TUGAS AKHIR – SS234862

KLASIFIKASI SPESIES BURUNG BERDASARKAN SUARA MENGGUNAKAN *DEEP LEARNING* DENGAN FITUR MEL-SPEKTOGRAM DAN *MEL FREQUENCY CEPTRAL COEFFICIENTS* (MFCC)

DANISWARA ADITYA PUTRA

NRP 5003211132

Dosen Pembimbing

Dr. Dra. Kartika Fithriasari, M.Si.

NIP 19691212 199303 2 002

Program Studi Sarjana Statistika

Departemen Statistika

Fakultas Sains dan Analitika Data

Institut Teknologi Sepuluh Nopember

Surabaya

2025

TUGAS AKHIR – SS234862

KLASIFIKASI SPESIES BURUNG BERDASARKAN SUARA MENGGUNAKAN *DEEP LEARNING* DENGAN FITUR MEL-SPEKTOGRAM DAN *MEL FREQUENCY CEPTRAL COEFFICIENTS* (MFCC)

DANISWARA ADITYA PUTRA

NRP 5003211132

Dosen Pembimbing

Dr. Dra. Kartika Fithriasari, M.Si.

NIP 19691212 199303 2 002

Program Studi Sarjana Statistika

Departemen Statistika

Fakultas Sains dan Analitika Data

Institut Teknologi Sepuluh Nopember

Surabaya

2025

FINAL PROJECT – SS234862

BIRD SPECIES CLASSIFICATION BASED ON SOUND USING DEEP LEARNING WITH MEL-SPECTROGRAM AND MEL FREQUENCY CEPSTRAL COEFFICIENTS (MFCC)

DANISWARA ADITYA PUTRA

NRP 5003211132

Advisor

Dr. Dra. Kartika Fithriasari, M.Si.

NIP 19691212 199303 2 002

Bachelor Program of Statistics

Department of Statistics

Faculty of Science and Data Analytics

Institut Teknologi Sepuluh Nopember

Surabaya

2025

LEMBAR PENGESAHAN

KLASIFIKASI SPESIES BURUNG BERDASARKAN SUARA MENGGUNAKAN *DEEP LEARNING* DENGAN FITUR MEL-SPEKTOGRAM DAN *MEL FREQUENCY CEPTRAL COEFFICIENTS (MFCC)*

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Statistika pada
Program Studi Sarjana Statistika
Departemen Statistika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember

Oleh : **DANISWARA ADITYA PUTRA**
NRP 5003211132

Tanggal Ujian: 03 Juli 2025
Periode Wisuda: September 2025

Disetujui oleh:
Pembimbing:

1. Dr. Dra. Kartika Fithriasari, M.Si. ……..
   NIP 19691212 199303 2 002

Penguji:

2. Adatul Mukarromah, S.Si., M.Si. ……..
   NIP 19800418 200312 2 001
3. Tintrim Dwi Ary Widhianingsih, S.Si., M.Stat., Ph.D. ……..
   NIP 19950520 202406 2 003

Kepala Departemen Statistika
Fakultas Sains dan Analitika Data

Dr.rer.pol. Dady Dwi Prastyo, S.Si., M.Si.
NIP 19831204 200812 1 002

(Halaman sengaja dikosongkan)

APPROVAL SHEET

BIRD SPECIES CLASSIFICATION BASED ON SOUND USING DEEP LEARNING WITH MEL-SPECTROGRAM AND MEL FREQUENCY CEPSTRAL COEFFICIENTS (MFCC)

FINAL PROJECT

Submitted to fullfill one of the requirements
For obtaining a degree in Bachelor of Statistics at
Bachelor Program of Statistics
Department of Statistics
Faculty of Science and Data Analytics
Institut Teknologi Sepuluh Nopember

by : **DANISWARA ADITYA PUTRA**
NRP 5003211132

Exam Date: 03 July 2025
Graduation Period: September 2025

Approved by:
Advisor:

1. Dr. Dra. Kartika Fithriasari, M.Si. .....
   NIP 19691212 199303 2 002

Examiners:

2. Adatul Mukarromah, S.Si., M.Si. .....
   NIP 19800418 200312 2 001
3. Tintrim Dwi Ary Widhianingsih, S.Si., M.Stat., Ph.D. .....
   NIP 19950520 202406 2 003

Head of Statistics Department
Faculty of Science and Data Analytics

Dr.rer.pol. Dedy Dwi Prastyo, S.Si., M.Si.
NIP 19831204 200812 1 002

(Halaman sengaja dikosongkan)

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP : Daniswara Aditya Putra / 5003211132

Departemen : Statistika FSAD - ITS

Dosen Pembimbing / NIP : Dr. Dra. Kartika Fithriasari, M.Si / 19691212 199303 2 002

dengan ini menyatakan bahwa Tugas Akhir dengan judul "**Klasifikasi Spesies Burung Berdasarkan Suara Menggunakan *Deep Learning* dengan Fitur Mel-Spektogram dan *Mel Frequency Cepstral Coefficients* (MFCC)**" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 30 Juni 2025

Mengetahui, Dosen Pembimbing	Mahasiswa
Dr. Dra. Kartika Fithriasari, M.Si NIP. 19691212 199303 2 002	Daniswara Aditya Putra NRP. 5003211132

(Halaman sengaja dikosongkan)

STATEMENT OF ORIGINALITY

The undersigned below:

Name of Student / NRP	: Daniswara Aditya Putra / 5003211132
Department	: Statistics FSAD - ITS
Advisor / NIP	: Dr. Dra. Kartika Fithriasari, M.Si / 19691212 199303 2 002

hereby declare that the Final Project with the title of "**Bird Species Classification Based on Sound Using Deep Learning with Mel-Spectogram dan Mel Frequency Cepstral Coefficients (MFCC)**" the result of my own work, is original, and written by following the rules scientific writing.

If in the future is an discrepancy with this statement, then I am willing to accept sanctions in a accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, 30 June 2025

Acknowledged,
Advisor

Dr. Dra. Kartika Fithriasari, M.Si
NIP. 19691212 199303 2 002

Student

Daniswara Aditya Putra
NRP. 5003211132

(Halaman sengaja dikosongkan)

ABSTRAK

KLASIFIKASI SPESIES BURUNG BERDASARKAN SUARA MENGGUNAKAN *DEEP LEARNING* DENGAN FITUR MEL-SPEKTOGRAM DAN *MEL FREQUENCY CEPTRAL COEFFICIENTS* (MFCC)

Nama Mahasiswa / NRP : Daniswara Aditya Putra / 5003211132
Departemen : Statistika FSAD ITS
Dosen Pembimbing : Dr. Dra. Kartika Fithriasari, M. Si

Abstrak

Burung merupakan indikator penting dalam menilai kesehatan ekosistem dan biodiversitas suatu wilayah. Perkembangan teknologi pengenalan suara berbasis deep learning memungkinkan pemantauan spesies burung dilakukan secara otomatis melalui data audio, tanpa perlu pengamatan langsung yang berisiko mengganggu habitat alami. Pada penelitian ini dilakukan klasifikasi suara enam spesies burung yang umum ditemukan di Indonesia, yaitu Kutilang, Gereja, Perkutut, Tekukur, Trucukan, dan Cendet, menggunakan lima arsitektur model deep learning, yaitu CNN (Mel-Spektrogram), CNN (MFCC), MobileNet (Mel-Spektrogram), MobileNet (MFCC), dan VGGish (waveform). Data yang digunakan merupakan rekaman suara dari situs xeno-canto.org yang telah melalui proses ekstraksi fitur. Model dibangun menggunakan framework TensorFlow dan Keras, kemudian dilatih dengan kombinasi hyperparameter yang bervariasi untuk memperoleh performa terbaik. Berdasarkan hasil evaluasi menggunakan metrik akurasi, precision, recall, dan F1-score pada data test, model CNN dengan fitur MFCC menunjukkan performa terbaik dengan akurasi mencapai 97,97%, precision 97,78%, recall 98,27%, dan F1-score 97,96%. Model ini tidak hanya mampu mengenali pola suara burung secara akurat, tetapi juga menunjukkan kemampuan generalisasi yang sangat baik terhadap data baru. Hasil penelitian ini menegaskan bahwa CNN dengan fitur MFCC merupakan pendekatan yang efektif dan andal untuk tugas klasifikasi suara burung, serta memiliki potensi besar untuk mendukung upaya konservasi berbasis teknologi di Indonesia.

Kata kunci: Burung, CNN, Mel-Spektogram, MFCC, MobileNet, VGGish

(Halaman sengaja dikosongkan)

ABSTRACT

BIRD SPECIES CLASSIFICATION BASED ON SOUND USING DEEP LEARNING WITH MEL-SPECTROGRAM AND MEL FREQUENCY CEPSTRAL COEFFICIENTS (MFCC)

Student Name / NRP : **Daniswara Aditya Putra / 5003211132**
Department : **Statistics F-Scientics - ITS**
Advisor : **Dr. Dra. Kartika Fithriasari, M. Si**

Abstract

Birds are important indicators for assessing the health of ecosystems and biodiversity in a region. Advances in deep learning-based sound recognition technology enable automatic monitoring of bird species through audio data, eliminating the need for direct observation that may disturb natural habitats. This study focuses on classifying the calls of six common bird species found in Indonesia—Kutilang, Gereja, Perkutut, Tekukur, Trucukan, and Cendet—using five deep learning architectures: CNN (Mel-Spectrogram), CNN (MFCC), MobileNet (Mel-Spectrogram), MobileNet (MFCC), and VGGish (waveform). The dataset consists of bird call recordings sourced from xeno-canto.org and processed through feature extraction. Models were developed using the TensorFlow and Keras frameworks and trained with various hyperparameter combinations to achieve optimal performance. Based on evaluation metrics including accuracy, precision, recall, and F1-score on the test data, the CNN model with MFCC features demonstrated the best performance, achieving an accuracy of 97.97%, precision of 97.78%, recall of 98.27%, and F1-score of 97.96%. This model not only accurately recognizes bird call patterns but also exhibits excellent generalization capability to unseen data. The results confirm that CNN with MFCC features is an effective and reliable approach for bird sound classification, with strong potential to support technology-based conservation efforts in Indonesia.

Keywords: Bird, CNN, Mel-Spektogram, MFCC, MobileNet, VGGish

(Halaman sengaja dikosongkan)

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan ke hadirat Allah Subhana Wa Ta'ala atas limpahan rahmat dan karunia-Nya yang tiada henti, sehingga penulis dapat menyelesaikan laporan Tugas Akhir yang berjudul "**Klasifikasi Spesies Burung Berdasarkan Suara Menggunakan *Deep Learning* dengan Fitur Mel-Spektogram dan *Mel Frequency Cepstral Coeffiicients* (MFCC)**" dengan baik dan lancar.

Terselesaikannya laporan ini tentunya tidak lepas dari dukungan, bantuan, serta doa dari banyak pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih dan penghargaan yang sebesar-besarnya kepada:

1. Kedua orang tua, saudara, dan keluarga yang tak henti-hentinya memberikan dukungan dan do'a kepada penulis.
2. Ibu Dr. Dra. Kartika Fithriasari, M.Si. selaku dosen pembimbing atas arahan, saran, serta bimbingannya selama penyusunan Tugas Akhir.
3. Ibu Adatul Mukarromah, S.Si., M.Si. selaku dosen penguji pertama yang memberikan saran dan masukan terkait penulisan Tugas Akhir.
4. Ibu Widhianingsih Tintrim Dwi Ary, S.Si., M.Stat., Ph.D. selaku dosen penguji kedua yang memberikan saran dan masukan terkait penulisan Tugas Akhir.
5. Bapak Dr.rer.pol. Dedy Dwi Prastyo, S.Si., M.Si. selaku Kepala Departemen Statistika, Ibu Dr. Wibawati, S.Si., M.Si. selaku Sekretaris Departemen Statistika, dan Ibu Shofi Andari, S.Stat., M.Si., Ph.D. selaku Kepala Program Studi Sarjana Statistika yang telah memberikan sarana dan prasarana selama perkuliahan.
6. Bapak dan Ibu dosen Departemen Statistika ITS yang telah memberikan ilmu yang bermanfaat bagi penulis, serta segenap staf karyawan Departemen Statistika ITS.
7. Teman-teman Statistika ITS Angkatan 2021, serta semua pihak yang turut membantu dalam pelaksanaan Tugas Akhir yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa dalam penulisan tugas akhir ini masih belum sempurna. Ketidak sempurnaan tersebut menjadi celah untuk pembelajaran dan pengembangan keilmuan kedepannya. Semoga Tugas Akhir ini bisa menjadi bahan referensi dan memberikan manfaat bagi pengembangan keilmuan statistika.

Surabaya, 12 Juni 2025

Penulis

(Halaman sengaja dikosongkan)

DAFTAR ISI

LEMBAR PENGESAHAN ..... **vii**
APPROVAL SHEET ..... **ix**
ABSTRAK ..... **xv**
ABSTRACT ..... **xvii**
DAFTAR ISI ..... **xxi**
DAFTAR GAMBAR ..... **xxiii**
DAFTAR TABEL ..... **xxv**
BAB 1 PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 3
1.5 Manfaat 3
BAB 2 TINJAUAN PUSTAKA **5**
2.1 Suara 5
2.2 *Voice Active Detection* (VAD) 5
2.3 Burung 6
2.4 *Deep Learning* 8
2.5 Mel-Spektogram dan *Mel Frequency Cepstrum Coefficients* (MFCC) 9
2.6 *Audio Processing* 14
2.6.1 Tensorflow 15
2.6.2 TF. Signal 15
2.7 *Convolutional Neural Network* (CNN) 16
2.7.1 *Convolutional Layer* 16
2.7.2 *Pooling Layer* 17
2.7.3 Fully Connected Layer 18
2.8 Fungsi Aktivasi 18
2.8.1 *Rectified Linear Unit* (ReLU) 18
2.8.2 *Softmax Classifier* 19
2.9 *Learning Rate* 20
2.10 *Adam Optimizer* 20
2.11 MobileNet 21
2.12 VGGish 22

2.13 Evaluasi Performa Model 23
2.14 Hasil Penelitian Terdahulu 25
BAB 3 METODOLOGI 27
3.1 Sumber Data 27
3.2 Variabel Penelitian 27
3.3 Struktur Data 28
3.4 Langkah Analisis 30
3.5 Diagram Alir 32
BAB 4 HASIL DAN PEMBAHASAN 33
4.1 Preprocessing Data 33
4.2 Ekstraksi Fitur Mel-Spektogram dan MFCC 34
4.3 Metode Pelatihan Model 42
4.4 Pemodelan dan Evaluasi Klasifikasi dengan CNN 43
4.4.1 Klasifikasi Model CNN dengan Fitur Mel-Spektogram 44
4.4.2 Klasifikasi Model CNN dengan Fitur MFCC 45
4.5 Pemodelan dan Evaluasi Klasifikasi dengan MobileNet 46
4.5.1 Klasifikasi Model *MobileNet* dengan Fitur Mel-Spektogram 48
4.5.2 Klasifikasi Model *MobileNet* dengan Fitur MFCC 49
4.6 Pemodelan dan Evaluasi Klasifikasi dengan VGGish 51
4.7 Pemilihan Model Terbaik 53
4.8 Implementasi Model Terbaik pada Data Test 54
BAB 5 PENUTUP 57
5.1 Kesimpulan 57
5.2 Saran 57
DAFTAR PUSTAKA 59
LAMPIRAN 63

DAFTAR GAMBAR

Gambar 2.1 Burung Kutilang 6
Gambar 2.2 Burung Gereja 7
Gambar 2.3 Burung Perkutut 7
Gambar 2.4 Burung Tekukur 7
Gambar 2.5 Burung Trucukan 8
Gambar 2.6 Burung Cendet 8
Gambar 2.7 Ilustrasi Arsitektur Deep Learning 9
Gambar 2.8 Alur Pembuatan MFCC 10
Gambar 2.9 Ilustrasi dari Data Suara, Spektrum Daya, Mel-Spektogram, dan MFCC 10
Gambar 2.10 Proses Framing 11
Gambar 2.11 Mel Filter Bank 14
Gambar 2.12 Ilustrasi *Convolutional Layer* 17
Gambar 2.13 Ilustrasi pada Max Pooling Layer 2x2 17
Gambar 2.14 Fungsi Softmax 19
Gambar 2.15 Efek Nilai Learning Rate terhadap Performa Model 20
Gambar 2. 16 Arsitektur MobileNet 22
Gambar 2.17 Arsitektur VGGish 22
Gambar 2.18 *Confussion Matrix Multiclass Classification* 23
Gambar 3.1 Spesies Burung yang Digunakan dalam Penelitian 27
Gambar 3.2 Contoh Representasi Visual dari Rekaman Suara 29
Gambar 3.3 Diagram Alir Penelitian 32
Gambar 4.1 Gelombang Audio Cendet2.wav 33
Gambar 4.2 Gelombang Audio Cendet2.wav setelah Trimming 33
Gambar 4.3 Gelombang Audio Segmen 1 Burung Cendet2.wav 38
Gambar 4.4 Visualisasi Mel-Spektogram Audio Segmen 1 Cendet2.wav 39
Gambar 4.5 Visualisasi MFCC Audio Segmen 1 Cendet2.wav 40
Gambar 4.6 Visualisasi Gambar Burung, Gelombang Audio, Mel-Spektogram dan MFCC untuk Setiap Jenis Burung (dari Kiri ke Kanan) , dengan Urutan Jenis Burung dari Atas ke Bawah: Cendet, Gereja, Kutilang, Perkutut, Tekukur, dan Trucukan 41
Gambar 4.7 Contoh Input 42
Gambar 4.8 Contoh Input dengan zero-padding pada Sisi Kanan dan Bawah 42
Gambar 4.9 Contoh Input dengan zero-padding pada Semua Sisi 42
Gambar 4.10 Nilai Akurasi dan Loss Model CNN Mel-Spektogram Kombinasi Pertama 45
Gambar 4.11 Nilai Akurasi dan Loss Model CNN MFCC Kombinasi Pertama 46
Gambar 4.12 Nilai Akurasi dan Loss Model MobileNet Mel-Spektogram Kombinasi Ketiga 49
Gambar 4.13 Nilai Akurasi dan Loss Model MobileNet MFCC Kombinasi Pertama 50
Gambar 4.14 Nilai Akurasi dan Loss Model VGGish Kombinasi Pertama 53
Gambar 4.15 Confusion Matrix Prediksi Model Terbaik pada Data Test 55
Gambar 4.16 Suara yang akan Diprediksi 56

(Halaman sengaja dikosongkan)

DAFTAR TABEL

Tabel 2.1 Penelitian Terdahulu Terkait Klasifikasi Suara Burung 25
Tabel 3.1 Jenis Spesies Burung 27
Tabel 3.2 Variabel Penelitian 28
Tabel 3.3 Struktur Data secara General 28
Tabel 3.4 Struktur Data Suara yang akan Diproses 28
Tabel 3.5 Struktur Data Fitur MFCC yang akan Diproses 29
Tabel 3.6 Struktur Data Fitur Mel-Spektogram yang akan Diproses 29
Tabel 4.1 Rekaman Suara Burung Sebelum dan Sesudah Proses Trimming 34
Tabel 4.2 Hasil pembagian Data Training dan Testing Per Jenis Burung 34
Tabel 4.3 Nilai Gelombang Audio Segmen 1 Cendet2.wav 36
Tabel 4.4 Nilai Pre-emphasis Audio Segmen 1 Cendet2.wav 36
Tabel 4.5 Hasil Proses Framing pada Audio Segmen 1 Cendet2.wav 36
Tabel 4.6 Hasil Proses Windowing pada Audio Segmen 1 Cendet2.wav 37
Tabel 4.7 Hasil Proses STFT pada Audio Segmen 1 Cendet2.wav 37
Tabel 4.8 Hasil Filter Bank untuk Audio Segmen 1 Cendet2.wav 37
Tabel 4.9 Mel-Spektogram untuk Audio Segmen 1 Cendet2.wav 38
Tabel 4.10 MFCC untuk Audio Segmen 1 Cendet2.wav 38
Tabel 4.11 Arsitektur Model CNN 43
Tabel 4.12 Performa Kombinasi Hyperparameter Model CNN dengan Input Mel-Spektogram 44
Tabel 4.13 Performa Kombinasi Hyperparameter Model CNN dengan Input MFCC 46
Tabel 4.14 Arsitektur Model MobileNet dari Library Tensorflow 47
Tabel 4.15 Performa Kombinasi Hyperparameter Model MobileNet dengan fitur Mel-Spektogram 48
Tabel 4.16 Performa Kombinasi Hyperparameter Model MobileNet dengan fitur MFCC 50
Tabel 4.17 Arsitektur Model VGGish 51
Tabel 4.18 Performa Kombinasi Hyperparameter Model VGGish 52
Tabel 4.19 Performa Terbaik Setiap Model 54
Tabel 4.20 Hasil Evaluasi Setiap Model pada Data Test 54
Tabel 4.21 Hasil Prediksi Berdasarkan Jumlah Deteksi masing-masing Jenis Burung 56

(Halaman sengaja dikosongkan)

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Burung memiliki peran ekologis yang signifikan dan dapat menjadi indikator kesehatan lingkungan. Populasi burung sering kali mencerminkan keadaan alam sekitar, termasuk kualitas udara, air, dan vegetasi. Ketika lingkungan alami terganggu, perubahan populasi burung dapat memberikan petunjuk awal terhadap gangguan ekosistem tersebut. Selain itu, burung juga merupakan bagian dari keanekaragaman hayati yang perlu dilestarikan, terutama di Indonesia yang memiliki kekayaan spesies burung yang melimpah (Soraya, 2024). Namun, penelitian terhadap spesies burung sering menghadapi tantangan, seperti keterbatasan akses ke habitat alami dan risiko mengganggu populasi burung. Salah satu pendekatan yang banyak digunakan adalah merekam suara burung di alam liar untuk mengidentifikasi spesiesnya (Rahman, 2022). Namun, rekaman suara ini memerlukan analisis lanjutan untuk mengidentifikasi spesies burung tertentu, yang dapat dilakukan menggunakan teknologi berbasis *deep learning* (Hatma, 2024).

Dalam lingkup Indonesia, sebagai negara dengan kekayaan biodiversitas burung yang melimpah, penelitian ini menjadi semakin relevan. Banyak spesies burung yang hidup di sekitar pemukiman penduduk atau kawasan hutan, yang keberadaannya sering kali tidak terdeteksi karena sulitnya akses atau gangguan aktivitas manusia. Dengan memanfaatkan algoritma *deep learning*, seperti *convolutional neural network* (CNN), sistem pengenalan suara burung dapat dikembangkan untuk mengidentifikasi spesies burung di wilayah tertentu secara akurat tanpa perlu kehadiran fisik manusia yang dapat mengganggu habitatnya (Afida, 2020). Pada penelitian ini akan difokuskan pada enam jenis burung yang sering ditemukan di sekitar pemukiman di Indonesia, yaitu Burung Kutilang (*Pycnonotus aurigaster)*, Burung Gereja (*Passer domesticus*), Burung Perkutut (*Geopelia striata),* Burung Tekukur (*Streptopelia chinensis),* Burung Trucukan (*Pycnonotus goiavier*), dan Burung Cendet (*Lanius cristatus*). Pemilihan keenam spesies burung ini didasari oleh beberapa alasan strategis. Pertama, karena spesies-spesies tersebut umum ditemukan di sekitar pemukiman dan kawasan hutan di Indonesia, sehingga memudahkan pengumpulan data rekaman suara secara berkelanjutan dan representatif. Kedua, variasi karakteristik vokal masing-masing spesies memberikan cakupan yang luas dalam hal pola suara, sehingga sistem pengenalan suara yang dikembangkan dapat mengidentifikasi perbedaan khas antar spesies secara akurat. Selain itu, sebagai indikator kesehatan lingkungan, spesies-spesies ini memiliki peran ekologis yang signifikan, yang memungkinkan pemantauan dan konservasi keanekaragaman hayati dilakukan secara efektif tanpa harus mengganggu habitat alami mereka. Dengan demikian, penggunaan spesies yang sering ditemui ini tidak hanya praktis untuk pengumpulan data, tetapi juga relevan untuk aplikasi konservasi dan pemetaan biodiversitas di Indonesia.

Klasifikasi data suara dapat dilakukan menggunakan beberapa metode berbasis *deep learning* (DL), terutama pada data yang tidak terstruktur seperti audio (Ponnusamy, 2017). Dalam pengolahan data suara, langkah pertama yang sering dilakukan adalah mengubah data audio menjadi representasi visual seperti mel-spektogram dan fitur *Mel Frequency Cepstral Coefficients* (MFCC). Representasi ini kemudian dapat digunakan sebagai input untuk model klasifikasi. Pada penelitian ini menggunakan model CNN, *pre-trained model* MobileNet, dan pre-trined audio VGGish. Metode CNN dipilih karena memiliki keunggulan yaitu dapat mendeteksi fitur-fitur penting dari data audio secara otomatis tanpa pengawasan manusia secara terus menerus. Selain itu, metode CNN juga efisien dalam melakukan proses komputasinya, sehingga proses klasifikasi dapat dilakukan dengan lebih cepat tanpa menggunakan sumber daya yang banyak. Model yang juga digunakan adalah VGGish, sebuah model *pre-trained* yang dikembangkan untuk tugas klasifikasi dengan arsitektur mendalam dan telah dilatih

sebelumnya pada dataset besar Audio (Alzubaidi, *et al.*, 2021). Dengan demikian, CNN, MobileNet, dan VGGish dapat mengolah data suara yang telah direpresentasikan dalam bentuk spektogram secara efisien untuk meningkatkan akurasi klasifikasi.

Penelitian terkait penggunaan metode DL untuk klasifikasi suara burung telah dilakukan oleh beberapa peneliti. Salah satunya adalah penelitian oleh Irwandi *et al.* (2005), yang mengeksplorasi pemanfaatan rekaman suara burung dan analisis spektrogram untuk menyusun metode klasifikasi berdasarkan suara, yang dikenal sebagai sonotaksonomi. Dalam penelitian ini, mereka berhasil mengidentifikasi lima spesies burung dengan menggunakan analisis karakteristik suara seperti jumlah elemen, frekuensi dominan, dan spektrum maksimum. Selain itu, penelitian oleh Ali (2020) berfokus pada klasifikasi suara burung lovebird menggunakan metode MFCC dan *Dynamic Time Warping* (DTW). Penelitian ini menghasilkan akurasi validasi suara sebesar 80%, yang menunjukkan potensi metode ini dalam mengenali kualitas suara burung berdasarkan kicauan mereka. Penelitian lain oleh Putra (2019) mengembangkan model klasifikasi suara burung menggunakan kombinasi MFCC dan CNN. Dataset yang digunakan mencakup berbagai jenis burung lokal di Indonesia, seperti burung trucukan, sirpu, cendet, dan kenari. Penelitian ini berhasil mencapai akurasi tinggi sebesar 94%, menunjukkan bahwa kombinasi MFCC sebagai fitur input dengan CNN sebagai model klasifikasi mampu menangkap pola suara burung dengan baik. Namun, penelitian ini membatasi durasi rekaman suara selama 1 detik yang dapat membatasi variasi data input. Selanjutnya, penelitian oleh Ihsanti dan Al Maki (2024) menggunakan fitur Mel-spektogram dan CNN untuk mengklasifikasikan suara burung hantu berdasarkan genusnya. Mereka melaporkan akurasi tinggi sebesar 99.8% dalam pengujian sistem, meskipun hasil *precision* dan *recall* yang rendah sebesar 16% yang menunjukkan perlunya dataset yang lebih seimbang untuk meningkatkan kinerja model. Berdasarkan penelitian-penelitian tersebut, penelitian ini akan memanfaatkan mel-spektrogram dan MFCC sebagai fitur representasi data suara, serta memanfaatkan CNN, MobileNet, dan VGGish dalam mengenali pola suara. Penelitian ini diharapkan dapat memberikan hasil yang lebih akurat dan efisien dalam mengidentifikasi spesies burung berdasarkan suara mereka. Pendekatan ini juga bertujuan untuk mengatasi keterbatasan penelitian terdahulu, seperti durasi data yang terbatas dan ketidakseimbangan dataset, sehingga menghasilkan model klasifikasi yang lebih baik.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang dijelaskan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut.

1. Bagaimana mengembangkan model klasifikasi suara burung yang akurat menggunakan *Convolutional Neural Network* (CNN) dan MobileNet dengan memanfaatkan fitur mel-spektrogram dan MFCC sebagai representasi data suara, serta model VGGish dengan input dari data suara?
2. Bagaimana perbandingan performa model CNN, MobileNet, dan VGGish dalam mengklasifikasikan spesies burung berdasarkan rekaman suara, dengan menggunakan fitur mel-spektogram dan *Mel Frequency Cepstral Coefficients* (MFCC)?

1.3 Batasan Masalah

Berikut merupakan batasan masalah dalam penelitian ini.

1. Model *deep learning* yang digunakan dalam melakukan klasifikasi yaitu *pre-trained model* MobileNet, VGGish, dan *Convolutional Neural Network* (CNN).
2. Kategori suara burung yang terdapat pada penelitian ini terdiri dari enam kategori spesies burung yaitu Kutilang (*Pycnonotus aurigaster*), Gereja (*Passer domesticus*), Perkutut

(*Geopelia striata*), Tekukur (*Streptopelia chinensis*), Trucukan (*Pycnonotus goiavier*), dan Cendet (*Lanius cristatus*).

3. Sumber data suara burung yang digunakan dalam penelitian ini berasal dari situs web xenocanto.org dan berformat .wav.

1.4 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut.

1. Mengembangkan model klasifikasi suara burung menggunakan metode CNN, MobileNet, dan VGGish untuk mengidentifikasi spesies burung berdasarkan rekaman suara dengan memanfaatkan fitur mel-spektrogram dan MFCC.
2. Membandingkan dan menentukan model terbaik antara metode CNN, MobileNet, dan VGGish dengan fitur audio mel-spektrogram dan MFCC dalam mengklasifikasikan spesies burung berdasarkan rekaman suara melalui analisis performa menggunakan metrik evaluasi *precision*, *recall*, *F1-score*, dan *accuracy*.

1.5 Manfaat

Penelitian ini diharapkan dapat memberikan kontribusi penting dalam pemahaman dan pengembangan teknik klasifikasi audio, khususnya menggunakan model *pre-trained* seperti VGG16. Selain memberikan wawasan baru tentang penggunaan CNN untuk klasifikasi suara, hasil dari penelitian ini diharapkan dapat memberikan dampak signifikan pada aplikasi yang berhubungan dengan konservasi satwa liar dan pemantauan lingkungan. Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut.

1. Penelitian ini berkontribusi pada literatur tentang klasifikasi audio dengan menggunakan *pre-trained model* MobileNet dan VGGish yang sudah terbukti efektif di berbagai tugas pengenalan pola.
2. Hasil penelitian ini dapat diaplikasikan pada pemantauan satwa liar, terutama untuk konservasi burung di lingkungan alami mereka.
3. Penelitian ini juga dapat menjadi acuan bagi pengembangan model klasifikasi audio lainnya untuk mendeteksi spesies burung atau suara alam lainnya, yang berperan penting dalam pelestarian keanekaragaman hayati.

(Halaman ini sengaja dikosongkan)

BAB 2 TINJAUAN PUSTAKA

2.1 Suara

Suara adalah fenomena fisik yang dihasilkan dari getaran benda yang merambat melalui medium, seperti udara, air, atau benda padat. Dalam konteks pengenalan pola dan pengolahan sinyal suara, suara sering diwakili oleh gelombang bunyi yang dapat diukur dalam berbagai dimensi, seperti frekuensi, amplitudo, dan durasi. Frekuensi gelombang suara diukur dalam satuan Hertz (Hz), yang menggambarkan jumlah getaran per detik. Amplitudo menggambarkan kekuatan atau intensitas suara, sedangkan durasi mengacu pada lamanya waktu suara tersebut terdengar. Pengenalan suara secara otomatis menggunakan algoritma pembelajaran mesin dan deep learning kini menjadi salah satu bidang yang sangat berkembang, terutama dengan meningkatnya kemampuan komputasi dan ketersediaan data suara (Mitchell, 2019).

Dalam pengolahan sinyal suara untuk keperluan klasifikasi atau pengenalan pola, proses awal melibatkan perekaman suara melalui mikrofon yang mengubah gelombang suara menjadi sinyal digital. Data digital ini kemudian dianalisis menggunakan metode pemrosesan sinyal seperti ekstraksi fitur mel-spektogram atau MFCC. Ekstraksi fitur ini memungkinkan model *deep learning*, seperti CNN, untuk mengenali pola-pola penting dalam data suara (Imaroh, 2024). CNN sering digunakan karena kemampuannya dalam mendeteksi fitur spasial pada data visual, termasuk data spektrogram yang merepresentasikan suara secara grafis (Wijaya, 2024). Dengan pendekatan ini, teknologi *deep learning* diharapkan dapat menunjukkan kemampuan signifikan dalam meningkatkan akurasi dan efisiensi sistem pengenalan suara (Lubis, 2024).

2.2 *Voice Active Detection* (VAD)

VAD adalah teknik dalam pemrosesan sinyal audio yang digunakan untuk mendeteksi keberadaan suara manusia atau suara tertentu dalam suatu rekaman audio. Tujuan utama VAD adalah memisahkan bagian rekaman yang mengandung suara dari bagian yang hanya berisi jeda atau noise. Proses ini dilakukan dengan menganalisis energi sinyal atau karakteristik spektral lainnya untuk menentukan apakah segmen tertentu dari audio mengandung suara atau tidak. VAD sering digunakan dalam berbagai aplikasi, seperti pengenalan suara, telekomunikasi, dan pengolahan data audio, karena membantu mengurangi beban pemrosesan dengan hanya memfokuskan analisis pada bagian yang relevan (Ramírez, *et al.*, 2004).

Dalam konteks penelitian klasifikasi suara burung, VAD dapat digunakan untuk memotong rekaman panjang menjadi beberapa segmen pendek berdasarkan keberadaan suara burung. Ini dilakukan dengan menetapkan ambang batas energi tertentu untuk mendeteksi aktivitas suara. Bagian rekaman yang memiliki energi di bawah ambang batas dianggap sebagai jeda atau *noise* dan diabaikan. Dengan cara ini, VAD tidak hanya meningkatkan efisiensi pemrosesan data tetapi juga membantu memperbaiki kualitas dataset dengan menghilangkan bagian-bagian yang tidak relevan. Teknik ini dapat diimplementasikan menggunakan *library* seperti Librosa melalui fungsi 'librosa.effects.split', yang mendeteksi interval suara berdasarkan energi sinyal. Persamaan (2.1) adalah rumus yang digunakan dalam fungsi 'librosa.effects.split' untuk mendeteksi aktivitas suara berdasarkan energi sinyal.

$$E_t = \frac{1}{T}\sum_{h=1}^{T} x(h)^2 \tag{2.1}$$

Dimana E_t adalah energi rata-rata sinyal suara pada frame ke-t, T adalah jumlah sampel dalam satu rekaman, dan $x(h)$ adalah nilai amplitudo sinyal pada sampel ke-h. Fungsi 'librosa.effects.split' bekerja dengan membandingkan E_t terhadap ambang batas tertentu ($E_{threshold}$) yang didefinisikan pengguna. Jika energi dalam suatu segmen melebihi nilai ambang batas ini, segmen tersebut dianggap sebagai bagian yang mengandung suara dan dipertahankan, sedangkan bagian lain yang berada di bawah ambang batas akan dihapus.

2.3 Burung

Burung adalah kelompok hewan vertebrata dari kelas *Aves* yang memiliki ciri khas berupa bulu, paruh tanpa gigi, dan kemampuan untuk bertelur. Burung memainkan peran penting dalam ekosistem, seperti menyebarkan biji, mengontrol populasi serangga, dan bertindak sebagai penyerbuk. Selain itu, burung sering digunakan sebagai objek penelitian di bidang ekologi, perilaku, dan konservasi karena memiliki variasi perilaku yang kompleks dan adaptasi unik terhadap lingkungan mereka (Gill, 2007). Dalam konteks penelitian ini, burung digunakan sebagai objek untuk menganalisis dan mengidentifikasi suara burung tertentu dalam dataset yang beragam karena setiap spesies memiliki karakteristik suara yang unik. Karakteristik ini memungkinkan penggunaan teknik pengenalan suara berbasis *deep learning*, seperti CNN dan VGG16 untuk mengklasifikasikan suara-suara tersebut secara akurat. Berikut merupakan spesies burung yang digunakan dalam penelitian ini.

a. Burung Kutilang (*Pycnonotus aurigaster*)

Gambar 2.1 Burung Kutilang

Gambar 2.1 merupakan ilustrasi dari Burung Kutilang. Karakteristik suara burung Kutilang sangat khas dan berperan penting dalam komunikasi spesies. Kicauannya terdengar nyaring namun merdu, seringkali diidentifikasi dengan suara "cuk-cuk" dan "cang-kur" yang diulang cepat. Selain itu, mereka juga dapat mengeluarkan suara seperti sedang bersiul dan suara "ke-ti-lang" yang diulang-ulang. Variasi suara ini berfungsi sebagai instruksi pertahanan wilayah, menarik pasangan, memberikan informasi tentang bahaya, serta menandakan adanya sumber pakan atau agresi. Kemampuan burung Kutilang dalam menghasilkan variasi suara yang luas juga menunjukkan adaptasinya yang baik terhadap lingkungan baru.

b. Burung Gereja (*Passer domesticus*)

Gambar 2.2 Burung Gereja

Gambar 2.2 merupakan ilustrasi dari Burung Gereja. Suara burung Gereja meningkat intensitasnya saat musim kawin tiba, menjadi lebih nyaring dan lantang sebagai upaya pejantan menarik perhatian betina. Meskipun tidak ada informasi spesifik tentang variasi suara seperti pada Burung Kutilang, kicauan burung Gereja yang meningkat intensitasnya memiliki peran penting dalam reproduksi. Burung ini memakan biji-bijian dan serangga kecil, serta memiliki kemampuan adaptasi yang tinggi terhadap perubahan kondisi cuaca, ketersediaan pakan, maupun kehadiran predator.

c. Burung Perkutut (*Geopelia striata*)

Gambar 2.3 Burung Perkutut

Gambar 2.3 merupakan ilustrasi dari Burung Perkutut. Kicauan perkutut memiliki nada yang lembut dan berirama, sering kali menjadi pengisi suasana di lingkungan sekitar. Banyak orang percaya bahwa kicauan perkutut dapat memberikan ketenangan dan kebahagiaan. Suara perkutut lokal cenderung lebih jernih dengan tempo cepat, sedangkan suara perkutut Bangkok lebih ngebas dan memiliki tempo yang lebih lambat. Perkutut jantan umumnya memiliki suara yang lebih keras, lantang, dan bertenaga dibandingkan betina, serta memiliki variasi suara yang lebih beragam.

d. Burung Tekukur (*Streptopelia chinensis*)

Gambar 2.4 Burung Tekukur

Gambar 2.4 merupakan ilustrasi dari Burung Tekukur. Karakteristik suara burung Tekukur adalah bunyi dekutan nyaring yang diulang-ulang, "te-kuk-kurrrr," dengan nada terakhir yang memanjang. Kicauan ini menjadi asal mula nama Indonesianya. Burung Tekukur mencari makan biji-bijian, benih rumput, dan buah-buahan yang jatuh di tanah. Burung ini mudah beradaptasi dengan lingkungan manusia dan sering ditemukan di hutan terbuka, lapangan, serta taman kota.

e. Burung Trucukan (*Pycnonotus goiavier*)

Gambar 2.5 Burung Trucukan

Gambar 2.5 merupakan ilustrasi dari Burung Trucukan. Kicauan Burung Trucukan terdengar nyaring dan berisik, dengan suara "cok, cok, cok" diikuti siulan pendek yang cepat berulang beberapa kali, kemudian terdengar seperti nyanyian lirih atau gumaman. Burung Trucukan jantan memiliki suara yang lebih keras, bervariasi, dan konsisten, serta sering mengangkat jambul. Makanan burung ini terdiri dari serangga, buah-buahan, dan nektar.

f. Burung Cendet (*Lanius cristatus*)

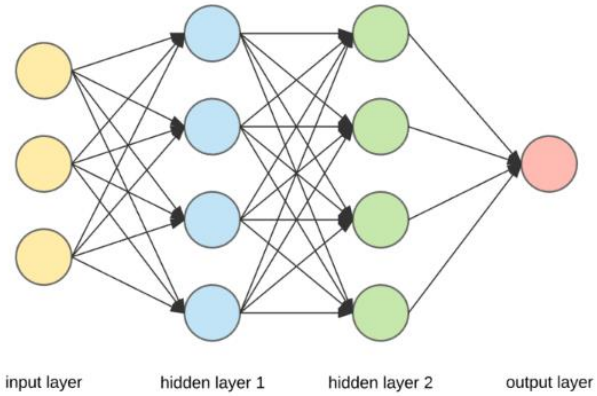
Gambar 2.6 Burung Cendet

Gambar 2.6 merupakan ilustrasi dari Burung Cendet. Karakteristik suara burung Cendet sangat bervariasi dan harmonis. Selain kicauan melodis, burung ini juga mampu menghasilkan berbagai tipe suara seperti siulan tajam, pekikan keras, dan suara-suara unik lainnya. Cendet jantan cenderung lebih agresif dan responsif terhadap suara, berkicau lebih lama dan lebih sering dibandingkan betina. Kicauannya berfungsi untuk menandai wilayah, menarik perhatian betina, dan merespons stimulus dari lingkungan.

2.4 *Deep Learning*

Deep Learning adalah salah satu cabang dari pembelajaran mesin (*machine learning*) yang berfokus pada algoritma berbasis jaringan saraf tiruan (*artificial neural networks*). Model *deep learning* memiliki arsitektur berlapis-lapis yang disebut dengan jaringan saraf dalam (*deep neural networks*), di mana tiap lapisan terdiri dari sejumlah neuron yang memproses input dan

menyaring informasi untuk ditransmisikan ke lapisan berikutnya. Gambar 2.7 menggambarkan mengenai ilustrasi arsitektur *deep learning* (Buono, 2020).



Gambar 2.7 Ilustrasi Arsitektur *Deep Learning*

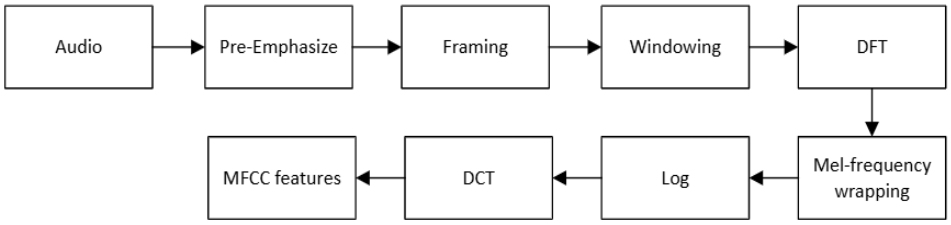
Keunggulan utama *deep learning* terletak pada kemampuannya untuk belajar secara hierarkis. Setiap lapisan dalam jaringan saraf dalam secara progresif mengekstraksi fitur tingkat yang lebih tinggi dari data mentah (van deer Laak, Litjens, & Ciompi, 2021). Sebagai contoh, dalam pengenalan gambar, lapisan awal mungkin mengenali tepi atau pola sederhana, sementara lapisan berikutnya mendeteksi bentuk objek yang lebih kompleks. Proses pembelajaran ini dilakukan melalui propagasi maju (*forward propagation*) dan penyesuaian bobot menggunakan *backpropagation* berdasarkan kesalahan prediksi. Dengan arsitektur mendalam yang dapat mencakup hingga ratusan lapisan tersembunyi, model *deep learning* mampu menangani data dengan kompleksitas tinggi dan menghasilkan akurasi yang lebih baik dibandingkan metode pembelajaran mesin tradisional (Baldock, Mennel, & Nesyhabur, 2021).

2.5 Mel-Spektogram dan *Mel Frequency Cepstrum Coefficients* (MFCC)

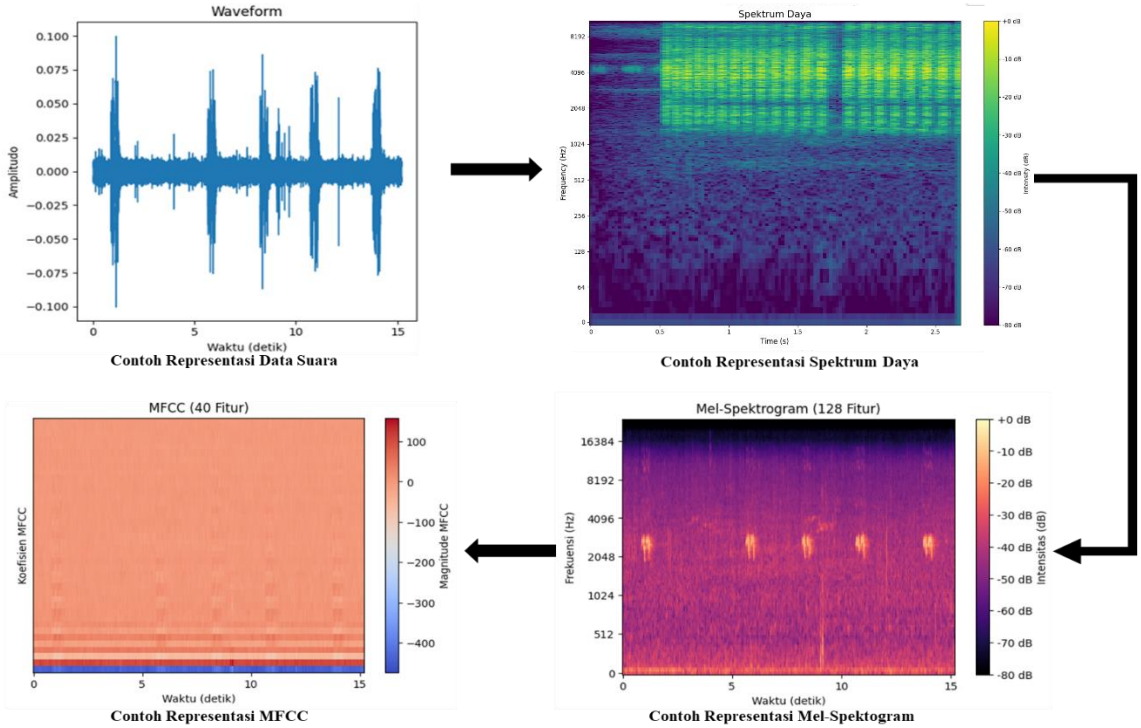
MFCC adalah informasi yang dihasilkan dari sinyal audio melalui serangkaian transformasi. MFCC merupakan fitur spektral jangka pendek yang didasarkan pada representasi cepstral dari audio. Yang membedakan *cepstrum* dengan *mel-frequency*, *cepstrum* adalah skala frekuensi pada MFCC yang disusun berdasarkan skala mel, yang lebih menyerupai respons pendengaran manusia dibandingkan dengan frekuensi yang menggunakan skala linier pada *cepstrum* biasa. Pembengkokan frekuensi ini memungkinkan representasi suara yang lebih akurat. MFCC umumnya diterapkan pada pemodelan suara dan pengenalan suara dalam *machine learning* (Vyas & Kumari, 2013). Sementara itu, mel-spektrogram adalah representasi visual dari sinyal audio yang menunjukkan variasi energi pada berbagai pita frekuensi seiring waktu. Ini adalah representasi waktu-frekuensi dari sinyal yang dihasilkan dengan membagi sinyal menjadi segmen-segmen kecil dari jendela waktu yang sangat pendek dan kemudian menghitung spektrum daya pada setiap segmen waktu menggunakan metode seperti *Short-Term Fourier Transform* (STFT). Mel-Spektrogram memiliki kemampuan untuk menangkap informasi temporal dan spektral dari sinyal audio, memungkinkan ekstraksi fitur yang lebih kompleks. Skala mel pada mel-spektrogram dirancang untuk meniru cara telinga manusia merespons frekuensi suara yang berbeda, sehingga fitur yang diekstrak lebih sesuai dengan persepsi manusia terhadap suara. Penggunaan sinyal dalam bentuk mel-spektogram telah menjadi metode yang sangat efektif dibandingkan dengan sinyal audio 1D (Amrilla, 2023).

Mel-spektrogram dan MFCC dipilih sebagai representasi data suara karena keduanya memiliki keunggulan dalam menangkap karakteristik penting suara. MFCC, dengan alur pemrosesan yang meniru sistem pendengaran manusia, mampu menghasilkan data seminimal mungkin tanpa menghilangkan informasi penting, membuatnya efektif untuk pengenalan suara.

Selain itu, MFCC adalah salah satu teknik ekstraksi fitur suara yang paling sering digunakan oleh para peneliti. Sementara itu, mel-spektrogram memberikan representasi visual dari spektrum frekuensi suara, memungkinkan kita melihat perubahan energi frekuensi dari waktu ke waktu. Yang membedakan keduanya, skala frekuensi pada MFCC disusun berdasarkan skala mel yang lebih menyerupai respons pendengaran manusia, sedangkan mel-spektrogram menampilkan spektrum frekuensi suara dalam skala waktu. MFCC dihitung dari mel-spektrogram dengan menerapkan *Discrete Cosine Transform* (DCT) untuk menghasilkan koefisien cepstral yang merepresentasikan amplop spektral dari suara, sehingga lebih menekankan pada representasi spektral yang ringkas dan menghilangkan beberapa detail temporal. Dengan menggunakan kedua fitur tersebut, diharapkan model *Deep Learning* dapat mempelajari fitur-fitur suara yang relevan dari berbagai perspektif, meningkatkan akurasi dalam klasifikasi suara burung. Gambar 2.8 menunjukkan alur transformasi dari audio hingga menjadi MFCC dan Gambar 2.9 menunjukkan ilustrasi dari data suara, spektrum daya, mel-spektogram, dan MFCC yang akan diproses.



Gambar 2.8 Alur Pembuatan MFCC



Gambar 2.9 Ilustrasi dari Data Suara, Spektrum Daya, Mel-Spektogram, dan MFCC

Sebelum masuk ke alur pembuatan, penting untuk memahami bahwa proses ini melibatkan beberapa langkah transformasi sinyal audio untuk mengubah data mentah menjadi fitur yang dapat diolah oleh model machine learning. MFCC bekerja dengan cara menangkap pola frekuensi suara yang relevan dengan karakteristik akustik, sehingga memungkinkan sistem

untuk membedakan spesies burung berdasarkan rekaman suara mereka. Proses ini menggabungkan prinsip-prinsip pemrosesan sinyal digital, termasuk analisis spektrum, pembentukan skala mel, dan representasi cepstral. Berikut adalah alur pembuatan mel-spektogram dan MFCC yang digunakan dalam penelitian ini.

1. *Pre-emphasis*

   Pre-emphasis merupakan tahap pertama dari MFCC, pada tahap ini suara difilter akan tetapi dalam proses filternya frekuensi-frekuensi yang tinggi pada sebuah spektrum akan dipertahankan. *Pre-emphasis* sendiri bertujuan untuk mengurangi *noise ratio* pada sinyal serta menyeimbangkan spektrum dari *voiced sound*. Secara perhitungan tahap ini dapat dirumuskan dalam dalam Persamaan (2.2).

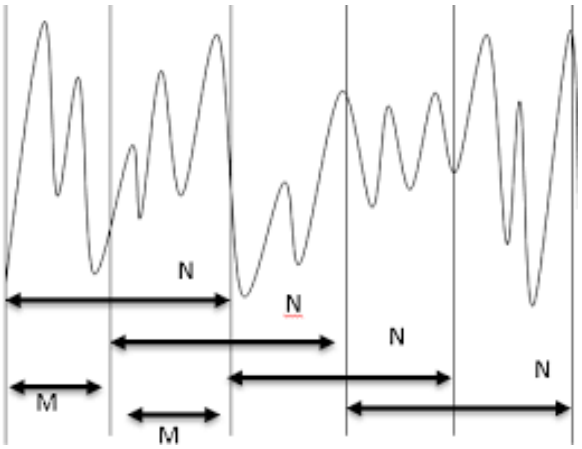
$$x_{pre}(t) = x(t) - \alpha x(t-1) \tag{2.2}$$

   dimana,

   $x_{pre}(t)$: sinyal hasil filter *pre-emphasis* ke-*t*
   $x(t)$: sinyal sebelum *pre-emphasis* ke-*t*
   α : koefisien *pre-emphasis* (0,95)
   $x(t-1)$: sinyal sebelum *pre-emphasis* ke (*t*-1)
   t : indeks waktu yang menunjukkan posisi sampel dalam sinyal audio

2. *Framing*

   Sinyal suara yang telah di *pre-emphasis* akan dilakukan proses *framing*. Pada langkah ini sinyal akan terbagi menjadi beberapa frame dengan masing-masing frame memuat N sampel sinyal dan *frame* yang saling berdekatan dipisahkan sejauh M sampel. Panjang *frame* yang membagi sampel menjadi beberapa *frame* berdasarkan waktu terletak di antara 20ms sampai 40ms. Gambar 2.10 menunjukan M adalah panjang *frame* sedangkan N menunjukan jumlah sampel.



Gambar 2.10 Proses *Framing*

   Proses *framing* seperti pada Gambar 2.10 merupakan langkah penting dalam pengolahan sinyal suara setelah proses *pre-emphasis*. Dalam tahap ini, sinyal suara yang telah diproses sebelumnya dibagi-bagi menjadi potongan-potongan yang lebih kecil atau *frame*. Agar informasi tidak hilang di batas antar *frame*, biasanya diterapkan tumpang tindih (*overlap*) antara *frame* yang berdekatan. Pembagian sinyal menjadi frame ini

bertujuan untuk memudahkan analisis spektral dan ekstraksi fitur dari sinyal suara, sehingga nantinya dapat digunakan dalam berbagai aplikasi pengenalan suara.

3. *Windowing*

Proses *windowing* mempunyai tujuan untuk mengurangi efek diskontinu pada ujung *frame* yang dihasilkan oleh *frame blocking*. Fungsi *window* (*w(t)*) sendiri ada banyak, contohnya adalah *rectangular window* dan *hamming window*. *Rectangular window* adalah fungsi yang paling mudah untuk diaplikasi. Fungsi ini menghasilkan *noise* yang paling rendah, akan tetapi fungsi ini memberikan *sidelobe level* yang paling rendah. Rendahnya *sidelobe* menyebabkan kebocoran *spectral* yang terjadi dalam proses ekstrasi ciri. Persamaan (2.3) menunjukan fungsi *rectangular window*.

$$w(t) = \begin{cases} 1\,, & 0 \leq t \leq T \\ 0\,, & lainnya \end{cases} \tag{2.3}$$

Fungsi *hamming window* adalah fungsi yang paling sering digunakan, berbeda dengan *rectangular window* fungsi *window* ini menghasilkan *sidelobe* yang tidak terlalu tinggi dan *noise* yang dihasilkan tidak terlalu besar. Persamaan (2.4) menunjukan fungsi *hamming window*.

$$w(t) = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi t}{T-1})\,, & 0 \leq t \leq T \\ 0 \qquad\qquad\qquad\qquad\qquad\quad , & lainnya \end{cases} \tag{2.4}$$

Persamaan (2.5) menunjukan fungsi window terhadap sinyal suara yang diinputkan.

$$x_{frame}(t) = x_{pre}(t).w(t) \tag{2.5}$$

dimana,

$x_{frame}(t)$: sinyal output hasil *windowing* ke-*t*
$w(t)$: fungsi *window* ke-*t*
T : jumlah sampel yang akan diproses

4. *Fast Fourier Transform* (FFT)

Fast Fourier Transform adalah pengembangan dari algoritma *Discrete Fourier Transform* (DFT) yang digunakan untuk mengubah sinyal yang semula *time domain* menjadi *frequency domain*. Secara perhitungan tahap ini dapat dirumuskan dalam dalam persamaan (2.6).

$$X(t,k) = \sum_{t=0}^{T-1} x_{frame}(t) e^{-j\left(\frac{2\pi}{T}\right)tk} \quad ; k = 0,1,2, \dots ,T-1 \tag{2.6}$$

Lalu dilakukan perhitungan magnitudo kuadrat dari spektrum untuk mendapatkan spektrum daya. Spektrum daya dapat dihitung dengan Persamaan (2.7).

$$SD(t,k) = |X(t,k)|^2 \tag{2.7}$$

dimana,

$X(t,k)$	: hasil perhitungan FFT indeks frame ke-t dan frekuensi ke-*k*
k	: indeks frekuensi diskrit yang bernilai ($k = \frac{T}{2}, k \in T$)
$SD(t,k)$	: spektrum daya indeks frame ke-*t* dan frekuensi ke-*k*

5. *Mel-scale* dan *Filter Bank*

Pada tahap ini dilakukan *wrapping* terhadap spectrum yang dihasilkan *mel-scale* untuk menyesuaikan resolusi frekuensi terhadap properti pendengaran manusia. Kemudian *mel-scale* dikelompokan menjadi sejumlah critical ban menggunakan filter bank. Filter bank adalah salah satu bentuk filter yang dilakukan dengan tujuan untuk mengetahui ukuran energi dari *frequency band* tertentu dalam sinyal suara. Pada MFCC filter ditetapkan dalam domain frekuensi. Adapun untuk mengkonversi frekuensi ke *mel-scale* dan sebaliknya dapat menggunakan Persamaan (2.8) dan (2.9).

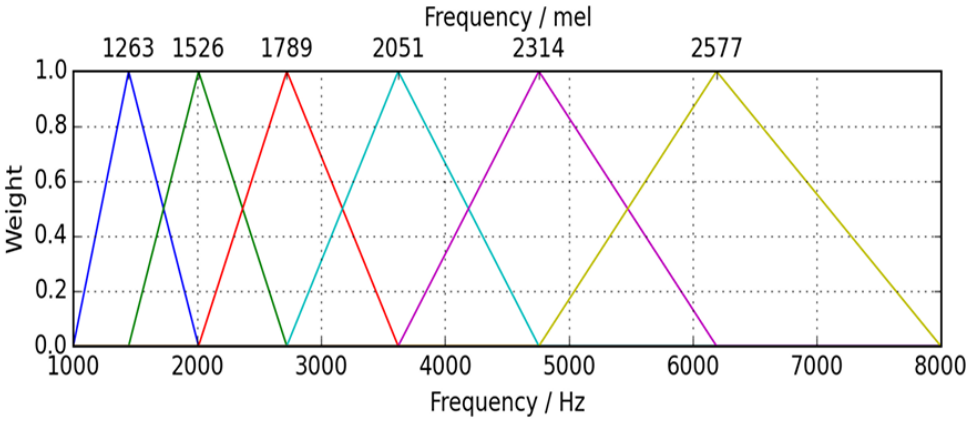
$$m = 2595 \log(1 + \frac{f}{700}) \tag{2.8}$$

$$f = 700(10^{\frac{m}{2595}} - 1) \tag{2.9}$$

Setelah memperoleh nilai mel dari frekuensi minimum dan maksimum, nilai-nilai tersebut kemudian digunakan untuk membagi spektrum menjadi sejumlah critical band melalui filter bank. *Mel bands* yang telah ditentukan sebelumnya menjadi dasar pembagian ini. Filter bank ini berisi serangkaian filter segitiga, yang dirancang untuk menangkap energi dari rentang frekuensi tertentu dalam sinyal suara. Filter segitiga ini memiliki tiga titik penting yaitu batas bawah (k_{m-1}), puncak (k_m), dan batas atas (k_{m+1}). Pembentukan masing-masing filter segitiga dilakukan dengan Persamaan (2.10).

$$H_m(k) = \begin{cases} 0, & k < k_{m-1} \\ \dfrac{k - k_{m-1}}{k_m - k_{m-1}}, & k_{m-1} \le k \le k_m \\ \dfrac{k_{m+1} - k}{k_{m+1} - k_m}, & k_m \le k \le k_{m+1} \\ 0, & k > k_{m+1} \end{cases} \tag{2.10}$$

Dimana $H_m(k)$ adalah filter segitiga untuk filter ke-*m*. Dengan demikian, frekuensi yang telah dipetakan ke dalam skala mel ini digunakan untuk mendistribusikan filter segitiga secara merata dalam bank filter mel. Proses ini memungkinkan setiap filter menangkap energi dari suatu band frekuensi tertentu, sehingga ukuran energi dari setiap *critical band* dapat diketahui. Dalam konteks MFCC, filter ini diterapkan pada domain frekuensi untuk memperoleh representasi yang sesuai dengan persepsi pendengaran manusia, di mana masing-masing filter segitiga mengintegrasikan nilai spektrum pada bin frekuensi yang telah dipetakan melalui persamaan di atas. Contoh dari bank filter mel ini dapat dilihat pada Gambar 2.11.



Gambar 2.11 *Mel Filter Bank*

Setelah mendapatkan bank filter Mel dengan menggunakan filter segitiga yang telah ditentukan berdasarkan titik-titik kritis langkah berikutnya adalah menghitung mel spektrogram. Proses ini dilakukan dengan mengalikan setiap filter pada bank tersebut dengan spektrum daya yang telah diperoleh dari Persamaan (2.7). Secara matematis, mel spektrogram dapat dihitung dengan Persamaan (2.11).

$$S(t,m) = \sum_{k=1}^{K} H_m(k).SD(t,k) \tag{2.11}$$

dimana,

$S(t,m)$	: mel spektogram indeks frame ke-*t* dan filter ke-*m*
$H_m(k)$	: filte rbank ke-*m* untuk frekuensi ke-*k* $(1 \leq m \leq M)$
m	: indeks koefesien mel filter bank
M	: jumlah channel dalam filter bank

6. *Discrete Cosine Transform* (DCT)

Discrete Cosine Transform (DCT) adalah langkah terakhir pada proses ekstrasi ciri MFCC. Konsep dasar dari DCT adalah mendekorelasi mel spectrum sehingga menghasilkan representasi yang baik dari properti spektral vokal (Patmadi, 2022). Adapun Persamaan (2.12) untuk menghitung DCT.

$$C(a,t) = \sum_{m=1}^{M} (\log_{10} S_{mel}(t,m) \cos[a\left(m - \frac{1}{2}\right)\frac{\pi}{M}]) \tag{2.12}$$

dimana,

$C(a,t)$	: koefisien MFCC ke-*a* dan frame ke-*t*
a	: indeks koefisien MFCC (a= 1,2, … , A)

2.6 *Audio Processing*

Audio Processing atau pengolahan audio adalah teknik pemrosesan yang memanfaatkan sinyal suara sebagai input dengan tujuan untuk memperbaiki kualitas sinyal tersebut atau mengekstrak informasi yang berguna untuk dijadikan dasar pengambilan keputusan. Salah satu operasi umum dalam pemrosesan sinyal audio adalah mengonversi sinyal suara menjadi

spektrogram, yaitu representasi dalam bentuk gambar dua dimensi. Dari gambar spektrogram ini, informasi penting dapat diolah lebih lanjut untuk mendapatkan hasil yang diinginkan.

Spektrogram merupakan salah satu representasi visual yang umum digunakan dalam pengolahan audio. Spektrogram menggambarkan perubahan energi atau intensitas frekuensi terhadap waktu dalam bentuk grafik dua dimensi, dengan sumbu horizontal menunjukkan waktu dan sumbu vertikal menunjukkan frekuensi. Intensitas energi pada frekuensi tertentu direpresentasikan dengan skala warna atau grayscale. Representasi ini mempermudah analisis pola suara secara visual dan sering digunakan sebagai input untuk model pembelajaran mesin berbasis *deep learning* (Putra, 2025). Dalam pengolahan audio modern, representasi seperti spektrogram atau Mel-spectrogram menjadi dasar penting untuk tugas-tugas seperti klasifikasi suara burung, pengenalan ucapan, dan analisis emosi dari suara (Ilmananda, 2012).

2.6.1 Tensorflow

TensorFlow adalah platform pembelajaran mesin yang dirancang untuk beroperasi dalam skala besar serta di lingkungan yang beragam. TensorFlow menggunakan grafik aliran data untuk merepresentasikan komputasi, status yang dibagikan, dan operasi yang mengubah status tersebut. TensorFlow mampu memetakan node dari grafik aliran data ke berbagai mesin dalam suatu *cluster*, dan juga memetakan ke berbagai perangkat komputasi dalam satu mesin, seperti *CPU multicore*, *GPU*, serta *ASIC* khusus yang disebut *Tensor Processing Unit (TPU)* (Abadi, *et al.*, 2016). Kemampuan ini menjadikan TensorFlow sangat fleksibel untuk menangani tugas-tugas pembelajaran mesin yang kompleks dan memproses data dalam jumlah besar dengan efisiensi tinggi.

Selain fleksibilitasnya, TensorFlow juga menawarkan skalabilitas yang tinggi, memungkinkan pelatihan model pada berbagai perangkat keras, mulai dari laptop hingga cluster komputasi besar. Ekosistemnya yang kuat mencakup berbagai alat dan pustaka pendukung seperti *TensorFlow Hub* dan *TensorFlow Extended*, yang mempercepat pengembangan model *deep learning*. TensorFlow digunakan secara luas dalam berbagai aplikasi, termasuk pengenalan gambar, pemrosesan bahasa alami (NLP), analisis data besar (*big data*), dan pengembangan sistem rekomendasi (Fahrizal, 2024). Dengan komunitas yang aktif dan dukungan ekstensif, TensorFlow terus menjadi salah satu *framework* utama untuk penelitian dan implementasi kecerdasan buatan di berbagai sektor (Fazry, 2024).

2.6.2 TF. Signal

TF. Signal adalah modul dari TensorFlow yang dirancang khusus untuk pemrosesan sinyal, termasuk audio. Modul ini menyediakan berbagai fungsi yang memungkinkan transformasi sinyal audio mentah menjadi representasi yang lebih informatif, seperti mel-spektrogram dan MFCC. Salah satu fungsi utamanya adalah 'tf.signal.stft', yang menerapkan *Short-Time Fourier Transform* (STFT) untuk mengubah sinyal audio dari domain waktu ke domain waktu-frekuensi. Representasi ini sangat berguna untuk analisis lebih lanjut, karena memungkinkan identifikasi pola frekuensi dalam potongan kecil sinyal audio. Selain itu, operator seperti 'tf.abs' dapat digunakan untuk menghilangkan fase sinyal dan hanya mempertahankan magnitudo, menghasilkan spektrogram yang lebih stabil untuk digunakan sebagai input ke model *deep learning* (Effendi & Saryadi, 2023).

Dalam konteks penelitian ini, modul tf.signal digunakan untuk menghasilkan mel-spektrogram dan menghitung MFCC. Perhitungan MFCC melibatkan penerapan *Discrete Cosine Transform* II (DCT-II) pada *log-magnitude* mel-spektrogram. MFCC adalah representasi fitur yang sangat efektif karena menskalakan spektrum frekuensi berdasarkan sensitivitas pendengaran manusia, sehingga lebih relevan untuk tugas pengenalan suara. Modul ini juga mengikuti konvensi penskalaan seperti pada HTK (*Hidden Markov Model Toolkit*), yang hampir menyerupai normalisasi ortogonal. Semua Mel bins dihitung, tetapi pengguna

dapat memilih subset tertentu sesuai kebutuhan aplikasi mereka. Sebagai contoh, dalam pengenalan suara, biasanya hanya beberapa koefisien MFCC pertama yang digunakan karena fitur ini memberikan representasi sinyal yang kurang lebih invarian terhadap nada (Tensorflow, 2024).

2.7 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah jenis arsitektur *deep learning* yang dirancang khusus untuk memproses data yang memiliki pola grid, seperti gambar dan data sejenis lainnya. CNN bekerja dengan menggunakan lapisan konvolusi yang mampu mengekstraksi fitur penting dari data masukan, seperti tepi, tekstur, atau bentuk dalam gambar, yang kemudian digunakan untuk melakukan klasifikasi atau deteksi objek. Lapisan konvolusi ini diikuti oleh lapisan *pooling* yang mengurangi dimensi data, dan lapisan *fully connected* yang menggabungkan fitur-fitur tersebut untuk membuat keputusan akhir. CNN telah terbukti sangat efektif dalam berbagai tugas seperti pengenalan wajah, klasifikasi gambar, dan bahkan pemrosesan sinyal suara, karena kemampuannya dalam menangkap pola-pola spasial dan lokal secara otomatis dari data (LeCun, Bottou, Bengio, & Haffner, 1998).

2.7.1 *Convolutional Layer*

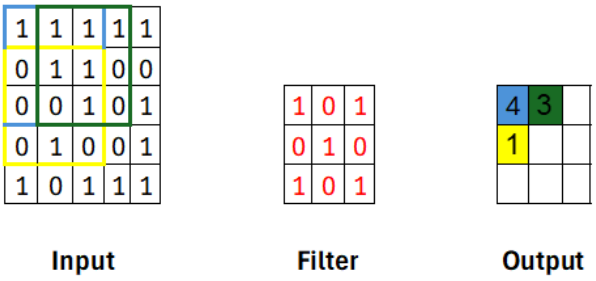
Convolutional layer adalah inti dari arsitektur *Convolutional Neural Network* (CNN) yang berfungsi untuk mengekstraksi fitur dari data masukan, seperti gambar atau audio. Pada layer ini, sebuah filter (atau kernel) kecil secara sistematis diterapkan pada data input untuk menghitung *dot* product antara filter dan bagian input yang dilaluinya. Proses ini menghasilkan sebuah peta fitur (*feature map*), yang mewakili berbagai karakteristik penting dari data seperti tepi, tekstur, atau pola tertentu. Filter tersebut bergerak melintasi data dengan langkah tertentu (*stride*), dan setiap filter dapat menangkap pola berbeda dari input. Konvolusi ini memungkinkan jaringan untuk mengenali fitur secara lokal dan menyusunnya menjadi representasi yang lebih kompleks di lapisan berikutnya. *Convolutional layer* sangat penting dalam deteksi pola spasial dan temporal dalam data (Goodfellow, Bengio, & Courville, 2016). *Convolutional Layer* dapat digambarkan dengan Persamaan (2.13).

$$c_{p^*,\,q^*}^{(layer)} = f_{ReLU}\left(B^{(layer)} + \sum_{i=1}^{I}\sum_{j=1}^{J} W_{i,j}^{(layer)} X_{p^*+i-1,\,q^*+j-1}^{(layer-1)}\right) \tag{2.13}$$

dimana,

$c_{p^*,\,q^*}^{(layer)}$	: output *convolutional layer* baris ke-p*, kolom ke-q*
$B^{(layer)}$	: nilai bias atau simpangan pada *feature map convolutional layer*
I	: panjang sisi kernel
J	: lebar sisi kernel
$W^{(layer)}$	: *weight* dari *convolution* kernel
$X_{p^*+i-1,\,q^*+j-1}^{(layer-1)}$	: input dari layer sebelumnya (input yang berupa mel-spektogram dari Persamaan (2.9) atau MFCC dari Persamaan (2.10))
p^*	: lebar *feature map*
q^*	: panjang *feature map*
f_{ReLU}	: fungsi aktivasi ReLU pada Persamaan (2.18)

selanjutnya, cara kerja lapisan konvolusi secara umum dapat diilustrasikan pada Gambar 2.12 berikut.



Gambar 2.12 Ilustrasi *Convolutional Layer*

2.7.2 *Pooling Layer*

Pooling adalah konsep penting lain dalam *Convolutional Neural Network* (CNN), yang berfungsi sebagai metode *non*-linear untuk melakukan *down-sampling* pada peta fitur (*feature map*) yang dihasilkan dari *convolutional layer*. Proses *pooling* bertujuan untuk mengurangi dimensi spasial dari data, sehingga mengurangi jumlah parameter dan komputasi dalam jaringan serta membantu menghindari *overfitting*. Ada beberapa jenis *pooling* yang dapat diterapkan, tetapi yang paling umum digunakan adalah *max-pooling*. Pada *max-pooling*, gambar atau peta fitur input dibagi menjadi beberapa sub-wilayah persegi panjang, dan dari setiap sub-wilayah tersebut, nilai maksimum yang dipilih menjadi output. Proses ini menghasilkan versi yang lebih kecil dari input dengan mempertahankan fitur-fitur yang paling penting. *Pooling layer* dengan metode *max-pooling* dapat digambarkan dengan Persamaan (2.14).

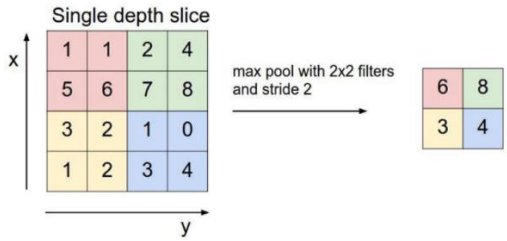
$$d_{p^*,\,q^*}^{(layer+1)} = \max_{i,j}\{c_{p^*+i-1,q^*+j-1}^{(layer)}\} \tag{2.14}$$

dimana,

$d_{p^*,\,q^*}^{(layer+1)}$: output pooling layer

$c_{p^*+i-1,q^*+j-1}^{(layer)}$: input dari layer sebelumnya

selanjutnya, cara kerja lapisan konvolusi secara umum dapat diilustrasikan pada Gambar 2.13.



Gambar 2.13 Ilustrasi pada *Max Pooling Layer* 2x2

Pooling layer beroperasi secara independen pada setiap irisan kedalaman dari input, yang berarti bahwa setiap channel atau fitur dari input diproses secara terpisah. Layer ini bertujuan untuk mengurangi dimensi spasial (lebar dan tinggi) dari data, tetapi tidak mempengaruhi dimensi kedalaman (*depth*). Jenis *pooling* yang paling umum adalah *max-pooling* dengan ukuran filter 2x2 dan *stride* 2, yang berarti filter ini melompati 2 unit pada setiap langkah. Proses ini mengecilkan ukuran input dengan memilih nilai maksimum dari setiap blok 2x2, sehingga menghilangkan 75% dari aktivasi (informasi yang tidak terlalu penting). Meskipun ukuran input secara spasial mengecil, jumlah *channel* (dimensi kedalaman) tetap tidak berubah,

yang memungkinkan model mempertahankan fitur penting sambil mengurangi kompleksitas komputasi (Md Anwar Hossain, *et al*., 2019).

2.7.3 Fully Connected Layer

Fully Connected Layer merupakan layer terakhir dalam arsitektur CNN, yang dapat terdiri dari beberapa lapisan. Output dari layer pooling akan diteruskan ke layer ini, dengan syarat bahwa data harus diubah menjadi format satu dimensi terlebih dahulu (Md Anwar Hossain, *et al.*, 2019). Sebelum data dari layer sebelumnya diproses oleh *layer fully connected*, output tersebut terlebih dahulu akan dipipihkan (*flatten*) menjadi vektor satu dimensi menggunakan Persamaan (2.15).

$$g = reshape(d_{p^*,\,q^*}^{(layer+1)}, (D,)) \tag{2.15}$$

dimana, D adalah vektor dengan dimensi p* x q* x 1 dan $d_{p^*,\,q^*}^{(layer+1)}$ merupakan output dari layer yang sebelumnya. Setelah dilakukan *flatten*, *layer fully connected* dapat dihitung menggunakan Persamaan (2.16).

$$\rho_r = \sum_{s=1}^{S} W_{r,s} g_s + B_r \tag{2.16}$$

dimana,

ρ_r	: nilai logit dari neuron ke-*r* pada fully connected layer
B_r	: nilai bias atau simpangan dari setiap neuron ke-*r* fully connected layer
$W_{r,s}$	: bobot fully connected layer untuk setiap neuron ke-*r* input ke-*s*
g_s	: input ke-s dari layer sebelumnya
s	: indeks input dari layer sebelumnya
r	: indeks neuron pada fully connected layer
S	: jumlah total neuron pada layer sebelumnya

2.8 Fungsi Aktivasi

Fungsi aktivasi adalah komponen penting dalam jaringan saraf tiruan (*neural network*) yang digunakan untuk memperkenalkan sifat non-linearitas ke dalam model. Fungsi ini menentukan bagaimana output dari suatu neuron diproses dan apakah neuron tersebut akan diaktifkan untuk memengaruhi proses berikutnya dalam jaringan. Tanpa fungsi aktivasi, jaringan saraf hanya akan merepresentasikan transformasi linier sederhana, yang membatasi kemampuan model untuk menangkap pola-pola kompleks dalam data. Fungsi aktivasi juga berperan dalam mengendalikan kontribusi neuron terhadap hasil akhir dengan mengatur nilai output ke dalam rentang tertentu, sehingga memungkinkan jaringan untuk belajar dan membuat prediksi lebih akurat. Pada penelitian ini, dua jenis fungsi aktivasi yang digunakan adalah ReLU dan *Softmax*.

2.8.1 *Rectified Linear Unit* (ReLU)

Rectified Linear Unit (ReLU) adalah fungsi aktivasi yang paling umum digunakan dalam jaringan saraf dalam model *deep learning*. Fungsi ini diakui karena kesederhanaannya dan kemampuannya dalam mengatasi masalah *vanishing gradient* yang sering terjadi pada

fungsi aktivasi lainnya, seperti *sigmoid* atau *tanh*. ReLU didefinisikan secara matematis pada Persamaan (2.17).

$$f_{ReLU}(Q) = \max(0, Q) \tag{2.17}$$

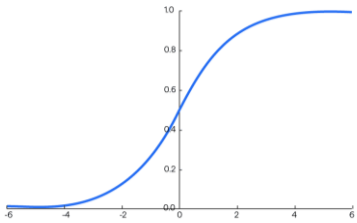
Artinya, jika input Q lebih besar dari nol, maka outputnya sama dengan Q. Sebaliknya, jika Q kurang dari atau sama dengan nol, maka outputnya adalah nol. Keunggulan ReLU terletak pada kemampuannya untuk mempercepat konvergensi jaringan saraf dan memberikan *sparsity* dalam representasi fitur, yang dapat meningkatkan efisiensi komputasi (Glorot, Bordes, & Bengio, 2011).

2.8.2 *Softmax Classifier*

Softmax Classifier adalah model dalam *machine learning* yang digunakan untuk klasifikasi multi-kelas. Model ini mengubah output dari layer sebelumnya menjadi distribusi probabilitas, di mana setiap nilai mewakili kemungkinan suatu input termasuk dalam masing-masing kelas. Fungsi *softmax* memastikan bahwa total probabilitas dari semua kelas adalah 1. Cara kerjanya dimulai dengan menerima vektor input, biasanya dari layer terakhir jaringan saraf (Ho & Wookey, 2020). Secara matematis fungsi softmax dapat dituliskan secara matematis pada Persamaan (2.18).

$$f_{softmax}(h) = \frac{e^{\rho_h}}{\sum_{r=1}^{R} e^{\rho_r}} \tag{2.18}$$

dimana, ρ_h adalah skor hasil lapisan *fully connected layer* dari neuron ke-h, ρ_r didapatkan dari Persamaan (2.16) dan $f_{softmax}(h)$ merupakan probabilitas output untuk kelas ke-h.



Gambar 2.14 Fungsi Softmax

Ilustrasi fungsi *softmax* dapat dilihat pada Gambar 2.14. Pada layer *softmax* dihitung nilai *loss* atau *error* untuk mengetahui perbedaaan antara nilai prediksi dengan label yang sebenarnya. Pada penelitian ini, karena dilakukan klasifikasi dengan kategori lebih dari dua, maka digunakan *categorical entropy loss function*. Penghitungan *categorical entropy loss* dilakukan menggunakan Persamaan (2.19).

$$L_{\mathrm{a}} = -\frac{1}{N}\sum_{h=1}^{H}\sum_{n=1}^{N} y_{n,h}\log(f_{softmax}(n,h)) \tag{2.19}$$

dimana,

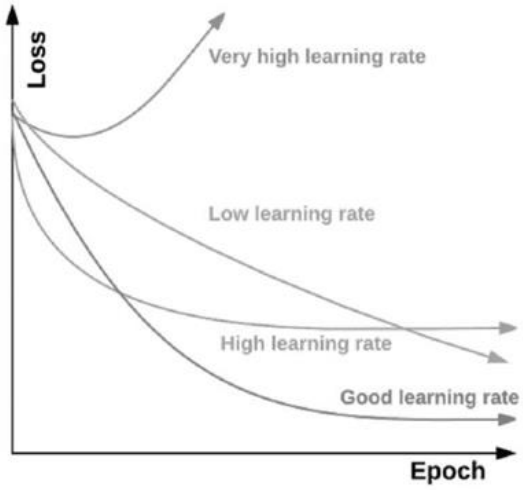
L_{a} : *loss function* untuk iterasi ke-a

N	: jumlah seluruh sampel pada dataset
$y_{n,h}$	: label asli untuk sampel ke-n dan kelas ke-h
$f_{softmax}(n,h)$	: nilai probabilitas prediksi sampel ke-n dan kelas ke-h dari Persamaan (2.18)
h	: indeks kelas
H	: banyak kelas atau kategori respon
n	: indeks rekaman suara
a	: (*timestep*) iterasi

2.9 *Learning Rate*

Learning rate adalah salah satu *hyperparameter* penting dalam proses pelatihan model *deep learning* yang berfungsi mengatur seberapa besar perubahan bobot model berdasarkan estimasi *error* selama pembaruan. Nilai *learning rate* umumnya berada dalam rentang 0 hingga 1 dan memengaruhi kecepatan serta arah konvergensi model. Jika *learning rate* terlalu kecil, perubahan bobot yang terjadi pada setiap iterasi akan sangat lambat, sehingga memerlukan jumlah *epoch* yang lebih banyak untuk mencapai performa optimal. Sebaliknya, jika *learning rate* terlalu besar, model akan melakukan perubahan signifikan pada setiap pembaruan bobot. Hal ini dapat mempercepat proses pelatihan tetapi berisiko model melewati solusi optimal atau berhenti di solusi suboptimal.

Pemilihan nilai *learning rate* yang tepat sangat penting untuk memastikan proses pelatihan berjalan efektif dan efisien. Nilai yang terlalu besar dapat menyebabkan model sulit mencapai konvergensi, sementara nilai yang terlalu kecil dapat membuat pelatihan stagnan atau membutuhkan waktu yang sangat lama. Oleh karena itu, eksperimen dan penyesuaian terhadap nilai *learning rate* sering dilakukan untuk mendapatkan performa model terbaik (Alzubaidi, *et al.*, 2021). Gambar 2.15 menunjukkan pengaruh beberapa nilai *learning rate* terhadap nilai loss selama proses pelatihan model.



Gambar 2.15 Efek Nilai *Learning Rate* terhadap Performa Model

2.10 *Adam Optimizer*

Adam (*Adaptive Moment Estimation*) *optimizer* adalah metode optimasi berbasis *stochastic gradient descent* yang dirancang untuk meminimalkan nilai *loss* dan dapat diterapkan pada berbagai kasus *deep learning*, termasuk pemrosesan sinyal audio. Dalam konteks

penelitian suara, *Adam* sangat cocok karena kemampuannya untuk menangani data dengan pola kompleks seperti rekaman audio. Proses optimasi menggunakan Adam dimulai dengan inisialisasi dua parameter utama, yaitu β_1 dan β_2 yang mewakili *exponential decay rate*. Biasanya β_1 diinialisasi dengan nilai 0,9 dan β_2 dengan nilai 0,999. *Exponential decay rate* adalah proses pengurangan nilai dengan persentase tetap secara konsisten dari waktu ke waktu. Selain itu, nilai awal *learning rate* diinisialisasi, misalnya α = 0.0001. Selanjutnya, ditentukan *stochastic objective function* dengan parameter θ yang dinotasikan sebagai $f(\theta)$. Kemudian, dilakukan inisialisasi momen pertama (m_0) dan momen kedua (v_0) serta *timestep* awal (a) dengan nilai nol. Setelah itu, dilakukan proses *forward* untuk setiap layer dalam arsitektur yang digunakan, seperti CNN atau VGG16. Pada tahap berikutnya, *gradient loss function* (g_a) dihitung berdasarkan Persamaan (2.20).

$$\mathrm{g_a} = \nabla_\theta L_\mathrm{a}(\theta_\mathrm{a}) = \frac{\partial L_\mathrm{a}}{\partial \theta_\mathrm{a}} \tag{2.20}$$

Dimana L_a merupakan *loss function* yang dijabarkan pada persamaan (2.18). Langkah selanjutnya yaitu melakukan update bias estimasi momen pertama (m_a) dan estimasi momen kedua (v_a) menggunakan Persamaan (2.21) dan Persamaan (2.22).

$$m_\mathrm{a} = \beta_1 m_{\mathrm{a}-1} + (1-\beta_1)\mathrm{g_a} \tag{2.21}$$
$$v_\mathrm{a} = \beta_2 v_{\mathrm{a}-1} + (1-\beta_2)\mathrm{g_a^2} \tag{2.22}$$

Kemudian dihitung estimasi koreksi bias momen pertama $(\widehat{m}_\mathrm{a})$ dan momen $(\hat{v}_\mathrm{a})$ menggunakan Persamaan (2.23) dan Persamaan (2.24).

$$\widehat{m}_\mathrm{a} = \frac{m_\mathrm{a}}{(1-\beta_1^\mathrm{a})} \tag{2.23}$$
$$\hat{v}_\mathrm{a} = \frac{v_\mathrm{a}}{(1-\beta_2^\mathrm{a})} \tag{2.24}$$

β_1^a dan β_2^a dilakukan *update* sesuai dengan nilai *timestep* yaitu a. Semakin tinggi a, maka nilai β_1^a dan β_2^a akan semakin mendekati 0 sehingga estimasi koreksi bias semakin mendekati 1. Selanjutnya, dilakukan *update* parameter dengan Persamaan (2.25).

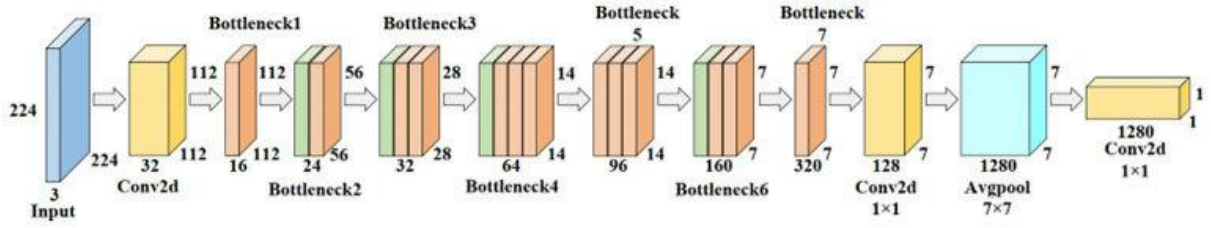
$$\theta_{\mathrm{a}+1} = \theta_\mathrm{a} - a\frac{\widehat{m}_\mathrm{a}}{\sqrt{\hat{v}_\mathrm{a}} + \varepsilon} \tag{2.25}$$

Dimana ε adalah konstanta kecil (misalnya 10^{-8}) untuk mencegah pembagian dengan nol. Iterasi dilakukan hingga didapatkan parameter bias maupun weight yang konvergen. Dengan langkah-langkah ini, Adam memungkinkan model untuk mencapai konvergensi yang lebih stabil dan efisien, khususnya dalam mengklasifikasikan suara berdasarkan data rekaman audio.

2.11 MobileNet

MobileNet merupakan salah satu arsitektur Convolutional Neural Network (CNN) yang dikembangkan oleh tim Google pada tahun 2017 dengan tujuan utama untuk menghasilkan model yang efisien dan ringan sehingga dapat digunakan pada perangkat mobile dan embedded.

Keunggulan utama MobileNet terletak pada penggunaan depthwise separable convolution, yaitu teknik yang memisahkan proses filtering dan penggabungan fitur sehingga jumlah parameter dan kebutuhan komputasi menjadi jauh lebih rendah dibandingkan arsitektur CNN konvensional. Dalam konteks transfer learning, MobileNet yang telah dilatih pada dataset besar seperti ImageNet dapat dimanfaatkan kembali untuk tugas klasifikasi spesifik dengan dataset yang lebih kecil. Proses transfer learning dilakukan dengan memanfaatkan bobot-bobot hasil pelatihan awal sebagai feature extractor, kemudian mengganti dan melatih ulang lapisan akhir sesuai jumlah kelas pada tugas baru, seperti klasifikasi suara burung berbasis citra mel-spektrogram dan MFCC. Arsitektur MobileNet ditunjukkan pada Gambar 2.16.

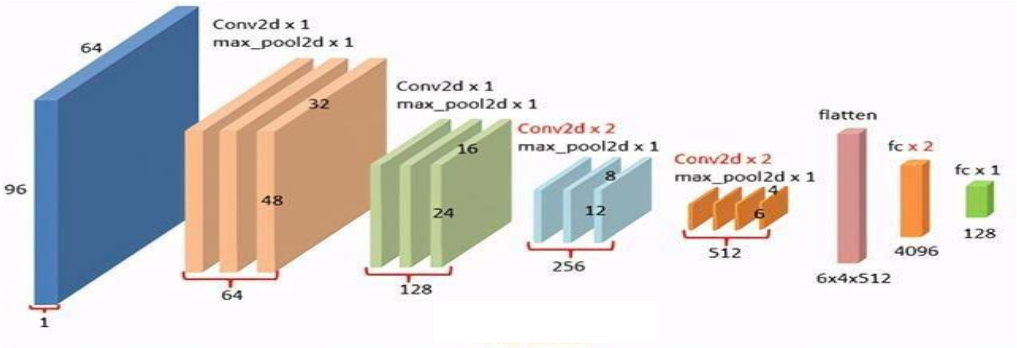


Gambar 2. 16 Arsitektur MobileNet

Pendekatan ini tidak hanya mempercepat proses pelatihan dan mengurangi kebutuhan data, tetapi juga mampu meningkatkan akurasi pada domain baru karena model telah memiliki pengetahuan awal dari dataset besar. MobileNet terbukti efektif untuk transfer learning pada berbagai aplikasi, terutama yang membutuhkan efisiensi komputasi tanpa mengorbankan akurasi, seperti pada penelitian klasifikasi benih padi yang berhasil mencapai akurasi tinggi dengan memanfaatkan transfer learning MobileNet (Howard, et al., 2017).

2.12 VGGish

VGGish adalah model deep learning yang dikembangkan oleh Google pada tahun 2017 sebagai adaptasi dari arsitektur VGG untuk ekstraksi fitur audio yang efektif. Model ini dilatih pada dataset AudioSet yang sangat besar dan beragam dari menggunakan data dari YouTube-8M, sehingga mampu menghasilkan representasi fitur audio yang kuat dan generalisasi baik untuk berbagai tugas pemrosesan suara. Input awal berupa waveform audio diubah terlebih dahulu menjadi Mel-spectrogram, yaitu representasi visual dari sinyal audio dalam domain frekuensi, dengan durasi frame sekitar 0,96 detik. Mel-spectrogram ini kemudian diproses oleh jaringan konvolusi VGGish yang memiliki struktur mirip VGG dengan lapisan konvolusi 3×3 dan pooling 2×2, menghasilkan embedding berdimensi 128 yang merepresentasikan fitur audio tingkat tinggi. Embedding ini dapat digunakan sebagai input untuk model klasifikasi atau deteksi lebih lanjut, sehingga memudahkan penerapan transfer learning pada berbagai aplikasi audio, termasuk pengenalan suara burung, musik, atau suara lingkungan. Arsitektur Model VGGish dapat dilihat pada Gambar 2.17.



Gambar 2.17 Arsitektur VGGish

Dengan menggunakan VGGish, proses pelatihan menjadi lebih efisien karena model sudah memiliki pengetahuan awal dari dataset besar, sekaligus meningkatkan akurasi pada dataset yang lebih kecil dan spesifik. Implementasi VGGish telah didukung oleh berbagai framework seperti TensorFlow dan PyTorch, yang menyediakan pipeline lengkap mulai dari ekstraksi fitur hingga fine-tuning model. Pendekatan ini telah terbukti efektif dalam berbagai penelitian klasifikasi audio dengan performa tinggi (Hershey, et al., 2017).

2.13 Evaluasi Performa Model

Evaluasi performa model adalah proses penting dalam pengembangan model machine learning untuk menilai seberapa baik model dapat memprediksi atau mengklasifikasikan data baru. Tujuan utama evaluasi ini adalah memastikan bahwa model tidak hanya bekerja dengan baik pada data pelatihan tetapi juga mampu melakukan generalisasi pada data yang belum pernah dilihat sebelumnya. Berikut merupakan beberapa metrik evaluasi yang sering digunakan dalam klasifikasi.

a. *Confusion Matrix*

Kinerja suatu algoritma klasifikasi perlu diukur dan dievaluasi untuk menentukan apakah algoritma klasifikasi tersebut dapat melakukan klasifikasi dengan baik atau tidak. Salah satu metode yang dapat digunakan untuk evaluasi performa dari algoritma klasifikasi yaitu *Confusion Matrix*. Prinsip dari *Confusion Matrix* yaitu membendingkan hasil klasifikasi dari algoritma dengan klasifikasi yang sebenarnya. Hasil klasifikasi yang diukur dengan *Confusion Matrix* dibedakan menjadi empat jenis yaitu *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN). Berdasarkan keempat jenis klasifikasi tersebut, dapat dihitung tiga pengukuran ketepatan klasifikasi yaitu presisi, akurasi, dan recall (Ali, Neagu, & Trundle, 2019). *Confusion matrix* untuk *multiclass classification* diilustrasikan pada Gambar 2.18. Pada Gambar 2.19 ditampilkan ilustrasi confusion matrix untuk kelas ke-1.

	Prediksi Kelas 1	**Prediksi Kelas 2**	...	**Prediksi Kelas N**
Aktual Kelas 1	$C_{1,1}$	$C_{1,2}$	...	$C_{1,N}$
Aktual Kelas 2	$C_{2,1}$	$C_{2,2}$	...	$C_{2,N}$
⋮	⋮	⋮	⋱	⋮
Aktual Kelas N	$C_{N,1}$	$C_{N,2}$	...	$C_{N,N}$

Gambar 2.18 *Confussion Matrix Multiclass Classification*

Keterangan:

- $C_{i,i}$ adalah jumlah data yang benar diklasifikasikan untuk kelas i (*True Positive* untuk kelas i).
- $C_{i,j}$ dengan $i \neq j$ adalah jumlah data dari kelas i yang salah diklasifikasikan sebagai kelas j (*False Negative* untuk kelas i, dan *False Positive* untuk kelas j).
- *True Negative* untuk kelas i yaitu jumlah data yang bukan kelas i dan tidak diprediksi sebagai kelas i.

	Prediksi Kelas 1	**Prediksi Kelas 2**	...	**Prediksi Kelas N**
Aktual Kelas 1	**TP**	**FN**		
Aktual Kelas 2	**FP**	**TN**		
⋮				
Aktual Kelas N				

Gambar 2.19 Ilustrasi *confusion matrix* untuk kelas ke-1

b. *Precision*

Precision adalah metrik yang digunakan untuk mengevaluasi kinerja model dalam konteks klasifikasi, terutama dalam pengenalan pola dan analisis data. *Precision* mengukur proporsi dari prediksi positif yang benar di antara semua prediksi positif yang dibuat oleh model. Secara matematis, *precision* untuk kelas ke-*i* didefinisikan pada Persamaan (2.25).

$$Precision_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i} = \frac{C_{i,i}}{\sum_{j=1}^{N} C_{j,i}} \tag{2.25}$$

Di mana *True Positives* (TP) adalah jumlah prediksi positif yang benar, dan *False Positives* (FP) adalah jumlah prediksi positif yang salah. *Precision* memberikan gambaran tentang keakuratan prediksi positif dan sangat berguna ketika biaya kesalahan positif tinggi, seperti dalam diagnosis medis atau deteksi penipuan (Confusion matrix in machine learning, 2021).

c. *Recall*

Recall adalah metrik yang digunakan untuk mengukur kemampuan model dalam mengidentifikasi semua *instance* positif dari suatu kelas dalam konteks klasifikasi. Dalam istilah sederhana, recall mengukur proporsi prediksi positif yang benar dari seluruh data positif yang ada. Secara matematis, *recall* untuk kelas ke-*i* didefinisikan pada Persamaan (2.26).

$$Recall_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i} = \frac{C_{i,i}}{\sum_{j=1}^{N} C_{i,j}} \tag{2.26}$$

Di mana *True Positives* (TP) adalah jumlah prediksi positif yang benar, dan *False Negatives* (FN) adalah jumlah *instance* positif yang salah diprediksi sebagai negatif. *Recall* memberikan wawasan tentang seberapa baik model dalam menangkap semua contoh yang relevan dan sangat berguna dalam situasi di mana kehilangan *instance* positif (*false negative*) memiliki konsekuensi yang serius, seperti dalam diagnosis penyakit atau deteksi penipuan. Dalam konteks klasifikasi yang tidak seimbang, di mana satu kelas lebih banyak daripada kelas lainnya, *recall* menjadi penting untuk dipertimbangkan bersama dengan *precision*. *Precision* mengukur akurasi prediksi positif, sedangkan *recall* mengukur kemampuan model untuk menemukan semua *instance* positif. Untuk memberikan gambaran yang lebih komprehensif tentang kinerja model, kedua metrik ini sering digunakan bersama untuk menghitung F1 Score, yang merupakan rata-rata harmonis dari *precision* dan *recall* (Confusion matrix in machine learning, 2021).

d. *F1-score*

F1-score adalah *harmonic mean* dari *precision* dan *recall*. F1-*score* terbaik adalah jika ada semacam keseimbangan antara *precision* dan *recall* dalam sistem. Sebaliknya Skor F1 tidak begitu tinggi jika satu ukuran ditingkatkan dengan mengorbankan yang lain. *F1-score* untuk kelas ke-*i* dapat dihitung dengan Persamaan (2.27) yang harus menghitung dulu *precison* dan *recall* pada Persamaan (2.25) dan Persamaan (2.26).

$$F1 - score_i = \frac{2(Precision_i \times Recall_i)}{Precision_i + Recall_i} \tag{2.27}$$

e. *Accuracy*

Accuracy (akurasi) adalah ukuran untuk berapa banyak prediksi yang benar yang dibuat model *machine learning* untuk set data pengujian yang lengkap. *Accuracy* adalah metrik dasar yang baik untuk mengukur kinerja model (Powers & Ailab, 2020). *Accuracy* untuk kelas ke-*i* dapat dihitung dengan Persamaan (2.28).

$$Accuracy = \frac{\mathrm{TP}_i + \mathrm{TN}_i}{\mathrm{TP}_i + \mathrm{FP}_i + \mathrm{TN}_i + \mathrm{FN}_i} = \frac{\sum_{i=1}^{N} C_{i,i}}{\sum_{i=1}^{N}\sum_{j=1}^{N} C_{i,j}} \tag{2.28}$$

Metrik ini sering digunakan karena kemudahannya dalam interpretasi, tetapi dapat memberikan gambaran yang menyesatkan, terutama dalam dataset yang tidak seimbang, di mana salah satu kelas lebih dominan. Dalam kasus seperti itu, meskipun akurasi tinggi, model mungkin tidak berkinerja baik dalam mengidentifikasi kelas minoritas. Oleh karena itu, penting untuk mempertimbangkan metrik lain seperti *precision, recall,* dan *F1-score* bersamaan dengan akurasi untuk mendapatkan evaluasi yang lebih komprehensif tentang kinerja model.

2.14 Hasil Penelitian Terdahulu

Pada penelitian ini, penulis merujuk pada beberapa penelitian terdahulu yang berkaitan dengan klasifikasi suara burung menggunakan metode Convolutional Neural Networks (CNN) dan arsitektur VGG16. Penelitian terdahulu ini bertujuan untuk memberikan bahan perbandingan dan acuan bagi penelitian yang dilakukan. Hasil-hasil penelitian terdahulu penulis cantumkan dalam Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu Terkait Klasifikasi Suara Burung

No.	Peneliti/Tahun	Metode	Dataset	Hasil
1	(Irwandi, *et al.*, 2005)	Analisis Spektrogram	Dataset suara burung cendet, suren, sigeundet, murai daun, dan kutilang emas.	Metode spektrogram berhasil digunakan untuk menyusun klasifikasi suara burung.Tetapi tidak ada laporan akurasi atau uji performa model.
2	(Putra, 2019)	MFCC dan DTW	Dataset suara lovebird terbagi atas tiga kelas berdasarkan patahan suara.	Metode ini mencapai akurasi 80% untuk klasifikasi suara lovebird. Tidak ada pengujian pada dataset burung lain sehingga belum teruji di luar lovebird.
3	(Ali, 2020)	Spektogram dan CNN	Dataset suara burung trucukan, sirpu, cendet, dan kenari.	Akurasi mencapai 99% dengan akurasi testing data sebesar 92,5% untuk klasifikasi spesies burung lokal di Indonesia. Namun durasinya hanya 1 detik, yang dapat membatasi variasi data input.
4	(Ihsanti, 2024)	CNN	Dataset suara burung hantu berdasarkan genus.	Mencapai akurasi 99.8% dalam klasifikasi suara burung hantu tetapi mendapat precision dan recall yang rendah (16%) akibat dataset tidak seimbang.

Berdasarkan Tabel 2.1, telah ada penelitian dengan berbagai pendekatan yang digunakan untuk klasifikasi suara burung. Irwandi *et al.* (2005) menggunakan analisis spektrogram untuk identifikasi lima spesies burung, namun metode ini kurang efisien untuk dataset kompleks. Putra (2019) mengaplikasikan MFCC dan DTW dengan akurasi 80%, tetapi kurang kompetitif dibandingkan pendekatan deep learning. Ali (2020) memanfaatkan kombinasi spektrogram dan CNN untuk klasifikasi suara burung lokal di Indonesia dengan akurasi mencapai 92,5%. Namun, durasi data suara yang dibatasi hanya 1 detik menjadi kendala dalam menangkap variasi pola suara yang lebih kompleks. Ihsanti dan Al Maki (2024) menggunakan CNN dengan akurasi tinggi 99.8%, namun precision dan recall yang rendah sebesesar 16% disebabkan oleh dataset yang tidak seimbang. Penelitian ini bertujuan untuk menggabungkan Mel-spektrogram dan MFCC sebagai metode ekstraksi fitur dengan CNN dan VGG16 sebagai model klasifikasi. Pendekatan ini diharapkan mampu menghasilkan akurasi yang tinggi dalam klasifikasi suara burung sekaligus mengatasi keterbatasan yang ditemukan pada penelitian terdahulu, seperti generalisasi model dan ketidakseimbangan data. Dengan demikian, penelitian ini dapat memberikan kontribusi signifikan dalam pengembangan teknologi bioakustik berbasis kecerdasan buatan serta pelestarian keanekaragaman hayati.

BAB 3 METODOLOGI

3.1 Sumber Data

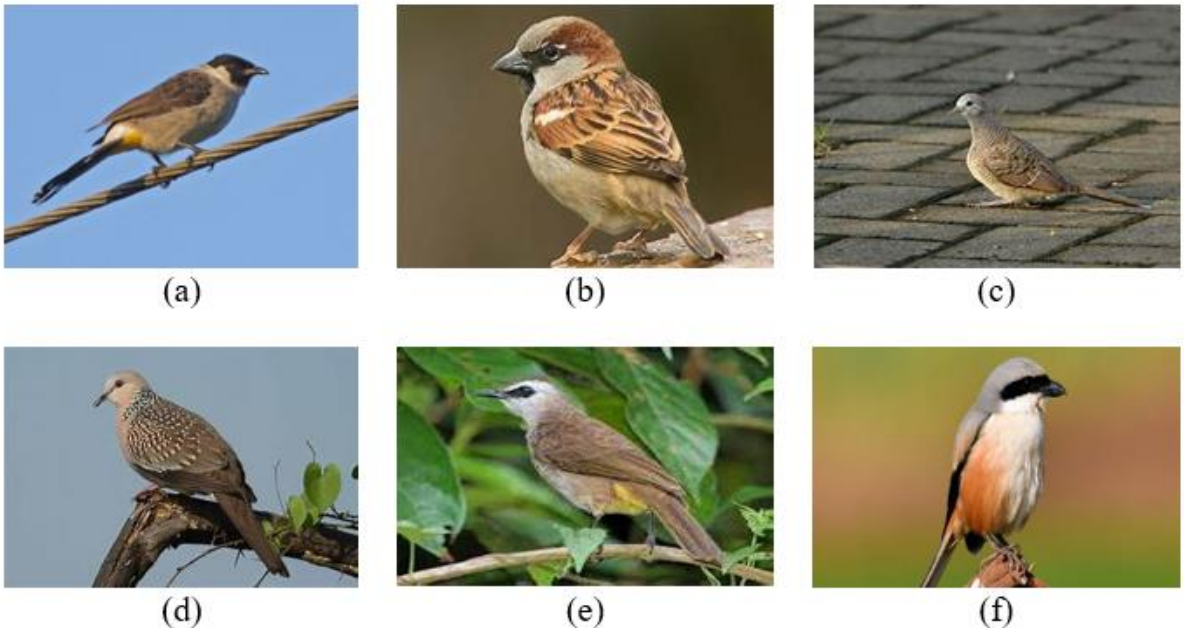
Data yang digunakan pada penelitian ini merupakan data sekunder berupa rekaman suara burung yang diperoleh dari situs https://xeno-canto.org. Dataset ini terdiri dari total 60 rekaman suara dengan 6 jenis spesies burung yang dipilih (10 rekaman per spesies) yang berdurasi antara 11 – 115 detik. Pemilihan spesies ini didasarkan pada karakteristik suara yang unik dari masing-masing burung, yang bertujuan untuk membangun model klasifikasi suara burung. Dataset ini digunakan untuk *training* dan *validation* pada model klasifikasi guna mengenali jenis burung berdasarkan ciri khas suaranya.

3.2 Variabel Penelitian

Variabel penelitian yang digunakan dalam penelitian ini adalah rekaman suara dari beberapa spesies burung di Indonesia. Tabel 3.1 menampilkan jenis burung yang digunakan sebagai variabel dalam penelitian ini.

Tabel 3.1 Jenis Spesies Burung

No	Jenis Burung
1	Burung Kutilang (*Pycnonotus aurigaster*)
2	Burung Gereja (*Passer domesticus*)
3	Burung Perkutut (*Geopelia striata*)
4	Burung Tekukur (*Streptopelia chinensis*)
5	Burung Trucukan (*Pycnonotus goiavier*)
6	Burung Cendet (*Lanius cristatus*)



Gambar 3.1 Spesies Burung yang Digunakan dalam Penelitian

Pada Gambar 3.1 menunjukkan ilustrasi dari keempat jenis burung yang digunakan dalam penelitian ini. Pada Gambar 3.1 (a), terlihat Burung Kutilang, yang memiliki ciri khas bulu

cokelat keabu-abuan dengan ekor hitam dan ujung kuning. Gambar 3.1 (b) menampilkan Burung Gereja, yang memiliki tubuh kecil dengan bulu cokelat bercorak hitam di punggung dan dada putih. Selanjutnya, Gambar 3.1 (c) menunjukkan Burung Perkutut, yang memiliki pola garis-garis pada bulunya dan sering ditemukan di area pedesaan. Pada Gambar 3.1 (d), terlihat Burung Tekukur, yang memiliki tubuh berukuran sedang dengan bulu cokelat keabu-abuan dan pola sisik di lehernya. Gambar 3.1 (e) menampilkan Burung Trucukan, yang memiliki warna bulu cokelat dengan dada putih dan garis kuning di bagian bawah tubuhnya. Terakhir, Gambar 3.1 (f) menunjukkan Burung Cendet (*Lanius cristatus*), yang memiliki tubuh ramping dengan warna bulu abu-abu, hitam, dan putih, serta paruh melengkung khas burung predator kecil.

Berikut ini adalah variabel penelitian yang digunakan dalam klasifikasi suara spesies burung, seperti yang tercantum pada Tabel 3.2.

Tabel 3.2 Variabel Penelitian

Variabel	Keterangan	Skala
$X_{n,t}$	Nilai amplitudo suara n pada waktu ke-t	Interval
Y_{sp}	Kategori spesies burung (1=Burung Kutilang, 2=Burung Gereja, 3=Burung Perkutut, 4=Burung Tekukur, 5=Burung Trucukan, 6=Burung Cendet).	Nominal

3.3 Struktur Data

Struktur data yang digunakan pada penelitian ini dapat terbagi menjadi 2, yaitu struktur data secara general dan struktur data secara spesifik yang akan digunakan pada proses training model. Struktur data secara general ditampilkan pada Tabel 3.3, sedangkan struktur data yang lebih spesifik tentang data suara yang akan diproses ditampilkan pada Tabel 3.4.

Tabel 3.3 Struktur Data secara General

No.	Suara (X)	Spesies Burung (Y)
1	Suara_1.wav	Y_1
2	Suara_2.wav	Y_1
⋮	⋮	⋮
N	Suara_N.wav	Y_{sp}

Tabel 3.4 Struktur Data Suara yang akan Diproses

Data	X_1	X_2	…	X_T	Y
1	$X_{1,1}$	$X_{1,2}$	…	$X_{1,T1}$	Y_1
2	$X_{2,1}$	$X_{2,2}$	…	$X_{2,T2}$	Y_2
	⋮	⋮	⋱	⋮	⋮
N	$X_{N,1}$	$X_{N,2}$	…	$X_{N,TN}$	Y_{sp}

Sementara untuk struktur data yang lebih detail dan yang akan digunakan untuk melakukan proses klasifikasi yaitu menggunakan fitur mel-spektogram dan MFCC yang telah diekstraksi dari data suara. Karena pada penelitian ini digunakan data suara yang diolah menjadi fitur MFCC (40 koefisien per frame) untuk klasifikasi, maka didapatkan struktur data pada Tabel 3.5.

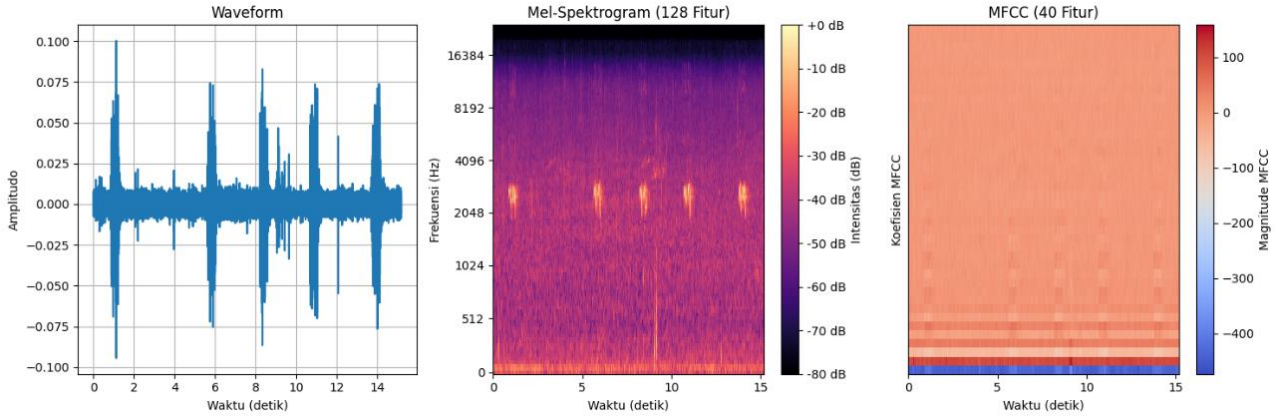
Tabel 3.5. Struktur Data Fitur MFCC yang akan Diproses

Data	*Frame*	C_1	C_2	…	C_{40}	Y
1	1	$C_{1,1,1}$	$C_{1,1,2}$	…	$C_{1,1,40}$	Y_1
	2	$C_{1,2,1}$	$C_{1,2,2}$	…	$C_{1,2,40}$	
	⋮	⋮	⋮	⋱	⋮	
	T_1	$C_{1,T1,1}$	$C_{1,T1,2}$	…	$C_{1,T1,40}$	
2	1	$C_{2,1,1}$	$C_{2,1,2}$	…	$C_{2,1,40}$	Y_1
	2	$C_{2,2,1}$	$C_{2,2,2}$	…	$C_{2,2,40}$	
	⋮	⋮	⋮	⋱	⋮	
	T_2	$C_{2,T2,1}$	$C_{2,T2,2}$	…	$C_{2,T2,40}$	
⋮	⋮	⋮	⋮	⋱	⋮	⋮
N	1	$C_{N,1,1}$	$C_{N,1,2}$	…	$C_{N,1,40}$	Y_{sp}
	2	$C_{N,2,1}$	$C_{N,2,2}$	…	$C_{N,2,40}$	
	⋮	⋮	⋮	⋱	⋮	
	T_N	$C_{N,TN,1}$	$C_{N,TN,2}$	…	$C_{N,TN,40}$	

Sementara untuk struktur data mel-spektogram (128 koefisien per frame) lebih detail didapatkan pada Tabel 3.6.

Tabel 3.6. Struktur Data Fitur Mel-Spektogram yang akan Diproses

Data	*Frame*	S_1	S_2	…	S_{128}	Y
1	1	$S_{1,1,1}$	$S_{1,1,2}$	…	$S_{1,1,128}$	Y_1
	2	$S_{1,2,1}$	$S_{1,2,2}$	…	$S_{1,2,128}$	
	⋮	⋮	⋮	⋱	⋮	
	T_1	$S_{1,T1,1}$	$S_{1,T1,2}$	…	$S_{1,T1,128}$	
2	1	$S_{2,1,1}$	$S_{2,1,2}$	…	$S_{2,1,128}$	Y_1
	2	$S_{2,2,1}$	$S_{2,2,2}$	…	$S_{2,2,128}$	
	⋮	⋮	⋮	⋱	⋮	
	T_2	$S_{2,T2,1}$	$S_{2,T2,2}$	…	$S_{2,T2,128}$	
⋮	⋮	⋮	⋮	⋱	⋮	⋮
N	1	$S_{N,1,1}$	$S_{N,1,2}$	…	$S_{N,1,128}$	Y_{sp}
	2	$S_{N,2,1}$	$S_{N,2,2}$	…	$S_{N,2,128}$	
	⋮	⋮	⋮	⋱	⋮	
	T_N	$S_{N,TN,1}$	$S_{N,TN,2}$	…	$S_{N,TN,128}$	



Gambar 3.2 Contoh Representasi Visual dari Rekaman Suara

Pada Gambar 3.2 menampilkan contoh representasi visual dari rekaman suara. Pada Gambar 3.2 (a) menampilkan *waveform* dari rekaman suara. *Waveform* merepresentasikan amplitudo sinyal audio terhadap waktu. Gambar 3.2 (b) menampilkan mel-spektrogram dengan 128 fitur yang diperoleh dengan menerapkan Persamaan (2.9) pada sinyal audio. Mel-spektrogram adalah representasi energi frekuensi dalam skala mel terhadap waktu. Pada mel-spektrogram, intensitas warna menunjukkan besarnya energi (dB) pada frekuensi tertentu pada waktu tertentu. Warna yang lebih cerah (putih) menunjukkan energi yang lebih tinggi, sedangkan warna yang lebih gelap (hitam) menunjukkan energi yang lebih rendah. Gambar 3.2 (c) menampilkan MFCC dengan 40 fitur yang yang diperoleh dengan menerapkan Persamaan (2.10) pada sinyal audio. MFCC adalah representasi numerik dari karakteristik akustik suara yang diperoleh melalui transformasi mel-spektrogram menggunakan DCT. Pada diagram MFCC, intensitas warna menunjukkan nilai koefisien MFCC. Warna merah mewakili nilai positif, sedangkan warna biru mewakili nilai negatif dan intensitas warna menunjukkan besarnya nilai koefisien tersebut.

3.4 Langkah Analisis

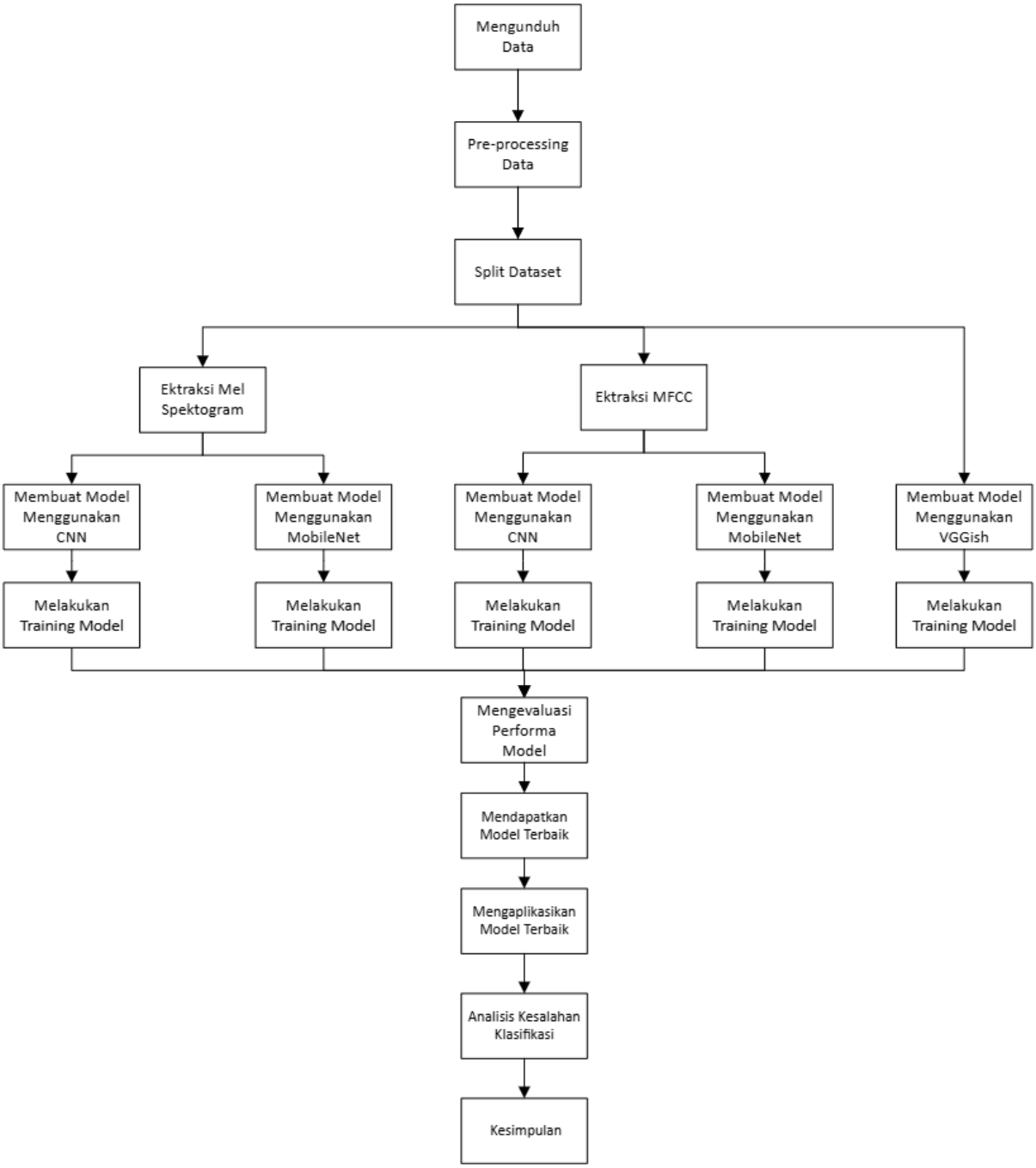
Langkah analisis yang dilakukan dalam penelitian ini untuk melakukan klasifikasi suara burung adalah sebagai berikut:

1. Mempersiapkan Dataset: Dataset yang digunakan terdiri dari rekaman suara burung dari situs https://xeno-canto.org.
2. Data Preprocessing:
   a. Melakukan pemrosesan awal pada rekaman suara, termasuk mengubah format suara jika diperlukan.
   b. Melakukan *trimming* pada rekaman suara untuk memotong bagian yang tidak relevan (jeda atau *noise*) dan membagi rekaman panjang menjadi beberapa segmen pendek (1-3 detik tergantung spesies burung) yang hanya berisi suara burung. Proses *trimming* dilakukan menggunakan metode deteksi aktivitas suara (*Voice Activity Detection*) untuk mendeteksi bagian sinyal audio yang memiliki energi di atas ambang batas (-45 dB). Setiap segmen suara yang terdeteksi disimpan sebagai file terpisah, sehingga dataset bertambah dari satu rekaman menjadi beberapa segmen suara burung.
   c. Menghitung fitur audio mel-spektogram dengan menerapkan Persamaan (2.9) pada sinyal audio dan MFCC dengan menerapkan Persamaan (2.10) pada sinyal audio untuk setiap rekaman suara agar dapat digunakan dalam model klasifikasi.
   d. Mengatur data agar berada dalam format yang sesuai untuk input model.
      1) *Resizing*: Mengubah ukuran representasi mel-spektrogram dari dimensi aslinya yaitu 128×T×1 dan MFCC sebesar 40×T×1 menjadi 224×224×1 untuk mel-spektogram dan 224×224×1. Hal ini dapat dilakukan dengan teknik *padding* yaitu dengan cara jika panjang frame (T) lebih pendek dari 224, akan ditambahkan *padding* nol di bagian akhir. Jika panjang frame (T) lebih panjang dari 224, akan dilakukan pemangkasan hingga 224.
      2) Normalisasi: Menerapkan normalisasi pada nilai-nilai mel-spektrogram dan MFCC agar berada dalam rentang yang sesuai dengan input standar MobileNet (misalnya antara 0 hingga 1 atau 0 hingga 255). Normalisasi ini penting agar distribusi nilai yang diinputkan konsisten dengan data yang digunakan untuk melatih model.
3. Statistika Deskriptif: Menampilkan beberapa contoh data suara serta visualisasi statistik dari fitur yang dihitung, seperti histogram dari nilai amplitudo dan spektrum frekuensi.

4. Membagi Dataset: Melakukan pembagian data secara terpisah untuk setiap kelas sesuai dengan proporsi 80% untuk *training validation* (80% untuk data *training* dan 20 % untuk data *validation*) dan 20% untuk data *test*. Tujuannya adalah agar setiap subset (*training, validation, test*) memiliki representasi dari semua kelas.
5. Konversi Data: Mengonversi rekaman suara dan fitur yang dihasilkan menjadi array NumPy untuk memudahkan pemrosesan dengan model yang diimpor.
6. Melakukan klasifikasi dengan metode Convolutional Neural Network (CNN):
   a. Menggunakan arsitektur CNN sederhana yang terdiri dari beberapa lapisan konvolusi, pooling, dan fully connected.
   b. Menentukan hyperparameter utama seperti *learning rate*, *batch size*, dan *dropout rate*.
   c. Melatih model CNN pada data *training* dan memvalidasi performanya dengan data *validation*.
7. Melakukan klasifikasi dengan metode MobileNet:
   a. Memuat model pre-trained MobileNet melalui API TensorFlow.
   b. Mengganti jumlah unit pada *output layer* agar sesuai dengan jumlah kategori spesies burung yang akan diklasifikasikan.
   c. Menentukan apakah layer terakhir model akan di-*freeze* atau di-*unfreeze* untuk penyesuaian lebih lanjut.
   d. Mengatur *hyperparameter* seperti *learning rate*, *batch size*, dan *dropout rate*.
   e. Menentukan strategi *early stopping* untuk mencegah overfitting selama proses pelatihan.
   f. Melakukan pelatihan model pada data *training* dan memvalidasi performanya dengan data *validation*.
8. Melakukan klasifikasi dengan metode VGGish dengan input waveform mentah.
   a. Memuat model pre-trained VGGish melalui API TensorFlow.
   b. Mengganti jumlah unit pada output layer agar sesuai dengan jumlah kategori spesies burung yang akan diklasifikasikan.
   c. Mengatur hyperparameter seperti learning rate, batch size, dan dropout rate.
   d. Menentukan strategi early stopping untuk mencegah overfitting selama proses pelatihan.
   e. Melakukan pelatihan model pada data training dan memvalidasi performanya dengan data validation.
9. Menghitung Kebaikan Model: Menghitung nilai akurasi dan loss untuk model baik pada data training maupun data validation.
10. Membandingkan Kinerja Model: Membandingkan hasil kinerja klasifikasi antara berbagai kombinasi hyperparameter dan arsitektur model yang digunakan.
11. Mengaplikasikan Model Terbaik: Menerapkan model dengan performa terbaik pada data test.
12. Menghitung Performa Model: Menghitung akurasi model pada data test untuk menilai efektivitas klasifikasi.
13. Analisis Kesalahan Klasifikasi: Melakukan analisis terhadap kesalahan klasifikasi yang terjadi pada data test untuk mengetahui kemungkinan perbaikan.
14. Menarik Kesimpulan dan Saran: Menyusun kesimpulan dari hasil analisis dan memberikan saran untuk penelitian selanjutnya atau pengembangan lebih lanjut.

3.5 Diagram Alir

Berdasarkan Langkah-langkah penelitian yang sudah dijelaskan, maka dapat disusun diagram alir untuk penelitian seperti pada Gambar 3.3.



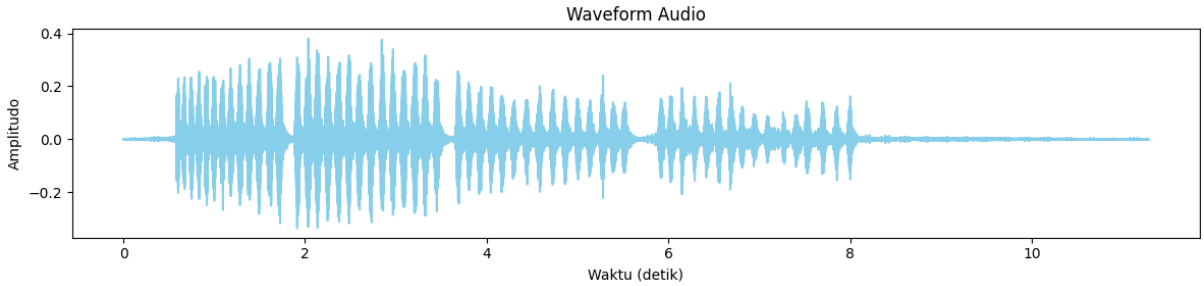
Gambar 3.3 Diagram Alir Penelitian

BAB 4 HASIL DAN PEMBAHASAN

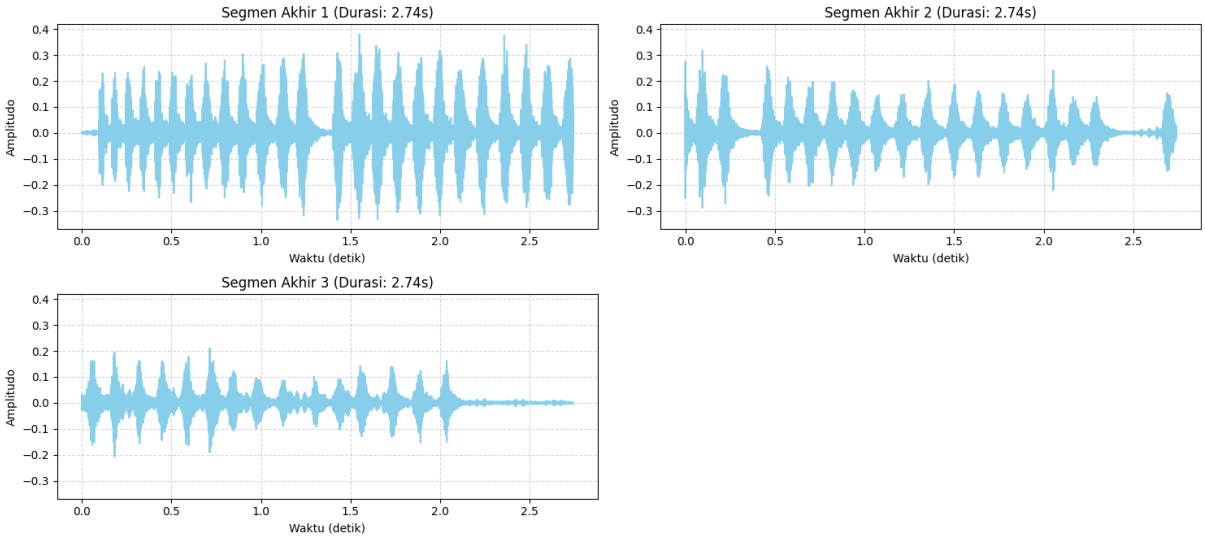
Bab ini menyajikan hasil klasifikasi suara enam spesies burung yang diperoleh dari situs xeno-canto.org, masing-masing sebanyak 10 rekaman dengan durasi 11–115 detik. Rekaman tersebut diolah dengan ekstraksi fitur Mel-Spectrogram dan MFCC lalu dimodelkan menggunakan CNN dan pre-trained model VGG16, serta waveform mentah untuk model VGGish. Evaluasi kinerja dilakukan dengan metrik akurasi, precision, recall, dan F1-score untuk mengukur efektivitas model dalam mengidentifikasi spesies burung berdasarkan ciri khas suaranya.

4.1 Preprocessing Data

Tahap awal dalam penelitian ini adalah preprocessing rekaman suara. Langkah pertama yang dilakukan adalah mengonversi format rekaman suara ke dalam format .wav jika diperlukan, untuk memastikan kompatibilitas dengan proses ekstraksi fitur audio. Selanjutnya, dilakukan proses *trimming* terhadap rekaman suara untuk menghilangkan bagian-bagian yang tidak relevan seperti jeda panjang atau noise di awal/akhir rekaman. Trimming ini menggunakan metode *Voice Activity Detection* (VAD), yang mendeteksi aktivitas suara berdasarkan energi sinyal di atas ambang batas -45 dB. Rekaman suara panjang kemudian dipotong menjadi segmen-segmen pendek berdurasi 1 hingga 3 detik (tergantung pada spesies burung) yang hanya mengandung bagian dengan suara aktif. Setiap segmen disimpan sebagai file terpisah, sehingga jumlah data dalam dataset meningkat secara signifikan dan lebih representatif untuk setiap kelas burung. Audio sebelum dan setelah tahap trimmed ditunjukkan pada Gambar 4.1.



Gambar 4.1 Gelombang Audio Cendet2.wav



Gambar 4.2 Gelombang Audio Cendet2.wav setelah Trimming

Setelah dilakukan trimming seperti pada Gambar 4.2, rekaman suara yang sebelumnya berdurasi panjang dan mengandung bagian tidak relevan kini telah dipotong menjadi beberapa segmen pendek yang hanya berisi suara burung aktif. Hal ini terlihat pada Gambar 4.1, di mana memperlihatkan bentuk gelombang (waveform) rekaman suara asli yang memiliki bagian sunyi di awal dan akhir. Sementara itu, Gambar 4.2 menunjukkan hasil pemotongan segmen (*trimming*) dari rekaman tersebut, yang kini terdiri dari bagian-bagian dengan durasi lebih pendek dan energi suara yang signifikan. Proses ini menghasilkan dataset yang lebih bersih, padat, dan efektif untuk proses pelatihan model klasifikasi, karena hanya menyertakan informasi akustik yang relevan dari setiap spesies burung. Segmentasi ini juga membantu dalam meningkatkan generalisasi model terhadap variasi suara antar individu dalam spesies yang sama.

Tabel 4. 1 Rekaman Suara Burung Sebelum dan Sesudah Proses Trimming

Jenis Burung	Rekaman Asli	Rekaman Hasil Trimming
Kutilang	10	97
Gereja	10	63
Perkutut	10	150
Tekukur	10	78
Trucukan	10	76
Cendet	10	95

Pada Tabel 4.1 memberikan informasi tentang jumlah file rekaman asli dan jumlah segmen hasil trimming untuk masing-masing spesies burung. Ini membantu menunjukkan peningkatan jumlah data akibat proses segmentasi, yang pada akhirnya memperkaya variasi data pelatihan dan meningkatkan performa model. Selanjutnya, dataset dibagi menjadi subset *training validation* dan *testing* dengan proporsi 80% (80% *training* dan 20% *validation*) dan 20% secara stratifikasi, sehingga masing-masing subset memiliki representasi yang seimbang dari semua kelas burung. Hasil pembagian dataset menjadi data *training* dan data *testing* ditunjukkan pada Tabel 4.2.

Tabel 4.2 Hasil pembagian Data *Training* dan *Testing* Per Jenis Burung

Jenis Burung	Data			Total
	Training	Validation	Testing	
Kutilang	62	16	19	97
Gereja	40	10	13	63
Perkutut	96	24	30	150
Tekukur	50	12	16	78
Trucukan	49	12	15	76
Cendet	61	15	19	95
Total	358	89	112	559

4.2 Ekstraksi Fitur Mel-Spektogram dan MFCC

Ekstraksi fitur merupakan tahap krusial dalam pemrosesan data audio, khususnya untuk tugas klasifikasi suara burung menggunakan metode deep learning. Pada penelitian ini, dua jenis fitur utama yang diekstraksi dari data audio adalah Mel-spektogram dan Mel Frequency Cepstral Coefficient (MFCC). Kedua fitur ini dipilih karena telah terbukti efektif dalam merepresentasikan karakteristik penting dari sinyal audio, sehingga dapat meningkatkan

performa model klasifikasi berbasis Convolutional Neural Network (CNN) maupun model transfer learning seperti MobileNet.

Mel-spektogram adalah representasi visual dari spektrum frekuensi suatu sinyal audio yang telah dipetakan ke dalam skala mel, yang meniru cara manusia mendengar perbedaan frekuensi. Proses ekstraksi Mel-spektogram diawali dengan membagi sinyal audio menjadi frame-frame kecil, kemudian menerapkan transformasi Fourier untuk mendapatkan spektrum frekuensi pada setiap frame. Selanjutnya, spektrum ini dipetakan ke dalam skala mel menggunakan filter bank mel, sehingga menghasilkan representasi yang lebih sesuai dengan persepsi manusia terhadap suara. Hasil ekstraksi Mel-spektogram biasanya divisualisasikan dalam bentuk gambar dua dimensi, di mana sumbu horizontal menunjukkan waktu, sumbu vertikal menunjukkan frekuensi mel, dan intensitas warna menggambarkan kekuatan energi pada setiap frekuensi dan waktu tertentu. Visualisasi ini sangat penting karena dapat langsung digunakan sebagai input pada arsitektur CNN, yang mampu mengenali pola-pola visual dari gambar spektogram untuk membedakan jenis suara burung.

Selain Mel-spektogram, fitur MFCC juga diekstraksi dari data audio. MFCC merupakan fitur yang sangat populer dalam pengenalan suara karena kemampuannya menangkap informasi spektral yang relevan dengan karakteristik sumber suara. Proses ekstraksi MFCC melibatkan beberapa tahapan, dimulai dari pre-emphasis untuk memperkuat frekuensi tinggi, framing, windowing, transformasi Fourier, penerapan filter bank mel, hingga akhirnya dilakukan Discrete Cosine Transform (DCT) untuk memperoleh koefisien MFCC. Dalam penelitian ini, digunakan 40 koefisien MFCC untuk setiap frame audio, yang bertujuan untuk menangkap detail spektral secara lebih menyeluruh. Koefisien pertama hingga ke-12 umumnya mewakili informasi dasar spektral, sedangkan koefisien berikutnya menangkap detail dan variasi frekuensi yang lebih kompleks, sehingga dapat membantu model dalam membedakan suara burung yang memiliki kemiripan spektral.

Proses ekstraksi fitur Mel-spektogram dan MFCC dilakukan menggunakan library Librosa, yang menyediakan fungsi-fungsi komprehensif untuk pemrosesan sinyal audio. Untuk memperjelas proses ekstraksi dan hasil fitur yang diperoleh, disajikan beberapa gambar contoh Mel-spektogram dan MFCC dari beberapa jenis suara burung yang berbeda pada subbab ini. Selain itu, tabel yang memuat nilai mel-spektogram dan koefisien MFCC untuk beberapa frame audio juga ditampilkan untuk memberikan gambaran mengenai variasi fitur yang dihasilkan.

Pada penelitian ini, proses ekstraksi fitur audio dilakukan dengan menggunakan *sampling rate* sebesar 22050 Hz dan frekuensi maksimum (fmax) sebesar 8000 Hz. Pemilihan sampling rate 22050 Hz dilakukan karena nilai ini sudah cukup untuk merepresentasikan informasi akustik yang umum ditemukan pada suara burung, sekaligus mengurangi beban komputasi dibanding *sampling rate* yang lebih tinggi. Sinyal audio dengan sampling rate 22050 Hz dapat merepresentasikan komponen frekuensi hingga 11025 Hz, sehingga batas atas frekuensi sebesar 8000 Hz masih berada dalam jangkauan aman dan efektif untuk analisis. Rentang ini juga dianggap ideal karena sebagian besar informasi penting dari vokalisasi burung berada pada rentang frekuensi rendah hingga menengah (< 8000 Hz). Oleh karena itu, nilai frekuensi maksimum sebesar 8000 Hz digunakan untuk membatasi skala frekuensi dalam perhitungan Mel-spektrogram dan MFCC untuk memastikan bahwa fitur yang diekstraksi fokus pada komponen suara yang paling relevan untuk klasifikasi jenis burung.

Proses ekstraksi fitur audio dalam penelitian ini dilakukan secara bertahap, dimulai dari sinyal suara mentah (waveform) yang contohnya ditampilkan pada Tabel 4.3 hingga menjadi representasi fitur dua dimensi seperti Mel-spektrogram dan MFCC yang digunakan sebagai input pada model klasifikasi. Tahapan pertama yaitu melakukan tahap *trimming* seperti yang ditunjukkan pada Gambar 4.2. Tahapan selanjutnya adalah pre-emphasis, yang dilakukan untuk

memperkuat frekuensi tinggi dalam sinyal. Secara alami, sinyal suara memiliki energi yang lebih tinggi pada frekuensi rendah, sehingga jika tidak diseimbangkan, model akan cenderung "buta" terhadap informasi penting pada frekuensi tinggi. Pre-emphasis dilakukan dengan menerapkan filter *high-pass* sederhana pada waveform dengan rumus 2.2 dan contoh hasilnya ditampilkan pada Tabel 4.4.

Tabel 4.3 Nilai Gelombang Audio Segmen 1 Cendet2.wav

No.	Waktu	Amplitudo
1	0	0,000519
2	0,000045	0,000488
3	0,000091	0,000549
⋮		
59007	2,676054	-0,189514

Tabel 4.4 Nilai Pre-emphasis Audio Segmen 1 Cendet2.wav

No.	Waktu	Amplitudo
1	0	0,000519
2	0,000045	0,000488
3	0,000091	0,000549
⋮		
59007	2,676054	-0,189514

Selanjutnya adalah proses *framing*, yaitu membagi sinyal audio menjadi frame-frame pendek berdurasi sekitar 20–40 ms. *Framing* diperlukan karena sinyal audio bersifat non-stasioner (berubah sepanjang waktu), sedangkan banyak teknik analisis seperti *Fourier Transform* mengasumsikan bahwa sinyal bersifat stasioner. Dengan membagi sinyal menjadi frame pendek, maka dalam setiap frame, sinyal bisa dianggap relatif stasioner dan dapat dianalisis secara frekuensi secara akurat. Contoh hasil proses *framing* dengan ukuran frame (*frame size*) sebesar 2048 sampel dan langkah perpindahan (*hop length*) sebesar 512 sampel ditampilkan pada Tabel 4.5.

Tabel 4.5 Hasil Proses *Framing* pada Audio Segmen 1 Cendet2.wav

Indeks Frame	Frame 1	Frame 2	Frame 3	…	Frame 2048
1	0,000519	-0,000005	0,000085	…	-0,001923
2	0,000607	-0,000278	-0,000720	…	0,001036
3	-0,000732	0,001276	-0,000551	…	-0,000380
⋮	⋮	⋮	⋮	⋱	⋮
112	0,052684	-0,009081	-0,001924	…	-0,135173

Setelah *framing*, diterapkan *windowing*, dalam penelitian ini menggunakan fungsi *Hamming window* yang persamaannya dapat dilihat pada persamaan 2.4 dan 2.5. Fungsi *window* ini mengurangi gangguan yang muncul di batas antara frame akibat pemotongan sinyal. Windowing memastikan bahwa transisi antar frame tetap halus dan tidak menghasilkan artefak pada spektrum yang dihitung. *Windowing* berfungsi untuk mengurangi *spectral leakage*, yaitu distorsi yang timbul ketika sinyal non-periodik dianalisis sebagai periodik dalam domain frekuensi. Contoh hasil proses windowing ditampilkan pada Tabel 4.6.

Tabel 4.6 Hasil Proses Windowing pada Audio Segmen 1 Cendet2.wav

Indeks Frame	Frame 1	Frame 2	Frame 3	…	Frame 2048
1	0,000042	-0,000001	0,000007	…	-0,000154
2	0,000049	-0,000022	-0,000058	…	0,000083
3	-0,000059	0,000102	-0,000044	…	-0,000030
⋮	⋮	⋮	⋮	⋱	⋮
112	0,004215	-0,000726	-0,000154	…	-0,010814

Langkah berikutnya adalah menerapkan *Short-Time Fourier Transform* (STFT) pada setiap frame untuk mengubah sinyal dari domain waktu ke domain frekuensi dengan menggunakan persamaan 2.6 lalu dilanjutkan menghitung spektrum daya dengan persamaan 2.7. STFT menghasilkan spektrum magnitudo dari setiap frame yang menunjukkan sebaran energi pada berbagai frekuensi. Contoh hasil proses STFT ditampilkan pada Tabel 4.7.

Tabel 4.7 Hasil Proses STFT pada Audio Segmen 1 Cendet2.wav

Indeks Frame	0 Hz	10,72 Hz	21,53 Hz	…	11025 Hz
1	0,000920	0,000319	0,000082	…	0,001225
2	0,000782	0,000387	0,000090	…	0,000135
3	0,000833	0,000322	0,000078	…	0,000317
⋮	⋮	⋮	⋮	⋱	⋮
112	0,002468	0,003682	0,003289	…	0,001278

Langkah selanjutnya adalah membangun dan menerapkan Mel filter bank terhadap spektrum daya yang telah diperoleh dari proses STFT. Mel filter bank terdiri dari sejumlah filter segitiga yang saling tumpang tindih, yang masing-masing berfungsi untuk menjumlahkan energi spektral dalam rentang frekuensi tertentu. Dalam konteks penelitian ini, digunakan sebanyak 128 filter, sesuai dengan jumlah fitur Mel yang ditentukan. Pertama-tama, nilai frekuensi minimum (0 Hz) dan frekuensi maksimum (8000 Hz) dikonversi ke skala Mel menggunakan Persamaan 2.8. Selanjutnya, rentang Mel ini dibagi menjadi 128 bagian yang sama, lalu tiap titik pada skala Mel ini dikonversi kembali ke skala frekuensi (Hz) menggunakan Persamaan 2.9. Setelah rentang frekuensi diperoleh, langkah berikutnya adalah menentukan posisi pusat masing-masing filter segitiga dalam domain frekuensi. Filter segitiga ini memiliki tiga titik penting yaitu batas bawah (k_{m-1}), puncak (k_m), dan batas atas (k_{m+1}), yang kemudian dikonversi ke indeks bin FFT berdasarkan resolusi spektral (NFFT) dan *sampling rate* (SR). Indeks bin ini menunjukkan di posisi mana pada spektrum daya masing-masing filter akan beroperasi. Setelah mengetahui lokasi bin tersebut, maka setiap filter segitiga dihitung menggunakan rumus filter segitiga pada Persamaan 2.10. Contoh dari hasil *filter bank* ditampilkan pada Tabel 4.8.

Tabel 4.8 Hasil *Filter Bank* untuk Audio Segmen 1 Cendet2.wav

Indeks Mel	FFT 1	FFT 2	FFT 3	…	FFT 1025
1	0,000000	0,019692	0,039384	…	0,000000
2	0,000000	0,000000	0,000000	…	0,000000
3	0,000000	0,000000	0,000000	…	0,000000
⋮	⋮	⋮	⋮	⋱	⋮
128	0,000000	0,000000	0,000000	…	0,000000

Nilai energi hasil filter bank yang didapat kemudian dikonversi ke skala logaritmik (dB) dengan persaamaan 2.11 untuk menekan dinamika amplitudo ekstrem dan menghasilkan representasi mel-spektogram yang lebih stabil dan informatif. Contoh hasil mel-spektogam ditampilkan pada Tabel 4.9.

Tabel 4.9 Mel-Spektogram untuk Audio Segmen 1 Cendet2.wav

Frame ke-	Mel ke-1	Mel ke-2	Mel ke-3	…	Mel ke-128
1	-69,199140	-75,34157	-74,350136	…	-72,787550
2	-68,113110	-78,81882	-76,259964	…	-69,254020
3	-68,803040	-77,26324	-76,02953	…	-69,282810
⋮	⋮	⋮	⋮	⋱	⋮
116	-43,600350	-43,77701	-43,519073	…	-20,992493

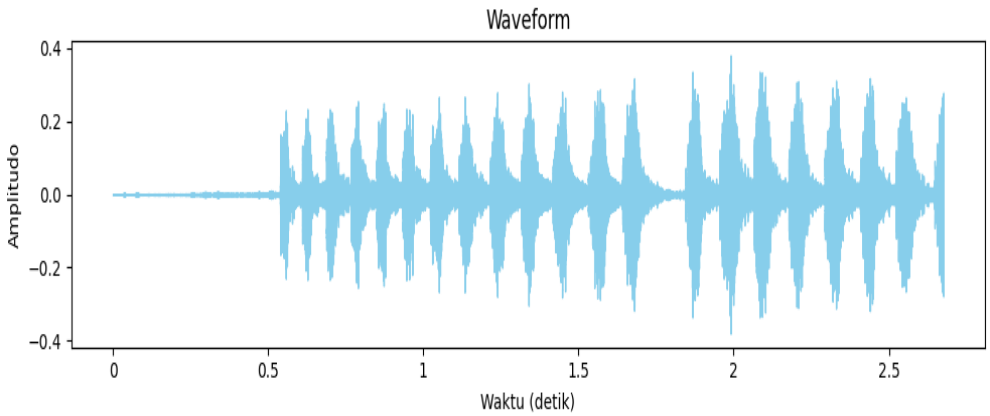
Untuk menghasilkan fitur MFCC, log-energi dari filter bank mel selanjutnya diproses dengan *Discrete Cosine Transform* (DCT) pada persamaan 2.12, sehingga diperoleh 40 koefisien MFCC yang merepresentasikan karakteristik spektral penting dari sinyal suara burung. Koefisien awal MFCC menggambarkan energi total, sedangkan koefisien lebih tinggi menangkap kontur dan variasi spektral yang lebih kompleks. Contoh hasil mel-spektogam ditampilkan pada Tabel 4.10.

Tabel 4.10 MFCC untuk Audio Segmen 1 Cendet2.wav

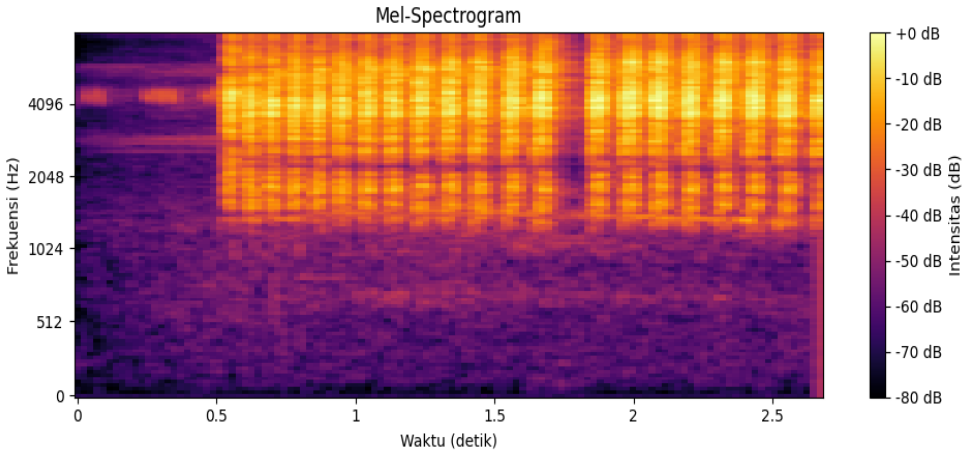
Frame ke-	MFCC ke-1	MFCC ke-2	MFCC ke-3	…	MFCC ke-40
1	-666,19385	-15,322249	4,257605	…	6,118926
2	-630,26996	-20,513931	-3,749663	…	6,198575
3	-619,50385	-29,231792	-12,351944	…	4,175071
⋮	⋮	⋮	⋮	⋱	⋮
116	-215,16031	-112,700010	-38,727203	…	0,149659

Seluruh tahapan ekstraksi fitur ini tidak hanya bersifat matematis, tetapi juga didasarkan pada pengetahuan tentang fisiologi dan persepsi pendengaran manusia, sehingga fitur yang dihasilkan sangat relevan untuk tugas klasifikasi suara burung.

Contoh hasil representasi bentuk gelombang (*waveform*) dari salah satu segmen suara ditunjukkan pada Gambar 4.3, sedangkan hasil penerapan mel-spektrogram dari segmen yang sama ditampilkan pada Gambar 4.4.



Gambar 4.3 Gelombang Audio Segmen 1 Burung Cendet2.wav

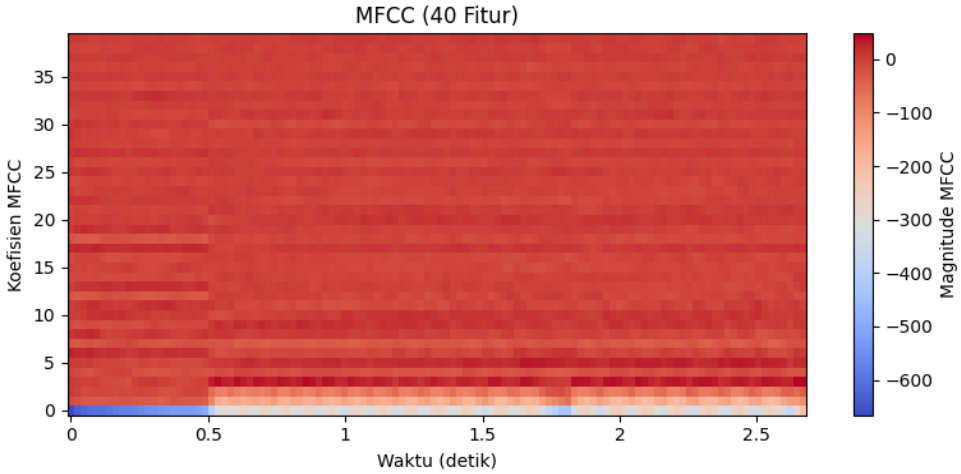


Gambar 4.4 Visualisasi Mel-Spektogram Audio Segmen 1 Cendet2.wav

Ekstraksi Mel-spektogram dilakukan dengan mengonversi sinyal audio menjadi representasi visual dua dimensi yang menangkap karakteristik spektral-temporal. Konfigurasi 128 filter dipilih untuk menyeimbangkan antara detail spektral (resolusi frekuensi) dan reduksi dimensi data, khususnya efektif dalam menangkap struktur harmonik unik pada vokalisasi burung. Visualisasi akhir berupa gambar spektogram dengan sumbu waktu (horizontal), frekuensi mel (vertikal), dan intensitas warna merepresentasikan energi desibel. Dalam penelitian ini digunakan 128 filter mel, sehingga setiap kolom pada matriks fitur Mel-spektogram merepresentasikan energi dalam rentang frekuensi tertentu, bukan frekuensi tunggal. Frekuensi maksimum (fmax) yang digunakan pada penelitian ini adalah 8000 Hz dan jumlah filter Mel adalah 128, maka rentang frekuensi yang dicakup oleh setiap filter Mel tidak sama secara linier, melainkan mengikuti skala Mel yang bersifat non-linear. Namun, secara kasar, jika diasumsikan rentang frekuensi dibagi rata secara linier, maka setiap filter akan mencakup sekitar 62,5 Hz. Namun, dalam praktiknya, filter Mel didesain dengan lebar yang semakin melebar pada frekuensi tinggi dan semakin sempit pada frekuensi rendah untuk meniru persepsi pendengaran manusia. Oleh karena itu, filter Mel pada frekuensi rendah mungkin hanya mencakup rentang sekitar 20–50 Hz, sedangkan pada frekuensi tinggi bisa mencapai beberapa ratus Hz. Nilai dari mel-spektogram ini menjadi input optimal untuk arsitektur CNN. Salah satu hasil perhitungan mel-spektogram menggunakan library Librosa dengan tambahan tahap *pre-emphasis* ditunjukkan pada Tabel 4.9.

Setiap dari 128 fitur Mel-spektogram merepresentasikan energi pada pita frekuensi tertentu dalam skala mel yang disusun secara non-linear, sehingga masing-masing fitur menangkap karakteristik spektral yang berbeda dari sinyal audio. Fitur-fitur pada pita frekuensi rendah (misalnya fitur Mel ke-1 hingga ke-40) cenderung menangkap komponen frekuensi fundamental dan harmonik utama yang dominan dalam vokalisasi burung, sedangkan fitur pada pita frekuensi menengah hingga tinggi (Mel ke-41 hingga ke-128) lebih sensitif terhadap detail spektral halus dan variasi frekuensi kompleks seperti perubahan cepat dan tekstur suara. Dengan menggunakan 128 filter Mel, ekstraksi fitur ini mampu memberikan representasi spektral yang sangat rinci, memungkinkan model untuk membedakan suara burung yang memiliki kemiripan tonal maupun pola frekuensi yang kompleks. Oleh karena itu, pemilihan jumlah filter sebanyak 128 ini memberikan keseimbangan optimal antara resolusi frekuensi yang tinggi dan efisiensi komputasi, sehingga model deep learning dapat menangkap informasi akustik yang lebih kaya dan meningkatkan akurasi klasifikasi suara burung. Setelah ekstraksi mel-spektogram dilakukan, selanjutnya melakukan ekstraksi fitur MFCC. Contoh hasil representasi bentuk gelombang (waveform) dari salah satu segmen suara ditunjukkan pada

Gambar 4.3, sedangkan hasil penerapan MFCC dari segmen yang sama ditampilkan pada Gambar 4.5.



Gambar 4.5 Visualisasi MFCC Audio Segmen 1 Cendet2.wav

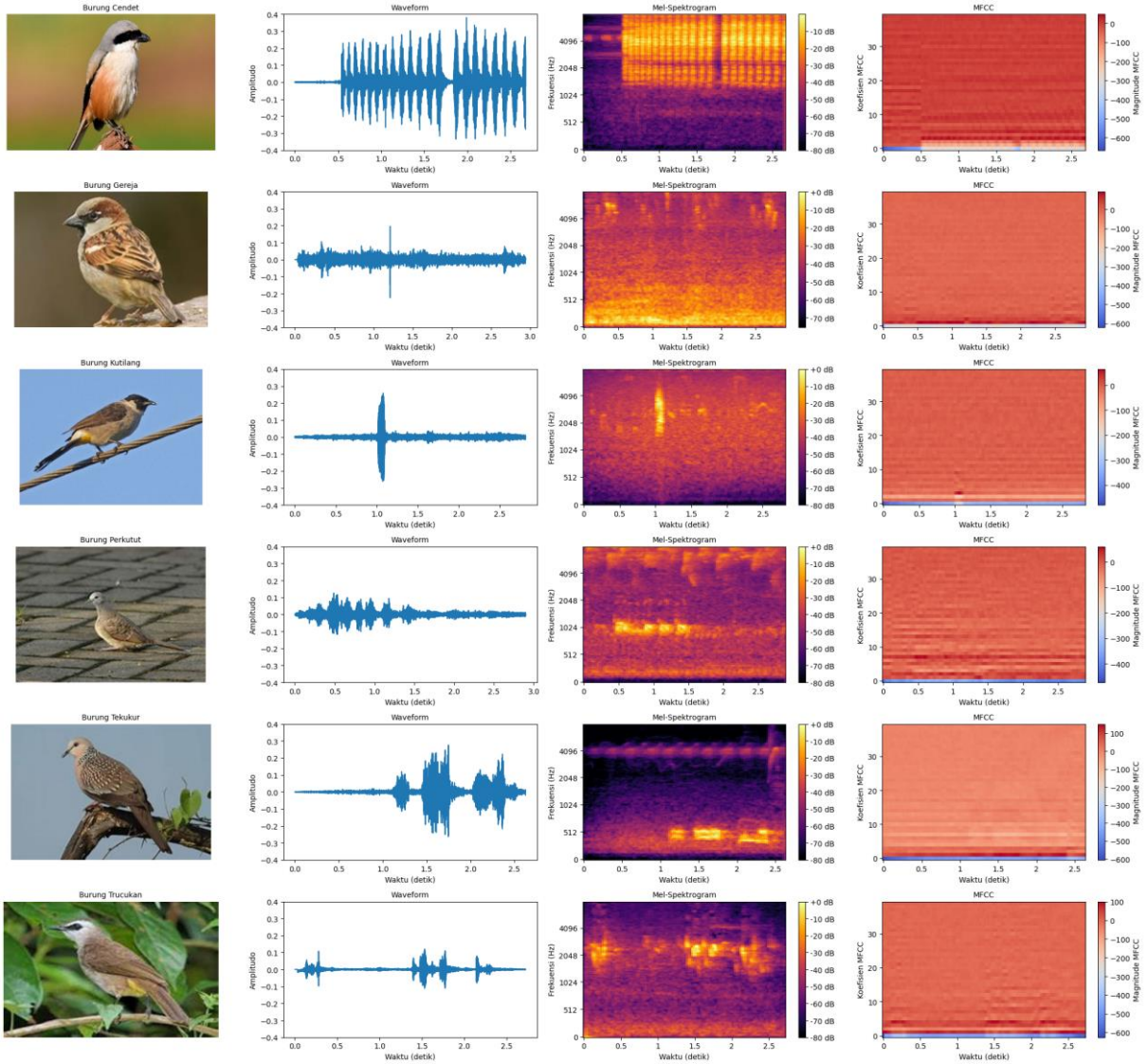
Tahap selanjutnya dalam ekstraksi fitur MFCC adalah menerapkan transformasi kosinus terdiskret (*Discrete Cosine Transform*) pada logaritma energi dari filter bank Mel tersebut. Transformasi ini bertujuan untuk mengubah representasi spektral yang telah dipetakan ke skala Mel menjadi domain cepstral, sehingga menghasilkan koefisien yang lebih terkompresi dan terpisah secara jelas antara informasi sumber suara dan karakteristik filter vokal. Dengan kata lain, DCT membantu mengurangi korelasi antar fitur dan mengekstrak pola-pola penting yang merepresentasikan ciri khas suara secara lebih efisien. Koefisien MFCC yang dihasilkan setelah DCT ini kemudian dipilih sejumlah tertentu (misalnya 40 koefisien) untuk digunakan sebagai fitur input dalam model pembelajaran mesin atau deep learning. Pemilihan jumlah koefisien ini bertujuan untuk menangkap detail spektral yang cukup tanpa menimbulkan redundansi yang berlebihan. Secara keseluruhan, proses ini memungkinkan MFCC untuk menghasilkan representasi fitur yang padat dan informatif, yang sangat berguna dalam membedakan karakteristik suara burung maupun suara lainnya dengan tingkat akurasi yang tinggi. Salah satu hasil perhitungan MFCC menggunakan library Librosa ditunjukkan pada Tabel 4.10.

Setiap koefisien MFCC menyimpan informasi spektral yang berbeda dan memiliki peran spesifik dalam merepresentasikan karakteristik suara. Koefisien MFCC pertama (MFCC ke-1) umumnya mencerminkan gambaran umum energi spektral secara keseluruhan, memberikan indikasi kasar tentang intensitas suara di seluruh rentang frekuensi. Selanjutnya, koefisien MFCC ke-2 hingga ke-12 berfungsi untuk menangkap variasi spektral yang lebih halus serta perubahan energi yang terjadi secara bertahap sepanjang waktu, sehingga mampu merepresentasikan pola dasar dari sinyal suara. Koefisien MFCC ke-13 hingga ke-24 lebih fokus pada detail spektral yang lebih kompleks, seperti perubahan frekuensi yang lebih dinamis dan fitur-fitur akustik yang lebih rumit. Sedangkan koefisien MFCC ke-25 hingga ke-40 berperan dalam menangkap detail spektral yang sangat halus dan variasi frekuensi tinggi yang mungkin tidak terdeteksi oleh koefisien yang lebih rendah. Dengan memilih sebanyak 40 koefisien MFCC, fitur yang dihasilkan menjadi lebih kaya dan detail, memungkinkan model untuk mengenali perbedaan kecil dalam karakteristik suara, seperti nuansa emosi atau keunikan vokalisasi, sehingga dapat meningkatkan akurasi dalam tugas klasifikasi suara.

Setelah melalui proses ekstraksi fitur menggunakan metode mel-spektogram dengan 128 filter mel dan MFCC sebanyak 40 fitur. Hasil ekstraksi tersebut kemudian digunakan sebagai input pada model klasifikasi berbasis *Convolutional Neural Network* (CNN) serta metode *transfer learning* dengan arsitektur *MobileNet*. Pendekatan ini memungkinkan model untuk

menangkap pola spektral dan temporal yang kompleks dalam suara burung, sehingga meningkatkan akurasi klasifikasi. Dengan memanfaatkan representasi fitur mel-spektogram dan MFCC yang kaya informasi, diharapkan model CNN dan *MobileNet* mampu mengenali ciri khas vokalisasi berbagai jenis burung secara efektif, sebagaimana telah dibuktikan dalam berbagai penelitian terkait klasifikasi suara burung menggunakan *deep learning*.

Setelah seluruh data audio suara burung melalui tahap proses ekstraksi fitur Mel-spektogram dan MFCC. Hasil ekstraksi tersebut kemudian divisualisasikan sebagai representasi spektral yang menjadi input utama untuk model klasifikasi berbasis CNN dan *transfer learning* dengan arsitektur *MobileNet* pada penelitian ini. Visualisasi pada Gambar 4.6 memperlihatkan karakteristik frekuensi, mel-spektogram, dan MFCC dari suara burung secara acak untuk berbagai jenis burung seperti Cendet, Gereja, Kutilang, Perkutut, Tekukur, dan Trucukan, sehingga memudahkan model dalam mengenali pola vokalisasi yang khas dari masing-masing spesies.



Gambar 4.6 Visualisasi Gambar Burung, Gelombang Audio, Mel-Spektogram dan MFCC untuk Setiap Jenis Burung (dari Kiri ke Kanan) , dengan Urutan Jenis Burung dari Atas ke Bawah: Cendet, Gereja, Kutilang, Perkutut, Tekukur, dan Trucukan

Pada Gambar 4.6 memperlihatkan visualisasi gelombang audio, mel-spektogram, dan MFCC dari masing-masing jenis burung, yang menunjukkan perbedaan karakteristik akustik pada setiap spesies. Visualisasi ini menjadi dasar penting dalam proses ekstraksi fitur dan pemodelan klasifikasi suara burung pada penelitian ini.

4.3 Metode Pelatihan Model

Pada peneilitian ini, normalisasi data seperti melakukan normalisasi berada di rentang 0-1 tidak diterapkan karena metode ini sangat rentan terhadap *outlier* dan dapat merusak pola internal yang krusial bagi setiap sinyal suara. Alasannya karena dalam dataset audio, sangat mungkin ada satu atau beberapa rekaman yang volumenya jauh lebih keras daripada yang lain. Jika kita menggunakan normalisasi global, rekaman yang paling keras ini akan menentukan nilai maksimum (skala 1), yang akibatnya memaksa semua rekaman dengan volume normal lainnya untuk berada di rentang nilai yang sangat kecil (misalnya antara 0 dan 0,1). Proses ini menghilangkan detail dan kontras penting dari sebagian besar data, sehingga model kehilangan informasi yang dibutuhkan untuk mengenali pola suara yang sebenarnya. Sehingga jika dilakukan normalisasi pada semua data fitur audio maka model jadi belajar tentang volume absolut, bukan tentang "bentuk" atau karakteristik suara itu sendiri.

Metode padding yang dilakukan adalah *zero-padding* pada sisi kanan dan bawah data audio. Metode padding pada penelitian ini memiliki konteks yang berbeda dibandingkan dengan padding pada data gambar yang biasanya diterapkan di semua sisi. Pada data gambar, padding di semua sisi dilakukan untuk menjaga keseimbangan spasial dan simetri fitur di sekitar objek, sehingga model dapat mengenali pola tanpa bias posisi dan mempertahankan informasi kontekstual di sekitar piksel. Sebaliknya, pada data audio, dimensi waktu dan frekuensi memiliki urutan dan arah yang sangat penting, padding di awal (kiri atau atas) dapat menggeser atau mendistorsi informasi temporal dan spektral yang krusial untuk klasifikasi suara. Oleh karena itu, padding salah satu sisi biasanya di kanan untuk fitur dan bawah untuk waktu dipilih agar bagian awal sinyal tetap utuh dan hanya bagian akhir yang diisi dengan nilai nol, sehingga urutan alami sinyal tetap terjaga. Pendekatan ini juga didukung oleh praktik umum dalam pustaka audio seperti ‘Librosa’ yang secara default menerapkan padding satu sisi melalui fungsi ‘librosa.util.fix_length’, yang menambahkan nol di bagian akhir array audio. Demikian pula, PyTorch/Torchaudio menggunakan padding satu sisi (right padding) dalam banyak pipeline audio, misalnya dengan fungsi ‘torch.nn.functional.pad’ yang menambahkan padding hanya di akhir sinyal untuk menyesuaikan panjang input. Dengan demikian, penggunaan padding satu sisi bukan hanya menjaga kualitas representasi audio, tetapi juga merupakan praktik standar yang diadopsi oleh banyak alat dan framework dalam domain pengolahan sinyal suara. Gambar 4.7 merupakan contoh suatu input, Gambar 4.8 merupakan input dengan *zero-padding* pada sisi kanan dan bawah, dan Gambar 4.9 merupakan *zero-padding* pada semua sisi.

1	3
3	4

Gambar 4.7 Contoh Input

1	3	0	0
3	4	0	0
0	0	0	0
0	0	0	0

Gambar 4.8 Contoh Input dengan *zero-padding* pada Sisi Kanan dan Bawah

0	0	0	0
0	1	3	0
0	3	4	0
0	0	0	0

Gambar 4.9 Contoh Input dengan *zero-padding* pada Semua Sisi

Proses *training* model pada penelitian ini dilakukan menggunakan dataset suara burung dengan fitur mel-spektogram dan MFCC sebagai input. Proses *training* ini melibatkan penyesuaian bobot jaringan melalui beberapa epoch untuk mengoptimalkan kemampuan model dalam mengenali pola akustik khas tiap jenis burung. Selama pelatihan, dilakukan monitoring terhadap metrik performa seperti akurasi dan loss pada data training maupun validasi untuk memastikan model tidak mengalami *overfitting*. Setiap model kombinasi dilatih selama maksimal 150 epoch dengan penerapan mekanisme *callbacks* untuk mengoptimalkan proses pelatihan dan mencegah *overfitting*. Salah satu *callback* yang digunakan adalah *EarlyStopping*, yang memonitor nilai *validation loss* dan secara otomatis menghentikan pelatihan jika tidak terjadi perbaikan selama 10 epoch berturut-turut. Selain itu, *EarlyStopping* juga mengembalikan bobot model ke kondisi terbaik saat nilai *validation loss* mencapai titik terendah, sehingga memastikan model yang dihasilkan memiliki kemampuan generalisasi yang optimal. Selain itu, digunakan juga *ReduceLROnPlateau*, yang secara dinamis menyesuaikan *learning rate* ketika *validation loss* tidak menunjukkan perbaikan dalam 5 epoch. Dengan mengurangi *learning rate* secara bertahap hingga batas minimum yang telah ditentukan, mekanisme ini membantu model keluar dari jebakan *local minima* dan mencapai konvergensi yang lebih baik. Kombinasi kedua *callbacks* ini memastikan pelatihan berjalan efisien, menghemat waktu, sekaligus menghasilkan model dengan performa terbaik pada data validasi.

4.4 Pemodelan dan Evaluasi Klasifikasi dengan CNN

Model klasifikasi pertama yang digunakan dalam penelitian ini adalah Convolutional Neural Network (CNN) dengan input berupa nilai-nilai numerik hasil ekstraksi fitur Mel-spektogram dan MFCC dari sinyal audio suara burung. Nilai fitur tersebut disusun dalam bentuk matriks dua dimensi yang merepresentasikan distribusi energi frekuensi dan koefisien spektral secara langsung. Model CNN ini terdiri dari tiga lapisan konvolusi (Conv2D) yang berfungsi mengekstraksi pola-pola penting dari data fitur tersebut, diikuti oleh lapisan MaxPooling2D untuk mengurangi dimensi spasial dan menjaga fitur dominan. Setiap lapisan konvolusi juga dilengkapi dengan normalisasi batch guna menjaga kestabilan proses pelatihan. Setelah melewati lapisan konvolusi, data fitur diubah menjadi vektor satu dimensi melalui flatten layer, kemudian diteruskan ke dua lapisan dense dengan fungsi aktivasi ReLU yang masing-masing diikuti oleh dropout untuk mencegah overfitting. Pada lapisan output, fungsi aktivasi softmax digunakan untuk menghasilkan probabilitas klasifikasi pada setiap kelas jenis burung. Detail arsitektur lengkap model CNN ini dapat dilihat pada Tabel 4.11.

Tabel 4.11 Arsitektur Model CNN

Layer Type	Output Shape	Parameter
input (input layer)	(224, 224, 3)	0
conv2d_1 (Conv2D)	(222, 222, 32)	320
max_pooling2d_1 (MaxPooling2D)	(111, 111, 32)	0
conv2d_2 (Conv2D)	(109, 109, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(54, 54, 64)	0
conv2d_3 (Conv2D)	(52, 52, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(26, 26, 128)	0
flatten_1 (Flatten)	(86528)	0
dense_1 (Dense)	(64)	5537856
dense_2 (Dense)	(6)	390
Total Parameter		**5630918**

Arsitektur Convolutional Neural Network (CNN) yang digunakan dalam penelitian ini terdiri dari tiga lapisan konvolusi (Conv2D), masing-masing diikuti oleh lapisan MaxPooling2D, serta dua lapisan fully connected (Dense) yang diakhiri dengan lapisan output softmax untuk klasifikasi enam kelas suara burung. Model ini menerima input berupa citra fitur audio berukuran 224×224 piksel dengan tiga saluran warna (RGB), yang merupakan representasi dari padding fitur Mel-spektrogram atau MFCC hasil ekstraksi dari data audio. Lapisan pertama (Conv2D_1) menggunakan 32 filter berukuran 3×3, stride 1, dan padding default 'valid' (tanpa padding), sehingga menghasilkan output berukuran 222×222×32. Jumlah parameter pada lapisan ini adalah 320. Setelah itu, MaxPooling2D_1 dengan ukuran pool 2×2 dan stride 2 mereduksi dimensi spasial menjadi 111×111×32. Lapisan konvolusi kedua (Conv2D_2) menggunakan 64 filter 3×3, stride 1, dan tanpa padding, sehingga menghasilkan output 109×109×64 dengan 18496 parameter, kemudian direduksi kembali oleh MaxPooling2D_2 menjadi 54×54×64. Lapisan ketiga (Conv2D_3) menerapkan 128 filter 3×3 (stride 1, tanpa padding) yang menghasilkan output 52×52×128, dengan jumlah parameter sebesar 73856. Setelah MaxPooling2D_3, dimensi output direduksi menjadi 26×26×128. Output dari bagian konvolusi kemudian diratakan (Flatten) menjadi vektor satu dimensi sepanjang 86528 elemen. Vektor ini diteruskan ke lapisan fully connected pertama (Dense_1) yang memiliki 64 neuron, sehingga menghasilkan 5.537.856 parameter. Lapisan ini berfungsi sebagai penghubung untuk mengabstraksikan fitur-fitur kompleks dari hasil konvolusi. Selanjutnya, lapisan output (Dense_2) memiliki 6 neuron (satu untuk tiap kelas burung) dan menghasilkan 390 parameter, menggunakan fungsi aktivasi softmax untuk menghasilkan probabilitas dari tiap kelas. Secara keseluruhan, model ini memiliki total 5630918 parameter yang mencerminkan kapasitas tinggi untuk mempelajari pola kompleks pada data input berbasis spektrum audio.

4.4.1 Klasifikasi Model CNN dengan Fitur Mel-Spektogram

Setelah model CNN dibuat dengan arsitektur yang telah ditentukan, pelatihan dilakukan menggunakan fitur Mel-spektogram dari dataset suara burung. Proses ini mencakup pengujian beberapa kombinasi hyperparameter untuk menemukan konfigurasi terbaik. Setiap konfigurasi dilatih sebanyak 30 kali secara independen, sehingga distribusi hasil evaluasi mendekati normal dan estimasi rata-rata performa lebih stabil. Pada setiap iterasi, bobot model diperbarui untuk mengenali pola karakteristik suara burung. Nilai rata-rata akurasi dan loss pada data training dan validasi digunakan sebagai dasar evaluasi performa dan pemilihan konfigurasi optimal, sebagaimana ditampilkan pada Tabel 4.12.

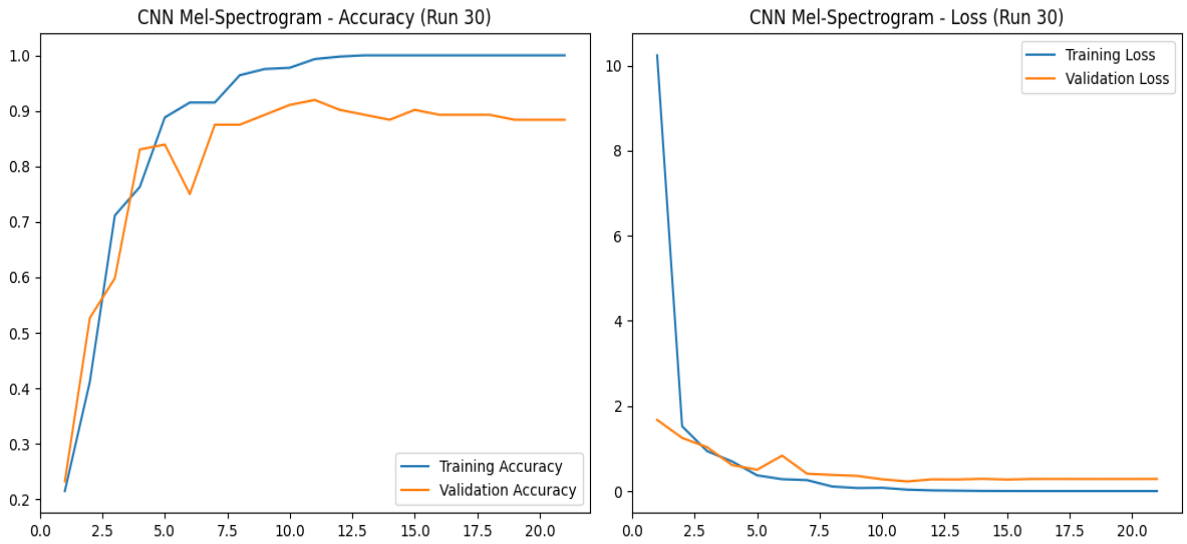
Tabel 4.12 Performa Kombinasi *Hyperparameter* Model CNN dengan Input Mel-Spektogram

No.	Hyperparameter			Hasil			
	Learning Rate	***Batch***	***Dropout***	***Mean* Akurasi *Training***	***Mean* Loss *Training***	***Mean* Akurasi *Validation***	***Mean* Loss *Validation***
1.	0,000001	16	0,5	0,99	0,002632	0,9134	0,453622
2.	0,000001	32	0,5	1,00	0,002472	0,9190	0,365826

Berdasarkan hasil pengujian performa model CNN dengan fitur Mel-spektogram yang ditampilkan pada Tabel 4.12, dapat dilihat bahwa variasi pengaturan hyperparameter memberikan pengaruh yang cukup signifikan terhadap hasil pelatihan. Pada kombinasi pertama, penggunaan batch size 16 dengan learning rate 0,000001 dan dropout 0,5 menghasilkan akurasi training sebesar 99% dengan nilai loss training yang rendah, yaitu 0,002632. Sementara itu, akurasi validasi yang diperoleh sebesar 91,34% dengan nilai loss

validasi 0,453622. Di sisi lain, kombinasi kedua dengan batch size 32, learning rate dan dropout yang sama, mampu meningkatkan akurasi training menjadi 100% dan menurunkan loss training menjadi 0,002472. Lebih penting lagi, kombinasi kedua juga menghasilkan akurasi validasi yang lebih tinggi, yaitu 91,90%, serta loss validasi yang lebih rendah sebesar 0,365826.

Perbedaan performa pada data validasi ini menunjukkan bahwa pemilihan batch size yang lebih besar memberikan dampak positif terhadap kemampuan generalisasi model. Nilai akurasi dan loss pada data validasi menjadi indikator utama dalam memilih konfigurasi terbaik, karena mencerminkan kemampuan model dalam mengklasifikasikan data yang belum pernah dilihat sebelumnya. Oleh karena itu, kombinasi kedua dianggap lebih optimal karena memberikan akurasi validasi yang lebih tinggi dan loss validasi yang lebih rendah. Hasil ini menegaskan pentingnya tuning hyperparameter secara cermat untuk mencapai keseimbangan antara performa training dan generalisasi model. Selanjutnya, grafik akurasi dan loss selama proses pelatihan pada kombinasi kedua pada salah satu pelatihan dapat diamati pada Gambar 4.10.



Gambar 4.10 Nilai Akurasi dan *Loss* Model CNN Mel-Spektogram Kombinasi Pertama

Nilai akurasi dan loss ini menunjukkan bahwa model mampu belajar dengan baik pada data training sekaligus mempertahankan performa yang cukup baik saat diuji pada data *validation*.

4.4.2 Klasifikasi Model CNN dengan Fitur MFCC

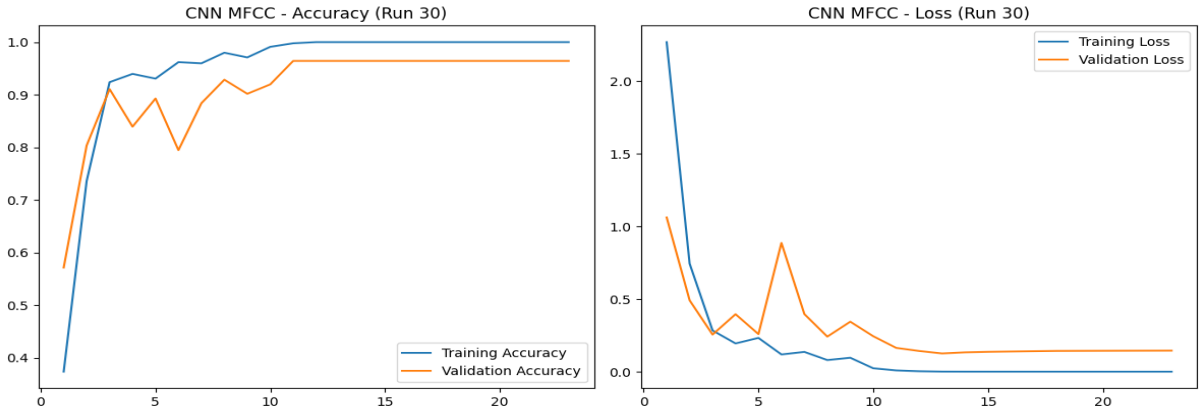
Setelah proses pelatihan model CNN menggunakan fitur Mel-spektogram, pelatihan serupa juga dilakukan dengan menggunakan fitur MFCC sebagai input pada arsitektur CNN yang sama. Proses training ini bertujuan untuk membandingkan efektivitas kedua jenis fitur dalam mendukung performa klasifikasi suara burung. Data MFCC yang telah diekstraksi disiapkan dan diolah agar sesuai dengan format input model, kemudian model dilatih dengan berbagai kombinasi hyperparameter untuk memperoleh konfigurasi terbaik. Setiap kombinasi hyperparameter diuji dengan menjalankan pelatihan sebanyak 30 kali secara independen , sehingga distribusi hasil evaluasi mendekati normal dan hasil rata-rata lebih representatif. Hasil pelatihan dengan fitur MFCC ini dievaluasi menggunakan metrik yang sama seperti akurasi dan loss, sehingga dapat dibandingkan secara langsung dengan hasil pelatihan menggunakan fitur Mel-spektogram untuk menentukan fitur mana yang memberikan kontribusi terbaik dalam klasifikasi jenis burung. Untuk menentukan konfigurasi dengan akurasi dan generalisasi terbaik pada model CNN dengan data MFCC, hasil performa rata-rata dari 30 kali pelatihan disajikan pada Tabel 4.13.

Tabel 4.13 Performa Kombinasi Hyperparameter Model CNN dengan Input MFCC

No.	Hyperparameter			Hasil			
	Learning Rate	***Batch***	***Dropout***	***Mean* Akurasi *Training***	***Mean* Loss *Training***	***Mean* Akurasi *Validation***	***Mean* Loss *Validation***
1.	0,000001	16	0,5	1,00	0,000368	0,9797	0,110764
2.	0,000001	32	0,5	1,00	0,000789	0,9762	0,109140

Berdasarkan hasil pengujian performa model CNN dengan fitur MFCC yang ditampilkan pada Tabel 4.13, terlihat bahwa kedua kombinasi hyperparameter memberikan hasil pelatihan yang sangat baik dengan akurasi training mencapai 100% dan loss training yang sangat rendah, masing-masing 0,000368 untuk batch size 16 dan 0,000789 untuk batch size 32. Pada data validasi, kombinasi pertama dengan batch size 16 menghasilkan akurasi validasi sedikit lebih tinggi yaitu 97,97% dengan loss validasi 0,110764, dibandingkan dengan kombinasi kedua yang memperoleh akurasi validasi 97,62% dan loss validasi 0,109140. Perbedaan performa ini menunjukkan bahwa meskipun batch size yang lebih besar sedikit menurunkan akurasi validasi, nilai loss validasi justru sedikit lebih rendah, mengindikasikan model tetap mampu melakukan generalisasi dengan baik pada data yang belum pernah dilihat sebelumnya.

Hasil ini menegaskan bahwa tuning hyperparameter, khususnya pemilihan batch size, perlu dilakukan secara cermat untuk mencapai keseimbangan optimal antara performa training dan generalisasi model. Kombinasi pertama dengan batch size 16 dianggap lebih unggul karena memberikan akurasi validasi yang sedikit lebih tinggi, meskipun perbedaan dengan batch size 32 tidak signifikan. Hal ini konsisten dengan beberapa penelitian yang menunjukkan bahwa MFCC sebagai fitur ekstraksi mampu mendukung performa CNN yang tinggi dalam klasifikasi suara, terutama ketika hyperparameter disesuaikan dengan tepat. Selanjutnya, grafik akurasi dan loss kombinasi pertama pada salah satu pelatihan dapat diamati pada Gambar 4.11, yang memperlihatkan peningkatan performa model secara stabil pada data training dan validasi.



Gambar 4.11 Nilai Akurasi dan Loss Model CNN MFCC Kombinasi Pertama

Nilai akurasi dan loss ini menunjukkan bahwa model mampu belajar dengan baik pada data *training* sekaligus mempertahankan performa yang cukup baik saat diuji pada data validasi.

4.5 Pemodelan dan Evaluasi Klasifikasi dengan MobileNet

Model klasifikasi kedua yang digunakan dalam penelitian ini adalah transfer learning dengan arsitektur MobileNet, yang menerima input berupa nilai numerik hasil ekstraksi fitur mel-spektogram dan MFCC dari sinyal audio suara burung, sama seperti pada model CNN sebelumnya. MobileNet merupakan model pra-latih yang ringan dan efisien, sehingga sangat cocok untuk aplikasi dengan keterbatasan sumber daya komputasi. Dalam implementasinya,

fitur Mel-spektogram dan MFCC disusun dalam format matriks dua dimensi yang kemudian diubah ukurannya agar sesuai dengan input yang diharapkan oleh MobileNet. Model ini memanfaatkan lapisan-lapisan konvolusi yang telah dilatih sebelumnya pada dataset besar dan hanya melakukan pelatihan ulang pada lapisan klasifikasi akhir untuk menyesuaikan dengan klasifikasi jenis burung dalam dataset. Dengan pendekatan transfer learning ini, MobileNet mampu mempelajari representasi fitur yang kompleks secara efisien dan menghasilkan performa klasifikasi yang kompetitif. Detail arsitektur dan konfigurasi pelatihan MobileNet dapat dilihat pada Tabel 4.14.

Tabel 4.14 Arsitektur Model MobileNet dari Library Tensorflow

Layer (Type)	**Output Shape**	**Parameter**
InputLayer	(224, 224, 3)	0
Conv2D (Conv1)	(112, 112, 32)	864
BatchNormalization	(112, 112, 32)	128
ReLU	(112, 112, 32)	0
DepthwiseConv2D (dw_1)	(112, 112, 32)	288
BatchNormalization	(112, 112, 32)	128
ReLU	(112, 112, 32)	0
Conv2D (pw_1)	(112, 112, 64)	2048
BatchNormalization	(112, 112, 64)	256
ReLU	(112, 112, 64)	0
⋮	⋮	⋮
Conv2D (pw_8)	(7, 7, 1280)	1310720
BatchNormalization	(7, 7, 1280)	5120
ReLU	(7, 7, 1280)	0
GlobalAveragePooling2D	(1280)	0
Dense (Softmax, 6 classes)	(6)	7686
Total Parameter		**4253894**

Untuk arsitektur MobileNet yang lebih jelas dapat dilihat pada Gambar 2.16 yang merupakan model convolutional neural network (CNN) efisien yang dirancang khusus untuk perangkat dengan keterbatasan komputasi. Model ini menerima input berukuran 224×224×3 dan memulai pemrosesan melalui layer Conv2D awal dengan filter 3×3 dan stride 2, yang menghasilkan output feature map berukuran 112×112×32. Selanjutnya, data diproses melalui serangkaian blok bottleneck (Bottleneck1 hingga Bottleneck6), yang merupakan implementasi dari depthwise separable convolution, kombinasi antara depthwise convolution dan pointwise convolution. Setiap blok bottleneck bertujuan untuk mengekstraksi fitur secara efisien dengan jumlah parameter yang lebih sedikit dibandingkan konvolusi standar. Setiap blok pada arsitektur MobileNet menggunakan depthwise separable convolution yang terdiri dari dua tahap utama. Pertama, dilakukan depthwise convolution (3×3) yang diterapkan secara terpisah pada tiap channel input (channel-wise), berfungsi untuk mengekstraksi fitur spasial. Selanjutnya, dilakukan pointwise convolution (1×1) untuk menggabungkan informasi antar channel dan menyesuaikan jumlah channel output. Setelah setiap konvolusi, biasanya diikuti dengan normalisasi (BatchNormalization) dan aktivasi ReLU. Pendekatan ini secara signifikan mengurangi jumlah parameter dan komputasi dibandingkan konvolusi konvensional. Proses ini

memungkinkan model mempertahankan akurasi yang tinggi dengan kompleksitas komputasi yang jauh lebih rendah. Dalam proses ini, dimensi spasial secara bertahap dikurangi (dari 112×112 menjadi 7×7), sementara jumlah channel meningkat (dari 16 hingga 320), yang mencerminkan peningkatan kompleksitas representasi fitur. Setelah blok bottleneck terakhir, terdapat layer Conv2D 1×1 yang mengubah dimensi channel menjadi 1280 tanpa mengubah ukuran spasial. Output ini kemudian diringkas melalui global average pooling yang menghasilkan vektor 1×1×1280, mewakili rata-rata dari setiap channel fitur. Setelah melewati seluruh proses ekstraksi fitur dan peringkasan representasi melalui global average pooling, arsitektur MobileNet pada penelitian ini dilengkapi dengan satu layer tambahan yaitu Dense (Softmax, 6 classes) yang memiliki 6 neuron output sesuai dengan jumlah kelas burung dalam tugas klasifikasi ini. Layer ini menerima input vektor berukuran 1280 (hasil dari global average pooling) dan memproyeksikannya ke dalam ruang kelas melalui fungsi aktivasi *softmax*, yang menghasilkan probabilitas untuk masing-masing kelas. Layer ini memiliki 7686 parameter trainable, dihitung dari 1280×6+6, yaitu bobot dan bias yang digunakan untuk memetakan representasi akhir ke output klasifikasi. Penambahan layer ini menjadikan MobileNet mampu menghasilkan prediksi akhir yang sesuai dengan konteks klasifikasi suara burung yang diteliti, sekaligus mempertahankan efisiensi arsitektur dengan total parameter yang tetap relatif ringan dengan jumlah total parameter sebanyak 4253894.

4.5.1 Klasifikasi Model *MobileNet* dengan Fitur Mel-Spektogram

Setelah arsitektur MobileNet ditentukan, proses pelatihan dilakukan dengan menggunakan data fitur ekstraksi Mel-spektogram dari dataset suara burung. Pelatihan model dilakukan dalam beberapa iterasi dengan kombinasi hyperparameter yang berbeda untuk menemukan konfigurasi terbaik. Selain itu, setiap konfigurasi diuji dengan menjalankan pelatihan sebanyak 30 kali secara independen, agar distribusi hasil evaluasi mendekati normal dan estimasi performa menjadi lebih stabil. Proses training ini bertujuan untuk menyesuaikan bobot model MobileNet agar mampu mengenali pola-pola khas pada fitur Mel-spektogram yang merepresentasikan karakteristik suara burung, dengan memanfaatkan pengetahuan yang telah diperoleh sebelumnya dari pelatihan pada dataset ImageNet. Hasil performa dari setiap kombinasi hyperparameter kemudian dihitung nilai rata-ratanya, termasuk mean accuracy dan mean loss pada data training maupun validasi. Nilai rata-rata inilah yang digunakan sebagai dasar evaluasi performa dan pemilihan konfigurasi terbaik pada model MobileNet dengan input Mel-spektogram yang ditampilkan pada Tabel 4.15.

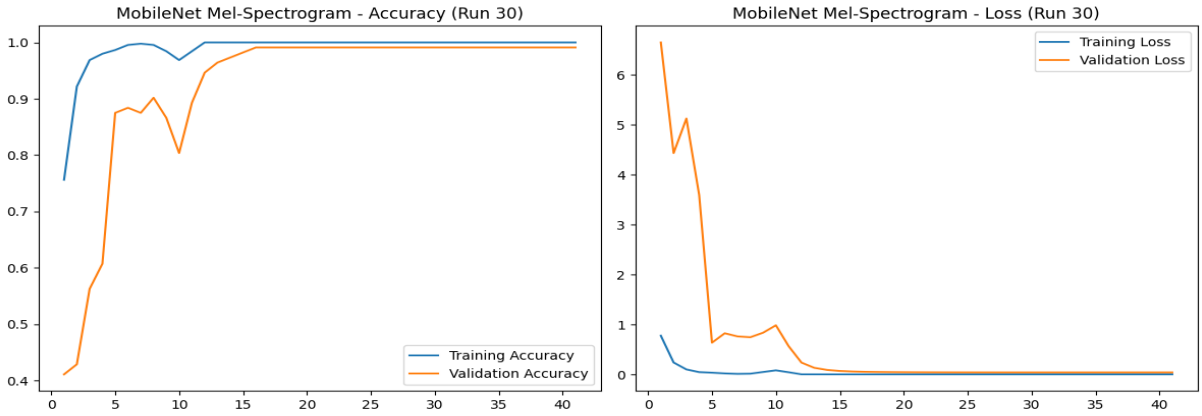
Tabel 4.15 Performa Kombinasi *Hyperparameter* Model MobileNet dengan fitur Mel-Spektogram

No.	*Freeze*	Hyperparameter			Hasil			
		Learning Rate	*Batch*	*Dropout*	*Mean* Akurasi *Training*	*Mean* Loss *Training*	*Mean* Akurasi *Validation*	Mean Loss *Validation*
1.	Ya	0,000001	16	0,5	0,99	0,030630	0,8958	0,339251
2.	Ya	0,000001	32	0,5	0,99	0,031818	0,8988	0,335059
3.	Tidak	0,000001	16	0,5	1,00	0,000039	0,9681	0,142865
4.	Tidak	0,000001	32	0,5	1,00	0,000013	0,9675	0,148537

Berdasarkan hasil pengujian performa model MobileNet dengan fitur Mel-Spektogram yang disajikan pada Tabel 4.15, terlihat bahwa variasi pada pengaturan hyperparameter serta status freeze layer memberikan pengaruh yang cukup signifikan terhadap hasil pelatihan dan validasi. Dari empat kombinasi yang diuji, dua kombinasi tanpa freeze layer menunjukkan performa yang jauh lebih unggul dengan akurasi training mencapai 100% dan nilai loss training

yang sangat kecil, mendekati nol. Kombinasi pertama tanpa freeze layer dengan batch size 16 dan dropout 0,5 menghasilkan akurasi validasi sebesar 96,81% dengan loss validasi 0,1429. Sementara itu, kombinasi kedua tanpa freeze layer dengan batch size 32 dan dropout 0,5 menunjukkan performa yang sedikit menurun pada akurasi validasi menjadi 96,75% dengan loss validasi 0,1485, meskipun perbedaannya sangat kecil.

Sebaliknya, pada dua kombinasi dengan freeze layer aktif, meskipun pelatihan dilakukan selama 100 epoch, akurasi training menurun menjadi sekitar 99% dengan loss training yang relatif lebih tinggi dibandingkan tanpa freeze layer. Akurasi validasi pada konfigurasi freeze layer juga mengalami penurunan, yaitu masing-masing sekitar 89,58% dan 89,88%, dengan loss validasi yang lebih besar dibandingkan konfigurasi tanpa freeze. Hal ini menunjukkan bahwa pembekuan lapisan pada MobileNet membatasi kemampuan model untuk beradaptasi dan belajar dari data pelatihan baru, yang berdampak pada penurunan performa generalisasi. Selain itu, batch size yang lebih besar dan jumlah epoch yang lebih banyak tanpa freeze layer cenderung memberikan hasil validasi yang lebih baik. Secara keseluruhan, kombinasi tanpa freeze layer dengan batch size 16 dan learning rate 0,000001 (Kombinasi 3) memberikan hasil terbaik dengan akurasi validasi yang tinggi dan loss validasi yang rendah, sehingga dianggap sebagai konfigurasi optimal untuk model MobileNet dengan fitur Mel-Spektogram. Grafik performa model pada kombinasi terbaik ini selama proses pelatihan dapat dilihat pada Gambar 4.12 yang menampilkan kurva akurasi dan loss pada data training dan validasi pada salah satu pelatihan.



Gambar 4.12 Nilai Akurasi dan Loss Model MobileNet Mel-Spektogram Kombinasi Ketiga

Dari Grafik yang ditunjukkan pada Gambar 4.12 diketahui bahwa model menunjukkan kestabilan dan keandalan dalam mengklasifikasikan fitur Mel-Spektogram, yang tercermin dari nilai evaluasi yang tinggi dan loss yang rendah.

4.5.2 Klasifikasi Model *MobileNet* dengan Fitur MFCC

Setelah proses pelatihan model MobileNet menggunakan fitur Mel-spektogram selesai, pelatihan serupa juga dilakukan dengan menggunakan fitur MFCC sebagai input pada arsitektur MobileNet yang sama. Tujuan dari proses ini adalah untuk membandingkan efektivitas kedua jenis fitur dalam meningkatkan performa klasifikasi suara burung. Data MFCC yang telah diekstraksi disiapkan dan diolah agar sesuai dengan format input model MobileNet, kemudian model dilatih dengan berbagai kombinasi hyperparameter untuk memperoleh konfigurasi terbaik. Selain itu, setiap konfigurasi dilakukan pelatihan sebanyak 30 kali secara independen agar hasil evaluasi memiliki distribusi yang mendekati normal dan estimasi performanya lebih stabil. Hasil pelatihan dengan fitur MFCC ini kemudian dievaluasi menggunakan metrik seperti akurasi dan loss, sehingga memungkinkan perbandingan langsung dengan hasil pelatihan menggunakan fitur Mel-spektogram. Performansi dari setiap konfigurasi hyperparameter pada

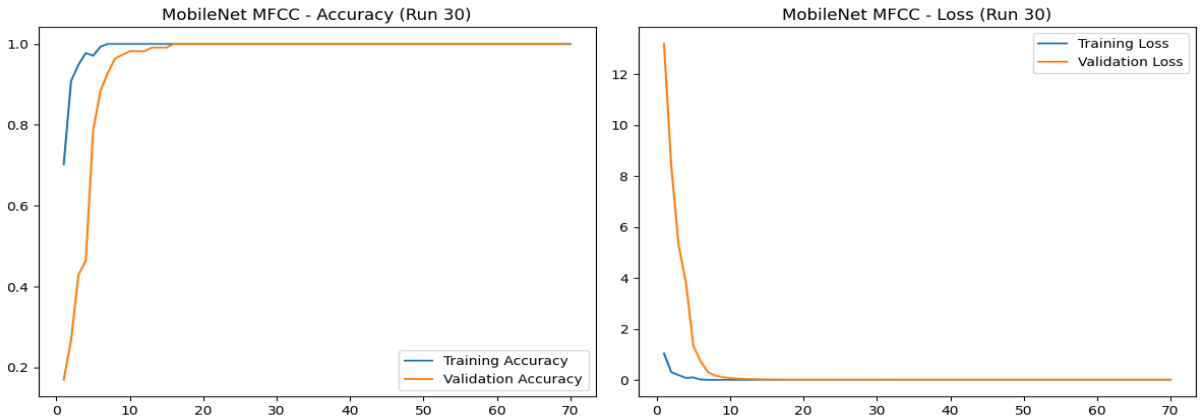
model MobileNet dengan data MFCC disajikan dalam Tabel 4.16 untuk menentukan konfigurasi yang memberikan akurasi dan generalisasi terbaik.

Tabel 4.16 Performa Kombinasi *Hyperparameter* Model MobileNet dengan fitur MFCC

No.	*Freeze*	Hyperparameter			Hasil			
		Learning Rate	*Batch*	*Dropout*	*Mean* Akurasi *Training*	*Mean* Loss *Training*	*Mean* Akurasi *Validation*	Mean Loss *Validation*
1.	Ya	0,000001	16	0,5	1,00	0,007361	0,9366	0,193864
2.	Ya	0,000001	32	0,5	1,00	0,006201	0,9300	0,202851
3.	Tidak	0,000001	16	0,5	1,00	0,000063	0,9646	0,171753
4.	Tidak	0,000001	32	0,5	1,00	0,000015	0,9631	0,175985

Berdasarkan hasil pengujian performa model MobileNet dengan fitur MFCC yang disajikan pada Tabel 4.16, terlihat bahwa variasi pengaturan hyperparameter dan status *freeze* layer memberikan pengaruh yang cukup signifikan terhadap hasil pelatihan dan validasi. Pada dua kombinasi tanpa freeze layer, model menunjukkan performa yang sangat baik dengan akurasi data training mencapai 100% dan loss training yang sangat rendah, mendekati nol. Kombinasi pertama tanpa freeze layer dengan batch size 16 dan dropout 0,5 berhasil mencapai akurasi validasi tertinggi sebesar 96,46% dengan loss validasi 0,1718. Sedangkan kombinasi kedua tanpa freeze layer dengan batch size 32 menghasilkan akurasi validasi sedikit menurun menjadi 96,31% dengan loss validasi 0,1760, meskipun perbedaannya sangat kecil.

Sebaliknya, pada kombinasi dengan freeze layer aktif selama 100 epoch, meskipun akurasi data training tetap tinggi sekitar 99%, terdapat penurunan yang cukup nyata pada akurasi validasi, yaitu menjadi 93,66% dan 93,00% untuk batch size 16 dan 32 berturut-turut. Loss validasi pada kedua konfigurasi ini juga mengalami peningkatan, yang mengindikasikan penurunan kemampuan generalisasi model ketika freeze layer diterapkan. Selain itu, penggunaan batch size yang lebih kecil (16) tanpa freeze layer memberikan hasil validasi yang sedikit lebih baik dibandingkan batch size lebih besar (32) dengan epoch lebih banyak. Dengan demikian, kombinasi ketiga tanpa freeze layer dengan batch size 16 dianggap sebagai konfigurasi paling optimal untuk model MobileNet dengan fitur MFCC, karena mampu menghasilkan keseimbangan terbaik antara akurasi dan loss pada data training dan validasi. Grafik performa model pada kombinasi terbaik ini selama proses pelatihan dapat diamati pada Gambar 4.13 yang menampilkan kurva akurasi dan loss pada data training dan validasi pada salah satu pelatihan.



Gambar 4.13 Nilai Akurasi dan Loss Model *MobileNet* MFCC Kombinasi Pertama

Setelah melihat grafik performa model MobileNet dengan fitur MFCC pada Gambar 4.13, terlihat bahwa konfigurasi hyperparameter tanpa freeze layer dengan dan batch size 16 memberikan keseimbangan yang baik antara kemampuan model dalam mempelajari data pelatihan dan menggeneralisasi pada data validasi.

4.6 Pemodelan dan Evaluasi Klasifikasi dengan VGGish

Model VGGish yang digunakan dalam penelitian ini merupakan model yang telah melalui proses pre-training pada dataset AudioSet. Model VGGish diimplementasikan menggunakan library TensorFlow dengan API Keras. Karena jumlah kelas pada dataset penelitian ini berbeda dengan jumlah kelas pada model pre-trained, lapisan fully connected bawaan dari VGGish tidak digunakan. Sebagai gantinya, lapisan fully connected baru ditambahkan dengan jumlah output yang disesuaikan dengan enam kelas jenis burung yang menjadi fokus penelitian. Penyesuaian ini memungkinkan model untuk melakukan klasifikasi suara burung secara spesifik berdasarkan fitur yang diekstraksi langsung dari waveform audio. Tabel 4.17 menunjukkan arsitektur model VGGish yang telah dimodifikasi dengan penambahan lapisan fully connected baru. Penambahan lapisan ini juga meningkatkan total parameter model dari jumlah awal menjadi sekitar 72,1 juta parameter, sehingga memberikan kapasitas yang lebih besar untuk mempelajari pola-pola kompleks dalam data suara burung.

Tabel 4.17 Arsitektur Model VGGish

Layer Type	Output Shape	Parameter
input (input layer)	(96, 64, 1)	0
conv1 (Conv2D)	(96, 64, 64)	640
MaxPooling2D-1	(48, 32, 64)	0
conv2 (Conv2D)	(48, 32, 128)	73856
MaxPooling2D-2	(24, 16, 128)	0
conv3_1 (Conv2D)	(24, 16, 256)	295168
conv3_2 (Conv2D)	(24, 16, 256)	590080
MaxPooling2D-3	(12, 8, 256)	0
conv4_1 (Conv2D)	(12, 8, 512)	1180160
conv4_2 (Conv2D)	(12, 8, 512)	2359808
MaxPooling2D-4	(6, 4, 512)	0
fc1_1 (Dense)	(12288)	50331648
fc1_2 (Dense)	(4096)	16777216
fc2 (Dense)	(128)	524416
dense_4 (Dense)	(64)	8256
dense_5 (Dense)	(6)	390
Total Parameter		**72141184**

Model VGGish yang digunakan dalam penelitian ini merupakan arsitektur CNN yang telah dipra-latih (*pretrained*) oleh Google pada dataset AudioSet, dan dirancang khusus untuk mengolah data audio mentah dalam bentuk waveform. Tidak seperti model CNN atau MobileNet yang membutuhkan fitur yang telah diekstraksi sebelumnya seperti Mel-spektrogram atau MFCC, VGGish menerima input berupa raw waveform yang kemudian diproses langsung menjadi log Mel-spectrogram secara internal. Proses ekstraksi fitur ini mencakup tahapan seperti framing, windowing, Fourier Transform, penghitungan Mel-spectrogram, dan konversi ke skala log, yang semuanya dilakukan oleh preprocessor VGGish

sebelum fitur-fitur tersebut diteruskan ke jaringan CNN. Input waveform dengan sampling rate tertentu (16 kHz) diubah menjadi blok fitur berukuran tetap yaitu (96, 64, 1), yang merepresentasikan 96 frame waktu dan 64 band frekuensi Mel, kemudian dimasukkan ke jaringan konvolusional. Blok konvolusi pertama terdiri dari Conv2D dengan 64 filter berukuran 3×3 dan menghasilkan 64 feature map, disusul dengan pooling. Parameter pada layer ini dihitung dari jumlah filter dikalikan ukuran kernel dan jumlah channel input, ditambah bias (misalnya: 3×3×1×64 + 64 = 640 parameter). Jumlah parameter terus meningkat seiring bertambahnya jumlah filter pada layer-layer berikutnya (128, 256, hingga 512 filter). Dua lapisan dense besar di akhir jaringan memiliki jumlah parameter yang sangat tinggi misalnya, dense dari 12288 ke 4096 neuron memerlukan lebih dari 50 juta parameter sendiri (12288 × 4096 + 4096). Total parameter dari seluruh jaringan dasar VGGish mencapai sekitar 72 juta, yang mencakup seluruh bobot konvolusi, dense, dan bias. Dalam penelitian ini, dua dense layer tambahan ditambahkan di atas VGGish, satu dengan 64 unit (menghasilkan 8256 parameter) dan satu lagi sebagai layer output dengan 6 unit kelas dan aktivasi softmax (menambahkan 390 parameter). Tambahan ini berfungsi menyesuaikan VGGish untuk klasifikasi enam jenis suara burung. Secara keseluruhan, total parameter model VGGish yang dimodifikasi dalam penelitian ini adalah sekitar 72141184.

Evaluasi performa model dilakukan berdasarkan metrik akurasi dan loss pada data training dan validation, yang kemudian dibandingkan untuk menentukan pengaturan hyperparameter optimal dalam klasifikasi jenis burung menggunakan input waveform melalui model VGGish. Untuk performa beberapa kombinasi model VGGish ditunjukkan pada Tabel 4.18.

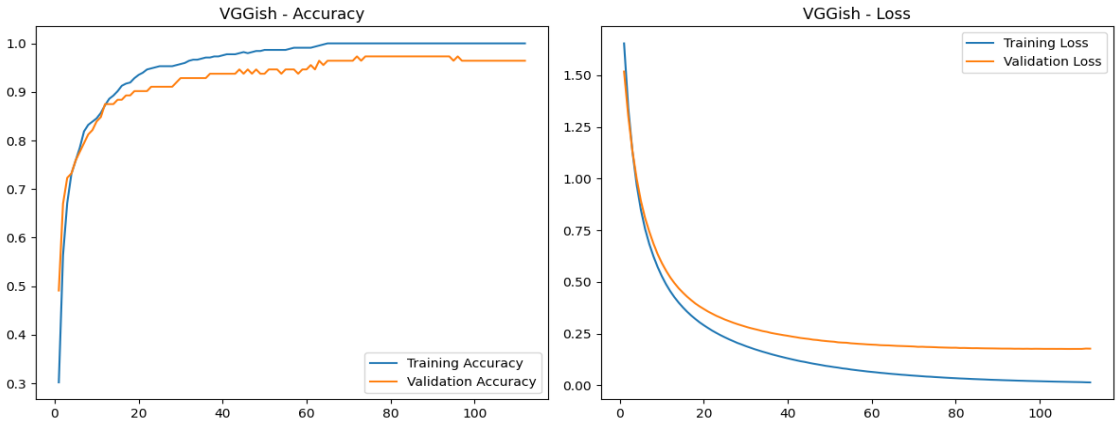
Tabel 4.18 Performa Kombinasi *Hyperparameter* Model VGGish

No.	**Hyperparameter**			**Hasil**			
	Learning Rate	***Batch***	***Dropout***	***Mean* Akurasi *Training***	***Mean* Akurasi *Training***	***Mean* Akurasi *Validation***	***Mean* Loss *Validation***
1.	0,000001	16	0,5	1,00	0,009292	0,9485	0,174700
2.	0,000001	32	0,5	1,00	0,008162	0,9467	0,169665

Berdasarkan hasil pengujian performa model VGGish dengan input waveform yang ditampilkan pada Tabel 4.18, terlihat bahwa variasi konfigurasi hyperparameter memberikan pengaruh terhadap kemampuan generalisasi model, meskipun akurasi training tetap konsisten mencapai 100%. Pada kombinasi pertama dengan learning rate 0,000001, batch size 16, dan dropout 0,5, model berhasil mencapai akurasi validasi tertinggi sebesar 94,85% dengan loss validasi sebesar 0,1747. Sedangkan pada kombinasi kedua dengan batch size 32 dan parameter yang sama, model memperoleh akurasi validasi sedikit menurun menjadi 94,67% dengan loss validasi 0,1697.

Perbedaan performa ini menunjukkan bahwa penggunaan batch size yang lebih kecil (16) memberikan kestabilan pembelajaran yang lebih baik pada data validasi, meskipun jumlah epoch pada kombinasi pertama lebih sedikit dibandingkan kombinasi kedua. Hal ini kemungkinan disebabkan oleh batch size kecil yang memungkinkan pembaruan bobot model lebih sering dan presisi, yang sangat penting dalam menangani kompleksitas temporal dari data waveform. Nilai loss validasi yang sedikit lebih tinggi pada kombinasi pertama masih menunjukkan kemampuan model yang baik dalam menghindari overfitting. Keunggulan kombinasi pertama ini menegaskan bahwa pemanfaatan bobot pre-trained VGGish pada dataset suara burung dapat berjalan optimal dengan pengaturan batch size kecil dan durasi pelatihan yang terkontrol. Grafik akurasi dan loss selama proses pada salah satu pelatihan dapat diamati

pada Gambar 4.14, yang memperlihatkan konvergensi model secara stabil pada data training dan validasi.



Gambar 4.14 Nilai Akurasi dan Loss Model VGGish Kombinasi Pertama

Dari grafik akurasi dan loss yang ditampilkan pada Gambar 4.14, dapat diamati bahwa model VGGish menunjukkan proses konvergensi yang stabil selama pelatihan. Akurasi pada data training dan validasi meningkat secara konsisten seiring bertambahnya epoch, sedangkan nilai loss pada kedua dataset cenderung menurun dan tidak menunjukkan lonjakan signifikan. Hal ini mengindikasikan bahwa model mampu belajar secara efektif dari data tanpa mengalami *overfitting*. Pola tren yang sejalan antara akurasi dan loss pada data training dan validation juga memperkuat keyakinan bahwa konfigurasi hyperparameter yang digunakan sudah sesuai dalam mendukung performa optimal model pada tugas klasifikasi suara burung.

4.7 Pemilihan Model Terbaik

Penelitian ini menggunakan tiga model klasifikasi utama, yaitu CNN, MobileNet, dan VGGish, dengan pendekatan input yang berbeda untuk mengoptimalkan pengenalan suara burung. Model CNN dan MobileNet menerima input berupa nilai numerik hasil ekstraksi fitur Mel-spektogram dan MFCC, yang masing-masing merepresentasikan distribusi energi frekuensi dan koefisien spektral dari sinyal audio secara langsung. Kedua model ini memanfaatkan kemampuan konvolusi untuk mengekstraksi pola-pola lokal penting dari fitur dua dimensi tersebut, yang secara luas digunakan dalam pengolahan suara. Sementara itu, model VGGish menggunakan input berupa waveform audio mentah yang diproses secara langsung oleh model untuk mengekstraksi fitur akustik secara hierarkis. VGGish merupakan model pra-latih yang telah dilatih pada dataset AudioSet besar, sehingga mampu menangkap representasi audio yang kompleks tanpa perlu ekstraksi fitur manual seperti Mel-spektogram atau MFCC. Pendekatan ini memungkinkan model untuk belajar langsung dari sinyal asli, yang dapat meningkatkan performa klasifikasi dalam kondisi data yang bervariasi.

Dengan kombinasi ketiga model dan dua jenis input fitur ini, penelitian bertujuan untuk membandingkan efektivitas masing-masing metode dalam mengklasifikasikan jenis burung berdasarkan suara, serta menentukan konfigurasi terbaik yang memberikan akurasi dan generalisasi optimal pada dataset yang digunakan. Tabel 4.19 menunjukkan hasil performa terbaik dari masing-masing model yang telah diuji, sedangkan hasil evaluasi akhir pada data *test* melalui metrik akurasi, recall, precision, dan F1-score ditunjukkan pada Tabel 4.20.

Tabel 4.19 Performa Terbaik Setiap Model

Model	*Mean Train Accuracy*	*Mean Train Loss*	*Mean Validation Accuracy*	*Mean Validation Loss*
CNN (Mel-Spektogram)	1,00	0,002472	0,9190	0,365826
CNN (MFCC)	1,00	0,000368	0,9797	0,110764
MobileNet (Mel-Spektogram)	1,00	0,000039	0,9681	0,142865
MobileNet (MFCC)	1,00	0,000063	0,9646	0,171753
VGGish (Waveform)	1,00	0,009292	0,9485	0,174700

Tabel 4.20 Hasil Evaluasi Setiap Model pada Data *Test*

Model	*Test Accuracy*	*Test Recall*	*Test Precision*	*Test F1-Score*
CNN (Mel-Spektogram)	0,91161	0,917915	0,90800	0,910245
CNN (MFCC)	0,97976	0,982743	0,97782	0,979612
MobileNet (Mel-Spektogram)	0,96934	0,971222	0,96598	0,967385
MobileNet (MFCC)	0,96607	0,968407	0,96376	0,964899
VGGish (Waveform)	0,95029	0,952212	0,94742	0,948519

Berdasarkan hasil evaluasi performa yang telah dilakukan, model terbaik dalam penelitian ini adalah CNN dengan fitur MFCC kombinasi pertama dengan batch size 16. Model ini berhasil mencapai akurasi pengujian tertinggi sebesar 97,97% dengan nilai F1-score 0,9796, yang menunjukkan kemampuan klasifikasi suara burung yang sangat baik. Keunggulan CNN dengan MFCC juga didukung oleh performa stabil pada data training dan validasi, dengan loss yang rendah serta akurasi validasi mencapai hampir 98%. Hal ini sejalan dengan berbagai penelitian sebelumnya yang menunjukkan bahwa ekstraksi fitur MFCC mampu menangkap karakteristik suara secara efektif, sementara CNN mampu mengenali pola-pola kompleks dalam data tersebut secara otomatis dan efisien.

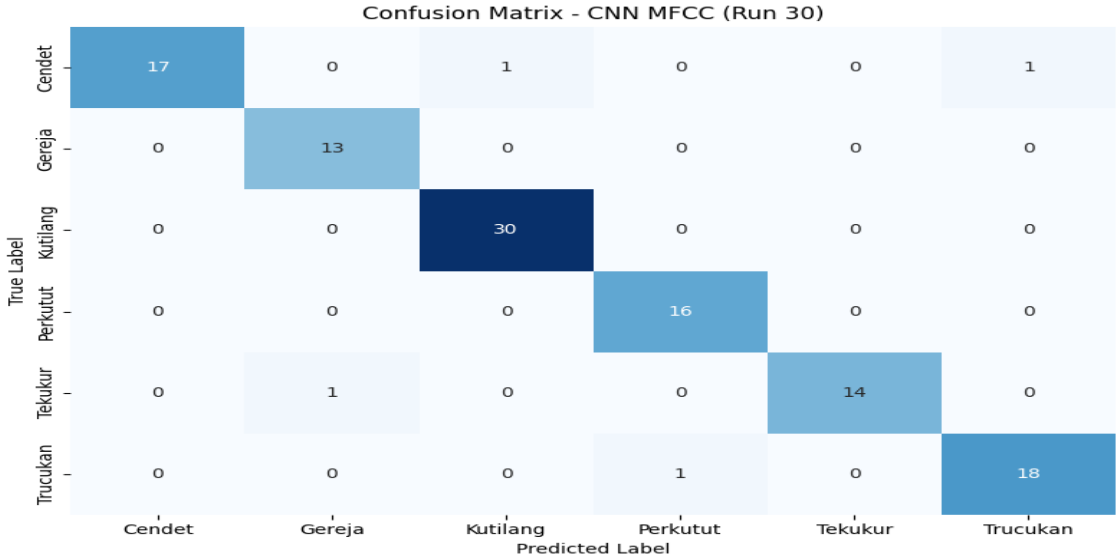
Selain itu, CNN dengan MFCC memiliki keunggulan dalam hal pemrosesan sinyal suara karena MFCC meniru cara telinga manusia menangkap frekuensi suara, sehingga fitur yang dihasilkan sangat representatif untuk klasifikasi suara burung. Dengan arsitektur CNN yang mampu mengekstraksi fitur secara hierarkis dan otomatis, model ini memberikan hasil yang akurat dan tahan terhadap variasi suara burung yang berbeda. Keunggulan kombinasi ini menjadi sangat penting dalam konteks penelitian ini, mengingat total data yang digunakan relatif terbatas, yaitu sebanyak 559 data suara. Dengan performa tinggi pada seluruh metrik evaluasi baik pada data *training*, *validation*, maupun data *test* serta efisiensi parameter dan arsitektur yang ringan, maka model CNN dengan fitur MFCC dipilih sebagai model terbaik dalam penelitian ini.

4.8 Implementasi Model Terbaik pada Data Test

Model dengan performa terbaik yang telah diperoleh dalam penelitian ini, yaitu CNN dengan fitur MFCC dengan batch size 16, selanjutnya diterapkan pada dataset yang tidak digunakan dalam proses pelatihan maupun validasi, yaitu data *test*. Data *test* ini terdiri dari rekaman suara enam jenis burung yang berbeda, yaitu Kutilang, Gereja, Perkutut, Tekukur, Trucukan, dan Cendet, dengan total sebanyak 112 sampel suara. Data *test* ini dipilih untuk mengevaluasi sejauh mana kemampuan model dalam mengklasifikasikan jenis suara burung pada kondisi nyata, khususnya dalam konteks lingkungan di Indonesia, karena seluruh data audio merupakan jenis burung lokal yang umum ditemukan di berbagai wilayah Indonesia.

Setelah data uji dimuat dan diproses menjadi representasi MFCC, model CNN digunakan untuk memprediksi kategori dari masing-masing sampel audio tersebut. Hasil prediksi tersebut

kemudian divisualisasikan dalam bentuk confusion matrix yang dapat dilihat pada Gambar 4.15. Visualisasi ini memberikan gambaran sejauh mana model mampu mengenali setiap jenis burung secara spesifik, serta memperlihatkan jika terdapat kesalahan klasifikasi antar kelas yang memiliki kemiripan suara.



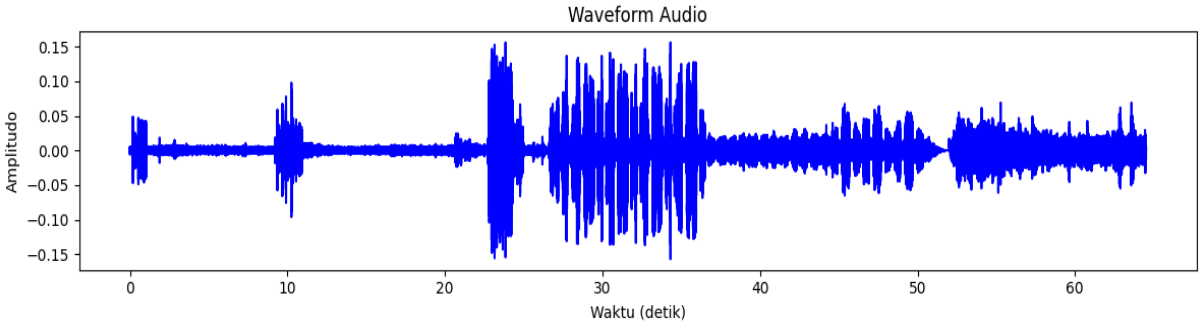
Gambar 4.15 *Confusion Matrix* Prediksi Model Terbaik pada Data Test

Confusion matrix pada Gambar 4.15 menunjukkan performa klasifikasi model CNN dengan fitur MFCC terhadap suara burung pada masing-masing kategori. Secara umum, model mampu mengklasifikasikan data dengan tingkat ketepatan yang sangat tinggi. Kategori Kutilang, Perkutut, dan Trucukan menunjukkan performa prediksi sempurna, di mana seluruh sampel pada masing-masing kelas berhasil diprediksi dengan benar oleh model. Pada kategori Cendet, dari 19 sampel, model berhasil mengklasifikasikan 17 sampel dengan benar, sementara 1 sampel salah diklasifikasikan sebagai Kutilang dan 1 sampel sebagai Trucukan. Untuk kategori Gereja, seluruh 13 sampel berhasil diprediksi dengan tepat tanpa kesalahan. Sedangkan pada kategori Tekukur, dari 15 sampel, 14 berhasil diklasifikasikan dengan benar dan 1 sampel salah diklasifikasikan sebagai Gereja. Hasil ini menunjukkan bahwa model CNN MFCC sangat baik dalam mengenali pola suara dari sebagian besar jenis burung, meskipun masih terdapat sedikit kesalahan klasifikasi pada beberapa kelas, khususnya pada Cendet dan Tekukur. Secara keseluruhan, hasil ini menegaskan keandalan model CNN dengan fitur MFCC dalam tugas klasifikasi suara burung pada dataset yang digunakan.

Setelah mendapatkan performa terbaik dari model CNN dengan fitur MFCC, langkah berikutnya adalah menerapkan model ini pada rekaman suara berdurasi panjang yang merekam aktivitas berbagai jenis burung di lingkungan asli mereka. Tujuannya adalah untuk menguji kemampuan model dalam mendeteksi jenis burung yang muncul dalam suatu rekaman serta menghitung berapa kali suara dari masing-masing jenis burung terdeteksi dalam satu periode waktu tertentu. Pendekatan ini lebih mencerminkan skenario dunia nyata, di mana suara burung terekam secara kontinu dalam lingkungan terbuka dan alami.

Untuk merealisasikan hal tersebut, rekaman suara akan dipotong menjadi beberapa segmen pendek dengan metode VAD (*Voice Active Detection*), kemudian setiap segmen akan dikonversi ke dalam bentuk Mel-Spektogram dan diprediksi menggunakan model. Hasil prediksi dari seluruh segmen kemudian akan direkapitulasi untuk mengetahui frekuensi kemunculan masing-masing jenis burung. Dengan demikian, model tidak hanya digunakan untuk klasifikasi tunggal, tetapi juga untuk analisis distribusi jenis burung dalam suatu wilayah atau waktu tertentu, yang

dapat dimanfaatkan dalam konteks monitoring biodiversitas dan konservasi lingkungan. Sebagai implementasi dari pendekatan tersebut, pada tahap ini akan dilakukan prediksi terhadap sebuah contoh rekaman suara burung berdurasi 53 detik. Rekaman tersebut akan diproses melalui tahap segmentasi menggunakan metode VAD, diikuti dengan ekstraksi fitur Mel-Spektogram untuk setiap segmen yang dihasilkan. Selanjutnya, model klasifikasi yang telah dilatih akan digunakan untuk mengidentifikasi jenis burung pada masing-masing segmen, sehingga diperoleh informasi mengenai jenis burung apa saja yang muncul serta seberapa sering kemunculannya dalam rekaman tersebut. Pada Gambar 4.16 menunjukkan bentuk gelombang (waveform) dari audio berdurasi 53 detik yang akan digunakan sebagai input dalam proses klasifikasi suara burung.



Gambar 4.16 Suara yang akan Diprediksi

Tabel 4.21 Hasil Prediksi Berdasarkan Jumlah Deteksi masing-masing Jenis Burung

No.	Jenis Burung	Jumlah Terdeteksi
1.	Trucukan	7
2.	Tekukur	9
3.	Gereja	5
4.	Kutilang	2

Berdasarkan hasil deteksi pada Tabel 4.21, dapat disimpulkan bahwa rekaman audio mengandung aktivitas vokalisasi dari burung Trucukan sebanyak 7 kali, burung Tekukur sebanyak 9 kali, burung Gereja terdeteksi sebanyak 5 kali, dan burung kutilang sebanyak 2 kali. Distribusi jumlah deteksi ini memberikan gambaran mengenai komposisi akustik lingkungan pada saat perekaman dilakukan. Jika pendekatan ini diterapkan secara kontinu dan sistematis, maka dapat digunakan untuk memonitor dinamika populasi burung di suatu lokasi secara non-invasif dan efisien. Dengan demikian, teknologi ini tidak hanya mendukung penelitian ilmiah, tetapi juga sangat potensial untuk diaplikasikan dalam sistem pemantauan keanekaragaman hayati secara otomatis.

BAB 5 PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, diperoleh beberapa kesimpulan sebagai berikut:

1. Penelitian ini menggunakan lima model klasifikasi untuk mengidentifikasi suara dari enam spesies burung, yaitu CNN (Mel-Spektrogram), CNN (MFCC), MobileNet (Mel-Spektrogram), MobileNet (MFCC) dan VGGish (waveform). Data yang digunakan berasal dari situs *xeno-canto.org*, dengan masing-masing spesies terdiri atas 10 rekaman berdurasi 11–115 detik. Seluruh rekaman diolah dengan proses ekstraksi fitur untuk menghasilkan representasi audio yang sesuai dengan input masing-masing model. Model-model tersebut dibangun dan dijalankan menggunakan framework TensorFlow dan Keras. Selanjutnya, dilakukan proses pelatihan dan tuning hiperparameter untuk mendapatkan kinerja optimal dari masing-masing arsitektur. Evaluasi performa dilakukan dengan menggunakan metrik akurasi dan loss pada data *training* dan *validation,* kemudian menggunakan metrik akurasi, precision, recall, dan F1-score pada data *test*, untuk menilai efektivitas masing-masing model dalam mengklasifikasikan jenis burung berdasarkan karakteristik suara. Setelah seluruh model dievaluasi, model dengan performa terbaik digunakan untuk proses klasifikasi lanjutan terhadap rekaman audio lingkungan.
2. Berdasarkan Berdasarkan hasil perbandingan performa dari masing-masing model yang diuji, diperoleh bahwa model CNN dengan input MFCC menunjukkan performa paling unggul dalam klasifikasi enam jenis suara burung, yaitu Kutilang, Gereja, Perkutut, Tekukur, Trucukan, dan Cendet. Model ini berhasil mencapai rata-rata akurasi pada data validasi sebesar 97,97% dengan nilai loss validasi 0,110764, serta akurasi pada data test sebesar 97,97%. Selain akurasi, model ini juga menunjukkan nilai precision sebesar 97,78%, recall sebesar 98,27%, dan F1-score sebesar 97,96% pada data test, yang mengindikasikan keseimbangan yang sangat baik antara ketepatan dan kemampuan deteksi model terhadap berbagai kelas suara burung. Kinerja ini mengungguli model-model lain seperti CNN dengan Mel-Spektogram, MobileNet dengan Mel-Spektogram dan MFCC, serta VGGish dengan input waveform mentah. Tingginya akurasi dan metrik evaluasi lain pada data *test* membuktikan bahwa model CNN dengan MFCC tidak hanya mampu mengenali pola-pola pada data pelatihan dan validasi, tetapi juga memiliki kemampuan generalisasi yang sangat baik terhadap data baru yang belum pernah dilihat sebelumnya. Hasil ini memperkuat bahwa CNN dengan fitur MFCC merupakan pendekatan yang sangat efektif dan andal untuk tugas klasifikasi suara burung, terutama dalam konteks dataset yang relatif terbatas. Dengan performa yang konsisten tinggi pada seluruh metrik evaluasi, model ini menjadi pilihan utama dalam penelitian ini untuk mengembangkan sistem klasifikasi suara burung yang akurat dan robust.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, berikut beberapa saran yang dapat dipertimbangkan untuk pengembangan lebih lanjut:

1. Penelitian ini masih dapat ditingkatkan dengan menambah jumlah dan variasi data suara burung, baik dari segi jumlah sampel maupun durasi rekaman. Semakin banyak data yang digunakan, semakin baik pula model dalam mempelajari karakteristik akustik masing-

masing burung. Selain itu, penting juga untuk memperhatikan kualitas pelabelan data agar model dapat belajar dari representasi yang tepat.

2. Untuk meningkatkan performa, penelitian selanjutnya dapat mengeksplorasi arsitektur CNN lainnya seperti EfficientNet, ResNet, atau VGGish, serta mempertimbangkan penggunaan model hybrid seperti CNN-LSTM yang mampu menangkap informasi spasial dan temporal sekaligus dari fitur audio.
3. Model klasifikasi yang dihasilkan dalam penelitian ini dapat dikembangkan lebih lanjut menjadi aplikasi sistem monitoring otomatis berbasis audio untuk deteksi spesies burung di alam liar, sehingga dapat membantu pekerjaan peneliti, konservasionis, atau pihak kehutanan dalam melakukan pengamatan populasi burung secara efisien dan non-invasif.

DAFTAR PUSTAKA

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., & Brain, G. (2016). Tensorflow: A system for large-scale machine learning. *12th USENIX Symposium*, 265.

Afida, A. (2020). Klasifikasi jenis burung berdasarkan suara menggunakan algoritme Support Vector Machine. *MATHunesa Jurnal Ilmiah Matematika 8(1)*, 1-6.

AI, G. (2025). *Audio classification guide*. Retrieved from https://ai.google.dev/edge/mediapipe/solutions/audio/audio_classifier

Ali, M. F. (2020). *Klasifikasi Jenis Burung Berdasarkan Suara Burung Berbasis Deep Learning.* Surabaya: Institut Teknologi Sepuluh Nopember.

Ali, N., Neagu, D., & Trundle, P. (2019). Evaluation of k-nearest neighbour classifier. *SN Applied Sciences, 1(12).*, 1(12). Retrieved from https://doi.org/10.1007/s42452-019-1356-9

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., . . . Farhan, L. (2021). Review of deep learning: concepts, CNN architechtures, challenges, applications, future directions. *Journal of Big Data*, 8(1).

Amrilla, M. F. (2023). *Speech Recognition Pada Dialek Bahasa Sunda Menggunakan Metode Convolutional Neural Network Dengan Ektraksi Ciri Mel-Spectogram.* Jakarta: UIN Syarif Hidayatullah.

Anderson, D. L. (2019). Acoustic Communication in Neotropical Birds. *Tropical Ecology and Conservation, 12(3)*, 134-151.

Baldock, R., Mennel, H., & Nesyhabur, B. (2021). Deep Learning Through the Lens of Example Difficulty. *35th Conference on Neural Information Processing Systems*, (pp. 1-14).

Buono, A. (2020). *Deep Learning: Dasar Teori dan Penerapannya.* Jakarta: Penerbit Informatika.

Confusion matrix in machine learning. (2021). Retrieved from https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ (Accessed: 2024-10-16).

Effendi, H., & Saryadi, R. (2023). Perancangan Sistem Deteksi dan Pengenalan Perintah Suara Menggunakan Modul ESP 32 dengan Metode CNN. *Sainstech Vol. 33 No. 4*.

Fahrizal. (2024). *Apa Itu Tensorflow dan Mengapa Penting?* Retrieved from TopKode: https://topkode.com/apa-itu-tensorflow-dan-mengapa-penting/

Fazry. (2024). *Pengantar TensorFlow: Membangun Model Machine Learning dengan Framework Populer – Bagian 1*. Retrieved from RumahCoding: https://rumahcoding.co.id/pengantar-tensorflow-membangun-model-machine-learning-dengan-framework-populer-bagian-1/

Gill, F. B. (2007). Ornithology (Third Edition). W.H. Freeman and Company.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. *In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, pp. 315-323.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

Hatma, G. A. (2024). *Klasifikasi Suara Jenis Burung Menggunakan Deep Learning Berbasis Compressive Sensing.* Bandung: Universitas Telkom.

Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., . . . Wilson, K. (2017). CNN architectures for large-scale audio classification. *2017 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. pp. 131–135). IEEE.

Ho, Y., & Wookey, S. (2020). *The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling.*

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.048614*.

Ihsanti, S. (2024). Klasifikasi genus burung hantu berdasarkan suara menggunakan Convolutional Neural Network. *e-Proceeding of Engineering 11, no. 4 (2024)*, 4532–4541.

Ilmananda, A. (2012). *Aplikasi Penganalisis Spektrum Sinyal Digital Secara Real-Time Menggunakan Algoritma Fast Fourier Transform (FFT) Pada Perangkat Mobile Phone.* Malang: Universitas Brawijaya.

Imaroh, F. (2024). *Cara Kerja Deep Learning dalam Pengenalan Suara*. Retrieved from Advan Blog: https://blog.advan.id/15138/cara-kerja-deep-learning-dalam-pengenalan-suara/

Irwandi, Marwan, Hadi Mahmud, A., & Abdullah, &. (2005). Upaya Pemanfaatan Rekaman Suara Burung dan Analisis Spektrogram untuk Menyusun Metode Klasifikasi Berdasarkan Suara (Sonotaksonomi). *Biosfera*, 18–24.

Krizhevsky, A. e. (2012). ImageNet Classification with Deep Convolutional Neural Network. *Advances in Neural Information Processing Systems, 25*, 1097-1105.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

Lubis, N. (2024). Implementasi Algoritma Deep Learning pada Aplikasi Speech to Text. *Sudo Jurnal Teknik Informatika, 3(3)*, 120-125.

Md Anwar Hossain, B., Shahriar Alam Sajib, M., Anwar Hossain, M., & Shahriar Alam Sajib, M. (2019). *Classification of Image using Convolutional Neural Network (CNN).*

Mitchell, T. (2019). *Machine Learning and Sound Processing.* Cambridge University Press.

Patmadi, F. (2022). *Klasifikasi Emosi pada Suara dengan Ektraksi Ciri Mel Fequency Ceptral Coefficients Menggunakan Metode 1D Convolutional Neural Network.* Jakarta: UIN Syarif Hidayatullah.

Ponnusamy, R. (2017). A Review of Image Classification Approaches and Techniques. *International Journal of Recent Trends in Engineering and Research*, 3(3), 1-5.

Powers, D., & Ailab. (2020). Evaluation: From precision, recall and f-measure to roc, informedness, markedness correlation. *J. Mach. Learn. Technol*, 2 , 2229-3981. doi:doi: 10.9735/2229-3981

Pratama, H., & Sam'an, M. (2023). Implementasi Metode Interpolasi Bilinear Untuk Perbesaran Skala Citra. *Jurnal Komunikasi dan Teknik Informatika (JKTI), 1(1)*, 21-25.

Putra, B. P. (2019). *Klasifikasi suara untuk menentukan kualitas burung lovebird menggunakan metode Mel Frequency Cepstral Coefficients dan Dynamic Time Warping.* Yogyakarta: Universitas Pembangunan Nasional Veteran Yogyakarta.

Putra, R. (2025). *Speech Processing in Python*. Retrieved from Bisa AI: https://bisa.ai/portofolio/detail/MzM4Ng

Rahman, A. Y. (2022). Klasifikasi Citra Burung Jalak Menggunakan Artificial Neural Network dan Random Forest. *Jurnal Edukasi dan penelitian Informatika*.

Ramírez, J., Segura, J. C., Benítez, C., García, L., & Rubio, A. (2004). Voice Activity Detection with Noise Reduction and Long-Term Spectral Divergence Estimation for Robust

Speech Recognition. *IEEE Transactions on Speech and Audio Processing, 12(3)*, 265–275.

Restackio. (2025, May 25). *Transfer learning techniques with YAMNet*. Retrieved from Restack io: https://www.restack.io/p/transfer-learning-answer-yamnet-techniques-cat-ai

Russell, B. e. (2020). The Role of Passive Acoustic Monitoring in Wildlife Conservation. *Journal of Wildlife Research, 45(2)*, 88-102.

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*.

Soraya. (2024). Klasifikasi Genus Burung Hantu Berdasarkan Suara Menggunakan Convolutional Neural Network. *Journal eproc*.

Tensorflow. (2024). *tf.signal.mfccs_from_log_mel_spectrograms*. Retrieved from tensorflow.org: https://www.tensorflow.org/api_docs/python/tf/signal/mfccs_from_log_mel_spectrograms

tensorflow.org. (2025).

van deer Laak, J., Litjens, G., & Ciompi, F. (2021). Deep learning in histopathology: the path to the clinic. *Nat Med 27*, 775–784. Retrieved from https://doi.org/10.1038/s41591-021-01343-4

Vyas, G., & Kumari, B. (2013). Speaker Recognition System Based on MFCC and DCT. *International Journal of engineering and advanced technology*, 2, 167-169.

Wijaya, H. (2024). Teknologi Pengenalan Suara tentang Metode, Bahasa dan Tantangan. *Bit-Tech, 7(2)*, 534–540.

(Halaman ini sengaja dikosongkan)

LAMPIRAN

Lampiran 1. Arsitektur Model MobileNet

Layer (Type)	Output Shape	Parameters
InputLayer	(224, 224, 3)	0
Conv2D (Conv1)	(112, 112, 32)	864
BatchNormalization	(112, 112, 32)	128
ReLU	(112, 112, 32)	0
DepthwiseConv2D (dw_1)	(112, 112, 32)	288
BatchNormalization	(112, 112, 32)	128
ReLU	(112, 112, 32)	0
Conv2D (pw_1)	(112, 112, 64)	2048
BatchNormalization	(112, 112, 64)	256
ReLU	(112, 112, 64)	0
DepthwiseConv2D (dw_2, stride=2)	(56, 56, 64)	576
BatchNormalization	(56, 56, 64)	256
ReLU	(56, 56, 64)	0
Conv2D (pw_2)	(56, 56, 128)	8192
BatchNormalization	(56, 56, 128)	512
ReLU	(56, 56, 128)	0
DepthwiseConv2D (dw_3)	(56, 56, 128)	1152
BatchNormalization	(56, 56, 128)	512
ReLU	(56, 56, 128)	0
Conv2D (pw_3)	(56, 56, 128)	16384
BatchNormalization	(56, 56, 128)	512
ReLU	(56, 56, 128)	0
DepthwiseConv2D (dw_4, stride=2)	(28, 28, 128)	1152
BatchNormalization	(28, 28, 128)	512
ReLU	(28, 28, 128)	0
Conv2D (pw_4)	(28, 28, 256)	32768
BatchNormalization	(28, 28, 256)	1024
ReLU	(28, 28, 256)	0
DepthwiseConv2D (dw_5)	(28, 28, 256)	2304
BatchNormalization	(28, 28, 256)	1024
ReLU	(28, 28, 256)	0
Conv2D (pw_5)	(28, 28, 256)	65536

Layer (Type)	Output Shape	Parameters
BatchNormalization	(28, 28, 256)	1024
ReLU	(28, 28, 256)	0
DepthwiseConv2D (dw_6, stride=2)	(14, 14, 256)	2304
BatchNormalization	(14, 14, 256)	1024
ReLU	(14, 14, 256)	0
Conv2D (pw_6)	(14, 14, 512)	131072
BatchNormalization	(14, 14, 512)	2048
ReLU	(14, 14, 512)	0
DepthwiseConv2D (dw_7)	(14, 14, 512)	4608
BatchNormalization	(14, 14, 512)	2048
ReLU	(14, 14, 512)	0
Conv2D (pw_7)	(14, 14, 512)	262144
BatchNormalization	(14, 14, 512)	2048
ReLU	(14, 14, 512)	0
(×5 blocks similar with 512 filters)	(14, 14, 512)	...
DepthwiseConv2D (dw_8, stride=2)	(7, 7, 512)	4608
BatchNormalization	(7, 7, 512)	2048
ReLU	(7, 7, 512)	0
Conv2D (pw_8)	(7, 7, 1024)	524288
BatchNormalization	(7, 7, 1024)	4096
ReLU	(7, 7, 1024)	0
DepthwiseConv2D (dw_9)	(7, 7, 1024)	9216
BatchNormalization	(7, 7, 1024)	4096
ReLU	(7, 7, 1024)	0
Conv2D (pw_9)	(7, 7, 1280)	1310720
BatchNormalization	(7, 7, 1280)	5120
ReLU	(7, 7, 1280)	0
GlobalAveragePooling2D	(1280)	0
Dense (Softmax, 6 classes)	(6)	7686
Total Parameters		**4253894**

Lampiran 2. Syntax untuk Preprocessing Data dan Trimmng Audio

```
def trim_audio(audio, sr, threshold_db=-45, min_duration=1, max_duration=3, hop_length=512):
    # Deteksi aktivitas suara
```

```
    intervals = librosa.effects.split(audio, top_db=-threshold_db, frame_length=2048, hop_length=hop_length)

    trimmed_segments = []
    for start, end in intervals:
        segment = audio[start:end]
        segment_duration = librosa.get_duration(y=segment, sr=sr)

        # Jika segmen terlalu panjang, potong menjadi beberapa bagian
        if segment_duration > max_duration:
            n_segments = int(np.ceil(segment_duration / max_duration))
            segment_length = int(len(segment) / n_segments)
            for i in range(n_segments):
                sub_segment = segment[i*segment_length:(i+1)*segment_length]
                if librosa.get_duration(y=sub_segment, sr=sr) >= min_duration:
                    trimmed_segments.append(sub_segment)
        elif segment_duration >= min_duration:
            trimmed_segments.append(segment)

    return trimmed_segments


def process_and_save_audio(input_path, output_path, class_name):
    audio, sr = librosa.load(input_path, sr=22050)
    trimmed_segments = trim_audio(audio, sr)

    # Buat folder output jika belum ada
    os.makedirs(os.path.join(output_path, class_name), exist_ok=True)

    # Simpan setiap segmen sebagai file terpisah
    for i, segment in enumerate(trimmed_segments):
        output_file = os.path.join(output_path, class_name, f"{os.path.splitext(os.path.basename(input_path))[0]}_segment_{i+1}.wav")
        sf.write(output_file, segment, sr)


def process_dataset(input_base_path, output_base_path):
    classes = ['Gereja', 'Kutilang', 'Perkutut', 'Tekukur']

    for class_name in classes:
        class_path = os.path.join(input_base_path, class_name)

        if not os.path.exists(class_path):
```

```
        print(f"Folder {class_name} tidak ditemukan!")
        continue

    for audio_file in os.listdir(class_path):
        if audio_file.endswith(('.wav', '.mp3', '.flac')):
            input_path = os.path.join(class_path, audio_file)
            process_and_save_audio(input_path, output_base_path, class_name)


# Jalankan proses trimming
input_base_path = r'Audio Dataset'
output_base_path = r'Trimmed Audio Dataset'
process_dataset(input_base_path, output_base_path)

print("Proses trimming selesai. Data hasil trimming telah disimpan di folder baru.")
```

Lampiran 3. Syntax untuk Ekstraksi Fitur Mel-Spektogram dan MFCC

```
# Fungsi ekstraksi fitur mel-spektogram
def extract_melspectrogram(audio, sr, n_mels=128):
    return librosa.feature.melspectrogram(y=audio, sr=sr, n_mels=n_mels, fmax=8000)

# Fungsi ekstraksi fitur MFCC
def extract_mfcc(audio, sr, n_mfcc=40):
    return librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=n_mfcc)

# Fungsi untuk menyeragamkan panjang fitur
def pad_or_trim(array, target_length):
    if array.shape[1] > target_length:
        return array[:, :target_length]
    elif array.shape[1] < target_length:
        pad_width = ((0, 0), (0, target_length - array.shape[1]))
        return np.pad(array, pad_width, mode='constant')
    else:
        return array

def load_data(base_path, mel_length=128, mfcc_length=128):
    X_mel, X_mfcc, y = [], [], []
    classes = ['Gereja', 'Kutilang', 'Perkutut', 'Tekukur']

    for i, bird_class in enumerate(classes):
```

```
        class_path = os.path.join(base_path, bird_class)

        if not os.path.exists(class_path):
            print(f"Folder {bird_class} tidak ditemukan!")
            continue

        for audio_file in os.listdir(class_path):
            if audio_file.endswith(('.wav', '.mp3', '.flac')):
                audio_path = os.path.join(class_path, audio_file)

                try:
                    audio, sr = librosa.load(audio_path, sr=22050)
                    trimmed_audio = trim_audio(audio, sr)

                    mel_spec = extract_melspectrogram(trimmed_audio, sr)
                    mfcc = extract_mfcc(trimmed_audio, sr)

                    mel_spec_norm = librosa.util.normalize(mel_spec)
                    mfcc_norm = librosa.util.normalize(mfcc)

                    mel_spec_padded = pad_or_trim(mel_spec_norm, mel_length)
                    mfcc_padded = pad_or_trim(mfcc_norm, mfcc_length)

                    X_mel.append(mel_spec_padded)
                    X_mfcc.append(mfcc_padded)
                    y.append(i)

                except Exception as e:
                    print(f"Error processing {audio_path}: {e}")

    return np.array(X_mel), np.array(X_mfcc), np.array(y)
```

Lampiran 4. Syntax untuk pemodelan CNN 30 iterasi beserta hasilnya

```
# Import library yang dibutuhkan
import os
import numpy as np
import librosa
import tensorflow as tf
import tensorflow_hub as hub
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.models import Model, Sequential
```

```
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Input, GlobalAveragePooling2D
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
from PIL import Image
import pandas as pd
import matplotlib.pyplot as plt

# Atur seed untuk reproduksibilitas awal
np.random.seed(42)
tf.random.set_seed(42)

# Sampling rate standar
SR = 22050
NUM_RUNS = 30 # Training masing-masing model 30 kali

# Fungsi untuk padding audio agar memiliki panjang yang sama
def pad_audio(y, target_length):
    """
    Padding dengan nol agar audio memiliki panjang `target_length`.
    """
    if len(y) < target_length:
        y = np.pad(y, (0, target_length - len(y)), mode='constant')
    else:
        y = y[:target_length]
    return y

import numpy as np

# def pad_audio_center(y, target_length):
#     """
#     Memusatkan audio di tengah array target_length,
#     dengan padding nol di kiri dan kanan.
#     """
#     if len(y) >= target_length:
#         # Potong jika terlalu panjang
#         start = (len(y) - target_length) // 2
#         y = y[start : start + target_length]
#     else:
#         # Hitung padding kiri dan kanan
#         total_padding = target_length - len(y)
```

```
#        pad_left = total_padding // 2
#        pad_right = total_padding - pad_left
#        y = np.pad(y, (pad_left, pad_right), mode='constant')
#    return y


# 1. Load Dataset
def load_audio_files(dataset_path, sr=SR, duration=None):
    """
    Memuat file audio dan label dari folder dataset.
    Asumsi: folder `dataset_path` memiliki subfolder untuk setiap kelas,
    masing-masing berisi file .wav.
    """
    audio_data = []
    labels = []
    class_names = []
    # Ambil daftar kelas (nama folder)
    for class_idx, class_name in enumerate(sorted(os.listdir(dataset_path))):
        class_folder = os.path.join(dataset_path, class_name)
        if not os.path.isdir(class_folder):
            continue
        class_names.append(class_name)
        # Iterasi file .wav dalam folder kelas
        for file_name in os.listdir(class_folder):
            if file_name.endswith('.wav'):
                file_path = os.path.join(class_folder, file_name)
                # Muat audio menggunakan librosa
                y, _ = librosa.load(file_path, sr=sr, duration=duration)
                audio_data.append(y)
                labels.append(class_idx)
    return audio_data, np.array(labels), class_names

# 2. Ekstraksi Fitur
def extract_mel_spectrogram(audio, sr=SR, n_mels=128, fmax=8000):
    """
    Menghitung mel-spectrogram log dari sinyal audio.
    """
    S = librosa.feature.melspectrogram(y=audio, sr=sr, n_mels=n_mels, fmax=fmax)
    S_db = librosa.power_to_db(S, ref=np.max)
    return S_db

def extract_mfcc(audio, sr=SR, n_mfcc=40):
    """
```

```
    Menghitung koefisien MFCC dari sinyal audio.
    """
    mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=n_mfcc)
    return mfcc

# 3. Membangun Model CNN untuk mel-spectrogram
def create_cnn_model_mel(input_shape, num_classes):
    """
    Membangun model CNN sederhana untuk klasifikasi berdasar mel-spectrogram.
    """
    model = Sequential([
        Conv2D(32, (3,3), activation='relu', input_shape=input_shape),
        MaxPooling2D((2,2)),
        Conv2D(64, (3,3), activation='relu'),
        MaxPooling2D((2,2)),
        Conv2D(128, (3,3), activation='relu'),
        MaxPooling2D((2,2)),
        Flatten(),
        Dense(64, activation='relu'),
        Dense(num_classes, activation='softmax')
    ])
    return model

# 4. Membangun Model CNN untuk MFCC
def create_cnn_model_mfcc(input_shape, num_classes):
    """
    Membangun model CNN sederhana untuk klasifikasi berdasar MFCC.
    """
    model = Sequential([
        Conv2D(32, (3,3), activation='relu', input_shape=input_shape),
        MaxPooling2D((2,2)),
        Conv2D(64, (3,3), activation='relu'),
        MaxPooling2D((2,2)),
        Conv2D(128, (3,3), activation='relu'),
        MaxPooling2D((2,2)),
        Flatten(),
        Dense(64, activation='relu'),
        Dense(num_classes, activation='softmax')
    ])
    return model

# Fungsi plotting
def plot_training_history(history, title="Model", run=None):
```

```
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs_range = range(1, len(acc)+1)

    plt.figure(figsize=(12,5))
    run_title = f" (Run {run})" if run else ""

    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, acc, label='Training Accuracy')
    plt.plot(epochs_range, val_acc, label='Validation Accuracy')
    plt.legend(loc='lower right')
    plt.title(f'{title} - Accuracy{run_title}')

    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label='Training Loss')
    plt.plot(epochs_range, val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title(f'{title} - Loss{run_title}')

    plt.tight_layout()
    plt.show()

def resize_feature(feature, target_shape=(224, 224)):
    """
    Resize/padding fitur agar berukuran target_shape (224x224).
    """
    current_shape = feature.shape
    pad_height = max(0, target_shape[0] - current_shape[0])
    pad_width = max(0, target_shape[1] - current_shape[1])

    # Padding ke target shape
    padded = np.pad(feature,
                    pad_width=((0, pad_height), (0, pad_width)),
                    mode='constant', constant_values=0)

    # Jika lebih besar, crop
    resized = padded[:target_shape[0], :target_shape[1]]

    return resized

# ----------------------------------------------------------------------
```

```
# Main pipeline
# ----------------------------------------------------------------------
if __name__ == '__main__':
    # Path ke dataset
    dataset_path = 'Trimmed Audio Dataset'

    # 1) Muat data audio dan label
    print("Memuat data audio...")
    audio_data, labels, class_names = load_audio_files(dataset_path, sr=SR)
    num_classes = len(class_names)

    # 1.a) Padding setiap audio ke panjang maksimum
    max_len = max(len(y) for y in audio_data)
    audio_data = [pad_audio(y, max_len) for y in audio_data]
    audio_data = np.array(audio_data)

    # Pra-pemrosesan fitur
    print("Mengekstrak fitur Mel-Spectrogram dan MFCC...")
    X_mel_full = []
    for y in audio_data:
        S_db = extract_mel_spectrogram(y, sr=SR, n_mels=128)
        resized_feature = resize_feature(S_db, target_shape=(224, 224))
        X_mel_full.append(resized_feature[..., np.newaxis])
    X_mel_full = np.array(X_mel_full)

    X_mfcc_full = []
    for y in audio_data:
        mfcc = extract_mfcc(y, sr=SR, n_mfcc=40)
        resized_feature = resize_feature(mfcc, target_shape=(224, 224))
        X_mfcc_full.append(resized_feature[..., np.newaxis])
    X_mfcc_full = np.array(X_mfcc_full)

    # Simpan metrik untuk semua model dari semua run
    all_run_results = []
    all_run_cms = []

    # Definisikan callbacks
    early_stop = EarlyStopping(
        monitor='val_loss',
        patience=10, # sedikit lebih sabar
        restore_best_weights=True,
        verbose=1
    )
```

```
    reduce_lr = ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.2,
        patience=5,
        min_lr=1e-6,
        verbose=1
    )
    callbacks = [early_stop, reduce_lr]

    EPOCHS = 150
    BATCH_SIZE = 16 #Kombinasi menggunakan 16 dan kombinasi 2 menggunkan 32 bat

    # Loop untuk melatih dan mengevaluasi model sebanyak NUM_RUNS
    for run in range(1, NUM_RUNS + 1):
        print(f"\n{'='*30} RUN {run}/{NUM_RUNS} {'='*30}")

        # 1.b) Tentukan pembagian train-test dengan stratifikasi label untuk setiap run
        # Menggunakan random_state berbeda agar split bervariasi tapi tetap reproduktif
        indices = np.arange(len(labels))
        train_idx, test_idx = train_test_split(indices, test_size=0.2, stratify=labels,
random_state=42 + run)

        y_train, y_test = labels[train_idx], labels[test_idx]

        #
==================================================================
=======
        # Model 1: CNN dengan fitur mel-spectrogram
        #
==================================================================
=======
        print(f"\n[RUN {run} - Model 1] CNN dengan fitur Mel-Spectrogram")

        # Bagi train/test
        X_train_mel, X_test_mel = X_mel_full[train_idx], X_mel_full[test_idx]

        # Bangun, compile, dan latih model (harus di dalam loop untuk reset weights)
        input_shape_mel = X_train_mel.shape[1:]
        model_mel = create_cnn_model_mel(input_shape_mel, num_classes)
        model_mel.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

        print(f"Training Mel-CNN model for run {run}...")
        history_mel = model_mel.fit(
```

```
        X_train_mel, y_train,
        epochs=EPOCHS,
        batch_size=BATCH_SIZE,
        validation_data=(X_test_mel, y_test),
        callbacks=callbacks,
        verbose=0 # Set ke 0 agar tidak terlalu ramai, atau 2 untuk ringkasan per epoch
    )

    # Tampilkan grafik untuk run ini
    plot_training_history(history_mel, title="CNN Mel-Spectrogram", run=run)

    # Evaluasi
    print(f"Evaluating Mel-CNN model for run {run}...")
    y_pred_mel = np.argmax(model_mel.predict(X_test_mel), axis=1)

    # Kumpulkan metrik performa
    run_results_mel = {
        "Run": run,
        "Model": "CNN Mel",
        "Accuracy": accuracy_score(y_test, y_pred_mel),
        "Precision": precision_score(y_test, y_pred_mel, average='macro', zero_division=0),
        "Recall": recall_score(y_test, y_pred_mel, average='macro', zero_division=0),
        "F1": f1_score(y_test, y_pred_mel, average='macro', zero_division=0),
        "Train Accuracy": history_mel.history['accuracy'][-1],
        "Train Loss": history_mel.history['loss'][-1],
        "Val Accuracy": history_mel.history['val_accuracy'][-1],
        "Val Loss": history_mel.history['val_loss'][-1]
    }
    all_run_results.append(run_results_mel)

    # Kumpulkan confusion matrix
    cm_mel = confusion_matrix(y_test, y_pred_mel)
    for true_idx, row in enumerate(cm_mel):
        for pred_idx, count in enumerate(row):
            all_run_cms.append({
                'Run': run,
                'Model': 'CNN Mel',
                'True_Label': class_names[true_idx],
                'Predicted_Label': class_names[pred_idx],
                'Count': count
            })

    print(f"Classification Report (Mel-CNN, Run {run}):")
```

```
    print(classification_report(y_test, y_pred_mel, target_names=class_names,
zero_division=0))

    #
==================================================================
=======
    # Model 2: CNN dengan fitur MFCC
    #
==================================================================
=======
    print(f"\n[RUN {run} - Model 2] CNN dengan fitur MFCC")
    X_train_mfcc, X_test_mfcc = X_mfcc_full[train_idx], X_mfcc_full[test_idx]

    model_mfcc = create_cnn_model_mfcc(X_train_mfcc.shape[1:], num_classes)
    model_mfcc.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

    print(f"Training MFCC-CNN model for run {run}...")
    history_mfcc = model_mfcc.fit(
        X_train_mfcc, y_train,
        epochs=EPOCHS,
        batch_size=BATCH_SIZE,
        validation_data=(X_test_mfcc, y_test),
        callbacks=callbacks,
        verbose=0
    )

    # Tampilkan grafik untuk run ini
    plot_training_history(history_mfcc, title="CNN MFCC", run=run)

    # Evaluasi
    print(f"Evaluating MFCC-CNN model for run {run}...")
    y_pred_mfcc = np.argmax(model_mfcc.predict(X_test_mfcc), axis=1)

    # Kumpulkan metrik performa
    run_results_mfcc = {
        "Run": run,
        "Model": "CNN MFCC",
        "Accuracy": accuracy_score(y_test, y_pred_mfcc),
        "Precision": precision_score(y_test, y_pred_mfcc, average='macro',
zero_division=0),
        "Recall": recall_score(y_test, y_pred_mfcc, average='macro', zero_division=0),
        "F1": f1_score(y_test, y_pred_mfcc, average='macro', zero_division=0),
        "Train Accuracy": history_mfcc.history['accuracy'][-1],
```

```
        "Train Loss": history_mfcc.history['loss'][-1],
        "Val Accuracy": history_mfcc.history['val_accuracy'][-1],
        "Val Loss": history_mfcc.history['val_loss'][-1]
    }
    all_run_results.append(run_results_mfcc)

    # Kumpulkan confusion matrix
    cm_mfcc = confusion_matrix(y_test, y_pred_mfcc)
    for true_idx, row in enumerate(cm_mfcc):
        for pred_idx, count in enumerate(row):
            all_run_cms.append({
                'Run': run,
                'Model': 'CNN MFCC',
                'True_Label': class_names[true_idx],
                'Predicted_Label': class_names[pred_idx],
                'Count': count
            })

    print(f"Classification Report (MFCC-CNN, Run {run}):")
    print(classification_report(y_test,        y_pred_mfcc,        target_names=class_names,
zero_division=0))


# ==========================================================================
# Selesai semua run, simpan hasil kompilasi
# ==========================================================================
print("\n\n" + "="*25 + " SEMUA RUN SELESAI " + "="*25)

# Simpan performa ke CSV
performance_df = pd.DataFrame(all_run_results)
performance_csv_path = 'all_runs_performance.csv'
performance_df.to_csv(performance_csv_path, index=False)
print(f"\nPerforma dari {NUM_RUNS} run disimpan ke: {performance_csv_path}")
print("Ringkasan Performa (rata-rata dari semua run):")
print(performance_df.groupby('Model')[['Accuracy', 'Precision', 'Recall', 'F1']].mean())

# Simpan confusion matrix semua run ke file CSV terpisah
cm_df = pd.DataFrame(all_run_cms)
cm_csv_path = 'all_runs_confusion_matrices.csv'
```

```
    cm_df.to_csv(cm_csv_path, index=False)
    print(f"\nConfusion matrices dari semua run disimpan ke: {cm_csv_path}")

    #
==================================================================
=======
    # Simpan model dari RUN TERAKHIR
    #
==================================================================
=======

    model_save_path = 'fix_Model/CNN_Kombinasi1'
    os.makedirs(model_save_path, exist_ok=True)

    # Simpan dalam format native Keras (.keras)
    model_mel.save(os.path.join(model_save_path, 'Mel1_final_run.keras'))
    model_mfcc.save(os.path.join(model_save_path, 'MFCC1_final_run.keras'))
    print(f"\nModel dari run terakhir ({NUM_RUNS}) disimpan di folder:
{model_save_path}")
```

Lampiran 5. Syntax untuk Model MobileNet 30 iterasi besert hasilnya

```
# Import library yang dibutuhkan
import os
import numpy as np
import librosa
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, GlobalAveragePooling2D
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, classification_report
import pandas as pd
import matplotlib.pyplot as plt

# Atur seed untuk reproduksibilitas awal
np.random.seed(42)
tf.random.set_seed(42)

# Konfigurasi
SR = 22050
NUM_RUNS = 30 # Training masing-masing model 30 kali
```

```
# --- Fungsi Helper (Sama seperti sebelumnya) ---

def pad_audio(y, target_length):
    if len(y) < target_length:
        y = np.pad(y, (0, target_length - len(y)), mode='constant')
    else:
        y = y[:target_length]
    return y

def load_audio_files(dataset_path, sr=SR):
    audio_data = []
    labels = []
    class_names = []
    for class_idx, class_name in enumerate(sorted(os.listdir(dataset_path))):
        class_folder = os.path.join(dataset_path, class_name)
        if not os.path.isdir(class_folder):
            continue
        class_names.append(class_name)
        for file_name in os.listdir(class_folder):
            if file_name.endswith('.wav'):
                file_path = os.path.join(class_folder, file_name)
                y, _ = librosa.load(file_path, sr=sr)
                audio_data.append(y)
                labels.append(class_idx)
    return audio_data, np.array(labels), class_names

def extract_mel_spectrogram(audio, sr=SR, n_mels=128, fmax=8000):
    S = librosa.feature.melspectrogram(y=audio, sr=sr, n_mels=n_mels, fmax=fmax)
    S_db = librosa.power_to_db(S, ref=np.max)
    return S_db

def extract_mfcc(audio, sr=SR, n_mfcc=40):
    mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=n_mfcc)
    return mfcc

def resize_feature(feature, target_shape=(224, 224)):
    current_shape = feature.shape
    # Untuk resize, kita bisa menggunakan tf.image.resize yang lebih efisien
    # Tambahkan dimensi channel sementara
    feature_with_channel = feature[..., np.newaxis]
    resized = tf.image.resize(feature_with_channel, target_shape).numpy()
    return np.squeeze(resized, axis=-1)
```

```
def plot_training_history(history, title="Model", run=None):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs_range = range(1, len(acc)+1)

    plt.figure(figsize=(12, 5))
    run_title = f" (Run {run})" if run else ""

    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, acc, label='Training Accuracy')
    plt.plot(epochs_range, val_acc, label='Validation Accuracy')
    plt.legend(loc='lower right')
    plt.title(f'{title} - Accuracy{run_title}')

    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label='Training Loss')
    plt.plot(epochs_range, val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title(f'{title} - Loss{run_title}')

    plt.tight_layout()
    plt.show()

# --- Fungsi Pembangunan Model ---

def create_transfer_mobilenet(input_shape, num_classes):
    """
    Membangun model klasifikasi dengan MobileNet (pra-terlatih) dari Keras Applications.
    """
    # Menggunakan `weights='imagenet'` secara otomatis akan mengunduh bobot yang sesuai.
    base_model = tf.keras.applications.MobileNet(input_shape=input_shape,
include_top=False, weights='imagenet')
    base_model.trainable = False

    # Membangun model baru di atas base_model
    inputs = Input(shape=input_shape)
    # MobileNet mengharapkan input ternormalisasi. Kita bisa menggunakan lapisan
normalisasi
    # atau membiarkannya belajar dari data float yang ada. Untuk konsistensi, kita biarkan
saja.
```

```
    x = base_model(inputs, training=False) #base layer bisa di freeze (training=False ) dan
tidak (training=False)
    x = GlobalAveragePooling2D()(x)
    outputs = Dense(num_classes, activation='softmax')(x)
    model = Model(inputs, outputs)
    return model

# ----------------------------------------------------------------------
# Main pipeline
# ----------------------------------------------------------------------
if __name__ == '__main__':
    dataset_path = 'Trimmed Audio Dataset'

    # 1) Muat dan pra-pemrosesan data (hanya sekali)
    print("Memuat data audio...")
    audio_data, labels, class_names = load_audio_files(dataset_path, sr=SR)
    num_classes = len(class_names)

    max_len = max(len(y) for y in audio_data)
    audio_data = [pad_audio(y, max_len) for y in audio_data]

    # Ekstraksi fitur dan persiapan untuk model (hanya sekali)
    print("Mengekstrak fitur dan mempersiapkannya untuk MobileNet...")
    X_mel_full = []
    for y in audio_data:
        S_db = extract_mel_spectrogram(y, sr=SR)
        resized_feature = resize_feature(S_db, target_shape=(224, 224))
        # Ulangi channel untuk membuat input 3-channel
        rgb_feature = np.repeat(resized_feature[..., np.newaxis], 3, axis=-1)
        X_mel_full.append(rgb_feature)
    X_mel_full = np.array(X_mel_full)

    X_mfcc_full = []
    for y in audio_data:
        mfcc = extract_mfcc(y, sr=SR)
        resized_feature = resize_feature(mfcc, target_shape=(224, 224))
        rgb_feature = np.repeat(resized_feature[..., np.newaxis], 3, axis=-1)
        X_mfcc_full.append(rgb_feature)
    X_mfcc_full = np.array(X_mfcc_full)

    print(f"Bentuk data Mel-Spectrogram: {X_mel_full.shape}")
    print(f"Bentuk data MFCC: {X_mfcc_full.shape}")
```

```
    # Inisialisasi list untuk menyimpan hasil dari semua run
    all_run_results = []
    all_run_cms = []

    # Definisikan callbacks
    early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True,
verbose=1)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=1e-
6, verbose=1)
    callbacks = [early_stop, reduce_lr]

    EPOCHS = 120
    BATCH_SIZE = 16 #bisa diganti 16 dan 32

    # Loop untuk melatih dan mengevaluasi model sebanyak NUM_RUNS
    for run in range(1, NUM_RUNS + 1):
        print(f"\n{'='*30} RUN {run}/{NUM_RUNS} {'='*30}")

        indices = np.arange(len(labels))
        train_idx, test_idx = train_test_split(indices, test_size=0.2, stratify=labels,
random_state=42 + run)
        y_train, y_test = labels[train_idx], labels[test_idx]

        # --- Model 1: MobileNet dengan Mel-Spectrogram ---
        print(f"\n[RUN {run} - Model 1] Transfer Learning MobileNet (Mel-spectrogram)")
        X_train_mel, X_test_mel = X_mel_full[train_idx], X_mel_full[test_idx]

        model_melnet = create_transfer_mobilenet(X_train_mel.shape[1:], num_classes)
        model_melnet.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

        history_melnet = model_melnet.fit(
            X_train_mel, y_train,
            epochs=EPOCHS, batch_size=BATCH_SIZE, validation_data=(X_test_mel, y_test),
            callbacks=callbacks, verbose=0
        )

        plot_training_history(history_melnet, title="MobileNet Mel-Spectrogram", run=run)

        y_pred_melnet = np.argmax(model_melnet.predict(X_test_mel), axis=1)

        # Kumpulkan Metrik Performa
        all_run_results.append({
```

```
        "Run": run, "Model": "MobileNet (Mel)",
        "Accuracy": accuracy_score(y_test, y_pred_melnet),
        "Precision": precision_score(y_test, y_pred_melnet, average='macro',
zero_division=0),
        "Recall": recall_score(y_test, y_pred_melnet, average='macro', zero_division=0),
        "F1": f1_score(y_test, y_pred_melnet, average='macro', zero_division=0),
        "Train Accuracy": history_melnet.history['accuracy'][-1], "Train Loss":
history_melnet.history['loss'][-1],
        "Val Accuracy": history_melnet.history['val_accuracy'][-1], "Val Loss":
history_melnet.history['val_loss'][-1]
    })

    # Kumpulkan Confusion Matrix
    cm_melnet = confusion_matrix(y_test, y_pred_melnet)
    for true_idx, row in enumerate(cm_melnet):
        for pred_idx, count in enumerate(row):
            all_run_cms.append({'Run': run, 'Model': 'MobileNet (Mel)', 'True_Label':
class_names[true_idx], 'Predicted_Label': class_names[pred_idx], 'Count': count})

    print(f"Classification Report (MobileNet-Mel, Run {run}):\n",
classification_report(y_test, y_pred_melnet, target_names=class_names, zero_division=0))

    # --- Model 2: MobileNet dengan MFCC ---
    print(f"\n[RUN {run} - Model 2] Transfer Learning MobileNet (MFCC)")
    X_train_mfcc, X_test_mfcc = X_mfcc_full[train_idx], X_mfcc_full[test_idx]

    model_mfccnet = create_transfer_mobilenet(X_train_mfcc.shape[1:], num_classes)
    model_mfccnet.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

    history_mfccnet = model_mfccnet.fit(
        X_train_mfcc, y_train,
        epochs=EPOCHS, batch_size=BATCH_SIZE, validation_data=(X_test_mfcc,
y_test),
        callbacks=callbacks, verbose=0
    )

    plot_training_history(history_mfccnet, title="MobileNet MFCC", run=run)

    y_pred_mfccnet = np.argmax(model_mfccnet.predict(X_test_mfcc), axis=1)

    # Kumpulkan Metrik Performa
    all_run_results.append({
        "Run": run, "Model": "MobileNet (MFCC)",
```

```
        "Accuracy": accuracy_score(y_test, y_pred_mfccnet),
        "Precision": precision_score(y_test, y_pred_mfccnet, average='macro',
zero_division=0),
        "Recall": recall_score(y_test, y_pred_mfccnet, average='macro', zero_division=0),
        "F1": f1_score(y_test, y_pred_mfccnet, average='macro', zero_division=0),
        "Train Accuracy": history_mfccnet.history['accuracy'][-1], "Train Loss":
history_mfccnet.history['loss'][-1],
        "Val Accuracy": history_mfccnet.history['val_accuracy'][-1], "Val Loss":
history_mfccnet.history['val_loss'][-1]
    })

    # Kumpulkan Confusion Matrix
    cm_mfccnet = confusion_matrix(y_test, y_pred_mfccnet)
    for true_idx, row in enumerate(cm_mfccnet):
        for pred_idx, count in enumerate(row):
            all_run_cms.append({'Run': run, 'Model': 'MobileNet (MFCC)', 'True_Label':
class_names[true_idx], 'Predicted_Label': class_names[pred_idx], 'Count': count})

    print(f"Classification Report (MobileNet-MFCC, Run {run}):\n",
classification_report(y_test, y_pred_mfccnet, target_names=class_names, zero_division=0))

  #
==================================================================
=======
  # Selesai semua run, simpan hasil kompilasi
  #
==================================================================
=======
  print("\n\n" + "="*25 + " SEMUA RUN SELESAI " + "="*25)

  # Simpan performa ke CSV
  performance_df = pd.DataFrame(all_run_results)
  performance_csv_path = 'mobilenet_kombinasi1_all_performance.csv'
  performance_df.to_csv(performance_csv_path, index=False)
  print(f"\nPerforma dari {NUM_RUNS} run disimpan ke: {performance_csv_path}")
  print("Ringkasan Performa (rata-rata dari semua run):")
  print(performance_df.groupby('Model')[['Accuracy', 'Precision', 'Recall', 'F1']].mean())

  # Simpan confusion matrix semua run ke file CSV terpisah
  cm_df = pd.DataFrame(all_run_cms)
  cm_csv_path = 'mobilenet_kombinasi1_all_confusion_matrices.csv'
  cm_df.to_csv(cm_csv_path, index=False)
  print(f"\nConfusion matrices dari semua run disimpan ke: {cm_csv_path}")
```

```
    #
==========================================================================
=======
    # Simpan model dari RUN TERAKHIR
    #
==========================================================================
=======
    model_save_path = 'fix_Model/MobileNet_Kombinasi1'
    os.makedirs(model_save_path, exist_ok=True)

    model_melnet.save(os.path.join(model_save_path, 'MobileNet_Mel_final_run.keras'))
    model_mfccnet.save(os.path.join(model_save_path,
'MobileNet_MFCC_final_run.keras'))
    print(f"\nModel dari run terakhir ({NUM_RUNS}) disimpan di folder:
{model_save_path}")
```

Lampiran 6. Syntax untuk Model VGGish 30 iterasi beserta hasilnya

```
# Import library yang dibutuhkan
import os
import numpy as np
import librosa
import tensorflow as tf
import tensorflow_hub as hub
from tensorflow.keras import layers, models
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, classification_report
import pandas as pd
import matplotlib.pyplot as plt

# Atur seed untuk reproduksibilitas awal
np.random.seed(42)
tf.random.set_seed(42)

# Konfigurasi
SR = 22050
NUM_RUNS = 30 #

# --- Fungsi Helper ---

def pad_audio(y, target_length):
```

```
    """Padding dengan nol agar audio memiliki panjang `target_length`."""
    if len(y) < target_length:
        y = np.pad(y, (0, target_length - len(y)), mode='constant')
    else:
        y = y[:target_length]
    return y

def load_audio_files(dataset_path, sr=SR):
    """Memuat file audio dan label dari folder dataset."""
    audio_data = []
    labels = []
    class_names = []
    for class_idx, class_name in enumerate(sorted(os.listdir(dataset_path))):
        class_folder = os.path.join(dataset_path, class_name)
        if not os.path.isdir(class_folder):
            continue
        class_names.append(class_name)
        for file_name in os.listdir(class_folder):
            if file_name.endswith('.wav'):
                file_path = os.path.join(class_folder, file_name)
                y, _ = librosa.load(file_path, sr=sr)
                audio_data.append(y)
                labels.append(class_idx)
    return audio_data, np.array(labels), class_names

def plot_training_history(history, title="Model", run=None):
    """Membuat plot akurasi dan loss dari riwayat training."""
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs_range = range(1, len(acc)+1)

    plt.figure(figsize=(12, 5))
    run_title = f" (Run {run})" if run else ""

    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, acc, label='Training Accuracy')
    plt.plot(epochs_range, val_acc, label='Validation Accuracy')
    plt.legend(loc='lower right')
    plt.title(f'{title} - Accuracy{run_title}')

    plt.subplot(1, 2, 2)
```

```
    plt.plot(epochs_range, loss, label='Training Loss')
    plt.plot(epochs_range, val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title(f'{title} - Loss{run_title}')

    plt.tight_layout()
    plt.show()

# --- Fungsi Spesifik untuk VGGish ---

def extract_vggish_embeddings(audio_list, sr=SR, target_sr=16000):
    """
    Menggunakan VGGish dari TensorFlow Hub untuk ekstraksi embedding.
    Proses ini memakan waktu dan hanya perlu dijalankan sekali.
    """
    print("Memuat model VGGish dari TensorFlow Hub...")
    vggish = hub.load('https://tfhub.dev/google/vggish/1')
    print("Model VGGish berhasil dimuat.")

    embeddings = []
    print(f"Mengekstrak VGGish embeddings dari {len(audio_list)} file audio...")
    for i, y in enumerate(audio_list):
        # Resample ke 16 kHz yang dibutuhkan oleh VGGish
        if sr != target_sr:
            y_res = librosa.resample(y, orig_sr=sr, target_sr=target_sr)
        else:
            y_res = y.astype(np.float32)

        # Dapatkan embeddings dari VGGish
        # VGGish menghasilkan embedding per 0.96 detik, jadi kita ambil rata-ratanya
        emb_frames = vggish(y_res)
        mean_emb = tf.reduce_mean(emb_frames, axis=0).numpy()
        embeddings.append(mean_emb)
        if (i + 1) % 10 == 0:
            print(f"  ... Selesai {i + 1}/{len(audio_list)}")

    print("Ekstraksi embedding selesai.")
    return np.stack(embeddings, axis=0)

def create_classifier_from_embeddings(input_dim, num_classes):
    """
    Membangun MLP sederhana untuk klasifikasi berdasarkan vektor embedding.
    """
```

```
    model = Sequential([
        Input(shape=(input_dim,)),
        Dense(128, activation='relu'),
        Dense(64, activation='relu'),
        Dense(num_classes, activation='softmax')
    ])
    return model

# ----------------------------------------------------------------------
# Main pipeline
# ----------------------------------------------------------------------
if __name__ == '__main__':
    dataset_path = 'Trimmed Audio Dataset'

    # 1) Muat data audio dan label (hanya sekali)
    print("Memuat data audio...")
    audio_data, labels, class_names = load_audio_files(dataset_path, sr=SR)
    num_classes = len(class_names)

    max_len = max(len(y) for y in audio_data)
    audio_data = [pad_audio(y, max_len) for y in audio_data]

    # 2) Ekstraksi fitur embedding VGGish (hanya sekali, karena prosesnya berat)
    X_vggish_full = extract_vggish_embeddings(audio_data, sr=SR)
    print(f"Bentuk data embedding VGGish setelah ekstraksi: {X_vggish_full.shape}")

    #
    all_run_results = []
    all_run_cms = []

    # Definisikan callbacks
    early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True,
verbose=1)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=1e-
6, verbose=1)
    callbacks = [early_stop, reduce_lr]

    EPOCHS = 200
    BATCH_SIZE = 16

    # Loop untuk melatih dan mengevaluasi model klasifikasi sebanyak NUM_RUNS
    for run in range(1, NUM_RUNS + 1):
        print(f"\n{'='*30} RUN {run}/{NUM_RUNS} {'='*30}")
```

```
    # Bagi data embedding yang sudah diekstrak
    X_train_vgg, X_test_vgg, y_train, y_test = train_test_split(
        X_vggish_full, labels, test_size=0.2, stratify=labels, random_state=42 + run
    )

    # Bangun, compile, dan latih model klasifikasi (di-reset setiap run)
      model_vgg = create_classifier_from_embeddings(input_dim=X_train_vgg.shape[1],
num_classes=num_classes)
            model_vgg.compile(optimizer='adam',   loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

    print(f"Training VGGish classifier for run {run}...")
    history_vgg = model_vgg.fit(
        X_train_vgg, y_train,
        epochs=EPOCHS, batch_size=BATCH_SIZE, validation_data=(X_test_vgg, y_test),
        callbacks=callbacks, verbose=0
    )

    # Tampilkan grafik untuk run ini
    plot_training_history(history_vgg, title="VGGish Classifier", run=run)

    # Evaluasi model
    print(f"Evaluating VGGish classifier for run {run}...")
    y_pred_vgg = np.argmax(model_vgg.predict(X_test_vgg), axis=1)

    # Kumpulkan metrik performa
    all_run_results.append({
        "Run": run, "Model": "VGGish",
        "Accuracy": accuracy_score(y_test, y_pred_vgg),
        "Precision": precision_score(y_test, y_pred_vgg, average='macro', zero_division=0),
        "Recall": recall_score(y_test, y_pred_vgg, average='macro', zero_division=0),
        "F1": f1_score(y_test, y_pred_vgg, average='macro', zero_division=0),
                "Train Accuracy": history_vgg.history['accuracy'][-1], "Train Loss":
history_vgg.history['loss'][-1],
                "Val Accuracy": history_vgg.history['val_accuracy'][-1], "Val Loss":
history_vgg.history['val_loss'][-1]
    })

    # Kumpulkan confusion matrix
    cm_vgg = confusion_matrix(y_test, y_pred_vgg)
    for true_idx, row in enumerate(cm_vgg):
        for pred_idx, count in enumerate(row):
```

```
                    all_run_cms.append({'Run': run, 'Model': 'VGGish', 'True_Label': class_names[true_idx], 'Predicted_Label': class_names[pred_idx], 'Count': count})

        print(f"Classification Report (VGGish, Run {run}):\n", classification_report(y_test, y_pred_vgg, target_names=class_names, zero_division=0))

                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
   # Selesai semua run, simpan hasil kompilasi
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
   print("\n\n" + "="*25 + " SEMUA RUN SELESAI " + "="*25)

   # Simpan performa ke CSV
   performance_df = pd.DataFrame(all_run_results)
   performance_csv_path = 'vggish1_all_run_performance.csv'
   performance_df.to_csv(performance_csv_path, index=False)
        print(f"\nPerforma VGGish dari {NUM_RUNS} run disimpan ke: {performance_csv_path}")
   print("Ringkasan Performa VGGish (rata-rata dari semua run):")
   print(performance_df[['Accuracy', 'Precision', 'Recall', 'F1']].mean())

   # Simpan confusion matrix semua run ke file CSV terpisah
   cm_df = pd.DataFrame(all_run_cms)
   cm_csv_path = 'vggish1_all_run_confusion_matrices.csv'
   cm_df.to_csv(cm_csv_path, index=False)
   print(f"\nConfusion matrices VGGish dari semua run disimpan ke: {cm_csv_path}")

                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
   # Simpan model klasifikasi dari RUN TERAKHIR
                                                                                                                                                                                                                                                                                                                                                                                                                                                
   model_save_path = 'fix_Model/VGGish'
   os.makedirs(model_save_path, exist_ok=True)

   # Simpan model klasifikasi (bukan model VGGish-nya)
   model_vgg.save(os.path.join(model_save_path, 'VGGish1_final_run.keras'))
   print(f"\nModel klasifikasi VGGish dari run terakhir ({NUM_RUNS}) disimpan di folder: {model_save_path}")
```

Lampiran 7. Surat Pernyataan Data

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini, mahasiswa Departemen Statistika FSAD ITS:

Nama : Daniswara Aditya Putra

NRP : 5003211132

menyatakan bahwa data yang digunakan dalam **Tugas Akhir** ~~/Thesis*~~ ini merupakan data sekunder yang diambil dari ~~**penelitian / buku / Tugas Akhir / Thesis /**~~ **publikasi lainnya*** yaitu :

Sumber : Data *website*

1. https://xeno-canto.org

Keterangan :

1. Data Suara Burung Kutilang (*Pycnonotus aurigaster*), Burung Gereja (*Passer domesticus*), Burung Perkutut (*Geopelia striata*), Burung Tekukur (*Streptopelia chinensis*), Burung Trucukan (*Pycnonotus goiavier*), dan Burung Cendet (*Lanius cristatus*)

Surat Pernyataan ini dibuat dengan sebenarnya. Apabila terdapat pemalsuan data maka saya siap menerima sanksi sesuai aturan yang berlaku.

Surabaya, 30 Juni 2025

Mengetahui, Dosen Pembimbing Tugas Akhir	Mahasiswa
Dr. Dra. Kartika Fithriasari, M.Si NIP. 19691212 199303 2 002	Daniswara Aditya Putra NRP. 5003211132

*(coret yang tidak perlu)

BIODATA PENULIS

Penulis dilahirkan di Jakarta, pada 14 September 2003, merupakan anak pertama dari tiga bersaudara. Penulis berasal dari Desa Talun, Kecamatan Talun, Kabupaten Blitar. Penulis mengawali pendidikan formal di SDN Talun 02, kemudian dilanjutkan ke jenjang menengah pertama di SMPN 1 Wlingi, lalu penulis melanjutkan ke jenjang menengah atas di SMAN 1 Talun. Setelah itu penulis melanjutkan pendidikan ke jenjang S1 di Departemen Statistika Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2021 melalui jalur Mandiri.

Selama berkuliah di Departemen Statistika, ITS, penulis aktif dalam berbagai kegiatan baik akademik maupun non-akademik. Dalam bidang akademik, penulis pernah mengikuti beberapa program, di antaranya menjadi peserta program Kampus Merdeka *Bangkit Academy* tahun 2024 dengan *learning path Machine Learning* dan mengerjakan *capstone project* berjudul *FreshGrade:* Identifikasi Kesegaran dan Penyakit Buah Melalui Citra Digital. Selain itu, pada tahun 2024 penulis juga mengikuti program magang di Dinas Pekerjaan Umum Sumber Daya Air Provinsi Jawa Timur, dengan fokus pada pengolahan dan visualisasi data pengairan serta produksi konten data videografis. Pada tahun 2025, penulis juga melaksanakan program magang di Tim Analytics Decision Science BTPN Syariah di Jakarta, di mana penulis terlibat langsung dalam pengembangan sistem deteksi fraud pada transaksi pencairan nasabah. Dalam bidang non-akademik, penulis pernah tergabung dalam beberapa kegiatan organisasi dan kepanitiaan di lingkungan kampus, seperti menjadi Staff Operational and Equipment serta Koordinator Roadshow pada kegiatan Pekan Raya Statistika tahun 2023, yang merupakan kompetisi tingkat Asia Tenggara dalam bidang statistika dan data science. Bagi pembaca yang ingin berdiskusi, memberikan kritik, serta masukan terkait Tugas Akhir ini dapat menghubungi penulis melalui email: daniswara.dap@gmail.com