233 - val_loss: 2.5590
Epoch 26/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 382s 230ms/step - accuracy: 0.4094 - loss: 2.4928 - val_accuracy: 0.4346 - val_loss: 2.5320
Epoch 27/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 387s 233ms/step - accuracy: 0.4093 - loss: 2.4807 - val_accuracy: 0.4346 - val_loss: 2.5648
Epoch 28/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 378s 229ms/step - accuracy: 0.4129 - loss: 2.4666 - val_accuracy: 0.4437 - val_loss: 2.5314
Epoch 29/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 386s 232ms/step - accuracy: 0.4085 - loss: 2.4587 - val_accuracy: 0.4368 - val_loss: 2.5448
Epoch 30/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 380s 230ms/step - accuracy: 0.4103 - loss: 2.4480 - val_accuracy: 0.4296 - val_loss: 2.6103
Epoch 31/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 381s 230ms/step - accuracy: 0.4182 - loss: 2.4498 - val_accuracy: 0.4260 - val_loss: 2.6411
Epoch 32/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 383s 230ms/step - accuracy: 0.4159 - loss: 2.4294 - val_accuracy: 0.4399 - val_loss: 2.5392
Epoch 33/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 340s 229ms/step - accuracy: 0.4092 - loss: 2.4583 - val_accuracy: 0.4503 - val_loss: 2.4801
Epoch 34/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 342s 230ms/step - accuracy: 0.4251 - loss: 2.4092 - val_accuracy: 0.4461 - val_loss: 2.5114
Epoch 35/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 382s 230ms/step - accuracy: 0.4179 - loss: 2.4251 - val_accuracy: 0.4513 - val_loss: 2.4919
Epoch 36/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 345s 233ms/step - accuracy: 0.4233 - loss: 2.3905 - val_accuracy: 0.4427 - val_loss: 2.5242
Epoch 37/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 340s 229ms/step - accuracy: 0.4278 - loss: 2.3746 - val_accuracy: 0.4593 - val_loss: 2.4901
Epoch 38/50
1482/1482 ━━━━━━━━━━━━━━━━━━━━ 340s 230ms/step - accuracy: 0.4245 - loss: 2.3801 - val_accuracy: 0.4441 - val_loss: 2.5309
0it [00:00, ?it/s]
```

[2]:

row_id	1139490	1192948	1194042	126247	1346504	134933	135045	1462711	1462737	...	yebfly1	yebsee1	yecspi2	yectyr1	yehbla2	yehcar1	yelori1

0 rows × 207 columns

3.4 Inovasi dalam Project Ini

A. Inovasi Kontekstual dan Local

Proyek ini tidak hanya menerapkan teknik klasifikasi audio yang umum, tetapi juga menawarkan inovasi yang sangat kontekstual, yaitu penerapan CNN (Convolutional Neural Network) untuk mendeteksi spesies berdasarkan suaranya di wilayah tropis Kolombia. Wilayah ini, khususnya Lembah Magdalena Tengah, merupakan salah satu pusat keanekaragaman hayati di dunia, namun masih sangat sedikit penelitian klasifikasi akustik berbasis pembelajaran mesin.

Sementara sebagian besar penelitian serupa berfokus pada wilayah subtropis atau beriklim sedang di Amerika Utara dan Eropa, proyek ini membawa pendekatan teknologi ke lingkungan tropis yang jauh lebih kompleks secara akustik. Hal ini termasuk tantangan kebisingan latar belakang alami seperti angin, hujan, serangga, dan tumpang tindihnya suara dari beberapa spesies pada satu waktu. Keberhasilan model dalam memproses data dari lingkungan ini menunjukkan bahwa arsitektur CNN fleksibel dan adaptif terhadap tantangan dunia nyata.

B. Pemanfaatan Data Tidak Berlabel (Semi-Supervised)

Inovasi lain yang diangkat dalam proyek ini adalah penggunaan lanskap suara yang tidak berlabel sebagai bagian dari strategi pembelajaran semi-pengawasan. Dalam banyak kasus konservasi, ketersediaan data berlabel sangat terbatas, terutama untuk spesies langka. Oleh karena itu, pendekatan ini menjadi sangat penting. Dengan menyertakan rekaman yang tidak berlabel dari lokasi yang sama, proyek ini membuka kemungkinan untuk menerapkan pelabelan semu, pelatihan mandiri, atau teknik pembelajaran representasi, yang memungkinkan model untuk memanfaatkan struktur alami data tanpa perlu anotasi manual secara penuh. Pendekatan ini menandai langkah pertama menuju solusi pemantauan keanekaragaman hayati berbasis AI yang lebih efisien, berbiaya rendah, dan terukur.

C. Respons terhadap Kebutuhan Konservasi Lapangan

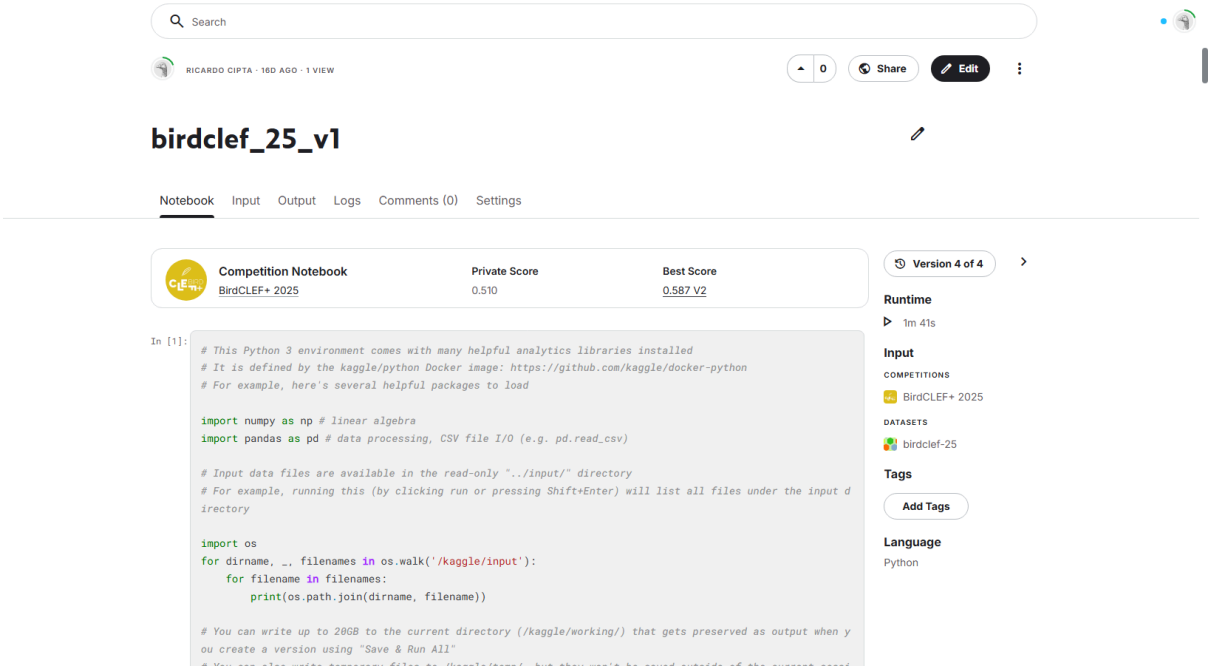
Penggunaan CNN untuk klasifikasi suara tidak hanya merupakan implementasi dari teknologi mutakhir, tetapi juga menjawab kebutuhan konservasi yang konkret di lapangan. Banyak daerah tropis seperti Cagar Alam El Silencio di Kolombia memiliki banyak spesies yang belum tercatat secara sistematis. Metode pengamatan langsung membutuhkan biaya dan waktu yang lama, sehingga sistem deteksi otomatis berbasis suara menjadi solusi yang sangat dibutuhkan oleh para peneliti dan praktisi

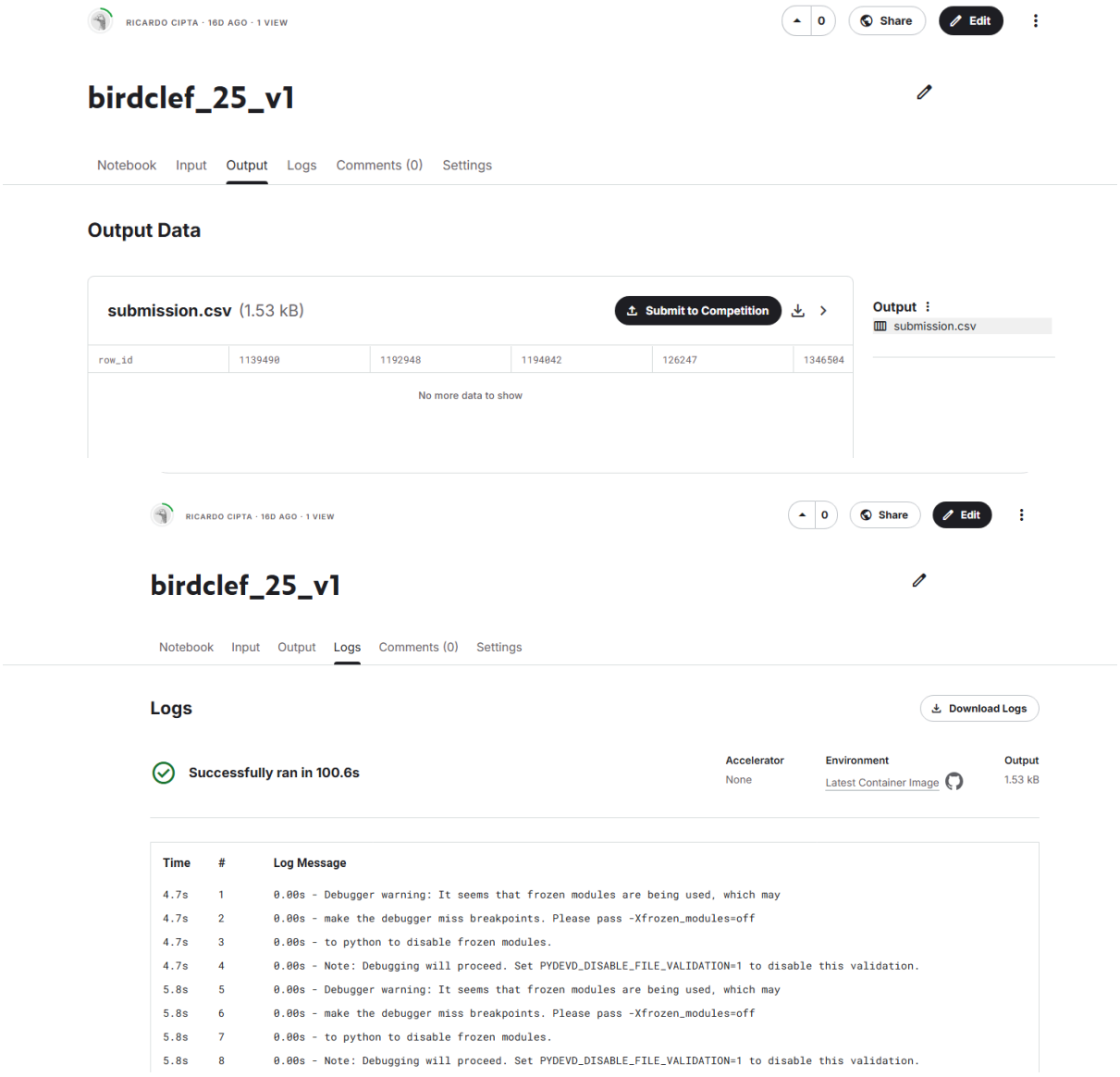
konservasi.Dengan menggunakan pendekatan ini, peneliti lapangan dapat memperoleh informasi yang lebih cepat dan lebih akurat mengenai keberadaan spesies, tren populasi, dan dampak intervensi restorasi ekosistem. Hal ini mendukung pengambilan keputusan berbasis data dan memungkinkan pemantauan keanekaragaman hayati secara real-time.

D. Kontribusi Terhadap Ilmu dan Teknologi Konservasi

Secara keseluruhan, proyek ini bukan hanya tentang keakuratan model atau penerapan CNN, tetapi juga tentang memberikan kontribusi nyata bagi ilmu konservasi dan teknologi konservasi alam. Dengan pendekatan berbasis konteks lokal, kemampuan untuk memanfaatkan data yang tidak berlabel, dan kepekaan terhadap kondisi ekologi tropis yang kompleks, proyek ini menunjukkan bahwa teknologi dapat digunakan secara strategis untuk mengatasi masalah global-dengan solusi yang relevan secara lokal.

3.5 Lampiran Screenshoot Bukti Submission Kompetisi





BAB IV

KESIMPULAN DAN SARAN

4.1 Kesimpulan

Penelitian ini bertujuan untuk mengembangkan model klasifikasi berbasis Convolutional Neural Network (CNN) untuk mengidentifikasi spesies hewan dari data audio yang direkam di Cagar Alam El Silencio, Lembah Magdalena Tengah, Kolombia. Berdasarkan tahapan metodologi CRISP-DM - mulai dari pemahaman masalah, eksplorasi dan persiapan data, pemodelan, evaluasi, hingga implementasi awal - dapat disimpulkan bahwa pendekatan ini memiliki potensi yang besar dalam mendukung kegiatan konservasi dan pemantauan keanekaragaman hayati di daerah tropis.

Model CNN berhasil dilatih menggunakan fitur MFCC yang diekstrak dari rekaman audio, dan mencapai akurasi public score sebesar 0.581. Meskipun akurasi ini masih rendah, proyek ini tetap menunjukkan kontribusi yang signifikan terutama dalam hal penerapan teknologi deep learning dalam konteks bioakustik tropis, serta eksplorasi awal strategi pembelajaran semi-supervised learning dengan menggunakan lanskap suara yang tidak berlabel. Selain itu, proyek ini juga menunjukkan pentingnya pemrosesan data audio yang benar dan representatif sebagai kunci keberhasilan klasifikasi akustik berbasis machine learning. Selain itu, penelitian ini memberikan pendekatan inovatif dalam memanfaatkan teknologi AI untuk mengatasi tantangan nyata di bidang konservasi - terutama di daerah yang memiliki keanekaragaman spesies yang tinggi tetapi tidak memiliki pemantauan otomatis. Dengan dukungan komunitas ilmiah, proyek ini membuka peluang kolaborasi antara teknologi dan ekologi, serta mempercepat proses pemantauan populasi satwa liar secara lebih efisien dan berkelanjutan.

4.2 Saran

Untuk meningkatkan kualitas hasil dan memperluas cakupan manfaat dari proyek ini, ada beberapa saran yang dapat digunakan sebagai referensi untuk pengembangan lebih lanjut:

A. **Perbaikan dan Validasi Ekstraksi Fitur Audio:**
   Penting untuk memastikan bahwa proses ekstraksi fitur (MFCC atau mel spektogram) dilakukan secara konsisten, termasuk segmentasi, normalisasi, pemangkasan, dan penyaringan. Kualitas fitur input sangat menentukan kinerja akhir model.
B. **Evaluasi Menggunakan Metrik Tambahan:**
   Selain akurasi, perlu dilakukan evaluasi dengan menggunakan metrik seperti precision, recall, F1-score, dan confusion matrix untuk mendapatkan gambaran yang lebih komprehensif mengenai performa model, terutama dalam menghadapi data yang tidak seimbang.
C. **Penggunaan Arsitektur Model yang Lebih Kompleks:**
   Direkomendasikan untuk mencoba arsitektur CNN yang lebih dalam atau menggunakan model yang sudah terlatih seperti VGGish atau YAMNet yang telah dioptimalkan untuk data audio. Transfer learning dapat membantu mengatasi keterbatasan data pelatihan.
D. **Penerapan Pembelajaran Semi Terawasi:**
   Potensi data yang tidak berlabel sangat besar, terutama dalam kondisi data spesies langka yang terbatas. Teknik seperti pelabelan semu dan pelatihan mandiri dapat mulai diterapkan untuk meningkatkan generalisasi model.
E. **Augmentasi dan Regularisasi Data:**
   Penambahan teknik augmentasi data seperti pergeseran pitch, peregangan waktu, dan injeksi noise dapat memperkaya variasi data dan meningkatkan kemampuan generalisasi model ke kondisi lapangan yang sebenarnya.

Dengan pengembangan lebih lanjut, proyek ini tidak hanya berpotensi sebagai alat bantu ilmiah, tetapi juga sebagai solusi teknologi nyata dalam upaya pelestarian lingkungan dan keanekaragaman hayati di daerah tropis.

4.3 Peran Anggota

A. Ricardo Cipta (00000098947) : Membuat Kodingan Project
B. Bonardo Gregorius B.S (00000097365): Membuat Kodingan Project
C. Jonathan Chandra - (00000094067): Membuat Laporan
D. Samuel Christophorus.W (00000100797): Membuat Laporan
E. Gabriel Yohanes.A (00000082326) - Mengedit Video

References

[1] A, R., & N, S. (2025). Sound-based bird classification using multiple features and machine learning paradigms. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-024-20565-5.

[2] Meng, H., Sethu, V., & Ambikairajah, E. (2023). What is Learnt by the LEArnable Front-end (LEAF)? Adapting Per-Channel Energy Normalisation (PCEN) to Noisy Conditions. , 2898-2902. https://doi.org/10.21437/Interspeech.2023-1617.

[3] Mushtaq, Z., & Su, S. (2020). Environmental sound classification using a regularized deep convolutional neural network with data augmentation. Applied Acoustics. https://doi.org/10.1016/j.apacoust.2020.107389.

[4] Noumida, A, & Rajan, R. (2021). Deep Learning-based Automatic Bird Species Identification from Isolated Recordings. 2021 8th International Conference on Smart Computing and Communications (ICSCC), 252-256. https://doi.org/10.1109/ICSCC51209.2021.9528234.

[5] Koutini, K., Eghbalzadeh, H., & Widmer, G. (2021). Receptive Field Regularization Techniques for Audio Classification and Tagging With Deep Convolutional Neural Networks. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 29, 1987-2000. https://doi.org/10.1109/TASLP.2021.3082307.

[6] Laska, B., Wallace, B., Hudak, A., & Goubran, R. (2023). Room Acoustic Characterization with Smartphone-Based Automated Speech Recognition. 2023 IEEE Sensors Applications Symposium (SAS), 1-5. https://doi.org/10.1109/SAS58821.2023.10254121.

[7] Michaud, F., Sueur, J., Cesne, M., & Haupert, S. (2022). Unsupervised classification to improve the quality of a bird song recording dataset. ArXiv, abs/2302.07560. https://doi.org/10.1016/j.ecoinf.2022.101952.

[8] Brooker, S., Stephens, P., Whittingham, M., & Willis, S. (2020). Automated detection and classification of birdsong: An ensemble approach. Ecological Indicators. https://doi.org/10.1016/j.ecolind.2020.106609.

[9] Silva, B., Mestre, F., Barreiro, S., Alves, P., & Herrera, J. (2022). soundClass: An automatic sound classification tool for biodiversity monitoring using machine learning. Methods in Ecology and Evolution, 13, 2356 - 2362. https://doi.org/10.1111/2041-210X.13964.

[10] Liu, X., Sahidullah, M., & Kinnunen, T. (2021). Learnable MFCCs for Speaker Verification. 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 1-5. https://doi.org/10.1109/ISCAS51556.2021.9401593.