PENGEMBANGAN APLIKASI *MOBILE* BERBASIS *DEEP LEARNING* UNTUK KLASIFIKASI VOKALISASI AYAM

TUGAS AKHIR

STELLA MAUREEN IGNACIA SANTOSO
NIM: 312110014

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI DAN DESAIN
UNIVERSITAS MA CHUNG
MALANG
2025

LEMBAR PENGESAHAN

TUGAS AKHIR

PENGEMBANGAN APLIKASI *MOBILE* BERBASIS *DEEP LEARNING* UNTUK KLASIFIKASI VOKALISASI AYAM

Oleh:

STELLA MAUREEN IGNACIA SANTOSO

NIM. 312110014

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI dan DESAIN

UNIVERSITAS MA CHUNG

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan dan berhak mendapatkan gelar Sarjana Teknik «GELAR»

Dosen Pembimbing I,

Hendry Setiawan, ST., M.Kom.
NIP. 20100006

Dosen Pembimbing II,

Dr. Kestrilia Rega Prilianti, M.Si.
NIP. 20120035

Dekan Fakultas Teknologi dan Desain,

Prof. Dr. Eng. Romy Budhi Widodo
NIP. 2007003

PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Skripsi saya dengan "Pengembangan Aplikasi *Mobile* Berbasis *Deep Learning* untuk Klasifikasi Vokalisasi Ayam" adalah benar benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Malang, 23 Juli 2025

Stella Maureen Ignacia Santoso

312110014

PENGEMBANGAN APLIKASI *MOBILE* BERBASIS *DEEP LEARNING* UNTUK KLASIFIKASI VOKALISASI AYAM

Stella Maureen Ignacia Santoso, Hendry Setiawan, Kestrilia Rega Prilianti

Universitas Ma Chung

Abstrak

Vokalisasi ayam memiliki peran penting dalam memahami perilaku dan kondisi biologis unggas, terutama dalam konteks peternakan modern yang menuntut efisiensi dan otomatisasi. Sistem klasifikasi suara ayam secara otomatis dapat menjadi solusi untuk mendeteksi situasi tertentu, seperti ancaman atau kondisi pasca bertelur, guna mendukung pengambilan keputusan yang lebih cepat dan akurat dalam manajemen ternak.

Penelitian ini mengembangkan sistem klasifikasi vokalisasi ayam menggunakan pendekatan *deep learning* serta mengimplementasikannya ke dalam aplikasi *mobile* berbasis *Android*. Vokalisasi ayam yang diklasifikasikan terdiri dari empat kelas, yaitu: suara ayam betina marah, suara ayam betina memanggil jantan, suara ayam ketika ada ancaman, dan suara ayam betina setelah bertelur. Terdapat 3 metode *deep learning* yang dibandingkan, yaitu *Convolutional Neural Network* (CNN), *Transformer*, dan *DeepSpeech*. *Dataset* terdiri dari 1.403 *file* audio 4 detik, dibagi menjadi data pelatihan dan validasi. Data pengujian menggunakan data tersendiri. Evaluasi dilakukan berdasarkan akurasi pelatihan, validasi, pengujian, serta ukuran *file* model untuk mempertimbangkan efisiensi *deployment*.

Hasil menunjukkan bahwa *Transformer* memberikan akurasi pengujian 100% pada semua kelas dengan ukuran file 3.35 MB, dan dipilih untuk implementasi aplikasi, yang menghasilkan akurasi per kelas: ayam betina marah 6.67%, ayam betina memanggil jantan 100%, ayam ketika ada ancaman 93.33%, dan ayam betina setelah bertelur 100%.

Kata kunci: *Android,* CNN, *Deep learning, DeepSpeech,* Klasifikasi vokaliasi ayam, *Transformer*

DEVELOPMENT OF A DEEP LEARNING-BASED MOBILE APPLICATION FOR CHICKEN VOCALIZATION CLASSIFICATION

Stella Maureen Ignacia Santoso, Hendry Setiawan, Kestrilia Rega Prilianti

Universitas Ma Chung

Abstract

Chicken vocalization plays an important role in understanding the behavior and biological condition of poultry, especially in the context of modern farming that demands efficiency and automation. An automatic chicken sound classification system can be a solution for detecting certain situations, such as threats or post-laying conditions, to support faster and more accurate decision-making in livestock management.

This study developed a chicken vocalization classification system using a deep learning approach and implemented it into an Android-based mobile application. The classified chicken vocalizations consist of four classes: angry female chicken sounds, female chicken calling male sounds, chicken sounds when there is a threat, and female chicken sounds after laying eggs. Three deep learning methods were compared: Convolutional Neural Network (CNN), Transformer, and DeepSpeech. The dataset consists of 1.403 4-second audio files, divided into training and validation data. The testing data uses separate data. Evaluation was based on training accuracy, validation accuracy, testing accuracy, and model file size to consider deployment efficiency.

The results show that the Transformer achieves 100% test accuracy across all classes with a file size of 3.35 MB, and was selected for application implementation, yielding class-specific accuracies of: angry hen 6.67%, hen calling a rooster 100%, chicken under threat 93.33%, and hen after laying an egg.

Keywords: Android, Chicken sound classification, CNN, Deep learning, DeepSpeech, Transformer

KATA PENGANTAR

Puji dan syukur dipanjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga penelitian Tugas Akhir berjudul "Pengembangan Aplikasi *Mobile* Berbasis *Deep Learning* untuk Klasifikasi Vokalisasi Ayam" ini bisa terselesaikan dengan baik. Penyusunan Tugas Akhir ini merupakan hasil dari serangkaian proses pembelajaran dan penelitian yang dilakukan secara mandiri maupun melalui bimbingan. Penulis berharap karya ini dapat memberikan kontribusi nyata dalam bidang pemrosesan suara dan pengembangan teknologi kecerdasan buatan, khususnya dalam penerapan sistem *monitoring* kondisi hewan berbasis suara. Dalam kesempatan ini, penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Bapak Hendry Setiawan, ST., M.Kom. selaku dosen pembimbing pertama pelaksanaan Tugas Akhir
2. Ibu Dr. Kestrilia Rega Prilianti, M.Si. selaku dosen pembimbing kedua pelaksanaan Tugas Akhir
3. Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd. selaku dosen penguji pelaksanaan Tugas Akhir
4. Teman-teman dan pihak lain yang telah memberikan bantuan baik secara langsung maupun tidak langsung selama proses penelitian

Penulis menyadari bahwa skripsi ini masih memiliki kekurangan. Oleh karena itu, penulis sangat terbuka terhadap saran dan kritik yang membangun guna menyempurnakan karya ini di masa mendatang.

Malang, 23 Juli 2025

Stella Maureen Ignacia Santoso

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Bab I
Pendahuluan

1.1. Latar Belakang

Ayam (*Gallus gallus domesticus*) merupakan salah satu unggas yang paling banyak dibudidayakan di Indonesia, baik secara tradisional maupun dalam skala industri untuk menghasilkan daging dan telur (Zaharo dkk., 2024). Sebagai hewan domestik yang telah dipelihara selama ribuan tahun, ayam memiliki sistem komunikasi yang kompleks, salah satunya melalui vokalisasi atau suara yang mereka hasilkan. Vokalisasi ini memiliki berbagai fungsi, mulai dari interaksi sosial, mengekspresikan kondisi emosional dan fisiologis, hingga merespons perubahan lingkungan (Safitra dkk., 2022). Berbagai jenis suara yang dihasilkan oleh ayam dapat mengindikasikan kondisi emosional mereka, seperti rasa lapar, stres, ketakutan, dominasi, atau bahkan kesakitan (Manikandan dan Neethirajan, 2024). Sebagai contoh, ayam jantan berkokok sebagai tanda dominasi dan batas wilayah, sementara induk ayam mengeluarkan suara khas untuk memanggil anak-anaknya.

Dalam industri peternakan modern, pemantauan kesehatan dan kesejahteraan ayam menjadi tantangan tersendiri. Peternakan dalam skala besar sering kali menghadapi kesulitan dalam mendeteksi kondisi individu setiap ayam karena jumlahnya yang sangat banyak (Mayasari dkk., 2023). Metode pemantauan tradisional yang mengandalkan pengamatan langsung sering kali memerlukan banyak tenaga kerja dan waktu, serta berisiko terhadap kesalahan manusia. Oleh karena itu, penggunaan teknologi berbasis kecerdasan buatan (*Artificial Intelligence* / AI) untuk mengenali dan mengklasifikasikan suara ayam menjadi solusi inovatif yang dapat meningkatkan efisiensi dan akurasi dalam pemantauan kondisi ayam secara otomatis. Penelitian terbaru menunjukkan bahwa kecerdasan buatan (AI) dapat digunakan untuk mengenali dan mengklasifikasikan suara ayam guna memahami kondisi mereka secara lebih akurat. Misalnya, peneliti di Jepang berhasil mengembangkan sistem AI yang mampu menerjemahkan suara ayam dan mengidentifikasi enam keadaan emosi yang berbeda (Liputan6, 2023).

Dalam beberapa tahun terakhir, kemajuan dalam bidang *deep learning* telah memungkinkan pengembangan sistem yang mampu mengenali pola suara dengan akurasi tinggi. Model *deep learning* dapat menganalisis vokalisasi ayam untuk mendeteksi perubahan perilaku atau kondisi kesehatan ayam secara *real-time* (Latif *et al.,* 2021). Beberapa metode *deep learning* yang umum digunakan dalam klasifikasi suara meliputi *Convolutional Neural Network* (CNN), *Transformer*, dan *Deepspeech*. Masing-masing metode ini memiliki keunggulan dan kelemahan dalam menangani data suara, sehingga diperlukan perbandingan performa untuk menentukan metode terbaik dalam mengklasifikasikan vokalisasi ayam.

Convolutional Neural Network (CNN) adalah salah satu model *deep learning* yang banyak digunakan dalam pemrosesan gambar dan suara. Dalam analisis suara, CNN bekerja dengan mengubah sinyal suara menjadi representasi spektrogram, yang kemudian diproses sebagai data citra untuk mengekstraksi fitur-fitur penting (Alberto dan Hermanto, 2023). CNN telah terbukti efektif dalam berbagai aplikasi pengenalan suara karena kemampuannya dalam menangkap pola spasial dari data suara. Model ini dapat mengenali perbedaan frekuensi dan durasi suara, sehingga memungkinkan klasifikasi vokalisasi dengan tingkat akurasi yang tinggi (Ferdiawan dan Hartono, 2022).

Transformer adalah model yang awalnya dikembangkan untuk pemrosesan bahasa alami (*Natural Language Processing* / NLP), tetapi belakangan ini mulai digunakan dalam analisis suara. Keunggulan utama *Transformer* terletak pada mekanisme *self-attention* yang memungkinkan model memahami hubungan jangka panjang dalam data sekuensial (Vaswani *et al.,* 2017). Dalam analisis vokalisasi ayam, *Transformer* dapat mengenali pola suara yang lebih kompleks dibandingkan dengan CNN, terutama dalam kasus di mana suara memiliki variasi yang lebih luas dalam hal intensitas dan durasi (Dosovitskiy *et al.,* 2021).

Deepspeech merupakan perangkat lunak sumber terbuka untuk konversi suara ke teks (*Speech-To-Text* / STT) yang memanfaatkan model yang dilatih menggunakan metode pembelajaran mesin. Teknologi ini awalnya dikembangkan berdasarkan penelitian *Deep Speech* oleh Baidu dan saat ini dikelola oleh Mozilla. *Deepspeech* dibangun menggunakan arsitektur *Recurrent Neural Network* (RNN)

yang terdiri dari beberapa lapisan *Long Short-Term Memory* (LSTM) sebagai komponen utamanya (GeeksforGeeks, 2024). Dengan pendekatan ini, *Deepspeech* memiliki keunggulan dalam memahami konteks suara secara berurutan, sehingga cocok untuk menganalisis vokalisasi ayam yang bersifat dinamis. Karena dibangun menggunakan arsitektur *Recurrent Neural Network* (RNN) dengan *Long Short-Term Memory* (LSTM), *Deepspeech* cenderung lebih membutuhkan daya komputasi yang tinggi (Howdy, 2025), sehingga perlu dilakukan analisis lebih lanjut mengenai efisiensinya dalam klasifikasi suara ayam.

Salah satu metode utama dalam ekstraksi fitur suara adalah *Mel-Frequency Cepstral Coefficients* (MFCC). MFCC berfungsi untuk mengubah sinyal suara menjadi representasi yang lebih mudah diolah oleh model *deep learning*. Dengan menggunakan MFCC, pola suara ayam dapat dikenali secara lebih efisien dan memberikan hasil yang lebih akurat dalam klasifikasi vokalisasi (Karina, 2020).

Penelitian mengenai pengenalan suara hewan masih tergolong terbatas dibandingkan dengan penelitian dalam bidang pengenalan suara manusia. Beberapa studi telah mengeksplorasi pengenalan suara pada hewan lain, seperti penelitian oleh Gumilang (2025) membahas klasifikasi suara katak menggunakan model *deep learning modified densenet-121* dan *densenet-169* dengan fitur ekstraksi MFCC. Selain itu, penelitian oleh Afida (2020) mengeksplorasi klasifikasi jenis burung berdasarkan suara menggunakan algoritme *support vector machine*. Namun, masih sedikit penelitian yang secara khusus membahas klasifikasi vokalisasi ayam menggunakan *deep learning*. Oleh karena itu, penelitian ini bertujuan untuk melakukan perbandingan antara 3 metode *deep learning*, yaitu CNN, *Transformer*, dan *Deepspeech*, dalam mengklasifikasikan suara ayam berdasarkan jenis vokalisasinya.

Studi ini diharapkan dapat memberikan kontribusi dalam pengembangan sistem pemantauan otomatis yang berbasis kecerdasan buatan untuk industri peternakan. Dengan adanya sistem klasifikasi yang akurat, peternak dapat lebih mudah mendeteksi tanda-tanda stres, penyakit, atau kondisi tidak normal lainnya pada ayam, sehingga memungkinkan intervensi lebih dini untuk meningkatkan kesejahteraan hewan. Selain itu, penelitian ini juga berpotensi menjadi referensi

bagi pengembangan lebih lanjut dalam bidang bioakustik dan teknologi AI untuk peternakan. Pemanfaatan AI dalam analisis vokalisasi ayam tidak hanya membantu meningkatkan produktivitas peternakan tetapi juga berkontribusi pada kesejahteraan hewan secara keseluruhan.

Penelitian ini akan diarahkan pada pengembangan aplikasi *mobile* yang memungkinkan peternak memantau kondisi ayam secara *real-time*. Aplikasi ini akan memanfaatkan model terbaik dari hasil perbandingan, memberikan informasi langsung mengenai kondisi ayam berdasarkan analisis suara. Dengan adanya aplikasi *mobile*, peternak mendapatkan akses lebih mudah terhadap data kesehatan ayam dan dapat mengambil tindakan preventif lebih cepat berdasarkan hasil analisis AI. Implementasi teknologi ini diharapkan membawa perubahan signifikan dalam industri peternakan dengan mengurangi risiko penyakit dan meningkatkan kesejahteraan hewan secara lebih efisien dan modern.

Penelitian ini juga merujuk pada studi sebelumnya yang dilakukan oleh Wijaya (2024) di KRPA (Kelompok Riset *Precision Agriculture*), khususnya dalam penggunaan *dataset* vokalisasi ayam. Penelitian tersebut menggunakan MFCC untuk mengekstraksi ciri fitur dari data audio, yang kemudian dihitung rata-ratanya dan dijadikan *input* untuk pelatihan model. Terdapat 3 jenis model yang dibandingkan, yaitu LSTM, Bi-LSTM, dan Conv2D-LSTM. Perbandingan dilakukan berdasarkan lima faktor utama, yaitu nilai akurasi, *precision*, *recall*, *F1-score*, serta ukuran dan efisiensi model saat diimplementasikan melalui API *server*. Model Conv2D-LSTM selanjutnya diuji untuk menentukan konfigurasi terbaik. Dari hasil pengujian, ditemukan dua model terbaik dengan arsitektur Conv2D-LSTM.

1.2. Identifikasi Masalah

Setiap jenis vokalisasi ayam memiliki karakteristik unik yang mencerminkan kondisi dan aktivitas tertentu, seperti rasa lapar, stres, atau adanya ancaman. Namun, perbedaan vokalisasi ini sering kali sulit dikenali secara konsisten oleh manusia karena keterbatasan pendengaran dan interpretasi yang

bersifat subjektif. Selain itu, variasi suara ayam juga dipengaruhi oleh faktor-faktor seperti usia, lingkungan, dan jenis ayam, yang semakin memperumit proses identifikasi secara manual. Kompleksitas ini menimbulkan tantangan dalam membedakan jenis vokalisasi secara akurat, sehingga diperlukan pendekatan yang lebih objektif dan sistematis untuk mengatasi permasalahan tersebut.

1.3. Batasan Masalah

a. Menggunakan bahasa pemrograman Python untuk pengolahan data dan pelatihan model.
b. 4 *Dataset* vokalisasi ayam yang digunakan pada penelitian ini didapatkan melalui *Youtube* dan *Github,*
c. *Dataset* memiliki 4 jenis vokalisasi, yaitu suara ayam betina marah, suara ayam betina memanggil jantan, suara ayam ketika ada ancaman, dan suara ayam setelah bertelur.
d. Ekstraksi fitur suara dilakukan menggunakan *Mel-Frequency Cepstral Coefficients* (MFCC).
e. Jenis ayam yang digunakan sampel suaranya adalah ayam yang sering ditemukan di peternakan yaitu ayam *broiler*.
f. Perbandingan dilakukan pada 3 jenis model *deep learning*, yaitu CNN, *Transformer*, dan *Deepspeech*.
g. Implementasi model dilakukan dalam bentuk aplikasi *mobile* untuk memudahkan akses dan pemantauan suara ayam.
h. Dalam proses pengembangan model, penulis menggunakan bantuan kecerdasan buatan seperti ChatGPT dan Claude.ai sebagai alat bantu dalam memahami konsep dan implementasi.

1.4. Perumusan Masalah

Berdasarkan latar belakang dan batasan masalah yang telah ditentukan, penelitian ini merumuskan pertanyaan sebagai berikut:

1. Bagaimana cara mengklasifikasikan vokalisasi ayam untuk mendeteksi kondisi ayam secara otomatis?
2. Model *deep learning* mana yang paling optimal dalam mengklasifikasikan suara ayam antara CNN, *Transformer*, dan *Deepspeech*?
3. Bagaimana implementasi sistem pemantauan suara ayam dalam bentuk aplikasi *mobile* agar dapat digunakan secara *real-time*?

1.5. Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Mengembangkan sistem klasifikasi vokalisasi ayam guna mendeteksi kondisi ayam dengan *deep learning*.
2. Mengevaluasi dan membandingkan performa 3 model *deep learning*, yaitu CNN, *Transformer*, dan *Deepspeech*, dalam mengklasifikasikan suara ayam.
3. Mengimplementasikan sistem klasifikasi vokalisasi ayam dalam aplikasi *mobile* yang dapat digunakan secara *real-time* untuk pemantauan kondisi ayam.

1.6. Luaran

Luaran dari penelitian ini adalah aplikasi *mobile* yang bisa mendeteksi jenis vokalisasi ayam, draf artikel untuk publikasi jurnal, dan laporan tugas akhir.

1.7. Manfaat penelitian

1. Bagi Universitas Ma Chung, khususnya program studi Teknik Informatika, penelitian ini dapat menjadi referensi dalam pengembangan studi terkait pemrosesan suara dan implementasi *deep learning*.
2. Bagi penulis, penelitian ini memberikan pengalaman dalam menerapkan CNN, *Transformer*, dan *Deepspeech* untuk klasifikasi suara ayam

berdasarkan vokalisasinya. Selain itu, penelitian ini membantu memperdalam pemahaman tentang pengolahan sinyal audio, ekstraksi fitur menggunakan *Mel-Frequency Cepstral Coefficients* (MFCC), serta penerapan *deep learning* dalam *speech processing*. Dengan penelitian ini, penulis juga memperoleh pengalaman dalam mengintegrasikan model *machine learning* ke dalam aplikasi *mobile.*

3. Bagi pembaca, penelitian ini dapat menjadi sumber wawasan dan inspirasi dalam mengembangkan sistem deteksi emosi dan kesehatan hewan berbasis suara, serta mendorong penelitian lanjutan di bidang serupa.

Bab II
Tinjauan Pustaka

2.1. Jenis Ayam

Industri peternakan unggas memegang peran penting dalam memenuhi kebutuhan pangan masyarakat, khususnya dalam penyediaan sumber protein hewani seperti daging dan telur. Seiring meningkatnya permintaan pasar terhadap produk unggas, efisiensi dan produktivitas dalam beternak menjadi aspek yang sangat diperhatikan. Salah satu tantangan utama dalam peternakan ayam adalah bagaimana mendeteksi kondisi kesehatan atau perilaku ayam secara dini agar tindakan pencegahan dapat segera dilakukan. Hal ini menjadi semakin krusial mengingat perubahan kondisi ayam sering kali ditandai dengan sinyal-sinyal non-verbal, seperti perubahan dalam vokalisasi (Adebayo et al., 2023).

Dalam konteks tersebut, pengembangan sistem klasifikasi vokalisasi ayam menjadi relevan, terutama jika diintegrasikan ke dalam platform yang mudah diakses seperti aplikasi *mobile*. Penelitian ini secara khusus memfokuskan diri pada jenis ayam *broiler* (ayam pedaging), yang merupakan jenis ayam yang diternakkan untuk kebutuhan konsumsi daging (Mao *et al.,* 2022).

Dalam industri peternakan modern, ayam *broiler* memegang peran sentral karena mampu menghasilkan daging dalam waktu yang relatif singkat dengan biaya produksi yang efisien. Permintaan pasar terhadap ayam *broiler* terus meningkat seiring bertambahnya jumlah penduduk dan tingginya konsumsi protein hewani oleh masyarakat (Zampiga *et al.,* 2021).

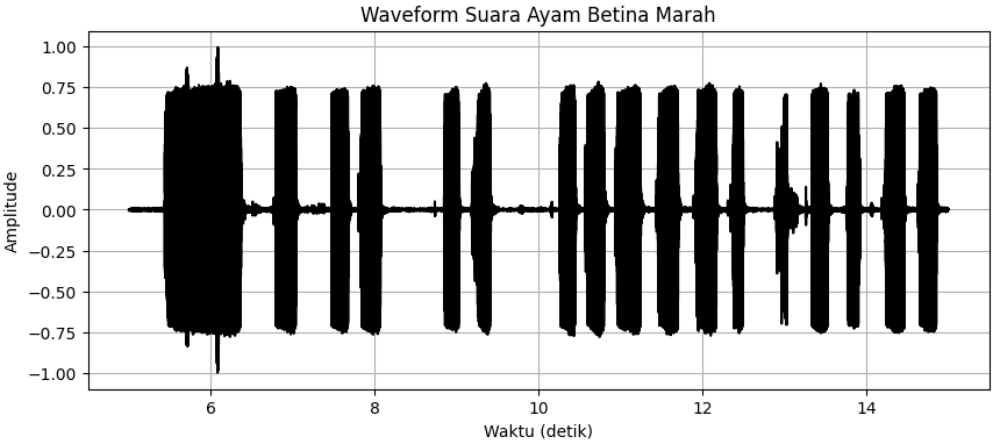
Karena perputaran siklus budidaya ayam *broiler* yang sangat cepat, peternak dituntut untuk dapat memantau kesehatan dan kondisi ayam secara efisien. Masalah seperti stres, penyakit, atau gangguan perilaku bisa berdampak langsung pada pertumbuhan dan kualitas daging ayam, serta menyebabkan kerugian ekonomi apabila tidak ditangani sejak dini (Apalowo *et al.,* 2024).

Di sinilah teknologi berperan penting. Salah satu pendekatan yang mulai banyak dikembangkan adalah memanfaatkan suara atau vokalisasi ayam sebagai

indikator kondisi mereka. Suara ayam dapat menjadi petunjuk terhadap keadaan tertentu, misalnya ketika ayam merasa lapar, terancam, atau sedang mengalami gangguan kesehatan (Soster *et al.,* 2025).

2.2. Vokalisasi Ayam

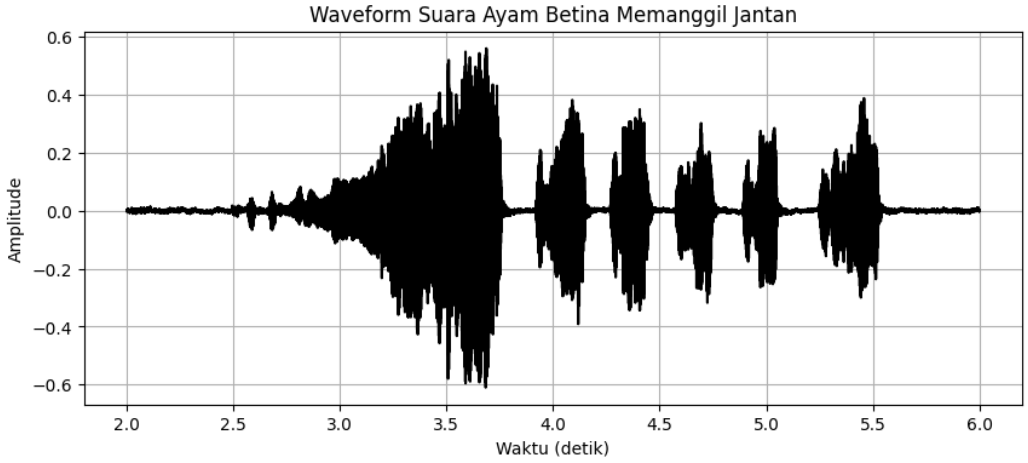
Ayam merupakan hewan yang berkomunikasi melalui berbagai jenis vokalisasi untuk menyampaikan informasi tertentu kepada sesamanya. Vokalisasi ini dapat mencerminkan kondisi emosional, tingkat stres, serta keadaan kesehatan ayam. Oleh karena itu, analisis suara ayam dapat menjadi metode yang efektif untuk memahami kondisi ayam secara otomatis (Merdeka.com, 2023). Penelitian yang dipimpin oleh Profesor Adrian David Cheok dari Universitas Tokyo menggunakan teknologi kecerdasan buatan untuk menganalisis vokalisasi ayam. Mereka berhasil mengidentifikasi enam keadaan emosi yang berbeda, termasuk rasa lapar, takut, marah, puas, gembira, dan kesusahan, dengan akurasi 80%. Penelitian ini menunjukkan bahwa analisis vokalisasi dapat digunakan untuk memahami kondisi emosional ayam (Cheok *et al.,* 2023). Berikut *waveform* 4 macam vokalisasi ayam yang akan digunakan di dalam penelitian ini, yaitu suara ayam betina marah, suara ayam betina memanggil jantan, suara ayam ketika ada ancaman, dan suara ayam setelah bertelur.



Gambar 2.1 *Waveform* vokalisasi ayam betina marah

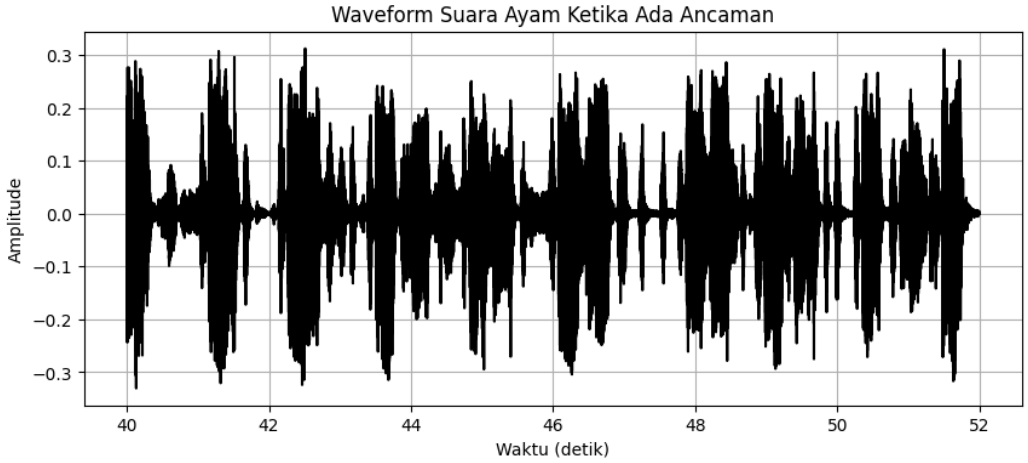
Gambar 2.1 menunjukkan *waveform* suara ayam betina yang sedang marah. Suara ini ditandai dengan pola gelombang yang memiliki amplitudo tinggi dan

durasi pendek secara berulang, yang mengindikasikan suara keras dan cepat. Berdasarkan karakteristik suara, ayam betina yang marah biasanya mengeluarkan vokalisasi dengan nada tinggi, seperti teriakan keras dan berulang (petok-petok cepat).



Gambar 2.2 *Waveform* vokalisasi ayam betina memanggil jantan

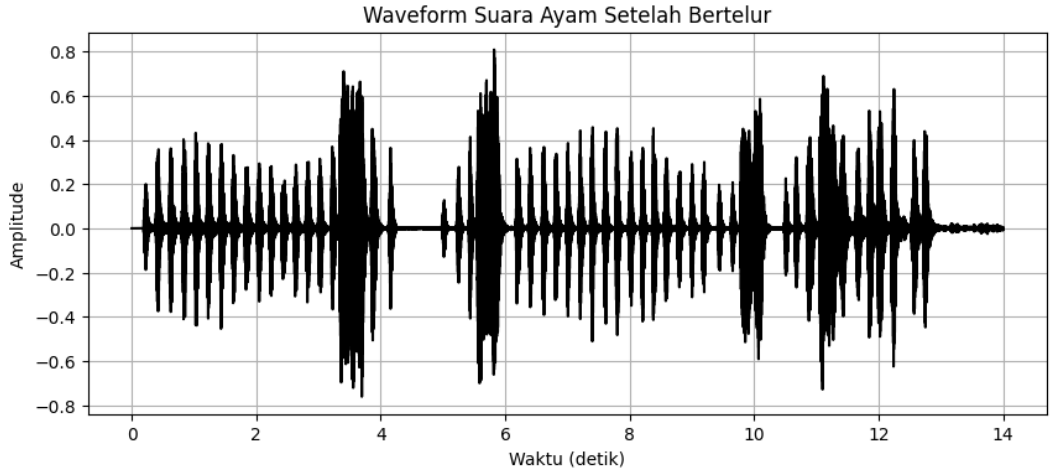
Gambar 2.2 menunjukkan pola vokalisasi ayam betina saat memanggil ayam jantan. Suara ini ditandai dengan pola gelombang yang terputus-putus dengan jeda di antara setiap panggilan. Dari grafik, dapat terlihat bahwa amplitudo suara bervariasi, dengan beberapa bagian menunjukkan lonjakan yang lebih tinggi, menandakan intensitas suara yang lebih kuat.



Gambar 2.3 *Waveform* vokalisasi ayam ketika ada ancaman

Gambar 2.3 menunjukkan *waveform* suara ayam ketika merasakan adanya ancaman. Vokalisasi ini ditandai dengan pola suara yang tidak teratur, memiliki

amplitudo yang cukup besar, dan terdiri dari suara keras yang berulang dalam waktu singkat. Suara ini dikenal sebagai "alarm calls" yang sering kali terdengar berisik, riuh, dan panik.



Gambar 2.4 *Waveform* vokalisasi ayam setelah bertelur

Gambar 2.4 menunjukkan *waveform* dari suara ayam betina setelah bertelur. Pola suara ini ditandai dengan serangkaian vokalisasi khas yang sering disebut sebagai "egg song." Vokalisasi ini memiliki pola amplitudo yang berulang dengan frekuensi yang cukup tinggi, yang menunjukkan nada yang nyaring dan jelas.

2.3. Augmentasi Suara

Augmentasi suara adalah teknik yang digunakan untuk memperkaya variasi data audio dalam proses pelatihan model pembelajaran mesin (ML). Dengan menerapkan berbagai transformasi pada sinyal audio, augmentasi dapat meningkatkan ketahanan model terhadap variasi suara dalam kondisi nyata. Teknik ini sering digunakan untuk memperbaiki kinerja model dalam tugas seperti klasifikasi suara, deteksi pola akustik, dan pengenalan suara di lingkungan bising (Mulyana dan Nurrohman, 2025). Beberapa metode augmentasi suara yang umum digunakan meliputi penambahan *noise* (*noise injection*) untuk membuat model lebih tahan terhadap gangguan suara latar, perubahan kecepatan (*time stretching*) untuk meningkatkan variasi temporal tanpa mengubah frekuensi, perubahan nada (*pitch shifting*) guna memperkaya variasi akustik, pergeseran waktu (*time shifting*)

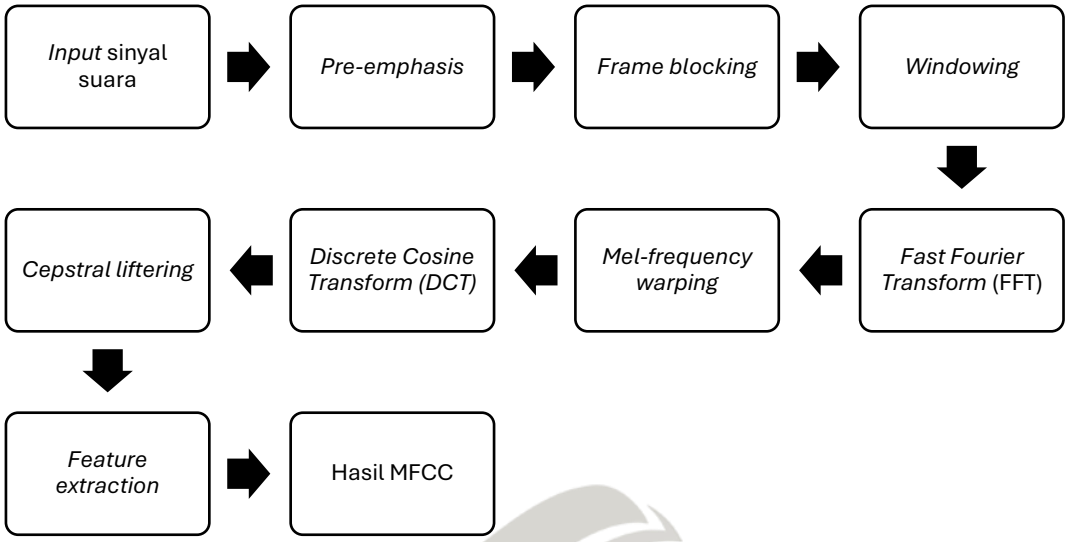
untuk mengubah posisi sinyal audio dalam skala waktu, serta penyesuaian *gain* secara acak (*random gain*) untuk mengubah intensitas suara secara acak (Halim dkk., 2024).

2.4. *Mel-Frequency Cepstral Coefficients* (MFCC)

Mel-Frequency Cepstral Coefficients (MFCC) adalah teknik ekstraksi fitur yang umum digunakan dalam pemrosesan sinyal suara, terutama dalam pengenalan ucapan, identifikasi pembicara, dan analisis akustik. MFCC pertama kali diperkenalkan oleh Davis dan Mermelstein pada akhir tahun 1980 dan telah menjadi metode standar dalam berbagai aplikasi pengolahan suara. MFCC bekerja dengan mengubah sinyal suara menjadi serangkaian koefisien yang merepresentasikan fitur-fitur utama dari sinyal tersebut. Koefisien ini meniru cara kerja sistem pendengaran manusia dengan memanfaatkan skala *mel*, yang lebih sesuai dengan persepsi frekuensi manusia dibandingkan skala linear (Ajinurseto dkk., 2023).

MFCC banyak digunakan dalam berbagai aplikasi, seperti pengenalan suara manusia, identifikasi spesies hewan melalui suara, dan analisis akustik dalam sistem kecerdasan buatan. Sebagai contoh, Adhinata dkk. (2021) melakukan penelitian tentang pengenalan jenis kelamin manusia berbasis suara menggunakan MFCC dan GMM. Lalu, ada juga penelitian yang dilakukan oleh Gultom dkk. (2025) yang membahas tentang klasifikasi suara nyamuk berbasis CNN untuk inovasi pengendalian hama dan penyakit, dimana pada proses ekstraksi fitur menggunakan MFCC.

Proses ekstraksi fitur MFCC umumnya terdiri dari beberapa tahap utama, yaitu *pre-emphasis*, *frame blocking*, *windowing*, *Fast Fourier Transform* (FFT), *mel-frequency wrapping*, *discrete cosine transforms, cepstral liftering,* dan *feature extraction*. Berikut Gambar 2.5 menampilkan diagram alur proses ekstraksi MFCC yang menggambarkan bagaimana sinyal suara diubah menjadi representasi fitur yang lebih kompak dan informatif.



Gambar 2.5 Diagram alur proses ekstrasi MFCC

2.4.1. *Pre-emphasis*

Pre-emphasis adalah salah satu tahap awal dalam pemrosesan sinyal suara yang bertujuan untuk memperkuat frekuensi tinggi dalam sinyal sebelum dilakukan analisis lebih lanjut. Teknik ini diterapkan untuk mengatasi atenuasi alami pada komponen frekuensi tinggi akibat karakteristik fisik alat perekam dan sistem vokal manusia (Hendry *et al.,* 2022).

Dalam sinyal suara mentah, energi pada frekuensi rendah cenderung lebih tinggi dibandingkan dengan frekuensi tinggi, yang dapat menyebabkan kehilangan informasi penting dalam analisis spektral. Dengan menerapkan *pre-emphasis*, perbedaan ini dapat dikurangi, sehingga fitur-fitur penting pada frekuensi tinggi tetap dapat dipertahankan. *Pre-emphasis* biasanya berbentuk filter digital dengan persamaan:

$$y(n) = x(n) - \alpha x(n-1) \tag{2-1}$$

Di mana $x(n)$ adalah sinyal *input*, $y(n)$ adalah sinyal *output* yang telah difilter, dan α adalah koefisien *pre-emphasis* yang biasanya bernilai antara 0.9 hingga 0.97 (Fauziah dkk., 2021).

Dengan menerapkan *pre-emphasis* sebelum proses ekstraksi fitur, informasi suara menjadi lebih seimbang, dan analisis spektral dapat menghasilkan representasi yang lebih akurat terhadap karakteristik suara asli.

2.4.2. *Frame Blocking*

Frame blocking adalah tahap awal dalam pemrosesan sinyal suara yang bertujuan untuk membagi sinyal menjadi segmen-segmen kecil (*frame*) agar dapat dianalisis secara lebih akurat dalam domain waktu dan frekuensi. Teknik ini diperlukan karena sinyal suara bersifat non-stasioner, yang berarti karakteristik spektralnya dapat berubah seiring waktu. Dengan melakukan *frame blocking*, sinyal dapat dianggap stasioner dalam interval waktu yang lebih pendek, memungkinkan ekstraksi fitur yang lebih representatif (Ranny dkk., 2019).

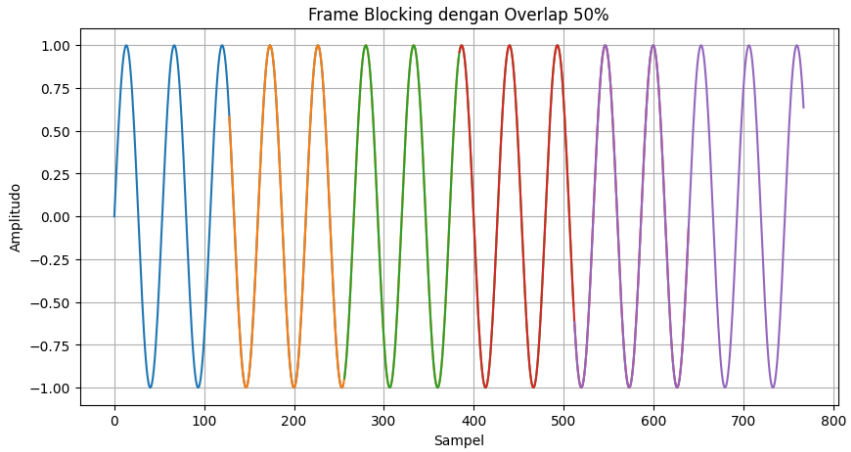
Dalam proses *frame blocking*, setiap *frame* memiliki ukuran tertentu yang disebut *frame size*, serta jarak antar *frame* yang disebut *frame shift*. Jika *frame size* terlalu besar, informasi detail dari sinyal dapat hilang, sedangkan jika terlalu kecil, maka sinyal dapat kehilangan kontinuitas. Oleh karena itu, umumnya *frame size* berkisar antara 20 hingga 40 milidetik, dengan *frame shift* sekitar 10 milidetik.

Secara matematis, *frame blocking* dapat direpresentasikan dengan rumus berikut:

$$X_f(n) = x(n + f \cdot S), 0 \leq n < N \quad (2\text{-}2)$$

Dalam rumus ini, $X_f(n)$ merupakan sampel sinyal pada *frame* ke-f, yang diambil dari sinyal asli $x(n)$. Parameter N menunjukkan ukuran *frame*, yaitu jumlah sampel yang terdapat dalam satu *frame*. Sementara itu, S merupakan jumlah sampel yang bergeser antar *frame*, yang disebut sebagai *frame shift*. Indeks f menunjukkan urutan *frame* dalam pemrosesan sinyal. Jika nilai S lebih kecil dari N, maka akan terjadi *overlap* antar *frame* yang bertujuan untuk mempertahankan kontinuitas informasi dalam sinyal suara.

Untuk memberikan ilustrasi lebih lanjut, berikut adalah grafik yang menggambarkan proses *frame blocking* dalam sinyal suara:



Gambar 2.6 Grafik *frame blocking* dengan *overlap* 50%

Gambar 2.6 menunjukkan bagaimana sinyal suara diproses dalam beberapa *frame* dengan beberapa segmen yang sedikit tumpang tindih untuk mempertahankan informasi suara. Dengan pendekatan ini, berbagai fitur suara dapat diekstraksi secara lebih optimal, yang mendukung proses seperti *speech recognition*, pengolahan audio, dan analisis spektrum suara.

2.4.3. *Windowing*

Windowing adalah teknik dalam pemrosesan sinyal yang bertujuan untuk mengurangi efek kebocoran spektral akibat pemotongan sinyal selama proses *frame blocking*. Ketika sinyal dibagi menjadi beberapa *frame*, bagian tepi setiap *frame* dapat mengalami diskontinuitas, yang kemudian menyebabkan distorsi dalam domain frekuensi saat dilakukan transformasi Fourier. Untuk mengatasi masalah ini, diterapkan fungsi *window* pada setiap *frame* guna membuat transisi antar *frame* menjadi lebih halus. Teknik ini sering digunakan dalam analisis sinyal suara, pemrosesan gambar, dan berbagai aplikasi berbasis frekuensi lainnya. (Hazmar dan Prasetio, 2023).

Dalam *windowing*, sinyal dalam satu *frame* dikalikan dengan fungsi *window* yang memiliki bentuk tertentu. Fungsi ini memberikan pembobotan pada amplitudo

sinyal sehingga bagian tengah *frame* tetap dominan, sementara bagian tepi secara bertahap mengecil mendekati nol. Dengan demikian, efek diskontinuitas pada batas *frame* dapat diminimalkan. Beberapa fungsi *window* yang umum digunakan antara lain *rectangular window, hamming window, hanning window,* dan *blackman window*. Setiap jenis memiliki karakteristik berbeda dalam menyeimbangkan resolusi frekuensi dan kebocoran spektral (Caroline *et al.,* 2020).

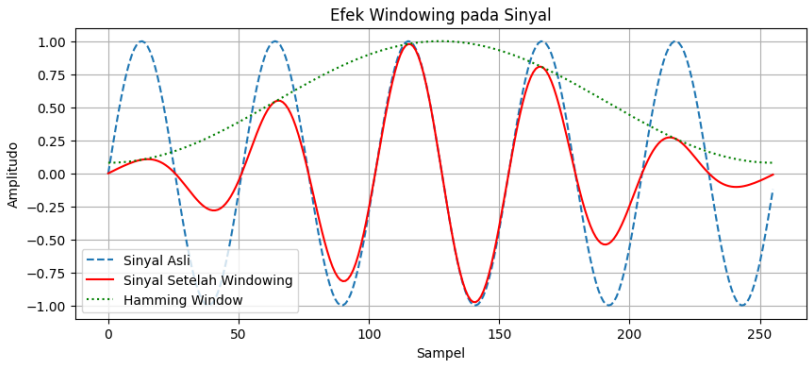
Secara matematis, proses *windowing* dapat dinyatakan dengan rumus:

$$X_w(n) = X_f(n) \cdot w(n), 0 \leq n < N \tag{2-3}$$

Dalam persamaan tersebut, $X_w(n)$ merupakan sinyal setelah diterapkan *windowing*, $X_f(n)$ adalah sinyal asli dalam satu *frame*, $w(n)$ adalah fungsi *window*, dan N adalah ukuran *frame*. Salah satu fungsi *window* yang banyak digunakan adalah *hamming window*, yang dirumuskan sebagai berikut:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), 0 \leq n < N \tag{2-4}$$

Fungsi *hamming window* memberikan transisi yang lebih lembut dibandingkan *rectangular window*, sehingga mampu mengurangi kebocoran spektral secara signifikan. Implementasi *windowing* dapat divisualisasikan dengan grafik yang menunjukkan perubahan amplitudo sinyal sebelum dan sesudah diterapkan *window*. Pada Gambar 2.7, sinyal asli berbentuk gelombang sinusoidal mengalami perubahan setelah dikalikan dengan fungsi *hamming window*.



Gambar 2.7 Grafik efek *windowing* pada sinyal

Dalam grafik tersebut, menunjukkan bagaimana fungsi *window* diterapkan pada sinyal. Sinyal asli (garis biru) memiliki tepi yang tajam, yang dapat

menyebabkan distorsi spektral saat dilakukan transformasi Fourier. Setelah diterapkan *hamming window* (garis merah), amplitudo sinyal berkurang secara bertahap di tepi *frame*, sehingga transisi antar *frame* menjadi lebih halus dan mengurangi kebocoran spektral.

2.4.4. *Fast Fourier Transform* (FFT)

Fast Fourier Transform (FFT) adalah algoritma yang digunakan untuk menghitung *Discrete Fourier Transform* (DFT) dan inversinya dengan lebih cepat dan efisien. Transformasi Fourier sendiri merupakan teknik matematika yang digunakan untuk mengubah sinyal dari domain waktu ke domain frekuensi, sehingga memungkinkan analisis komponen frekuensi dalam sinyal. Algoritma FFT mempercepat proses ini dengan mengurangi kompleksitas komputasi dari $O(N^2)$ menjadi $O(N \log N)$, di mana N adalah jumlah sampel dalam sinyal (Ramezanian *et al.,* 2020).

Dalam pemrosesan sinyal digital, FFT digunakan secara luas dalam berbagai aplikasi seperti pengolahan audio, pengenalan suara, komunikasi digital, radar, dan analisis spektrum. Dengan FFT, informasi frekuensi dalam suatu sinyal dapat diekstraksi dengan lebih cepat, memungkinkan sistem untuk merespons dalam waktu nyata.

Implementasi FFT didefinisikan sebagai berikut:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, k = 0,1,\ldots,N-1 \quad (2\text{-}5)$$

Persamaan FFT menyatakan bahwa $X(k)$ merupakan representasi sinyal dalam domain frekuensi, yang diperoleh melalui transformasi dari sinyal dalam domain waktu $x(n)$. Jumlah sampel dalam sinyal dinyatakan sebagai N, yang menentukan resolusi frekuensi dalam analisis spektrum. Indeks frekuensi k menunjukkan komponen frekuensi tertentu dalam hasil transformasi, sementara j adalah bilangan imajiner dengan sifat $j^2 = -1$, yang digunakan dalam perhitungan eksponensial kompleks dalam transformasi Fourier.

2.4.5. *Mel-Frequency Warping*

Mel-frequency warping adalah teknik yang digunakan dalam analisis dan pengolahan sinyal suara, khususnya dalam konversi frekuensi untuk meningkatkan pemrosesan suara. Teknik ini memanfaatkan skala frekuensi *mel* untuk merubah spektrum sinyal sehingga lebih sesuai dengan cara manusia mendengar suara. Skala frekuensi *mel* ini didasarkan pada pengamatan bahwa telinga manusia lebih sensitif terhadap perbedaan frekuensi pada frekuensi rendah dibandingkan dengan frekuensi tinggi (Yehezkiel dan Suyanto, 2022).

Mel-frequency warping berfokus pada transformasi dari skala frekuensi linear ke skala frekuensi *mel*. Pada skala frekuensi linear, perbedaan antara dua frekuensi yang terletak dekat di bagian bawah spektrum sangat mudah dideteksi oleh telinga manusia. Namun, perbedaan serupa pada frekuensi yang lebih tinggi sulit untuk dikenali. Oleh karena itu, *mel-frequency warping* mengubah spektrum frekuensi menjadi bentuk yang lebih sesuai dengan persepsi pendengaran manusia (Heriyanto dkk., 2018).

Skala *mel*, yang digunakan dalam *mel-frequency warping*, didefinisikan sebagai berikut:

$$f_{\text{mel}} = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{2-6}$$

dimana f adalah frekuensi dalam *hertz* (*Hz*), dan f_{mel} adalah frekuensi dalam skala *mel*. *Mel-frequency warping* ini digunakan untuk mengurangi distorsi yang muncul ketika informasi spektrum sinyal frekuensi tinggi diproses.

Mel-frequency warping banyak digunakan dalam bidang pengolahan suara dan pengenalan ucapan, terutama dalam ekstraksi fitur dari sinyal suara untuk analisis lebih lanjut. Salah satu aplikasi utamanya adalah dalam perhitungan *Mel-Frequency Cepstral Coefficients* (MFCCs), yang digunakan dalam pengenalan ucapan. Dengan menggunakan *mel-frequency warping*, MFCC dapat memberikan representasi yang lebih akurat mengenai informasi penting dalam sinyal suara, serta memungkinkan pemrosesan yang lebih efisien (Hartayu dkk., 2022).

2.4.6. *Discrete Cosine Transform* (DCT)

Discrete Cosine Transform (DCT) adalah metode transformasi matematis yang digunakan untuk merepresentasikan sinyal dalam domain frekuensi dengan hanya menggunakan komponen kosinus. DCT sering digunakan dalam pemrosesan sinyal suara, termasuk dalam ekstraksi fitur untuk pengenalan suara dan kompresi data audio (Heriyanto, 2021).

Dalam proses ekstraksi fitur seperti MFCC, DCT digunakan setelah penerapan *mel filterbank* untuk mereduksi dimensi dan menghilangkan korelasi antar-koefisien. DCT bekerja pada hasil dari *mel filterbank* dan hanya mempertimbangkan komponen *real*, sehingga efektif dalam merepresentasikan energi sinyal dalam jumlah koefisien yang lebih sedikit. (Siregar dkk., 2018).

Persamaan umum untuk 1D-DCT dari suatu sinyal diskrit $x(n)$ dengan panjang N didefinisikan sebagai:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi k}{N}\left(n + \frac{1}{2}\right)\right] \quad (2\text{-}7)$$

dengan $k = 0,1,2,\ldots,N-1$. Transformasi ini menghasilkan serangkaian koefisien yang merepresentasikan distribusi energi dari sinyal asli dalam domain frekuensi.

Dalam pemrosesan sinyal suara, terutama dalam ekstraksi (MFCC), DCT digunakan untuk mengubah spektrum *log-magnitude* dari *mel filterbank* menjadi representasi yang lebih kompak. Langkah ini bertujuan untuk menghilangkan korelasi antar koefisien dan mempertahankan informasi penting dalam jumlah fitur yang lebih sedikit.

Proses umum penerapan DCT dalam MFCC melibatkan langkah-langkah berikut:

1. Penerapan *mel-frequency warping*: Sinyal suara yang telah melalui transformasi Fourier dikonversi ke skala *mel* dengan *filterbank mel*.
2. Pengambilan log-amplitudo: Hasil dari *filterbank mel* dikonversi ke bentuk logaritmik untuk mendekati cara manusia merasakan suara.

3. Penerapan DCT: DCT diterapkan untuk mengubah spektrum logaritmik menjadi representasi yang lebih terkompresi, menghasilkan MFCC.

Koefisien hasil DCT yang pertama (koefisien nol) biasanya menggambarkan energi keseluruhan sinyal dan sering kali diabaikan dalam pengenalan suara, karena tidak berkontribusi banyak pada fitur spesifik suara.

Keunggulan utama DCT adalah kemampuannya untuk merepresentasikan sinyal dengan lebih sedikit koefisien tanpa kehilangan banyak informasi. DCT sangat efisien dalam merepresentasikan data karena sebagian besar energi sinyal dapat dipertahankan dalam hanya beberapa koefisien pertama. Selain itu, DCT membantu menghilangkan korelasi antar fitur, sehingga memudahkan pemrosesan lebih lanjut dalam algoritma pengenalan pola. Teknik ini juga memungkinkan reduksi dimensi yang signifikan, seperti dalam aplikasi MFCC, di mana hanya beberapa koefisien pertama yang digunakan untuk ekstraksi fitur suara, menjadikannya lebih ringkas namun tetap informatif (Ali *et al.,* 2020).

2.4.7. *Cepstral Liftering*

Setelah proses transformasi dengan DCT untuk menghilangkan korelasi antar-koefisien dan menghasilkan representasi yang lebih kompak, langkah selanjutnya dalam ekstraksi MFCC adalah menerapkan *cepstral liftering*. *Cepstral liftering* bertujuan untuk meningkatkan kualitas fitur suara dengan menyeimbangkan distribusi energi pada koefisien *cepstral*. Proses ini dilakukan dengan menerapkan sebuah fungsi jendela (*lifter*) pada koefisien MFCC agar informasi yang lebih relevan dapat dipertahankan dan koefisien yang kurang signifikan dapat diminimalkan (Taoufiq *et al.,* 2022).

Dalam hasil ekstraksi MFCC, koefisien yang lebih rendah (misalnya, koefisien pertama) sering kali memiliki nilai yang besar dan mendominasi, sedangkan koefisien yang lebih tinggi memiliki nilai yang lebih kecil dan cenderung mengandung *noise* atau informasi yang kurang relevan. Hal ini menyebabkan ketidakseimbangan dalam distribusi energi fitur. Untuk mengatasi

masalah ini, dilakukan proses *cepstral liftering* dengan menerapkan faktor skala tertentu pada koefisien MFCC (Lu *et al.,* 2024).

Secara matematis, proses *cepstral liftering* dapat dinyatakan sebagai:

$$\hat{c}(n) = \left(1 + \frac{L}{2}\sin\left(\frac{\pi n}{L}\right)\right) c(n) \tag{2-8}$$

Di dalam persamaan ini, $\hat{c}(n)$ merupakan koefisien MFCC setelah *liftering*, sedangkan $c(n)$ adalah koefisien MFCC sebelum *liftering*. Parameter L berperan sebagai faktor *lifter* yang menentukan seberapa besar efek dari *cepstral liftering* yang diterapkan. Sementara itu, n menunjukkan indeks dari masing-masing koefisien cepstral dalam urutan hasil ekstraksi fitur.

Terdapat dua jenis utama *cepstral liftering* yang sering digunakan dalam ekstraksi MFCC, yaitu:

1. *Low-pass liftering*: Bertujuan untuk menghilangkan koefisien dengan indeks tinggi yang cenderung mengandung *noise* atau informasi yang kurang signifikan. Teknik ini memperhalus representasi fitur dengan mempertahankan koefisien yang lebih rendah, yang umumnya lebih informatif.
2. *High-pass liftering*: Digunakan untuk menghilangkan koefisien dengan indeks rendah, termasuk koefisien pertama (c_0) yang merepresentasikan energi sinyal secara keseluruhan. Teknik ini berguna dalam pengenalan suara yang berfokus pada pola spektral daripada intensitas sinyal.

Salah satu efek utama *cepstral liftering* adalah mengurangi dominasi energi pada koefisien pertama dalam MFCC. Koefisien pertama (c_0) sering kali memiliki nilai yang jauh lebih besar dibandingkan koefisien lainnya, sehingga dapat mendominasi representasi fitur. Dengan menerapkan *cepstral liftering*, efek dominasi ini dapat dikurangi, sehingga fitur yang dihasilkan menjadi lebih seimbang. Selain itu, *cepstral liftering* juga meningkatkan ketahanan terhadap variasi suara antar pembicara. Dalam sistem pengenalan suara, variasi frekuensi dan intonasi dapat menyebabkan perbedaan signifikan pada fitur MFCC. Dengan

menerapkan teknik ini, fitur yang diekstraksi menjadi lebih *robust* terhadap perbedaan individu, sehingga meningkatkan akurasi sistem pengenalan suara.

2.4.8. *Feature Extraction*

Tahap terakhir dalam proses ekstraksi MFCC adalah *feature extraction*. Langkah ini bertujuan untuk mengekstrak dan memilih fitur yang paling representatif dari sinyal suara agar dapat digunakan dalam proses analisis dan klasifikasi lebih lanjut, seperti dalam sistem pengenalan suara (*speech recognition*) atau identifikasi pembicara (*speaker recognition*) (Noughabi *et al.,* 2024).

Feature extraction dalam konteks MFCC merupakan proses pemilihan dan pengolahan koefisien cepstral yang telah dihasilkan agar lebih sesuai untuk digunakan dalam tahap selanjutnya, seperti klasifikasi atau pengenalan pola. Tujuan utama dari tahap ini adalah untuk memperoleh fitur yang kompak, informatif, dan diskriminatif, sehingga dapat meningkatkan kinerja sistem pengenalan suara (Noughabi *et al.,* 2024).

Dalam sistem berbasis MFCC, fitur yang diekstrak meliputi:

1. Koefisien cepstral MFCC: Merupakan koefisien utama yang diperoleh setelah *cepstral liftering*. Biasanya, hanya sejumlah koefisien pertama yang digunakan karena mengandung informasi yang paling signifikan mengenai karakteristik akustik sinyal suara.
2. *Delta coefficients*: Merupakan turunan pertama dari koefisien MFCC, yang digunakan untuk menangkap perubahan dinamis dari fitur suara seiring waktu.
3. *Delta-delta Coefficients*: Merupakan turunan kedua dari koefisien MFCC, yang digunakan untuk menangkap akselerasi perubahan dalam fitur suara.

Dengan menambahkan koefisien *delta* dan *delta-delta*, fitur yang diekstrak menjadi lebih *robust* terhadap perubahan tempo dan variasi dalam pengucapan. Untuk meningkatkan ketahanan terhadap perubahan temporal, koefisien *delta* dan *delta-delta* dihitung menggunakan rumus sebagai berikut:

a) Koefisien *delta*

Koefisien *delta* dihitung dengan formula:

$$\Delta c(n) = \frac{\sum_{k=1}^{K} k\big(c(n+k) - c(n-k)\big)}{2\sum_{k=1}^{K} k^2} \tag{2-9}$$

Koefisien *delta*, yang dilambangkan dengan $\Delta c(n)$, merupakan turunan pertama dari koefisien MFCC pada indeks n. Nilai ini dihitung berdasarkan perbedaan nilai koefisien MFCC di sekitar titik waktu tersebut. Dalam rumusnya, $c(n)$ merepresentasikan koefisien MFCC asli pada indeks ke-n, sedangkan K menunjukkan ukuran jendela temporal yang digunakan dalam perhitungan. Nilai K ini umumnya dipilih antara 2 hingga 3 untuk menjaga kestabilan perhitungan serta menghindari pengaruh gangguan lokal pada sinyal suara. Dengan demikian, koefisien *delta* mampu menangkap pola perubahan fitur suara dari waktu ke waktu secara lebih akurat.

b) Koefisien *delta-delta*

Koefisien *delta-delta* dihitung dengan cara yang sama seperti *delta*, tetapi menggunakan nilai *delta* sebagai *input*-nya:

$$\Delta^2 c(n) = \frac{\sum_{k=1}^{K} k\big(\Delta c(n+k) - \Delta c(n-k)\big)}{2\sum_{k=1}^{K} k^2} \tag{2-10}$$

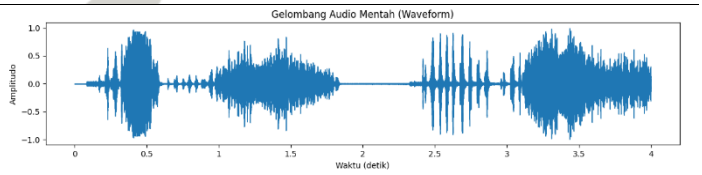
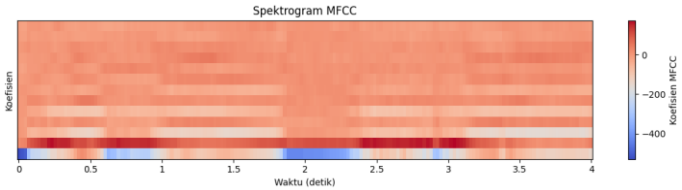
Koefisien *delta-delta*, yang dilambangkan dengan $\Delta^2 c(n)$, merupakan turunan kedua dari koefisien MFCC dan menggambarkan percepatan perubahan fitur suara pada indeks n. Nilai ini dihitung dengan menggunakan koefisien *delta* sebelumnya, yang dilambangkan dengan $\Delta c(n)$. Dengan memperhitungkan perubahan dari koefisien *delta*, fitur *delta-delta* memberikan informasi yang lebih kaya mengenai dinamika temporal dari sinyal suara, sehingga memperkuat kemampuan sistem dalam membedakan pola suara yang bersifat kompleks dan variatif.

Proses *feature extraction* memberikan sejumlah dampak penting dalam sistem berbasis suara. Pertama, proses ini dapat meningkatkan akurasi model dengan menambahkan koefisien *delta* dan *delta-delta*, yang memungkinkan sistem untuk mengenali perubahan pola suara secara lebih baik. Kedua, *feature extraction*

mampu mengurangi dimensi data dengan hanya memilih fitur yang paling relevan, seperti 12 atau 13 koefisien MFCC pertama, sehingga ukuran data menjadi lebih ringkas tanpa kehilangan informasi penting. Ketiga, proses ini meningkatkan ketahanan sistem terhadap variasi antar pembicara, karena fitur turunan seperti *delta* dan *delta-delta* dapat menangkap perbedaan karakteristik vokal. Terakhir, *feature extraction* juga mempercepat proses pengolahan sinyal karena data yang digunakan dalam proses klasifikasi sudah dalam bentuk yang terstruktur dan efisien. Seluruh efek ini menjadikan *feature extraction* sebagai tahap krusial dalam sistem pengenalan suara dan pemrosesan sinyal ucapan. Untuk memberikan gambaran lebih jelas, berikut disajikan perbandingan antara bentuk awal data audio (*waveform*) dengan hasil representasinya setelah melalui proses ekstraksi fitur menggunakan MFCC.

Untuk memberikan gambaran yang lebih jelas, berikut disajikan perbandingan antara data audio mentah (*waveform*) dengan hasil representasi fitur setelah melalui proses ekstraksi menggunakan MFCC. Tabel 2.1 menunjukkan perbedaan antara bentuk awal data audio dan hasil akhirnya setelah diproses dengan MFCC.

Tabel 2.1 Perubahan Representasi Data Suara Akibat Proses MFCC

Tahap	**Representasi Data**	**Visualisasi**
Awal (audio mentah)	Gelombang sinyal waktu (amplitudo vs waktu)	
Setelah MFCC	Spektrogram MFCC	

Gambar tahap pertama menunjukkan gelombang audio mentah yang direkam dari suara ayam betina sedang marah. Sumbu horizontal mewakili waktu

dalam satuan detik, sedangkan sumbu vertikal menunjukkan amplitudo atau kekuatan sinyal suara. Puncak-puncak naik turun pada grafik tersebut menggambarkan intensitas suara ayam pada waktu tertentu. Area dengan amplitudo tinggi menunjukkan bahwa ayam sedang bersuara, sedangkan area datar menunjukkan jeda atau keheningan.

Sementara itu, tahap kedua menampilkan hasil transformasi MFCC dalam bentuk spektrogram. Sumbu horizontal menunjukkan waktu, sedangkan sumbu vertikal menunjukkan urutan koefisien MFCC yang dihasilkan (misalnya, 13 koefisien per *frame*). Warna yang ditampilkan menunjukkan nilai dari setiap koefisien, dengan gradasi dari warna biru (nilai rendah), putih (nilai netral), hingga merah (nilai tinggi).

2.5. Spektrogram

Spektrogram adalah representasi visual dari spektrum frekuensi suatu sinyal audio yang berubah terhadap waktu. (Tanci dan Hekim, 2023). Spektrogram menunjukkan bagaimana energi dalam berbagai frekuensi berubah seiring waktu dengan memanfaatkan analisis frekuensi berbasis transformasi Fourier. Dengan kata lain, spektrogram memungkinkan kita untuk memahami distribusi energi suara dalam domain waktu-frekuensi (Suprapto dan Yandra, 2021).

Spektrogram digunakan secara luas dalam berbagai bidang. Dalam pengenalan suara dan ucapan, spektrogram dimanfaatkan dalam sistem *speech recognition* untuk menganalisis pola suara manusia. Sebagai contoh, penelitian oleh Permana dkk. (2018) membahas implementasi metode MFCC dan DTW untuk pengenalan jenis suara pria dan wanita. Di bidang bioakustik, spektrogram digunakan dalam penelitian vokalisasi hewan, termasuk untuk mengidentifikasi spesies berdasarkan karakteristik suara mereka. Penelitian oleh Lubis dkk. (2015) meneliti suara stridulasi ikan gupi (*Poecilia reticulata*) menggunakan analisis spektrogram untuk memahami spektrum suara yang dihasilkan dari gerakan sirip ikan tersebut. Dalam diagnostik medis, spektrogram membantu dalam analisis sinyal suara guna mendeteksi gangguan bicara atau penyakit neurologis. Misalnya,

penelitian oleh Helmiyah dkk. (2020) mengidentifikasi emosi manusia berdasarkan ucapan menggunakan metode ekstraksi ciri LPC dan metode *euclidean distance.* Selain itu, dalam musik dan akustik, spektrogram berperan dalam analisis nada, harmoni, serta karakteristik instrumen musik. Pada penelitian Pamungkas dan Djohan (2022) dilakukan eksperimen *beat frequency* sebagai liminal ritme dan timbre dalam karya musik spektral.

Secara umum, spektrogram dibuat dengan menerapkan *Short-Time Fourier Transform* (STFT) pada sinyal audio. Proses ini membagi sinyal menjadi jendela kecil (*window*) dan menghitung spektrum frekuensinya untuk setiap segmen. Hasilnya ditampilkan dalam bentuk grafik dua dimensi, di mana sumbu horizontal mewakili waktu, sumbu vertikal menunjukkan frekuensi, dan intensitas warna atau tingkat kecerahan menggambarkan amplitudo atau energi sinyal pada frekuensi tertentu. (Salsabillah dkk., 2023).

Rumus matematis dari STFT adalah sebagai berikut:

$$X(\tau, \omega) = \sum_{n=-\infty}^{\infty} x(n) \cdot w(n-\tau) \cdot e^{-j\omega n} \qquad (2\text{-}11)$$

Dalam rumus tersebut, $X(\tau, \omega)$ merupakan hasil transformasi STFT pada waktu τ dan frekuensi ω. Variabel $x(n)$ adalah sinyal *input* dalam domain waktu, sementara $w(n-\tau)$ adalah fungsi jendela (*window function*) yang bergeser sepanjang waktu τ untuk membatasi bagian sinyal yang akan dianalisis pada setiap interval waktu tertentu. Fungsi basis Fourier ditunjukkan oleh $e^{-j\omega n}$, yang bertugas memproyeksikan sinyal ke dalam domain frekuensi. Dengan membagi sinyal menjadi potongan kecil dan menerapkan transformasi Fourier secara lokal pada tiap segmen, STFT memungkinkan analisis spektrum frekuensi secara dinamis terhadap waktu.

Salah satu keunggulan utama dari spektrogram adalah kemampuannya dalam merepresentasikan informasi sinyal dalam domain waktu dan frekuensi secara bersamaan. Hal ini sangat bermanfaat dalam analisis sinyal non-stasioner seperti sinyal suara, di mana karakteristik frekuensinya dapat berubah seiring waktu. Dengan menggunakan spektrogram, perubahan energi dan pola frekuensi

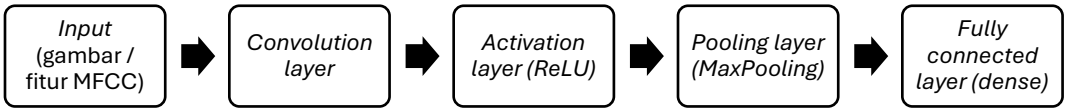
dari sinyal dapat diamati secara visual, yang mempermudah proses interpretasi dan ekstraksi fitur. Selain itu, spektrogram juga mendukung berbagai jenis jendela dan parameter, sehingga dapat disesuaikan dengan kebutuhan analisis tertentu. Keunggulan-keunggulan ini menjadikan spektrogram sebagai alat penting dalam bidang pemrosesan sinyal digital, terutama dalam aplikasi seperti pengenalan suara, musik, dan analisis ucapan.

2.6. *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah jenis jaringan saraf tiruan yang dirancang untuk memproses data berbentuk *grid*, seperti gambar dan sinyal suara. Dalam analisis audio, CNN sering digunakan untuk mengenali pola dalam spektrogram, yaitu representasi visual dari sinyal audio yang menunjukkan intensitas frekuensi terhadap waktu. Pendekatan ini memungkinkan CNN untuk mengekstraksi fitur penting dari data audio, yang kemudian digunakan dalam berbagai aplikasi seperti pengenalan ucapan, klasifikasi suara lingkungan, dan identifikasi instrumen musik. (Sinha *et al.,* 2020).

Penerapan CNN dalam analisis audio telah menunjukkan hasil yang signifikan dalam berbagai studi. Misalnya, penelitian oleh Palanisamy *et al.* (2020) menunjukkan bahwa model CNN yang dilatih sebelumnya pada *dataset ImageNet* dapat digunakan secara efektif untuk klasifikasi audio, meskipun terdapat perbedaan signifikan antara sampel gambar dan spektrogram audio. Selain itu, studi oleh Ansari dan Hasan (2022) memperkenalkan *SpectNet*, sebuah model yang mengintegrasikan lapisan *front-end* untuk mengekstraksi fitur spektrogram dalam arsitektur CNN, yang menunjukkan peningkatan kinerja dalam tugas klasifikasi sinyal audio.

CNN terdiri dari beberapa lapisan utama, yaitu *convolutional layer*, *activation layer* (biasanya menggunakan fungsi aktivasi *ReLU*), *pooling layer*, dan *fully connected layer*. Struktur ini memungkinkan CNN untuk mengekstraksi fitur dari tingkat rendah hingga tingkat tinggi secara hierarkis (Yamashita *et al.,* 2018).



Gambar 2.8 Diagram alur proses CNN

2.6.1. *Convolutional Layer*

Convolutional layer adalah inti dari CNN yang berfungsi untuk mengekstraksi fitur dari data *input*, seperti citra atau spektrogram. Lapisan ini bekerja dengan menggeser filter (juga disebut *kernel*) berukuran kecil di seluruh *input* dan melakukan operasi konvolusi antara filter dan bagian lokal dari *input*. Hasil dari operasi ini menghasilkan *feature map* yang merepresentasikan fitur penting dari data. (Yamashita *et al.,* 2018).

Secara matematis, operasi konvolusi dapat dituliskan sebagai berikut:

$$S(i,j) = (X * K)(i,j) = \sum_{m}\sum_{n} X(i+m, j+n) \cdot K(m,n) \qquad (2\text{-}12)$$

Dalam konteks ini, $X(i,j)$ menyatakan nilai piksel dari *input* (misalnya citra atau spektrogram) pada posisi (i,j). Kemudian, $K(m,n)$ adalah *kernel* atau filter konvolusi berukuran $m \times n$, yang digunakan untuk mengekstraksi fitur lokal dari *input*. Operasi konvolusi dilakukan dengan menggeser *kernel* melintasi seluruh area *input* dan menghitung hasil perkalian antara elemen *kernel* dan elemen *input* yang bersesuaian, lalu menjumlahkan hasilnya. Nilai yang diperoleh dari proses ini adalah $S(i,j)$, yaitu hasil konvolusi pada posisi (i,j). Hasil ini disusun menjadi peta fitur (*feature map*) yang menjadi keluaran dari lapisan konvolusi, dan akan digunakan dalam proses pembelajaran lebih lanjut untuk mengenali pola atau fitur penting dalam data.

2.6.2. *Activation Layer*

Setelah proses konvolusi, hasilnya biasanya akan dilewatkan melalui fungsi aktivasi, dan yang paling umum digunakan adalah *Rectified Linear Unit* (*ReLU*).

Fungsi aktivasi ini bersifat non-linear dan berperan penting dalam membantu jaringan mengenali hubungan kompleks dalam data (Goh *et al,* 2024).

Fungsi aktivasi *ReLU* didefinisikan sebagai:

$$f(x) = \max(0, x) \quad (2\text{-}13)$$

Fungsi ini akan mengubah nilai negatif menjadi nol dan mempertahankan nilai positif, sehingga mempercepat proses pelatihan dan mengurangi masalah *vanishing gradient*.

2.6.3. *Pooling Layer*

Pooling layer digunakan untuk mengurangi dimensi spasial dari *feature map*, yang berfungsi untuk mengurangi jumlah parameter, meningkatkan efisiensi komputasi, dan mencegah *overfitting*. Terdapat beberapa jenis *pooling*, namun yang paling umum adalah *max pooling* dan *average pooling* (Skourt *et al.,* 2022).

Contoh *max pooling* dengan ukuran jendela 2×2 akan mengambil nilai maksimum dari setiap jendela dan menghasilkannya sebagai *output*:

$$y(i,j) = \max\{x(2i, 2j), x(2i, 2j+1), x(2i+1,2j), x(2i+1,2j+1)\} \quad (2\text{-}14)$$

Pooling tidak memiliki parameter yang dipelajari, namun sangat penting dalam membangun representasi hierarkis dari fitur-fitur.

2.6.4. *Fully Connected Layer*

Fully connected layer (lapisan terhubung penuh) merupakan komponen penting dalam arsitektur CNN yang biasanya ditempatkan setelah beberapa lapisan konvolusi dan *pooling*. Lapisan ini berperan dalam pengambilan keputusan akhir dengan memproses fitur-fitur hasil ekstraksi dari lapisan sebelumnya menjadi representasi vektor yang dapat digunakan untuk klasifikasi atau regresi (Lahouaoui *et al.,* 2022).

Berbeda dengan lapisan konvolusi yang hanya terhubung secara lokal, pada *fully connected layer*, setiap neuron dihubungkan dengan semua neuron di lapisan sebelumnya. Hal ini memungkinkan lapisan ini untuk menggabungkan informasi spasial dari seluruh *feature map* dan melakukan pemetaan dari fitur ke *output* target. Struktur seperti ini menyerupai jaringan saraf tiruan (*artificial neural network*) konvensional, yang terdiri dari bobot, bias, dan fungsi aktivasi (Tejeda dan Mayer, 2024).

Secara matematis, proses pada *fully connected layer* dapat dijelaskan dengan rumus:

$$y = f(Wx + b) \tag{2-15}$$

Pada persamaan tersebut, y merupakan *output* dari lapisan, sedangkan x adalah *input* berupa vektor fitur hasil dari proses *flattening* terhadap *feature map*. Selanjutnya, W adalah matriks bobot (*weight matrix*) yang menghubungkan setiap neuron *input* ke neuron *output*, dan b merupakan vektor bias yang ditambahkan ke setiap neuron untuk memberikan fleksibilitas dalam proses pembelajaran. Fungsi f merepresentasikan fungsi aktivasi yang diterapkan pada hasil perkalian bobot dan *input* yang telah dijumlahkan dengan bias. Fungsi aktivasi ini bisa berupa *ReLU*, *sigmoid*, atau *softmax*, tergantung pada kebutuhan arsitektur jaringan dan jenis tugas klasifikasi yang dihadapi.

Fungsi aktivasi yang digunakan dalam lapisan ini bergantung pada jenis tugas yang dihadapi. Untuk klasifikasi biner, biasanya digunakan fungsi aktivasi *sigmoid*, sedangkan untuk klasifikasi multi-kelas, digunakan fungsi *softmax*. Lapisan ini akan menyesuaikan bobot dan bias selama proses pelatihan menggunakan algoritma *backpropagation*, dengan tujuan meminimalkan nilai *loss function* melalui pembaruan parameter.

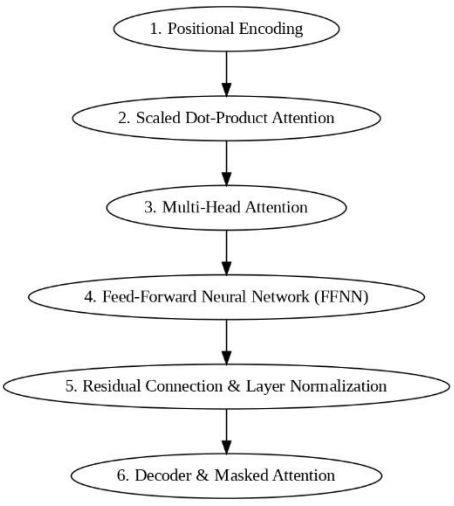
2.7. *Transformer*

Transformer adalah arsitektur model dalam pembelajaran mesin yang diperkenalkan oleh Vaswani *et al.* (2017). Model ini dirancang untuk memproses

data sekuensial dengan lebih efisien dibandingkan model konvensional seperti *Recurrent Neural Network* (RNN) dan *Long Short-Term Memory* (LSTM). Salah satu keunggulan utama *Transformer* adalah penggunaan mekanisme *self-attention*, yang memungkinkan model untuk menimbang hubungan antar elemen dalam suatu sekuens tanpa harus memprosesnya secara berurutan.

Dalam bidang pengolahan sinyal suara, *Transformer* telah digunakan dalam berbagai aplikasi, termasuk *speech recognition*, sintesis suara, dan klasifikasi suara. Penelitian oleh Gong *et al.* (2021) mengembangkan model AST (*Audio Spectrogram Transformer*), yang menunjukkan bahwa arsitektur berbasis *Transformer* dapat mencapai performa tinggi dalam tugas klasifikasi suara. Selain itu, penelitian oleh Gulati *et al.* (2020) memperkenalkan *conformer*, yaitu kombinasi antara CNN dan *Transformer* yang dioptimalkan untuk pemrosesan sinyal suara, khususnya dalam tugas pengenalan ucapan.

Arsitektur *Transformer* terdiri dari dua komponen utama, yaitu *encoder* dan *decoder*. Masing-masing komponen terdiri atas beberapa lapisan identik. Di dalam setiap lapisan terdapat beberapa komponen utama seperti *multi-head attention*, *feed-forward network*, *residual connection*, dan *layer normalization* (Dong *et al.,* 2018).



Gambar 2.9 Diagram blok *Transformer*

1. *Positional encoding*

Karena *Transformer* tidak memiliki struktur berurutan secara alami seperti RNN, maka dibutuhkan informasi tambahan mengenai posisi kata dalam sekuens. Hal ini dilakukan dengan *positional encoding*, yang menambahkan nilai posisi menggunakan fungsi sinus dan kosinus terhadap dimensi representasi kata:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000\,\frac{2i}{d_{\text{model}}}}\right), PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000\,\frac{2i}{d_{\text{model}}}}\right) \quad (2\text{-}16)$$

Di mana pos adalah posisi token, i adalah indeks dimensi, dan d_{model} adalah jumlah dimensi pada representasi vektor.

2. *Scaled dot-product attention*

   Mekanisme inti dalam *Transformer* adalah *self-attention*, yang memungkinkan setiap posisi dalam *input* untuk memperhatikan posisi lain dalam sekuens. Rumus dasar dari mekanisme ini adalah sebagai berikut:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2\text{-}17)$$

   Pada persamaan di atas, Q (*query*), K (*key*), dan V (*value*) merupakan hasil proyeksi linier dari *input*. Nilai d_k merupakan dimensi dari vektor *key*. Operasi *softmax* digunakan untuk menghitung bobot perhatian, kemudian dikalikan dengan *value* untuk menghasilkan *output* akhir.

3. *Multi-head attention*

   Untuk menangkap berbagai representasi informasi secara paralel, digunakan *multi-head attention*. Dalam mekanisme ini, proses *attention* dilakukan sebanyak beberapa kepala secara simultan, kemudian hasilnya digabungkan:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \dots, head_h)W^O \quad (2\text{-}18)$$

   Setiap $head_i$ dihitung dengan rumus yang sama seperti *attention* standar, dan W^O adalah matriks proyeksi *output* yang dilatih selama pelatihan.

4. *Feed-Forward Neural Network* (FFNN)

Setiap posisi dalam keluaran dari *multi-head attention* diproses lebih lanjut menggunakan jaringan saraf *feed-forward* dua lapis:

$$\text{FFNN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2\text{-}19)$$

Proses ini bersifat independen pada tiap posisi dalam sekuens dan berfungsi sebagai pemrosesan lanjutan terhadap hasil *attention*.

5. *Residual connection* dan *layer normalization*

   Untuk menjaga stabilitas pelatihan, setiap sub-lapisan dalam *Transformer* menggunakan *residual connection*, yaitu penjumlahan antara *input* awal dengan *output* dari sub-lapisan tersebut. Hasilnya kemudian dinormalisasi menggunakan *layer normalization*:

   $$\text{LayerNorm}\big(x + \text{Sublayer}(x)\big) \quad (2\text{-}20)$$

   Penggunaan teknik ini mempercepat konvergensi dan mencegah degradasi informasi selama propagasi lapisan.

6. *Decoder* dan *masked attention*

   Bagian *decoder* memiliki struktur serupa dengan *encoder*, namun dengan tambahan *masked self-attention*. Tujuannya adalah untuk mencegah informasi dari posisi kata berikutnya digunakan saat melakukan prediksi selama pelatihan, sehingga model hanya mengandalkan konteks sebelumnya.

Dengan kemampuannya yang fleksibel dan efisien dalam menangani data sekuensial, *Transformer* semakin banyak diadopsi dalam berbagai penelitian dan aplikasi terkait pemrosesan sinyal suara, menggantikan model tradisional berbasis RNN dan LSTM yang memiliki keterbatasan dalam menangani dependensi jangka panjang.

2.8. *Deepspeech*

Deepspeech adalah model pengenalan suara berbasis *deep learning* yang dikembangkan oleh Mozilla dengan arsitektur *end-to-end*. Model ini terinspirasi

dari penelitian yang dilakukan oleh Baidu Research dalam proyek *Deep Speech* (Hannun *et al.,* 2014). Berbeda dengan pendekatan tradisional yang menggunakan beberapa tahap pemrosesan sinyal secara terpisah, *Deepspeech* langsung mengonversi sinyal suara mentah menjadi teks dengan memanfaatkan jaringan saraf dalam (*Deep Neural Network*).

Arsitektur *Deepspeech* menggunakan *Recurrent Neural Network* (RNN), khususnya *Bidirectional Long Short-Term Memory* (BiLSTM), yang mampu menangkap konteks dari kedua arah dalam data audio. Model ini juga menerapkan algoritma *Connectionist Temporal Classification* (CTC) sebagai fungsi kehilangan (*loss function*), yang memungkinkan pelatihan model tanpa memerlukan pemetaan eksplisit antara *input* dan *output*. Selain itu, *Deepspeech* menggunakan bahasa pemodelan berbasis *n-gram* untuk meningkatkan akurasi dalam transkripsi kata dan kalimat. (Amodei *et al.,* 2016).

Gambar 2.10 Diagram alur proses *Deepspeech*

Secara umum, arsitektur *Deepspeech* terdiri dari beberapa lapisan utama, yaitu:

1. Lapisan *input*

   Lapisan ini menerima representasi fitur dari sinyal audio yang telah diproses sebelumnya, biasanya berupa spektrogram atau MFCC. Setiap kerangka waktu pada sinyal suara direpresentasikan sebagai vektor fitur.
2. Beberapa lapisan *dense* (*fully connected layer*)

   Lapisan-lapisan ini melakukan transformasi linier pada *input*. Biasanya terdiri dari 3 hingga 5 lapisan dengan fungsi aktivasi seperti *ReLU* yang diterapkan setelah transformasi linier.
3. Lapisan RNN

   RNN digunakan untuk menangkap ketergantungan temporal dalam data sekuensial, yaitu bagaimana urutan fonem atau kata muncul dalam sinyal suara. Versi *Deepspeech 2* mengganti RNN tradisional dengan arsitektur *Bidirectional Gated Recurrent Units* (Bi-GRU) atau *Bidirectional LSTM*

untuk meningkatkan pemahaman konteks dari dua arah—masa lalu dan masa depan.

4. Lapisan *output softmax*

   Lapisan ini menghasilkan distribusi probabilitas atas semua kemungkinan karakter (termasuk karakter kosong atau *blank*) untuk setiap kerangka waktu *input*.

Untuk mengatasi permasalahan pelabelan sekuensial yang tidak teratur, *Deepspeech* menggunakan fungsi *loss* bernama *Connectionist Temporal Classification* (CTC). Fungsi ini memungkinkan pelatihan model tanpa perlu pelabelan waktu yang tepat untuk setiap fonem atau kata. CTC akan mengevaluasi semua kemungkinan penyelarasan (*alignment*) antara *input* dan target *output* untuk menghitung *loss*.

Secara matematis, CTC *loss* dapat dituliskan sebagai:

$$\mathcal{L}_{CTC} = -\log \sum_{\pi \in B^{-1}(y)} P(\pi|x) \tag{2-21}$$

Dalam persamaan tersebut, x adalah *input* (fitur audio), y adalah label target (transkrip), π adalah urutan label termasuk *blank* yang dapat dikonversi ke y, dan $B^{-1}(y)$ adalah himpunan semua kemungkinan urutan π yang akan diredusir menjadi y oleh fungsi *collapse* B. Fungsi $P(\pi|x)$ dihitung sebagai hasil keluaran dari jaringan setelah fungsi aktivasi *softmax*.

Deepspeech telah diterapkan dalam berbagai aplikasi, termasuk sistem asisten suara, transkripsi otomatis, serta pengenalan suara pada perangkat lunak berbasis *speech-to-text*. Dengan perkembangan teknologi, versi terbaru dari *Deepspeech* semakin dioptimalkan untuk mendukung berbagai bahasa dan meningkatkan efisiensi pemrosesan suara secara *real-time*. (Hannun *et al.,* 2014).

2.9. Python

Python adalah bahasa pemrograman tingkat tinggi yang bersifat dinamis, interpretatif, dan multi-paradigma. Ciri khas utama dari Python adalah sintaksnya

yang bersih dan mudah dipahami, menjadikannya sangat cocok bagi pemula yang baru mempelajari pemrograman maupun profesional yang mengembangkan sistem berskala besar. Bahasa ini pertama kali dikembangkan oleh Guido van Rossum dan dirilis secara publik pada tahun 1991. Sejak awal, Python dirancang dengan filosofi untuk memberikan sintaks yang ekspresif dan struktur kode yang lebih ringkas dan mudah dibaca dibandingkan dengan banyak bahasa pemrograman lainnya.

Salah satu keunggulan Python terletak pada fleksibilitas dan skalabilitasnya dalam berbagai bidang pengembangan perangkat lunak. Python tidak hanya digunakan dalam pengembangan perangkat lunak umum, tetapi juga telah menjadi tulang punggung dalam berbagai domain modern seperti kecerdasan buatan (*Artificial Intelligence* / AI), pembelajaran mesin (*Machine Learning* / ML), pemrosesan bahasa alami (*Natural Language Processing* / NLP), analisis data (*data analytics*), komputasi ilmiah (*scientific computing*), serta pemrosesan sinyal dan citra digital (Saptadi *et al.,* 2025).

Ekosistem pustaka (*library*) Python yang sangat kaya menjadi faktor utama yang mendukung popularitas dan kapabilitasnya. Dalam bidang analisis data, pustaka seperti *NumPy*, *pandas*, dan *Matplotlib* memungkinkan manipulasi data, analisis statistik, serta visualisasi yang efisien dan efektif. Dalam *machine learning* dan *deep learning*, Python menyediakan pustaka tingkat lanjut seperti *scikit-learn*, *TensorFlow*, *Keras*, dan *PyTorch* yang digunakan secara luas oleh peneliti dan praktisi untuk membangun serta menguji model-model prediktif yang kompleks (McFee *et al.,* 2015).

Dalam konteks pemrosesan sinyal audio, pustaka seperti *librosa* memberikan dukungan yang komprehensif untuk ekstraksi fitur akustik seperti MFCC, spektrogram, *chroma features*, dan lain-lain. Hal ini sangat penting dalam pengembangan sistem pengenalan suara (*speech recognition*), deteksi emosi berbasis audio, dan klasifikasi suara. McFee *et al,* (2015) menyatakan bahwa *librosa* menyediakan antarmuka pemrosesan audio yang fleksibel serta dokumentasi yang baik, menjadikannya alat yang sangat berharga dalam penelitian berbasis audio digital.

Selain itu, Python juga digunakan secara luas dalam bidang pemrosesan citra (*image processing*), pengembangan aplikasi robotika, serta pengembangan antarmuka pengguna berbasis *web* dan *desktop*. Berbagai pustaka seperti *OpenCV* untuk pengolahan citra, *Flask* dan *Django* untuk pengembangan *web*, serta *Tkinter* atau *PyQt* untuk *desktop GUI*, menjadikan Python sebagai bahasa pemrograman serba guna yang sangat relevan dalam berbagai skenario pengembangan perangkat lunak modern.

Dukungan terhadap pengembangan prototipe yang cepat (*rapid prototyping*) juga merupakan alasan mengapa Python banyak digunakan dalam lingkungan penelitian dan pendidikan tinggi. Dengan sintaksis yang intuitif dan dokumentasi yang lengkap, pengembang dapat dengan mudah menulis, menguji, dan memodifikasi kode tanpa banyak hambatan. Selain itu, Python kompatibel dengan berbagai *Integrated Development Environment* (IDE) seperti *Jupyter Notebook*, *Spyder*, *VS Code*, dan *PyCharm*, yang semakin mempermudah proses eksplorasi data dan eksperimen model dalam pengembangan sistem kecerdasan buatan.

Secara keseluruhan, Python terus menjadi salah satu bahasa pemrograman yang paling banyak digunakan di dunia. Menurut survei tahunan dari *Stack Overflow* dan berbagai lembaga teknologi lainnya, Python secara konsisten menempati peringkat teratas sebagai bahasa pemrograman yang paling dicintai dan digunakan oleh para pengembang. (Raschka *et al.,* 2020).

2.10. *Flask* API

Flask adalah sebuah *micro web framework* berbasis Python yang digunakan untuk membangun aplikasi *web* dan *RESTful* API dengan struktur yang sederhana dan fleksibel. *Flask* dikembangkan oleh Armin Ronacher dan pertama kali dirilis pada tahun 2010 sebagai bagian dari proyek Pocoo. *Framework* ini bersifat ringan karena tidak menyertakan komponen eksternal seperti ORM (*Object Relational Mapping*) atau validasi *form* secara *default*, sehingga memberikan keleluasaan bagi pengembang dalam menentukan kebutuhan proyeknya sendiri. (Grinberg, 2018)

Flask banyak digunakan dalam pengembangan *backend* aplikasi berbasis *machine learning* atau *deep learning* karena kemampuannya dalam menangani *request* HTTP, serta kemudahan dalam mengintegrasikan model prediktif berbasis Python yang telah dilatih sebelumnya.

2.11. Etika Penggunaan Data / Hak Cipta Sumber *Dataset*

Dataset pada penelitian ini sebagian besar diperoleh dari platform *Youtube*, serta sebagian kecil dari repositori publik di *GitHub*. Semua data yang digunakan berupa rekaman suara ayam yang tersedia secara terbuka (*publicly available*) untuk umum.

Penggunaan data dari *Youtube* dilakukan hanya untuk keperluan penelitian akademik (*fair use*), tanpa tujuan komersial. Setiap *file* audio yang diambil telah dicantumkan sumber dan nama *channel*/pengunggahnya secara jelas, sebagai bentuk penghargaan terhadap hak cipta.

Untuk data dari *GitHub*, hanya repositori dengan lisensi terbuka (*open source*) atau yang mengizinkan penggunaan untuk riset yang digunakan.

2.12. Penelitian Sebelumnya

Penelitian mengenai pengenalan vokalisasi ayam telah mengalami perkembangan signifikan dalam beberapa tahun terakhir, terutama dengan kemajuan teknologi dalam bidang *machine learning* dan *signal processing*. Berbagai studi telah dilakukan untuk memahami dan mengklasifikasikan suara ayam, baik untuk tujuan pemantauan kesehatan, perilaku, maupun kesejahteraan hewan. Berikut adalah beberapa penelitian terkini yang relevan dengan topik ini:

1. Mao *et al.,* (2022) model *deep learning* berbasis CNN untuk mengidentifikasi vokalisasi stres pada ayam secara otomatis. Dalam penelitian ini, data suara ayam dikonversi menjadi spektrogram, yang kemudian digunakan sebagai *input* untuk model CNN. Model yang

dinamakan *light-VGG11* ini berhasil mencapai akurasi sebesar 95,07% dalam mendeteksi suara stres, menunjukkan potensi besar dalam penerapan pemantauan kesejahteraan ayam secara *real-time* di lingkungan peternakan intensif.

2. Ginovart-Panisello *et al.* (2020) melakukan analisis akustik terhadap vokalisasi ayam pedaging di peternakan intensif untuk menilai dampaknya terhadap kesejahteraan hewan. Studi ini menyoroti bagaimana variasi dalam vokalisasi dapat mencerminkan kondisi lingkungan dan kesejahteraan ayam, memberikan wawasan penting untuk pengembangan sistem pemantauan berbasis suara.
3. Penelitian oleh Wijaya (2024) merupakan salah satu rujukan utama dalam pengembangan topik klasifikasi vokalisasi ayam berbasis *deep learning*. Dalam penelitian ini, penulis menggunakan pendekatan RNN dan varian arsitektur *Long Short-Term Memory* (LSTM) untuk mengklasifikasikan berbagai jenis vokalisasi ayam.
4. Penelitian oleh Manikandan dan Neethirajan (2024) memperkenalkan pendekatan inovatif dengan menerapkan model NLP dan *Transformer* untuk menganalisis vokalisasi ayam. Dengan menggunakan model seperti Wave2Vec 2.0 dan BERT yang disesuaikan untuk tugas-tugas terkait unggas, penelitian ini berhasil mengklasifikasikan berbagai jenis vokalisasi ayam, termasuk panggilan stres, sinyal makan, dan vokalisasi kawin, dengan akurasi mencapai 92%. Studi ini menunjukkan potensi besar dalam menerjemahkan vokalisasi ayam ke dalam informasi yang dapat dimengerti manusia.
5. Hassan *et al.* (2024) memperkenalkan pendekatan baru berbasis *deep learning* untuk mengklasifikasikan sinyal audio unggas, dengan mengintegrasikan lapisan khusus yang disebut *burn layer* untuk meningkatkan ketahanan model terhadap variasi sinyal *input*. Metodologi ini menggabungkan pemrosesan sinyal audio digital, CNN, dan *burn layer* inovatif, yang menyuntikkan *noise* acak terkontrol selama pelatihan untuk memperkuat ketahanan model terhadap variasi sinyal *input*. Model ini menunjukkan efisiensi dengan mengurangi parameter yang dapat dilatih

menjadi 191.235, dibandingkan dengan arsitektur tradisional yang memiliki lebih dari 1,7 juta parameter.

Penelitian-penelitian di atas menunjukkan bahwa penggunaan teknologi *deep learning* dalam analisis vokalisasi ayam memiliki potensi besar dalam meningkatkan kesejahteraan hewan dan efisiensi peternakan. Namun, diperlukan lebih banyak penelitian yang secara khusus fokus pada pengenalan vokalisasi ayam untuk mengembangkan sistem pemantauan otomatis yang lebih canggih di masa depan.

Bab III
Analisis dan Perancangan Sistem

Penelitian ini bertujuan untuk mengembangkan sistem klasifikasi suara ayam berbasis *deep learning* guna mengidentifikasi kondisi-kondisi tertentu pada ayam. Sistem ini diuji menggunakan 3 metode *deep learning* yang berbeda, yaitu *Convolutional Neural Network* (CNN), *Transformer*, dan *Deepspeech*. Kemudian salah satu dari metode akan diimplementasikan dalam aplikasi *mobile*.

3.1. Analisis Sistem

Proses analisis kebutuhan adalah tahap pertama yang sangat penting dalam perancangan sistem. Analisis ini bertujuan untuk memahami berbagai aspek yang dibutuhkan oleh pengguna dan peneliti untuk memastikan bahwa sistem yang dikembangkan dapat berjalan dengan baik dan memenuhi kebutuhan yang ada.

Analisis kebutuhan dibagi menjadi dua bagian utama: kebutuhan pengguna dan kebutuhan peneliti.

3.1.1. Kebutuhan Pengguna

Sistem yang akan dikembangkan dirancang agar mudah digunakan oleh penggunanya, dengan fokus pada kemudahan pengoperasian dan keakuratan dalam memberikan hasil. Berikut adalah beberapa kebutuhan pengguna yang perlu dipertimbangkan:

1. Kemudahan penggunaan: Aplikasi harus memiliki antarmuka yang sederhana dan mudah dimengerti oleh pengguna. Pengguna hanya perlu merekam suara dan aplikasi secara otomatis akan memberikan hasil klasifikasi suara tersebut. Sistem harus dapat mengidentifikasi suara secara *real-time* dan memberikan *feedback* langsung kepada pengguna.

2. Akurasi klasifikasi: Pengguna mengharapkan agar aplikasi dapat mengklasifikasikan suara dengan akurat, seperti membedakan antara berbagai jenis suara berdasarkan kondisi yang telah ditentukan. Hasil klasifikasi harus mudah dipahami oleh pengguna, misalnya dengan memberikan label atau deskripsi tentang kondisi suara yang terdeteksi.

3.1.2. Kebutuhan Peneliti

Selain kebutuhan pengguna, peneliti juga memiliki kebutuhan teknis yang harus dipenuhi untuk memastikan pengembangan sistem berjalan sesuai rencana. Berikut adalah beberapa kebutuhan yang perlu dipertimbangkan oleh peneliti:

1. Perangkat Keras
   a. *Smartphone*: Xiaomi Redmi Note 8 Pro
      - CPU: MTK Helio G90T Octa-core Max2.05GHz
      - RAM: 6 GB
      - ROM: 128 GB
      - Sistem Operasi: MIUI Global 12.5.7
      - GPU: ARM Mali G76 MC4
   b. Komputer
      - CPU: Intel(R) Core(TM) i7-10700F CPU @ 2.90GHz (16 CPUs), ~2.9GHz
      - RAM: 16 GB
      - HDD: 2TB
      - SSD: 256 GB
      - Sistem Operasi: Windows 11 Pro 64-bit (10.0, Build 26100)
      - GPU: NVIDIA GeForce RTX 2060
2. Perangkat Lunak
   a. Windows 11
   b. Google Colab
   c. *Android Studio*
   d. Python

e. Java
f. *TensorFlow* dan *Keras*
g. *Librosa*
h. *NumPy*
i. *Matplotlib*
j. *Scikit-learn*
k. *Scipy*
l. *File* .h5
m. *Flask*

3.2. Pengumpulan Data

Data yang digunakan dalam penelitian ini terdiri dari empat jenis vokalisasi ayam, yaitu: suara ayam betina marah, suara ayam betina memanggil jantan, suara ayam ketika ada ancaman, dan suara ayam setelah bertelur. Seluruh data dikumpulkan dari berbagai sumber di internet. Setiap *file* audio diseleksi dan dikategorikan berdasarkan jenis vokalisasi sesuai dengan perilaku ayam yang diamati dalam rekaman. Bentuk *file* audio yang digunakan dalam penelitian ini berformat .wav.

Tabel 3.1 Data Vokalisasi Ayam yang Dikumpulkan

Jenis Vokalisasi	Sumber	Judul	Kredit	Durasi
Memanggil jantan	*Youtube*	bunyi pikat ayam betina	Yopi_semende_SDU	9 detik
Memanggil jantan	*Youtube*	suara ayam betina minta kawin	didikh84channel66	9 detik
Memanggil jantan	*Youtube*	suara pikat betina ayam hutan memanggil jantan 100% ampuh	brugoQtubabaofficial	6 menit 2 detik
Ketika ada ancaman	*Github*	Chickens alarm calling	zebular13	30 detik

Tabel 3.1 (Lanjutan)

Jenis Vokalisasi	Sumber	Judul	Kredit	Durasi
Ketika ada ancaman	*Youtube*	Chicken Alarm Call	Roamer17	15 detik
Ketika ada ancaman	*Youtube*	My hens are dust bathing when all of my roosters give their predator warning alert call	rosssrurallife504	4 detik
Ketika ada ancaman	*Youtube*	rooster crowing warning call	willielacalle4991	46 detik
Ketika ada ancaman	*Youtube*	Rooster on Alert	ChickenChick	46 detik
Ketika ada ancaman	*Youtube*	Rooster's distress call	JenCalvertND	19 detik
Ketika ada ancaman	*Youtube*	Suara Ayam saat ada bahaya mengancam(Riweh, rusuh dan menggemparkan)	pejuangquraan6788	1 menit 19 detik
Betina marah	*Youtube*	Angry chickens	minicharioteer	29 detik
Betina marah	*Youtube*	Angry hen	JacobSimpson	22 detik
Betina marah	*Youtube*	Angry Hen	Silverwolf387	1 menit 2 detik
Betina marah	*Youtube*	Broody hen growling/ hissing	Veer67	13 detik
Betina marah	*Youtube*	Dale annoying Squirrely the broody hen	thecasadet	46 detik
Betina marah	*Youtube*	Mad Chicken on her egg	HeirloomSeedhouse	57 detik

Tabel 3.1 (Lanjutan)

Jenis Vokalisasi	Sumber	Judul	Kredit	Durasi
Betina marah	*Github*	Privacy_Please	zebular13	1 detik
Betina marah	*Github*	Privacy_Please2	zebular13	2 detik
Betina marah	*Youtube*	What a broody hen looks and sounds like when you get near her.	susanarmstrong3438	2 menit 4 detik
Betina marah	*Youtube*	What does a broody hen sound like?	home_chicken_keeping	7 detik
Setelah bertelur	*Youtube*	A Chicken Singing the Egg	emj3334	57 detik
Setelah bertelur	*Youtube*	Chicken Egg Song - Including rooster imitating.	FMC351	24 detik
Setelah bertelur	*Youtube*	Chicken Singing The Egg Song	Ttater Tott	42 detik
Setelah bertelur	*Youtube*	Chicken sings after laying egg!	ChixnSkratch	14 detik
Setelah bertelur	*Youtube*	Chicken's Egg Song	lauranickerson09	2 menit 20 detik
Setelah bertelur	*Youtube*	I Laid an Egg! Chicken Hen Egg Song	EveryPeachInReach	39 detik
Setelah bertelur	*Youtube*	Just laid an egg! "The Egg Song" 🎶🥚	NativeMomwithaVlog	8 detik
Setelah bertelur	*Youtube*	Mystic Maran hen singing her egg song! Hens each have their own song.	michiganchicken	15 detik
Setelah bertelur	*Youtube*	Orpington chicken singing her Egg Song	CyclingChicken	31 detik
Setelah bertelur	*Youtube*	Suara ayam betina selesai bertelur	Caraternak	1 menit 51 detik

Berdasarkan identifikasi sumber suara dalam *dataset*, diketahui bahwa sebagian besar data berasal dari ayam betina, dengan beberapa berasal dari ayam

jantan, dan sebagian kecil merupakan campuran. Dari total 30 sumber suara yang dianalisis:

1. Ayam betina: 22 sumber (73.3%)
2. Ayam jantan: 5 sumber (16.7%)
3. Campuran (tidak dapat ditentukan spesifik jantan/betina): 3 sumber (10%)

Hal ini menunjukkan bahwa *dataset* memiliki dominasi vokalisasi dari ayam betina, sesuai dengan fokus utama penelitian pada suara ayam betina dalam beberapa kondisi (seperti marah, memanggil jantan, dan setelah bertelur).

Setelah dikumpulkan, data dikategorikan ke dalam *folder* masing-masing dan diberi label untuk memudahkan proses pelatihan model.

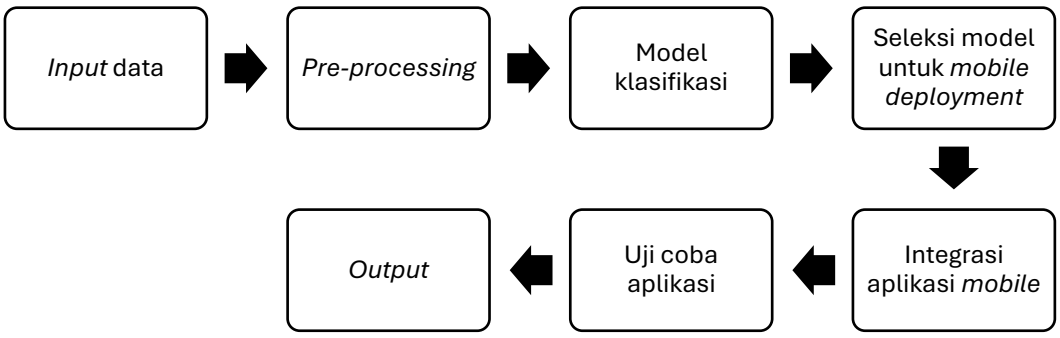
3.3. Desain dan Perancangan Sistem

Tahapan ini dilakukan untuk menggambarkan alur kerja sistem secara menyeluruh, mulai dari proses *input* data suara hingga *output* berupa hasil klasifikasi. Sistem harus dapat mengambil data suara ayam, memproses suara tersebut, kemudian melakukan klasifikasi untuk menentukan jenis suara tersebut, dan memberikan *output* yang berguna bagi pengguna.

Secara keseluruhan, sistem yang dibangun terdiri dari beberapa tahap utama, yaitu:

1. *Input* data
2. *Pre-processing*
3. Model klasifikasi
4. Seleksi model untuk *mobile deployment*
5. Integrasi aplikasi *mobile*
6. Uji coba aplikasi
7. *Output*

Desain alur sistem ditunjukkan melalui Gambar 3.1 berikut ini.



Gambar 3.1 Diagram alur sistem

3.3.1. *Input* Data

Dataset yang digunakan dalam penelitian ini merupakan hasil pengolahan dari rekaman suara ayam yang sebelumnya telah dikumpulkan dan diproses oleh peneliti sebelumnya (Wijaya, 2024). Berdasarkan identifikasi, terdapat 30 sumber suara yang menjadi asal data vokalisasi ayam, terdiri dari 22 sumber ayam betina, 5 sumber ayam jantan, dan 3 sumber campuran. Jumlah akhir data tidak hanya bergantung pada jumlah sumber, melainkan pada jumlah segmen audio hasil pemrosesan. Data mentah dari masing-masing jenis vokalisasi digabungkan untuk di-*import* ke satu *file Audacity*. Setelah itu, audio di-*export* dengan bentuk mono, *sampe rate* 16.000 *Hz*, dan *encoding* 16 bit. Selanjutnya, dilakukan penambahan *noise* dan *pitch shifting* untuk meningkatkan variasi data. Dari proses augmentasi menghasilkan 2 *file* untuk setiap jenis vokalisasi yang durasinya 6 sampai 8 menit. Dari observasi data audio, data-data ini kemudian dipotong menjadi segmen-segmen kecil dengan durasi 4 detik dan *overlap* 1 detik. Hasil total dari proses tersebut adalah 1.403 *file* audio yang terbagi dalam empat kategori vokalisasi ayam. *Dataset* ini kemudian digunakan dalam pelatihan dan validasi model klasifikasi dalam penelitian ini.

Dengan menggunakan Google Colab, *dataset* diunggah ke Google Drive dan disimpan dalam format direktori '/content/drive/MyDrive/skripsi/dataset/' dengan struktur *folder* sebagai berikut:

Gambar 3.2 Struktur *folder dataset*

Berikut distribusi *dataset* untuk masing-masing kategori vokalisasi:

1. Ayam betina marah: 352 *file*
2. Ayam betina memanggil jantan: 375 *file*
3. Ketika ada ancaman: 331 *file*
4. Setelah bertelur: 345 *file*

Masing-masing kategori dibagi ke dalam data pelatihan dan validasi secara proporsional. Sebanyak 80% dari total data digunakan untuk pelatihan dan 20% untuk validasi. Distribusi data yang relatif seimbang antar kategori memungkinkan model untuk belajar dengan baik tanpa bias yang signifikan terhadap kelas tertentu.

Selain itu, sebanyak 60 *file* audio terpisah (15 per kategori) digunakan sebagai data uji (*testing*) untuk mengevaluasi performa akhir model (dapat dilihat pada Gambar 3.2 terdapat *folder testing*).

3.3.2. *Pre-processing*

Tahap ini dilakukan dengan mengekstrak fitur suara menggunakan MFCC. MFCC dipilih karena kemampuannya dalam merepresentasikan karakteristik spektral audio yang penting untuk pengenalan suara. Dalam proses ekstraksi, parameter *sample rate* tidak diubah (`sr=None`) sehingga mengikuti nilai asli dari *dataset*, yaitu 16.000 *Hz*. Hal ini dilakukan untuk menjaga keaslian sinyal suara dan menghindari potensi distorsi akibat *resampling*.

Proses ekstraksi MFCC dilakukan pada setiap *file* audio dengan konfigurasi parameter yang dapat disesuaikan dengan kebutuhan. Parameter-parameter MFCC yang dimodifikasi meliputi:

1. `n_mfcc`: Jumlah koefisien MFCC
2. `hop_length`: Hop *length*
3. `n_fft`: Fast *Fourier Transform* (FFT)
4. `max_length`: Panjang *frame*

Tabel 3.2 menampilkan konfigurasi MFCC yang digunakan untuk masing-masing metode.

Tabel 3.2 Konfigurasi MFCC pada CNN

Parameter	Konfigurasi	Keterangan
`n_mfcc`	20	Jumlah koefisien MFCC yang diambil dari setiap *frame* audio. Semakin besar nilainya, semakin banyak informasi yang ditangkap
`hop_length`	256	Jumlah sampel per langkah saat membagi sinyal audio menjadi *frame*. Semakin kecil nilai ini, semakin banyak *frame* yang dihasilkan
`n_fft`	1024	Ukuran jendela untuk transformasi *Fourier*. Mempengaruhi resolusi frekuensi dari fitur MFCC
`max_length`	250	Jumlah *frame* tetap untuk setiap audio. Audio yang menghasilkan *frame* lebih dari 250 akan dipotong, sedangkan yang kurang dari 250 akan ditambahkan *padding*

Jumlah koefisien MFCC yang digunakan adalah 20, yang sudah umum digunakan dalam banyak aplikasi pengenalan suara karena cukup untuk menangkap karakteristik utama dari sinyal audio. Nilai `hop_length` sebesar 256 dan `n_fft` sebesar 1024 dipilih untuk memberikan keseimbangan antara resolusi waktu dan frekuensi. Dengan `hop_length` yang tidak terlalu besar, model dapat menangkap

perubahan suara yang cepat dalam vokalisasi ayam. Dalam penelitian ini, *sample rate* ditetapkan sebesar 16.000 *Hz*, sedangkan durasi dari *file* audio adalah 4 detik. Maka, total jumlah sampel dalam satu *file* audio maksimal adalah:

$$4 \times 16000 = 64000 \text{ sampel} \quad (3\text{-}1)$$

Dengan nilai `hop_length` sebesar 256, maka jumlah *frame* MFCC yang dihasilkan dari *file* berdurasi maksimal tersebut adalah:

$$\frac{64000}{256} = 256\ frame \quad (3\text{-}2)$$

Oleh karena itu, parameter `max_length` = 250 digunakan untuk memotong atau menyamakan panjang *output* MFCC dari semua *dataset* menjadi ukuran yang seragam, yaitu 250 *frame*. Ini dilakukan agar semua *input* ke model CNN memiliki bentuk yang konsisten.

MFCC hasil ekstraksi kemudian dibentuk menjadi *array* berdimensi (20, 250, 1) sebagai *input* ke model CNN, dengan penambahan *channel* sebagai dimensi ke-3.

Tabel 3.3 Konfigurasi MFCC pada *Transformer*

Parameter	Konfigurasi	Keterangan
`n_mfcc`	20	Jumlah koefisien MFCC yang diambil dari setiap *frame* audio. Semakin besar nilainya, semakin banyak informasi yang ditangkap
`hop_length`	512	Jumlah sampel per langkah saat membagi sinyal audio menjadi *frame*. Semakin kecil nilai ini, semakin banyak *frame* yang dihasilkan
`n_fft`	1024	Ukuran jendela untuk transformasi *Fourier*. Mempengaruhi resolusi frekuensi dari fitur MFCC
`max_length`	250	Jumlah *frame* tetap untuk setiap audio. Audio yang menghasilkan *frame* lebih dari 250 akan dipotong, sedangkan yang kurang dari 250 akan ditambahkan *padding*

Pada dasarnya, konfigurasi parameter MFCC yang digunakan dalam penelitian ini disamakan untuk ketiga metode. Parameter seperti `n_mfcc` sebanyak 20, dan `n_fft` sebesar 1024 dipilih karena telah terbukti cukup representatif dalam mengekstraksi informasi dari audio. Selain itu, penggunaan nilai `max_length` yang disamakan (yaitu 250) bertujuan untuk menyeragamkan dimensi *input* antar metode agar proses pelatihan lebih konsisten.

Perbedaan hanya terdapat pada `hop_length`, yaitu sebesar 256 untuk CNN dan 512 untuk *Transformer*. Nilai `hop_length` = 256 pada CNN dipilih agar menghasilkan resolusi waktu yang lebih rapat, sesuai dengan karakteristik CNN yang sensitif terhadap pola spasial. Sementara itu, pada *Transformer* digunakan `hop_length` = 512 untuk mengurangi panjang urutan (*sequence length*) dari hasil MFCC, sehingga lebih efisien diproses oleh mekanisme *self-attention*. Penggunaan `hop_length` yang lebih besar ini menghasilkan jumlah *frame* yang lebih sedikit, membantu mengurangi beban komputasi tanpa menghilangkan informasi penting yang dibutuhkan oleh model *Transformer*.

Tabel 3.4 Konfigurasi MFCC pada *Deepspeech*

Parameter	Konfigurasi	Keterangan
`n_mfcc`	20	Jumlah koefisien MFCC yang diambil dari setiap *frame* audio. Semakin besar nilainya, semakin banyak informasi yang ditangkap
`hop_length`	256	Jumlah sampel per langkah saat membagi sinyal audio menjadi *frame*. Semakin kecil nilai ini, semakin banyak *frame* yang dihasilkan
`n_fft`	1024	Ukuran jendela untuk transformasi *Fourier*. Mempengaruhi resolusi frekuensi dari fitur MFCC
`max_length`	250	Jumlah *frame* tetap untuk setiap audio. Audio yang menghasilkan *frame* lebih dari 250 akan dipotong, sedangkan yang kurang dari 250 akan ditambahkan *padding*

Konfigurasi yang digunakan pada *Deepspeech* sama seperti konfigurasi yang digunakan pada CNN.

3.3.3. Model Klasifikasi

Pada tahap ini, dilakukan pembangunan dan pelatihan model klasifikasi menggunakan tiga metode *deep learning* yang berbeda, yaitu CNN, *Transformer*, dan *Deepspeech*. Masing-masing metode memiliki karakteristik arsitektur yang berbeda, sehingga format *input* MFCC yang digunakan juga disesuaikan dengan kebutuhan masing-masing model.

Sebelum dijelaskan konfigurasi spesifik dari setiap metode pada subbab berikutnya, Tabel 3.5 berikut menyajikan bentuk *input* yang digunakan oleh setiap model berdasarkan hasil ekstraksi fitur MFCC.

Tabel 3.5 *Input* Tiap Metode dari Hasil MFCC

Metode	Deskripsi	Bentuk *Input*
CNN	Spektrogram MFCC dalam bentuk matriks MFCC 3D	`(n_mfcc, max_length, 1)` → `(20, 250, 1)`
Transformer	Matriks urutan	`(max_length, n_mfcc)` → `(250, 20)`
Deepspeech	MFCC urutan	`(max_length, n_mfcc)` → `(250, 20)`

Untuk CNN, data MFCC diperlakukan seperti gambar yang memiliki *channel* tunggal (*grayscale*), sehingga dibentuk dalam format (20, 250, 1). Pada *Transformer* dan *Deepspeech*, MFCC diubah menjadi urutan vektor berdimensi (250, 20) untuk mencerminkan struktur urutan sinyal audio. Format ini sesuai untuk pemrosesan berbasis *time series* seperti pada *Transformer* dan RNN (*Deepspeech*).

3.3.3.1. CNN

Arsitektur CNN yang digunakan dalam penelitian ini adalah *custom* CNN. Arsitektur ini menggunakan pendekatan *sequential* model dari *TensorFlow Keras*, dimana *layer-layer* disusun secara berurutan dan data mengalir dari *input* hingga *output* melalui serangkaian operasi konvolusi, *pooling*, dan *fully connected layers*.

Sebelum model dibangun, dilakukan pembagian *dataset* untuk pelatihan dan validasi. Pembagian *dataset* hanya dilakukan untuk pelatihan dan validasi karena untuk pengujiannya menggunakan data tersendiri. *Dataset* dibagi secara *stratified* untuk memastikan distribusi kelas yang seimbang pada setiap *subset*. *Dataset* dibagi menjadi 80% untuk pelatihan dan 20% untuk validasi. Pembagian *dataset* dilakukan menggunakan `train_test_split` dari *scikit-learn*.

Model dilatih menggunakan konfigurasi dan parameter tertentu yang disusun berdasarkan percobaan (*trial and error*). Terdapat konfigurasi parameter tetap yang digunakan pada model, yang artinya konfigurasi ini digunakan secara konsisten pada setiap percobaan. Selain parameter MFCC (yang sudah dijabarkan pada Tabel 3.2), parameter-parameter tersebut meliputi:

Tabel 3.6 Konfigurasi Parameter Tetap Model CNN

Parameter	Konfigurasi	Keterangan
Epoch	30	Jumlah iterasi penuh terhadap seluruh *dataset* selama proses pelatihan
Ukuran *batch*	32	Jumlah sampel yang diproses dalam satu *batch* saat pelatihan. Seimbang antara akurasi dan efisiensi memori
Ukuran *kernel*	(3, 3)	Ukuran *kernel* 3×3 merupakan pilihan umum karena seimbang antara detail lokal dan efisiensi komputasi
Strides	1	Menggunakan pergeseran *filter default*
Padding	*Same*	Menjaga dimensi *output* tetap sama dengan *input* agar ukuran tidak cepat mengecil setelah konvolusi

Tabel 3.6 (Lanjutan)

Parameter	Konfigurasi	Keterangan
Fungsi aktivasi	*ReLU*	Aktivasi di blok *layer* konvolusi untuk menambahkan non-linearitas
Pooling	*MaxPooling2D* (2×2)	Mengurangi dimensi *feature map* dengan tetap mempertahankan fitur paling penting. Membantu efisiensi dan mencegah *overfitting*
Aktivasi *output*	*Softmax*	Mengubah *output* menjadi probabilitas antar kelas
Fungsi optimasi	*Adam*	*Optimizer* adaptif yang umum digunakan karena stabil dan cepat konvergen
Fungsi *loss*	*Categorical crossentropy*	Fungsi *loss* yang sesuai untuk klasifikasi multi-kelas dengan label *one-hot*

Tabel 3.6 merangkum konfigurasi parameter yang tidak diubah selama proses percobaan. Parameter-parameter tersebut dipilih berdasarkan praktik umum dalam pelatihan model klasifikasi audio dan telah disesuaikan melalui beberapa percobaan untuk memperoleh hasil yang optimal.

Selanjutnya, variasi untuk percobaan dilakukan pada beberapa konfigurasi parameter model, yaitu jumlah *filter* pada *convolutional layer* (*Conv2D*) di setiap blok *layer*, jumlah unit pada *layer dense*, nilai *dropout*, serta *learning rate*. Konfigurasi disusun secara eksploratif dengan mencoba kombinasi parameter secara acak.

Konfigurasi dari masing-masing percobaan yang dilakukan disajikan pada Tabel 3.7.

Tabel 3.7 Konfigurasi Model CNN pada Setiap Percobaan

No	***Filter* (Blok 1-3)**	***Dense***	***Dropout***	***Learning rate***	Keterangan
1	32-32-64	256	0.1	0.0005	*Baseline* awal

Tabel 3.7 (Lanjutan)

No	*Filter* (Blok 1-3)	*Dense*	*Dropout*	*Learning rate*	Keterangan
2	32-32-32	64	0.1	0.0005	Menurunkan *filter Conv2D* blok 3 dan *dense*
3	32-64-64	128	0.1	0.0005	Meningkatkan *filter Conv2D* blok 2 dan 3, serta *dense*
4	32-32-32	64	0.1	0.0001	Menurunkan *filter Conv2D* blok 2 dan 3, *dense*, dan *learning rate*
5	32-32-32	64	0.1	0.001	Meningkatkan *learning rate*
6	32-32-64	128	0.1	0.0001	Meningkatkan *filter Conv2D* blok 3, *dense* dan menurunkan *learning rate*
7	32-64-128	128	0.1	0.0001	Meningkatkan *filter Conv2D* blok 2 dan 3
8	32-64-128	128	0.1	0.0005	Meningkatkan *learning rate*
9	32-64-128	256	0.1	0.0005	Meningkatkan *dense*
10	32-64-128	256	0.2	0.0005	Meningkatkan *dropout*
11	32-64-128	256	0.1	0.001	Menurunkan *dropout* dan meningkatkan *learning rate*
12	16-32-64	128	0.1	0.001	Menurunkan *filter Conv2D* semua blok dan *dense*

Dapat dilihat bahwa percobaan dilakukan dengan mengubah beberapa parameter inti untuk mengevaluasi pengaruhnya terhadap performa model. Dalam penelitian ini, pengubahan konfigurasi parameter dilakukan secara eksploratif. Pada

setiap percobaan, hasil dievaluasi dari 4 metrik yaitu *accuracy*, *precision*, *recall*, dan *F1-score* untuk data pelatihan, validasi, dan pengujian, serta *confusion matrix* dari masing-masing *subset*. Kriteria penghentian didasarkan pada analisis terhadap seluruh metrik tersebut, dengan mempertimbangkan apakah model sudah menunjukkan performa yang baik secara konsisten antar data (tidak mengalami *overfitting*), serta ukuran *file* model yang masih tergolong wajar untuk kebutuhan *deployment*. Percobaan dihentikan ketika model menunjukkan keseimbangan antara akurasi tinggi dan generalisasi yang baik, serta efisiensi ukuran *file*.

3.3.3.2. *Transformer*

Arsitektur *Transformer* yang digunakan dalam penelitian ini merupakan *custom Transformer* yang dirancang khusus untuk klasifikasi data audio MFCC. Model ini memproses *input* sebagai urutan *time frame* (*sequence*) dan menggunakan mekanisme *self-attention* untuk menangkap hubungan temporal antar *frame*.

Model terdiri dari empat tahap utama: *patch embedding*, *positional encoding*, *encoder*, dan *classification head*. MFCC dipecah menjadi *patch*, diubah menjadi *embedding*, lalu ditambahkan *positional encoding*. Selanjutnya, *patch* ini diproses oleh beberapa *encoder layer* dengan *self-attention*. Hasil akhirnya diringkas dengan *global average pooling* dan diklasifikasikan oleh MLP (*Multilayer Perceptron*).

Dataset dibagi menjadi 80% untuk pelatihan dan 20% untuk validasi. Pembagian *dataset* dilakukan menggunakan `train_test_split` dari *scikit-learn*.

Model dilatih menggunakan konfigurasi dan parameter tertentu yang disusun berdasarkan percobaan (*trial and error*). Terdapat pula konfigurasi parameter konsisten yang digunakan pada model di setiap percobaan. Selain parameter MFCC (yang sudah dijabarkan pada Tabel 3.3), parameter-parameter tersebut meliputi:

Tabel 3.8 Konfigurasi Parameter Tetap Model *Transformer*

Parameter	Konfigurasi	Keterangan
Epoch	30	Jumlah iterasi penuh terhadap seluruh *dataset* selama proses pelatihan
Ukuran *batch*	32	Jumlah sampel yang diproses dalam satu *batch* saat pelatihan
Dropout	0.2	Untuk mengurangi *overfitting* dengan mematikan sebagian neuron saat pelatihan. Nilai 0.2 cukup umum dan seimbang
Epsilon layer normalization	1e-6	Nilai *epsilon* kecil untuk stabilitas *layer normalization*
Dense sebelum *output*	256	256 unit digunakan agar representasi akhir cukup informatif sebelum diklasifikasikan ke *output* kelas
Aktivasi MLP	*ReLU*	Untuk non-linearitas
Aktivasi *output*	*Softmax*	Mengubah *output* menjadi probabilitas antar kelas
Fungsi optimasi	*Adam*	*Optimizer* adaptif yang umum digunakan karena stabil dan cepat konvergen
Learning rate	0.0005	Kecepatan pembelajaran saat pelatihan model
Fungsi *loss*	*Categorical crossentropy*	Fungsi *loss* yang sesuai untuk klasifikasi multi-kelas dengan label *one-hot*

Tabel 3.8 merangkum konfigurasi parameter yang tidak diubah selama proses percobaan. Parameter-parameter tersebut disesuaikan melalui beberapa percobaan untuk memperoleh hasil yang optimal.

Selanjutnya, variasi untuk percobaan dilakukan pada beberapa konfigurasi parameter model, yaitu ukuran *patch*, ukuran dimensi *embedding*, jumlah *head* pada *self-attention*, jumlah *layer encoder*, dan ukuran dimensi dalam MLP. Konfigurasi disusun secara eksploratif dengan mencoba kombinasi parameter secara acak.

Konfigurasi dari masing-masing percobaan disajikan pada Tabel 3.9.

Tabel 3.9 Konfigurasi Model *Transformer* pada Setiap Percobaan

No	Ukuran *Patch*	Dimensi *Embedding*	*Head*	*Encode Layer*	Dimensi MLP	Keterangan
1	10	96	8	3	256	*Baseline* awal
2	5	96	8	3	256	Menurunkan ukuran *patch*
3	10	64	8	3	256	Meningkatkan ukuran *patch* dan menurunkan dimensi *embedding*
4	10	96	6	3	256	Meningkatkan dimensi *embedding* dan menurunkan *head*
5	10	96	4	3	256	Menurunkan *head*
6	10	96	6	2	256	Meningkatkan *head* dan menurunkan *encode layer*
7	10	96	6	3	2x dimensi *embedding*	Meningkatkan *encode layer* dan mengubah dimensi MLP menjadi 2x dari dimensi *embedding*
8	10	96	8	3	2x dimensi *embedding*	Meningkatkan *head*

Dapat dilihat bahwa percobaan dilakukan dengan mengubah beberapa parameter inti untuk mengevaluasi pengaruhnya terhadap performa model. Sama seperti CNN, pengubahan konfigurasi parameter dilakukan secara eksploratif. Pada setiap percobaan, hasil dievaluasi dari 4 metrik yaitu *accuracy*, *precision*, *recall*, dan *F1-score* untuk data pelatihan, validasi, dan pengujian, serta *confusion matrix* dari masing-masing *subset*. Kriteria penghentian didasarkan pada analisis terhadap seluruh metrik tersebut, dengan mempertimbangkan apakah model sudah menunjukkan performa yang baik secara konsisten antar data (tidak mengalami *overfitting*), serta ukuran *file* model yang masih tergolong wajar untuk kebutuhan *deployment*. Percobaan dihentikan ketika model menunjukkan keseimbangan antara akurasi tinggi dan generalisasi yang baik, serta efisiensi ukuran *file*.

3.3.3.3. *Deepspeech*

Arsitektur *Deepspeech* yang digunakan dalam penelitian ini merupakan *custom Deepspeech* yang dirancang khusus untuk klasifikasi data audio MFCC.

Arsitektur pertama diawali dengan *layer Conv1D* untuk mengekstraksi fitur lokal dari urutan MFCC. Setelah itu, fitur diproses oleh tiga *layer BiLSTM* untuk menangkap informasi temporal dari dua arah (maju dan mundur). Dua LSTM pertama mengembalikan seluruh urutan (`return_sequences=True`) untuk melanjutkan informasi ke *layer* berikutnya, sedangkan LSTM ketiga hanya mengembalikan *output* dari waktu terakhir untuk keperluan klasifikasi. *Output* dari LSTM kemudian diproses melalui *dense layer* sebelum diklasifikasikan ke dalam empat kelas menggunakan fungsi aktivasi *softmax*.

Dataset dibagi menjadi 80% untuk pelatihan dan 20% untuk validasi. Pembagian *dataset* dilakukan menggunakan `train_test_split` dari *scikit-learn*.

Model dilatih menggunakan konfigurasi dan parameter tertentu yang disusun berdasarkan percobaan (*trial and error*). Terdapat pula konfigurasi parameter konsisten yang digunakan pada model di setiap percobaan. Selain parameter MFCC (yang sudah dijabarkan pada Tabel 3.4), parameter-parameter tersebut meliputi:

Tabel 3.10 Konfigurasi Parameter Tetap Model *Deepspeech*

Parameter	Konfigurasi	Keterangan
Epoch	30	Jumlah iterasi penuh terhadap seluruh *dataset* selama proses pelatihan
Ukuran *batch*	32	Jumlah sampel yang diproses dalam satu *batch* saat pelatihan. Seimbang antara akurasi dan efisiensi memori
Ukuran *kernel*	5	Umum digunakan pada pemrosesan sinyal
Strides	2	Mengurangi panjang sekuens dengan tetap mempertahankan informasi penting, sekaligus mempercepat proses

Tabel 3.10 (Lanjutan)

Parameter	Konfigurasi	Keterangan
Padding	*Same*	Menjaga dimensi *output* tetap sama dengan *input* agar ukuran tidak cepat mengecil setelah konvolusi
Fungsi aktivasi	*ReLU*	Untuk non-linearitas
Dropout LSTM	0.1	10% neuron akan dinonaktifkan secara acak selama pelatihan
Aktivasi *output*	*Softmax*	Mengubah *output* menjadi probabilitas antar kelas
Fungsi optimasi	*Adam*	*Optimizer* adaptif yang umum digunakan karena stabil dan cepat konvergen
Fungsi *loss*	*Categorical crossentropy*	Fungsi *loss* yang sesuai untuk klasifikasi multi-kelas dengan label *one-hot*

Tabel 3.10 merangkum konfigurasi parameter yang tidak diubah selama proses percobaan. Parameter-parameter tersebut disesuaikan melalui beberapa percobaan untuk memperoleh hasil yang optimal.

Selanjutnya, variasi untuk percobaan dilakukan pada beberapa konfigurasi parameter model, yaitu *filter* pada *convolutional layer* (*Conv1D*), jumlah unit di setiap *layer BiLSTM*, jumlah unit pada *layer dense*, nilai *dropout* sebelum *output*, dan *learning rate*. Konfigurasi disusun secara eksploratif dengan mencoba kombinasi parameter secara acak.

Konfigurasi dari masing-masing percobaan disajikan pada Tabel 3.11.

Tabel 3.11 Konfigurasi Model *Deepspeech* pada Setiap Percobaan

No	***Filter***	**Unit (*BiLSTM* 1-3)**	***Dense***	***Dropout***	***Learning Rate***	**Keterangan**
1	128	128-128-128	64	0.1	0.0005	*Baseline* awal

Tabel 3.11 (Lanjutan)

No	*Filter*	Unit (*BiLSTM* 1-3)	*Dense*	*Dropout*	*Learning Rate*	Keterangan
2	64	64-64-128	32	0.1	0.0005	Menurunkan *filter Conv1D*, unit *BiLSTM* 1 dan 2, serta *dense*
3	128	128-128-128	64	0.2	0.0005	Meningkatkan *filter Conv1D*, unit *BiLSTM* 1 dan 2, *dense, dan dropout*
4	128	128-128-128	64	0.1	0.0001	Menurunkan *dropout* dan *learning rate*
5	128	128-128-128	64	0.1	0.001	Meningkatkan *learning rate*
6	128	128-128-128	64	0.2	0.0001	Meningkatkan *dropout* dan menurunkan *learning rate*

Dapat dilihat bahwa percobaan dilakukan dengan mengubah beberapa parameter inti untuk mengevaluasi pengaruhnya terhadap performa model. Sama seperti sebelumnya, pengubahan konfigurasi parameter dilakukan secara eksploratif. Pada setiap percobaan, hasil dievaluasi dari 4 metrik yaitu *accuracy*, *precision*, *recall*, dan *F1-score* untuk data pelatihan, validasi, dan pengujian, serta *confusion matrix* dari masing-masing *subset*. Kriteria penghentian didasarkan pada analisis terhadap seluruh metrik tersebut, dengan mempertimbangkan apakah model sudah menunjukkan performa yang baik secara konsisten antar data (tidak mengalami *overfitting*), serta ukuran *file* model yang masih tergolong wajar untuk kebutuhan *deployment*. Percobaan dihentikan ketika model menunjukkan keseimbangan antara akurasi tinggi dan generalisasi yang baik, serta efisiensi ukuran *file*.

3.3.3.4 *Output* Model

Setiap model menghasilkan distribusi probabilitas (*softmax*) untuk empat kelas vokalisasi ayam, yaitu: "Ayam Betina Marah", "Ayam Betina Memanggil Jantan", "Terancam", dan "Setelah Bertelur". *Output* ini berupa vektor probabilitas dengan jumlah total bernilai 1.0, di mana setiap elemen menunjukkan tingkat kepercayaan model terhadap masing-masing kelas. Dari distribusi probabilitas, kelas dengan probabilitas tertinggi (*confidence*) diambil sebagai hasil prediksi akhir.

3.3.4. Seleksi Model untuk *Mobile Deployment*

Proses pemilihan model untuk *deployment* pada aplikasi *mobile* sangat bergantung pada beberapa faktor penting, terutama dalam konteks penggunaan model *deep learning* untuk klasifikasi vokalisasi ayam. Untuk memastikan aplikasi berjalan optimal di perangkat *mobile* dengan keterbatasan sumber daya, model yang dipilih harus memenuhi dua kriteria utama: akurasi yang tinggi dan ukuran aplikasi yang kecil.

Dalam mengevaluasi performa model, digunakan beberapa indikator kinerja utama dari bidang klasifikasi, yaitu:

1. *Accuracy* (Akurasi)

   Menunjukkan persentase prediksi model yang benar dibandingkan dengan seluruh data uji.
2. *Precision*

   Mengukur seberapa banyak dari hasil klasifikasi yang positif itu benar. Dalam konteks ini, *precision* menunjukkan seberapa sering model tidak salah dalam mengklasifikasikan suatu jenis vokalisasi tertentu.
3. *Recall (Sensitivity)*

   Mengukur seberapa banyak dari semua data yang seharusnya diklasifikasikan sebagai suatu kelas tertentu berhasil dikenali dengan benar. *Recall* penting untuk memastikan model tidak melewatkan suara ayam yang penting.

4. *F1-Score*

   Merupakan rata-rata harmonis dari *precision* dan *recall*. *F1-score* sangat berguna ketika terjadi *trade-off* antara *precision* dan *recall*, karena memberi penilaian menyeluruh terhadap performa model, terutama pada data yang tidak seimbang antar kelas vokalisasi.

Dalam tahap seleksi model, pemilihan model terbaik untuk diterapkan pada aplikasi *mobile* dilakukan berdasarkan evaluasi kinerja menggunakan sejumlah metrik klasifikasi. Proses ini membutuhkan dukungan dari beberapa *library* dalam Python. *Scikit-learn* menjadi *library* utama untuk evaluasi, di mana fungsi `classification_report()` digunakan untuk menghasilkan laporan klasifikasi lengkap yang mencakup *precision*, *recall*, *F1-score*, dan *support* untuk setiap kategori vokalisasi. Fungsi `confusion_matrix()` dimanfaatkan untuk membuat matriks konfusi yang menunjukkan detail prediksi benar dan salah untuk setiap kelas, sedangkan `accuracy_score()` memberikan nilai akurasi keseluruhan model.

Penggunaan indikator-indikator ini bertujuan untuk mendapatkan gambaran yang lebih holistik dari kinerja model dibanding hanya mengandalkan akurasi saja.

3.3.5. Integrasi Aplikasi *Mobile*

Setelah memilih dan mengoptimalkan model *deep learning* terbaik berdasarkan akurasi dan ukuran aplikasi untuk klasifikasi vokalisasi ayam, langkah selanjutnya adalah mengintegrasikan model tersebut ke dalam aplikasi *mobile*. Proses integrasi ini bertujuan untuk memastikan bahwa model yang telah dipilih dapat memberikan hasil klasifikasi yang akurat dan memberikan pengalaman pengguna yang baik melalui antarmuka *mobile*.

Model klasifikasi yang telah dilatih disimpan dalam format .h5 beserta *file* normalisasi (.npz) yang berisi nilai rata-rata (*mean*) dan standar deviasi untuk *pre-processing*. Model tersebut kemudian di-*deploy* menggunakan *Flask* sebagai *web server* yang menyediakan API *endpoint* untuk menerima *request* klasifikasi dari aplikasi *mobile*. Aplikasi *Android* dibangun menggunakan *Android Studio* dengan

antarmuka pengguna yang memungkinkan *user* merekam suara ayam secara *real-time* menggunakan *AudioRecord* API. Audio yang direkam kemudian dikirim ke *Flask server* melalui *HTTP request* dalam format yang sesuai. Di sisi *server*, *Flask* menerima audio, melakukan ekstraksi fitur MFCC menggunakan *librosa*, menormalisasi data sesuai dengan parameter yang tersimpan, dan menjalankan prediksi menggunakan model .h5 yang telah dimuat. Hasil klasifikasi berupa probabilitas untuk setiap kategori vokalisasi dikirim kembali ke aplikasi *Android* dalam format JSON, yang kemudian ditampilkan kepada pengguna. Pendekatan ini memungkinkan model tetap menggunakan *library* Python lengkap untuk akurasi maksimal, sementara aplikasi *mobile* tetap ringan dan responsif.

3.3.6. Uji coba aplikasi

Tahapan uji coba aplikasi dilakukan untuk mengevaluasi performa dan fungsionalitas sistem secara menyeluruh. Tujuan utama dari uji coba ini adalah untuk memastikan bahwa:

1. Aplikasi dapat merekam atau menerima *input* suara ayam dengan baik.
2. Model klasifikasi dapat memproses *input* dengan cepat dan akurat.
3. Hasil klasifikasi ditampilkan secara jelas dan informatif di antarmuka pengguna.

Uji coba dilakukan terhadap seluruh data pengujian untuk mengevaluasi kinerja sistem klasifikasi saat dijalankan melalui aplikasi *Android*.

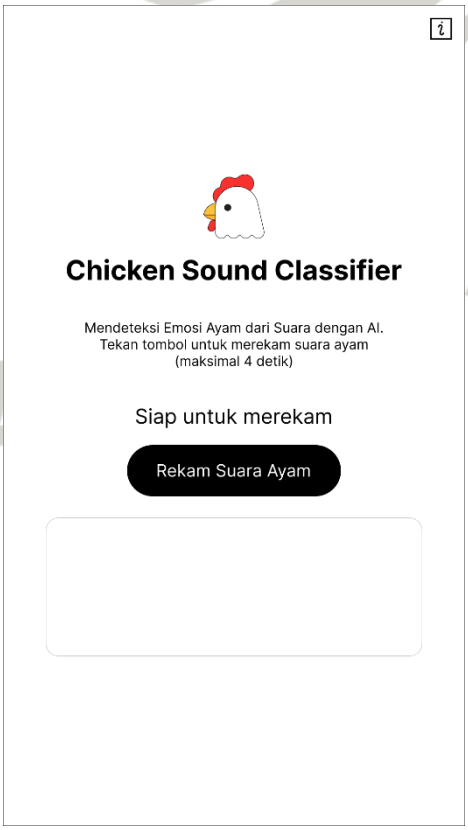
3.3.7. *Output*

Output dari aplikasi dirancang untuk memberikan informasi yang jelas dan mudah dipahami oleh pengguna. Setelah pengguna merekam suara ayam dan sistem melakukan proses klasifikasi melalui *Flask* API, aplikasi akan menampilkan hasil dalam dua format yang informatif:

1. Hasil klasifikasi: Menampilkan kategori vokalisasi yang terdeteksi dari empat kemungkinan kelas, yaitu "ayam betina marah", "ayam betina memanggil jantan", "terancam", dan "setelah bertelur".
2. Tingkat keyakinan: Menampilkan persentase tingkat keyakinan (*confidence level*) model terhadap hasil klasifikasi yang diberikan, memungkinkan pengguna mengetahui seberapa yakin sistem terhadap prediksi yang dibuat.

3.4. Perancangan Aplikasi *Mobile*

Aplikasi ini dirancang agar sederhana dan mudah digunakan oleh pengguna, dengan fitur utama berupa tombol untuk merekam suara ayam dan tampilan hasil klasifikasi secara *real-time*. Untuk memberikan gambaran visual mengenai aplikasi yang dikembangkan, berikut ditampilkan *mockup* antarmuka pengguna (*user interface*) dari aplikasi *mobile* klasifikasi vokalisasi ayam.



Gambar 3.3 *Mockup* aplikasi

3.5. Evaluasi Kinerja Aplikasi

Evaluasi kinerja aplikasi dilakukan untuk menilai seberapa baik aplikasi dalam menjalankan fungsinya. Evaluasi ini mencakup beberapa aspek, antara lain ukuran *file* aplikasi (APK), waktu respons dalam memproses *input* suara, stabilitas aplikasi saat dijalankan di berbagai perangkat, serta kemudahan penggunaan antarmuka pengguna (*UI/UX*).

Bab IV

Hasil dan Pembahasan

4.1. *Pre-processing*

Ekstraksi MFCC dilakukan pada setiap *file* audio menggunakan pustaka *librosa* di Python. Berikut potongan kode program yang digunakan untuk melakukan ekstraksi MFCC:

```
# Load dan ekstraksi MFCC
audio, sr = librosa.load(audio_path, sr=None)
mfcc = librosa.feature.mfcc(
    y=audio,
    sr=sr,
    n_mfcc=n_mfcc,
    hop_length=hop_length,
    n_fft=n_fft
)

# Padding/truncate seperti di CNN
if mfcc.shape[1] < max_length:
    pad_width = max_length - mfcc.shape[1]
    mfcc = np.pad(mfcc, ((0, 0), (0, pad_width)), mode='constant')
elif mfcc.shape[1] > max_length:
    mfcc = mfcc[:, :max_length]
```

Gambar 4.1 *Coding* Ekstraksi MFCC

Hasil ekstraksi MFCC berupa matriks dua dimensi yang merepresentasikan 20 koefisien MFCC pada setiap *frame* waktu (20 koefisien, 250 *frame*). Sebagai ilustrasi, Tabel 4.1 menampilkan hasil MFCC salah satu audio kelas ayam betina marah:

Tabel 4.1 Hasil Ekstraksi MFCC

Koefisien ke-	*Frame*
1	[-4.05263947e+02 -3.42682404e+02 -2.96657837e+02 -2.79135590e+02 -2.80144806e+02 …]
2	[3.42825851e+01 3.14657402e+01 2.56993275e+01 2.13720226e+01 1.98987236e+01 …]
3	[-1.83060951e+01 -4.65324173e+01 -7.45234299e+01 -7.93467712e+01 -6.82691956e+01 …]

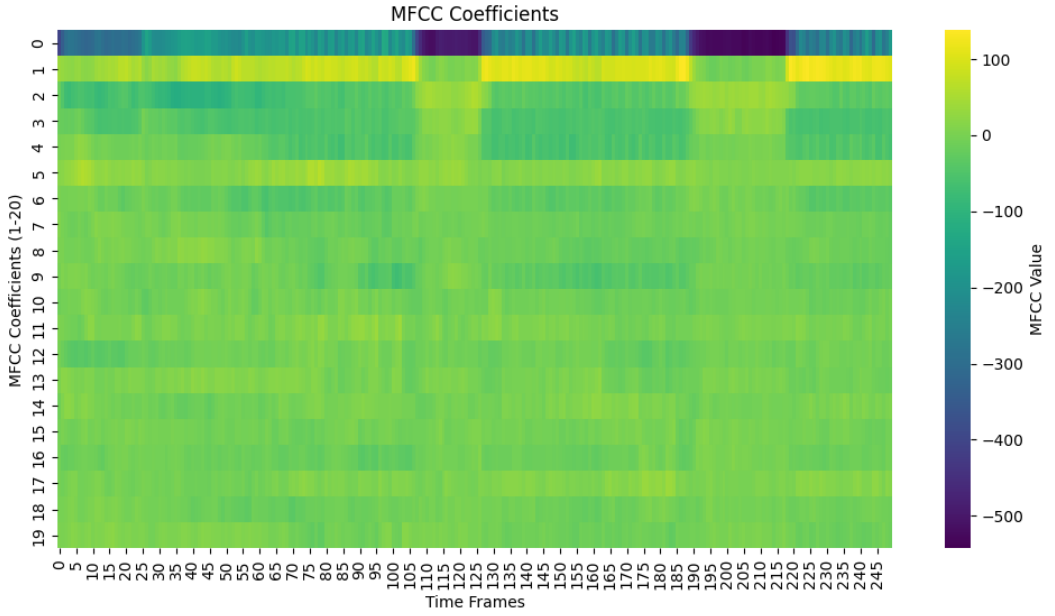
Tabel 4.1 (Lanjutan)

Koefisien ke-	*Frame*
4	[-1.66044197e+01 -2.09083824e+01 -2.20306816e+01 -2.15178108e+01 -2.20429096e+01 …]
5	[-2.39160085e+00 2.15873694e+00 4.61269712e+00 3.94952130e+00 4.48588085e+00 …]
6	[1.37115316e+01 1.70227776e+01 2.49543762e+01 2.83265018e+01 3.27465973e+01 …]
7	[1.52676487e+00 -3.56907606e+00 -4.20817709e+00 -3.16811681e-01 -1.22337759e+00 …]
8	[8.61397076e+00 -2.80311155e+00 -8.14572334e+00 -1.30763655e+01 -1.54029045e+01 …]
9	[3.35793883e-01 -2.23366052e-01 -7.50040770e+00 -6.21227074e+00 -7.18458700e+00 …]
10	[3.35793883e-01 -2.23366052e-01 -7.50040770e+00 -6.21227074e+00 -7.18458700e+00 …]
11	[-1.27342911e+01 -7.16390991e+00 -5.89129639e+00 -1.54993868e+00 -4.06783915e+00 …]
12	[4.73840523e+00 7.50699759e-01 -4.02826881e+00 -7.11853504e+00 -8.68721104e+00 …]
13	[4.73840523e+00 7.50699759e-01 -4.02826881e+00 -7.11853504e+00 -8.68721104e+00 …]
14	[4.57100153e+00 1.76209755e+01 1.82477188e+01 1.04738932e+01 5.68750191e+00 …]
15	[-1.10152779e+01 7.55600357e+00 1.85362778e+01 2.00215054e+01 1.35895252e+01 …]
16	[-6.40777206e+00 -5.59238195e-02 -3.41734695e+00 -8.94355583e+00 -1.19058018e+01 …]
17	[-5.97834539e+00 -2.00247746e+01 -2.24168091e+01 -1.83361168e+01 -1.65042133e+01 …]

Tabel 4.1 (Lanjutan)

Koefisien ke-	*Frame*
18	[-4.86296749e+00 -1.02683449e+01 -5.10086060e+00 4.12329388e+00 5.67534781e+00 …]
19	[-1.82686973e+00 -1.13553643e+00 7.60793746e-01 6.45622444e+00 8.26092815e+00 …]
20	[4.79829216e+00 5.53040981e-01 4.03854656e+00 8.72061634e+00 1.30499973e+01 …]

Sebagai pelengkap, matriks MFCC divisualisasikan menggunakan skema warna dari pustaka *matplotlib*. Warna pada gambar spektrogram mewakili nilai-nilai intensitas dari koefisien MFCC: nilai rendah ditampilkan dengan warna gelap, sedangkan nilai tinggi dengan warna terang. Visualisasi ini membantu manusia memahami pola yang terbentuk.



Gambar 4.2 Visualisasi spektrogram MFCC

Dengan cara ini, semua *dataset* diubah menjadi bentuk yang seragam dan siap digunakan sebagai input CNN. Dilakukan 12 kali percobaan pada model CNN dengan menggunakan proses dan konfigurasi MFCC yang sama pada seluruh percobaan.

4.2. Hasil Pemodelan CNN

Pada bagian ini disajikan hasil pelatihan dan pengujian model CNN setiap percobaan yang telah dilakukan. Setiap percobaan dilakukan dengan mengubah konfigurasi parameter tertentu yang telah dijelaskan sebelumnya pada Bab 3, tepatnya pada Tabel 3.7. Total terdapat 12 percobaan yang dilakukan, setiap percobaan dibahas dalam subbab tersendiri secara rinci.

Pada setiap subbab percobaan akan ditampilkan hasil evaluasi model dalam bentuk tabel akurasi, tabel metrik evaluasi per kelas (*precision*, *recall*, dan *F1-score*), visualisasi *Confusion matrix* masing-masing *subset* (pelatihan, validasi, dan pengujian), serta ukuran *file* .h5.

4.2.1. Percobaan 1

Pada percobaan pertama, model CNN dikonfigurasi dengan jumlah *filter* 32-32-64, 256 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0005. Percobaan ini digunakan sebagai *baseline* awal untuk membandingkan performa dengan konfigurasi pada percobaan lainnya.

Tabel 4.2 Akurasi Model pada Percobaan 1 CNN

Subset	**Akurasi**
Pelatihan	46.08%
Validasi	41.99%
Pengujian	65.00%

Akurasi pada data pengujian lebih tinggi dibandingkan data pelatihan maupun validasi. Secara umum, kondisi ini tergolong tidak biasa karena model cenderung memiliki performa terbaik pada data pelatihan yang menjadi acuan proses pembelajaran. Beberapa kemungkinan penyebab kondisi ini antara lain:

1. Distribusi data uji yang lebih mudah diprediksi secara kebetulan, sehingga model tanpa sengaja menyesuaikan dengan pola sederhana di data uji.

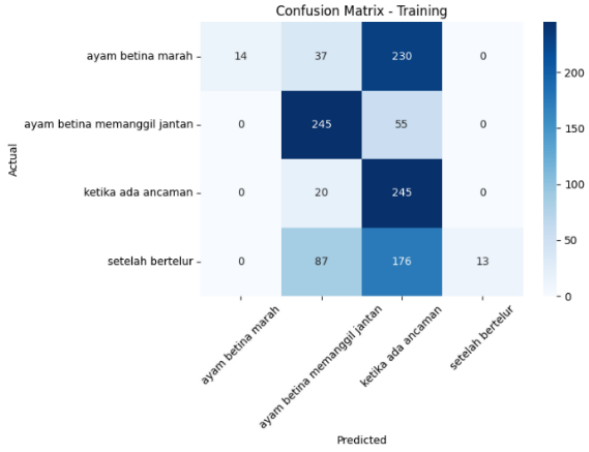
2. *Underfitting* pada pelatihan dan validasi, yang justru tidak berdampak besar terhadap data uji karena data uji tidak ikut mempengaruhi proses.
3. Ukuran data uji kecil, menyebabkan fluktuasi nilai akurasi lebih besar dan tidak selalu representatif.

Tabel 4.3 Metrik Evaluasi per Kelas pada Percobaan 1 CNN

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	62.98%	34.70%	100.00%
	Validasi	100.00%	55.21%	32.96%	100.00%
	Pengujian	100.00%	68.75%	51.72%	84.62%
Recall	Pelatihan	4.98%	81.67%	92.45%	4.71%
	Validasi	2.82%	70.67%	89.39%	5.80%
	Pengujian	13.33%	73.33%	100.00%	73.33%
F1-score	Pelatihan	9.49%	71.12%	50.46%	9.00%
	Validasi	5.48%	61.99%	48.16%	10.96%
	Pengujian	23.53%	70.97%	68.18%	78.57%

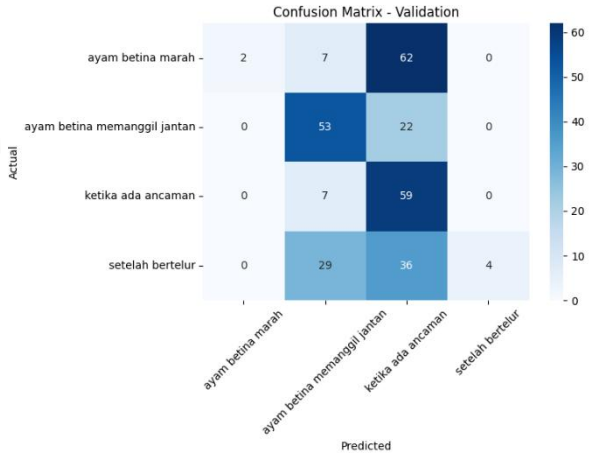
Kelas ketika ada ancaman dan ayam betina memanggil jantan cenderung lebih dikenali dengan baik (*recall* tinggi), sementara kelas lainnya memiliki *recall* yang sangat rendah, terutama ayam betina marah dan setelah bertelur. Hal ini menunjukkan bahwa model mengalami ketidakseimbangan dalam mengenali pola antar kelas.

Meskipun *precision* untuk beberapa kelas cukup tinggi, seperti ayam betina marah dan setelah bertelur, nilai *recall* yang sangat rendah menyebabkan *F1-score*-nya juga rendah, yang berarti model cenderung hanya mengenali sebagian kecil sampel dari kelas-kelas tersebut. Kondisi ini bisa disebabkan oleh distribusi fitur yang kurang representatif atau kemiripan karakteristik akustik antar kelas tertentu.



Gambar 4.3 *Confusion matrix* pelatihan CNN percobaan 1

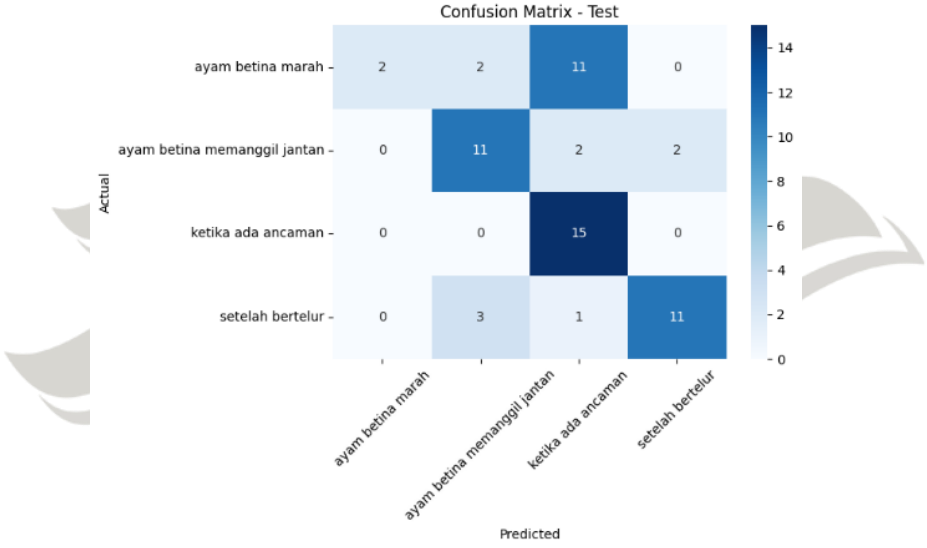
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 14 data benar diprediksi sebagai ayam betina marah, 37 data salah diprediksi sebagai ayam betina memanggil jantan, dan 230 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, ada 245 data benar diprediksi sebagai ayam betina memanggil jantan dan 55 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ketika ada ancaman, ada 245 data benar diprediksi sebagai ketika ada ancaman dan 20 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi setelah bertelur, ada 13 data benar diprediksi sebagai setelah bertelur, 87 data salah diprediksi sebagai ayam betina memanggil jantan, dan 176 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.4 *Confusion matrix* validasi CNN percobaan 1

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 2 data benar diprediksi sebagai ayam betina marah, 7 data salah

diprediksi sebagai ayam betina memanggil jantan, dan 62 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, ada 53 data benar diprediksi sebagai ayam betina memanggil jantan dan 22 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ketika ada ancaman, ada 59 data benar diprediksi sebagai ketika ada ancaman dan 7 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi setelah bertelur, ada 4 data benar diprediksi sebagai setelah bertelur, 29 data salah diprediksi sebagai ayam betina memanggil jantan, dan 36 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.5 *Confusion matrix* pengujian CNN percobaan 1

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, ada 2 data benar diprediksi sebagai ayam betina marah, 2 data salah diprediksi sebagai ayam betina memanggil jantan, dan 11 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, ada 11 data benar diprediksi sebagai ayam betina memanggil jantan, 2 data salah diprediksi sebagai ketika ada ancaman, dan 2 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi ketika ada ancaman, semua data benar diprediksi sebagai ketika ada ancaman. Pada vokalisasi setelah bertelur, ada 11 data benar diprediksi sebagai setelah bertelur, 3 data salah diprediksi sebagai ayam betina memanggil jantan, dan 1 data salah diprediksi sebagai ketika ada ancaman.

Ukuran *file* model hasil percobaan pertama ini sebesar 12 MB, yang mencerminkan jumlah parameter dan kompleksitas arsitektur yang digunakan.

4.2.2. Percobaan 2

Pada percobaan kedua, model CNN dikonfigurasi dengan jumlah *filter* 32-32-32, 64 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0005.

Tabel 4.4 Akurasi Model pada Percobaan 2 CNN

Subset	**Akurasi**
Pelatihan	100.00%
Validasi	99.64%
Pengujian	90.00%

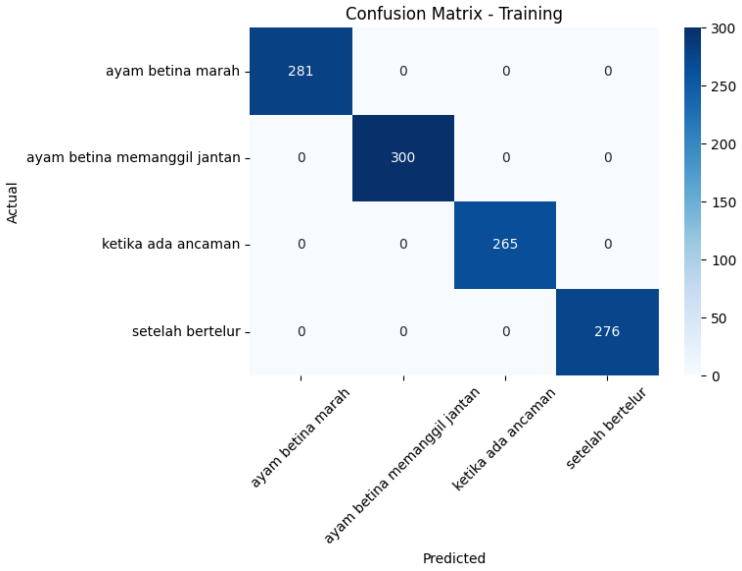
Nilai akurasi pada data pelatihan dan validasi menunjukkan hasil hampir sempurna, yang mengindikasikan bahwa model sangat baik dalam mengenali pola pada data yang telah dilihatnya. Namun, akurasi yang turun menjadi 90% pada data uji menandakan adanya *overfitting*, di mana model terlalu menyesuaikan diri dengan data latih dan kurang mampu menggeneralisasi ke data baru.

Tabel 4.5 Metrik Evaluasi per Kelas pada Percobaan 2 CNN

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	100.00%	98.51%	100.00%
	Pengujian	71.43%	100.00%	100.00%	100.00%
Recall	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	100.00%	100.00%	98.55%
	Pengujian	100.00%	60.00%	100.00%	100.00%
F1-score	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	100.00%	99.25%	99.27%
	Pengujian	83.33%	75.00%	100.00%	100.00%

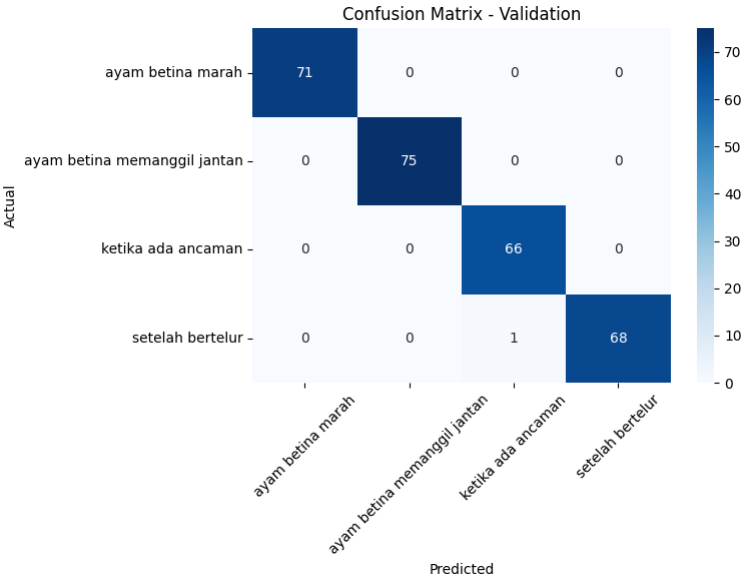
Precision, *recall*, dan *F1-score* pada masing-masing kelas menunjukkan hasil yang sangat baik untuk data pelatihan dan validasi. Terdapat sedikit penurunan

pada *precision* pengujian kelas ayam betina marah (71.43%) dan *recall* pengujian kelas ayam betina memanggil jantan (60%). Hal ini menyebabkan nilai *F1-score* pengujian pada kedua kelas tersebut menurun.



Gambar 4.6 *Confusion matrix* pelatihan CNN percobaan 2

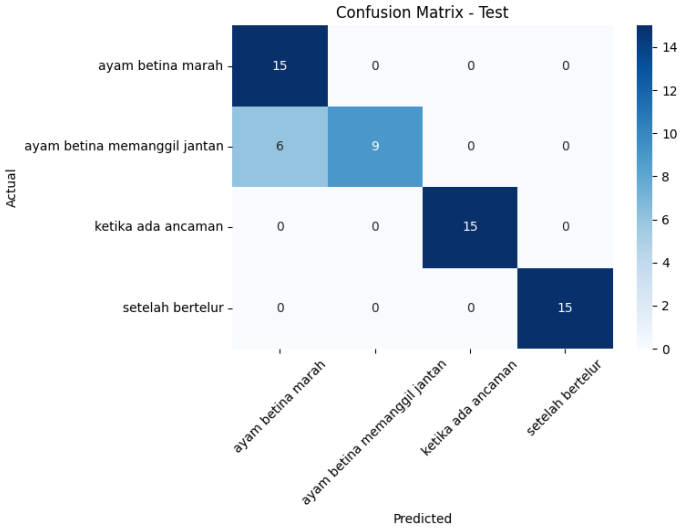
Berdasarkan *confusion matrix* hasil pelatihan, semua data vokalisasi ayam berhasil diprediksi dengan benar.



Gambar 4.7 *Confusion matrix* validasi CNN percobaan 2

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ayam betina memanggil jantan, dan ketika ada ancaman, semua data

berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.8 *Confusion matrix* pengujian CNN percobaan 2

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, ketika ada ancaman, dan setelah bertelur, semua data berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 9 data benar diprediksi sebagai ayam betina memanggil jantan dan 6 data salah diprediksi sebagai ayam betina marah.

Ukuran *file* model hasil percobaan kedua ini sebesar 1.75 MB, jauh lebih kecil dibanding percobaan pertama (12 MB). Ini menunjukkan bahwa konfigurasi parameter yang lebih kecil memberikan efisiensi signifikan dalam ukuran *file*.

4.2.3. Percobaan 3

Pada percobaan ketiga, model CNN dikonfigurasi dengan jumlah *filter* 32-64-64, 128 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0005.

Tabel 4.6 Akurasi Model pada Percobaan 3 CNN

Subset	**Akurasi**
Pelatihan	62.57%

Tabel 4.6 (Lanjutan)

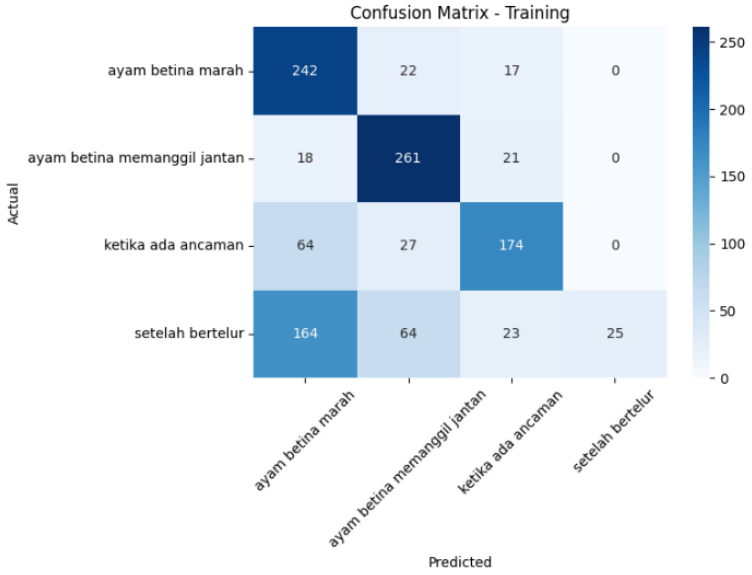
Subset	Akurasi
Validasi	60.14%
Pengujian	80.00%

Hasil akurasi yang rendah pada data pelatihan dan validasi menandakan bahwa kemampuan model dalam mempelajari pola dari data masih terbatas. Akurasi pengujian yang tinggi (80%) tidak menjamin generalisasi yang baik, karena hasil tersebut tidak selaras dengan performa pada data sebelumnya. Hal ini berpotensi disebabkan oleh ketidakseimbangan kelas atau distribusi data uji yang lebih mudah dikenali.

Tabel 4.7 Metrik Evaluasi per Kelas pada Percobaan 3 CNN

Metrik	*Subset*	Ayam Betina Marah	Ayam Betina Memanggil Jantan	Ketika Ada Ancaman	Setelah Bertelur
Precision	Pelatihan	49.59%	69.79%	74.04%	100.00%
	Validasi	52.46%	61.62%	70.91%	100.00%
	Pengujian	70.00%	90.91%	70.59%	100.00%
Recall	Pelatihan	86.12%	87.00%	65.66%	9.06%
	Validasi	90.14%	81.33%	59.09%	7.25%
	Pengujian	93.33%	66.67%	80.00%	80.00%
F1-score	Pelatihan	62.94%	77.45%	69.60%	16.61%
	Validasi	66.32%	70.11%	64.46%	13.51%
	Pengujian	80.00%	76.92%	75.00%	88.89%

Performa antar kelas sangat tidak seimbang. Kelas setelah bertelur memiliki *recall* yang sangat rendah pada data pelatihan dan validasi (9.06% dan 7.25%). Ini menunjukkan model hanya mengenali sebagian kecil data pada kelas tersebut, kemungkinan karena ketidakseimbangan distribusi fitur atau *overconfidence* pada prediksi tertentu.

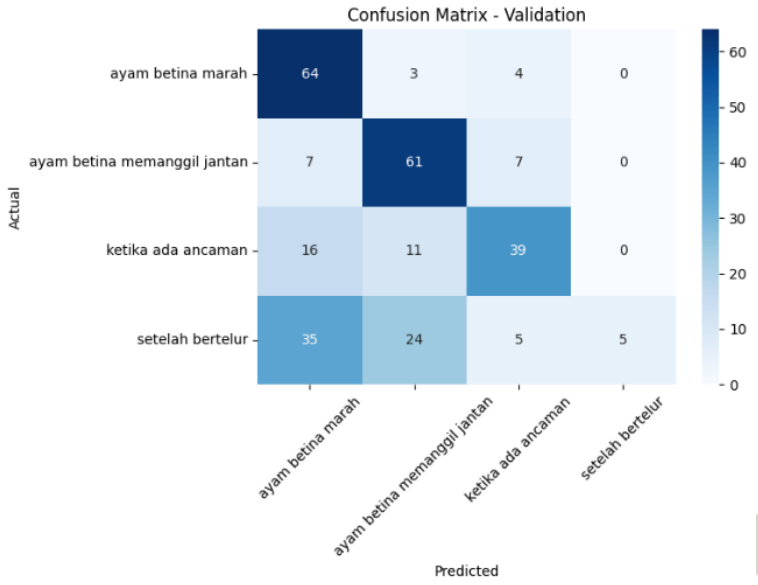


Gambar 4.9 *Confusion matrix* pelatihan CNN percobaan 3

Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 242 data benar diprediksi sebagai ayam betina marah, 22 data salah diprediksi sebagai ayam betina memanggil jantan, dan 17 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, ada 261 data benar diprediksi sebagai ayam betina memanggil jantan, 18 data salah diprediksi sebagai ayam betina marah, dan 21 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ketika ada ancaman, ada 174 data benar diprediksi sebagai ketika ada ancaman, 64 data salah diprediksi sebagai ayam betina marah, dan 27 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi setelah bertelur, ada 25 data benar diprediksi sebagai setelah bertelur, 164 data salah diprediksi sebagai ayam betina marah, 64 data salah diprediksi sebagai ayam betina memanggil jantan, dan 23 data salah diprediksi sebagai ketika ada ancaman.

Dari hasil tersebut terlihat bahwa performa model cukup baik dalam mengenali vokalisasi ayam betina marah dan ayam betina memanggil jantan, ditunjukkan oleh jumlah prediksi benar yang lebih tinggi dibandingkan prediksi salah. Namun, masih terdapat banyak kesalahan saat memprediksi kelas lainnya. Hal ini mengindikasikan bahwa model belum cukup representatif. Oleh karena itu, diperlukan evaluasi lebih lanjut terhadap konfigurasi parameter model maupun

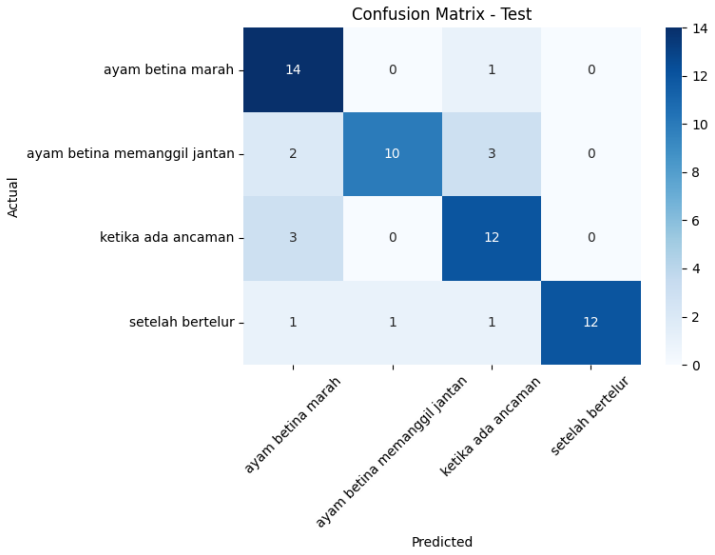
karakteristik suara tiap kelas agar model dapat membedakan vokalisasi secara lebih akurat.



Gambar 4.10 *Confusion matrix* validasi CNN percobaan 3

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 64 data benar diprediksi sebagai ayam betina marah, 3 data salah diprediksi sebagai ayam betina memanggil jantan, dan 4 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, ada 61 data benar diprediksi sebagai ayam betina memanggil jantan, 7 data salah diprediksi sebagai ayam betina marah, dan 7 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ketika ada ancaman, ada 39 data benar diprediksi sebagai ketika ada ancaman, 16 data salah diprediksi sebagai ayam betina marah, dan 11 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi setelah bertelur, ada 5 data benar diprediksi sebagai setelah bertelur, 35 data salah diprediksi sebagai ayam betina marah, 24 data salah diprediksi sebagai ayam betina memanggil jantan, dan 5 data salah diprediksi sebagai ketika ada ancaman.

Pola kebingungan antar kelas terlihat masih cukup tinggi. Hasil validasi ini mirip seperti hasil pelatihan, di mana jika dilihat dari warna *confusion matrix*, kelas ketika ada ancaman dan setelah bertelur banyak salah diprediksi sebagai kelas lain. Temuan ini penting untuk dijadikan pertimbangan dalam perbaikan model selanjutnya.



Gambar 4.11 *Confusion matrix* pengujian CNN percobaan 3

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, ada 14 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, ada 10 data benar diprediksi sebagai ayam betina memanggil jantan, 2 data salah diprediksi sebagai ayam betina marah, dan 3 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ketika ada ancaman, ada 12 data benar diprediksi sebagai ketika ada ancaman dan 3 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi setelah bertelur, ada 12 data benar diprediksi sebagai setelah bertelur, 1 data salah diprediksi sebagai ayam betina marah, 1 data salah diprediksi sebagai ayam betina memanggil jantan, dan 1 data salah diprediksi sebagai ketika ada ancaman.

Ukuran *file* model hasil percobaan ketiga ini sebesar 6.53 MB, lebih besar dibanding percobaan kedua (1.75 MB). Hal ini menunjukkan tingkat kompleksitas arsitektur yang sedang, namun masih tergolong efisien untuk skenario *deployment mobile*.

4.2.4. Percobaan 4

Pada percobaan keempat, model CNN dikonfigurasi dengan jumlah *filter* 32-32-32, 64 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0001.

Tabel 4.8 Akurasi Model pada Percobaan 4 CNN

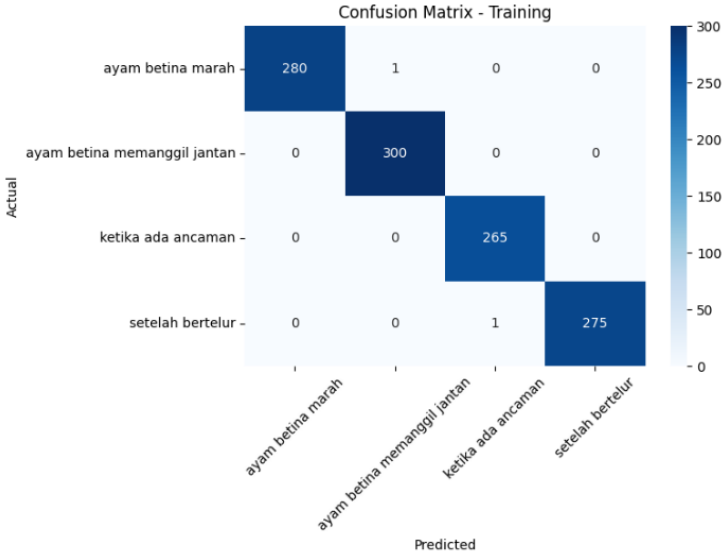
Subset	**Akurasi**
Pelatihan	99.82%
Validasi	98.93%
Pengujian	95.00%

Model menunjukkan performa akurasi yang sangat tinggi pada seluruh *subset* data. Hal ini menandakan bahwa model mampu mengenali pola dengan sangat baik dan juga berhasil mengeneralisasikan ke data baru, tanpa tanda-tanda *overfitting* atau *underfitting* yang mencolok.

Tabel 4.9 Metrik Evaluasi per Kelas pada Percobaan 4 CNN

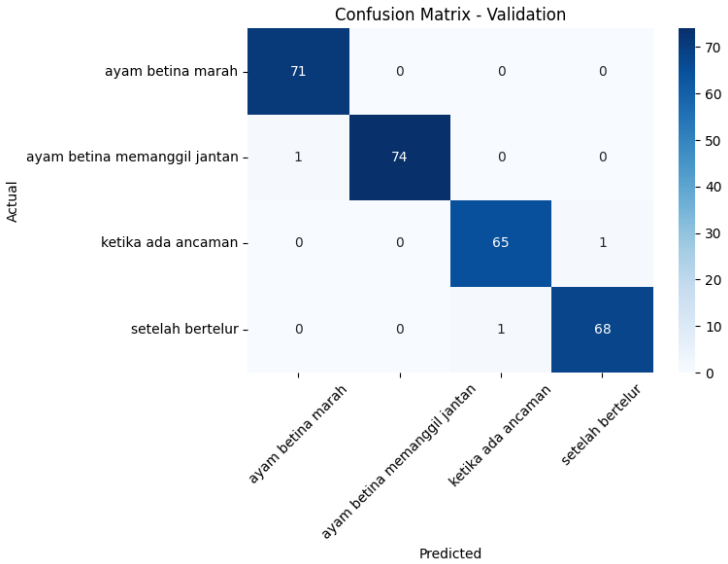
Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	99.67%	99.62%	100.00%
	Validasi	98.61%	100.00%	98.48%	98.55%
	Pengujian	88.24%	100.00%	100.00%	93.75%
Recall	Pelatihan	99.64%	100.00%	100.00%	99.64%
	Validasi	100.00%	98.67%	98.48%	98.55%
	Pengujian	100.00%	93.33%	86.67%	100.00%
F1-score	Pelatihan	99.82%	99.83%	99.81%	99.82%
	Validasi	99.30%	99.33%	98.48%	98.55%
	Pengujian	93.75%	96.55%	92.86%	96.77%

Terdapat sedikit penurunan pada *precision* pengujian kelas ayam betina marah (88.24%) dan *recall* pengujian kelas ketika ada ancaman (86.67%). Nilai *precision*, *recall*, dan *F1-score* yang mendekati atau di atas 90% di hampir seluruh kelas menunjukkan bahwa model berhasil mengenali pola vokalisasi dengan cukup baik. Hasil ini mencerminkan peningkatan yang signifikan dibanding percobaan sebelumnya.



Gambar 4.12 *Confusion matrix* pelatihan CNN percobaan 4

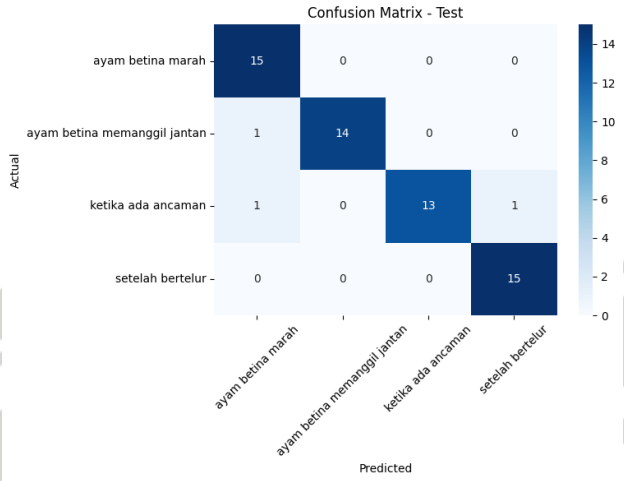
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 280 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan dan ketika ada ancaman, semua data berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 275 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.13 *Confusion matrix* validasi CNN percobaan 4

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, semua data berhasil diprediksi dengan benar. Pada vokalisasi ayam betina

memanggil jantan, ada 74 data benar diprediksi sebagai ayam betina memanggil jantan dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 65 data benar diprediksi sebagai ketika ada ancaman dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.14 *Confusion matrix* pengujian CNN percobaan 4

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 14 data benar diprediksi sebagai ayam betina memanggil jantan dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 13 data benar diprediksi sebagai ketika ada ancaman, 1 data salah diprediksi sebagai ayam betina marah, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, semua data berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan keempat ini sebesar 1.75 MB, lebih kecil dibanding percobaan ketiga (6.53 MB).

4.2.5. Percobaan 5

Percobaan kelima dilatarbelakangi oleh hasil yang cukup memuaskan pada percobaan keempat. Dengan mempertahankan konfigurasi CNN yang sama—

jumlah *filter* 32–32–32, 64 unit *dense*, dan *dropout* sebesar 0.1—perubahan dilakukan pada bagian *learning rate*, yaitu dinaikkan dari 0.0001 menjadi 0.001. Tujuan dari eksperimen ini adalah untuk mengamati pengaruh peningkatan laju pembelajaran terhadap konvergensi dan kestabilan model.

Tabel 4.10 Akurasi Model pada Percobaan 5 CNN

Subset	**Akurasi**
Pelatihan	100.00%
Validasi	99.29%
Pengujian	86.67%

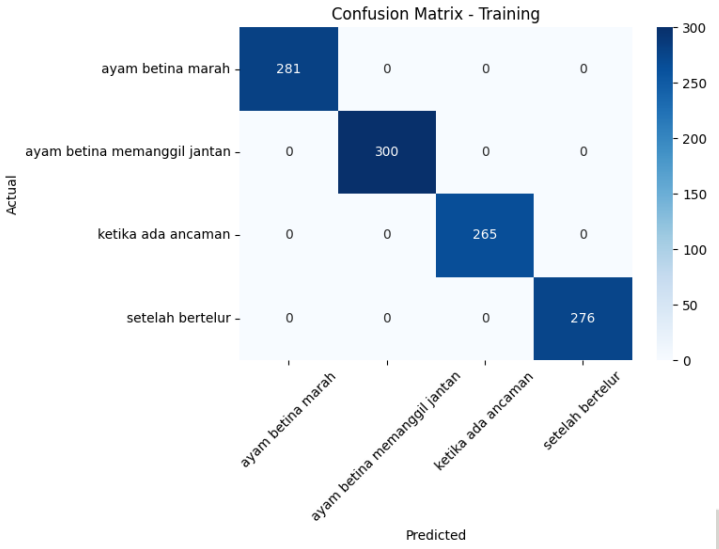
Akurasi pada data pelatihan dan validasi menunjukkan hasil yang sangat tinggi. Namun, penurunan signifikan terjadi pada data uji, yang mengindikasikan adanya kemungkinan *overfitting*.

Tabel 4.11 Metrik Evaluasi per Kelas pada Percobaan 5 CNN

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	98.68%	100.00%	98.57%
	Pengujian	71.43%	100.00%	100.00%	88.24%
Recall	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	98.59%	100.00%	98.48%	100.00%
	Pengujian	100.00%	46.67%	100.00%	100.00%
F1-score	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	99.29%	99.34%	99.24%	99.28%
	Pengujian	83.33%	63.64%	100.00%	93.75%

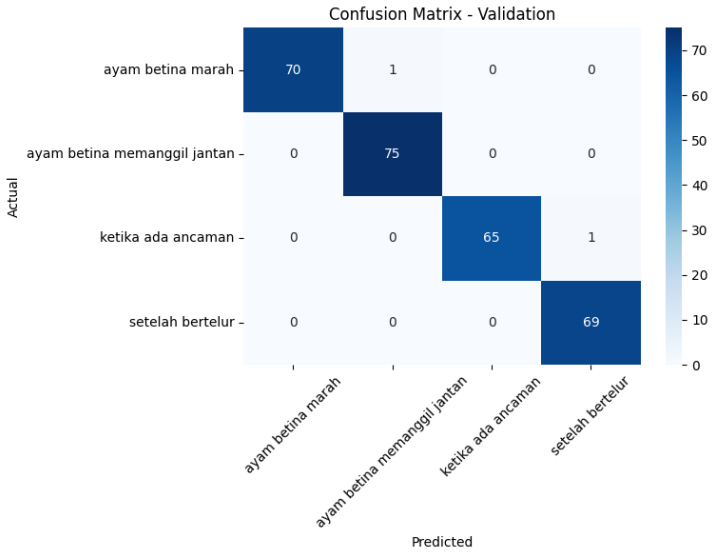
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (71.43%) dan setelah bertelur (86.67%). Terdapat pula penurunan pada *recall*

pengujian kelas ayam betina memanggil jantan (46.67%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada kedua kelas tersebut.



Gambar 4.15 *Confusion matrix* pelatihan CNN percobaan 5

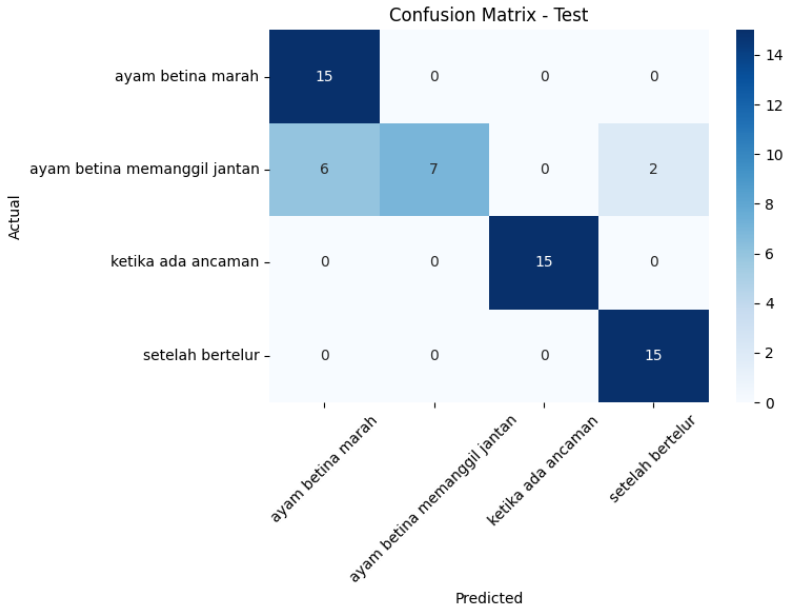
Berdasarkan *confusion matrix* hasil pelatihan, semua data vokalisasi ayam berhasil diprediksi dengan benar.



Gambar 4.16 *Confusion matrix* validasi CNN percobaan 5

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 70 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan, semua data berhasil diprediksi dengan benar. Pada vokalisasi

ketika ada ancaman, ada 65 data benar diprediksi sebagai ketika ada ancaman dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, semua data berhasil diprediksi dengan benar.



Gambar 4.17 *Confusion matrix* pengujian CNN percobaan 5

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 7 data benar diprediksi sebagai ayam betina memanggil jantan, 6 data salah diprediksi sebagai ayam betina marah, dan 2 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data berhasil diprediksi dengan benar.

Ukuran *file* sama dengan percobaan keempat, yaitu sebesar 1.75 MB, karena struktur arsitektur tidak diubah, hanya nilai *learning rate*-nya saja yang dimodifikasi. Eksperimen ini menunjukkan bahwa *learning rate* yang terlalu besar belum tentu lebih baik.

4.2.6. Percobaan 6

Pada percobaan keenam, model CNN dikonfigurasi dengan jumlah *filter* 32-32-64, 128 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0001.

Tabel 4.12 Akurasi Model pada Percobaan 6 CNN

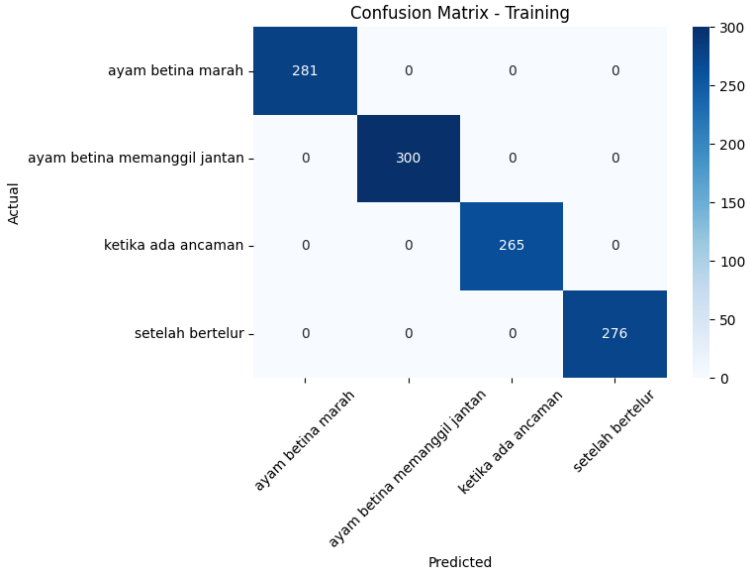
Subset	**Akurasi**
Pelatihan	100.00%
Validasi	97.51%
Pengujian	83.33%

Meskipun akurasi pelatihan mencapai 100%, dan validasi juga sangat tinggi, akurasi pengujian menurun dan lebih rendah dibandingkan percobaan kelima. Penurunan ini menunjukkan kemungkinan *overfitting*.

Tabel 4.13 Metrik Evaluasi per Kelas pada Percobaan 6 CNN

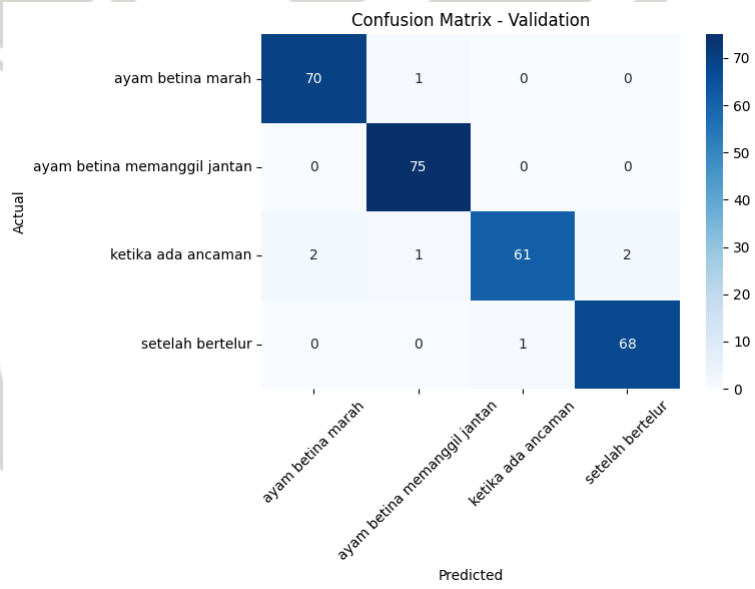
Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	97.22%	97.40%	98.39%	97.14%
	Pengujian	63.64%	100.00%	100.00%	88.24%
Recall	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	98.59%	100.00%	92.42%	98.55%
	Pengujian	93.33%	53.33%	86.67%	100.00%
F1-score	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	97.90%	98.68%	95.31%	97.84%
	Pengujian	75.68%	69.57%	92.86%	93.75%

Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (63.64%) dan setelah bertelur (88.24%). Terdapat pula penurunan pada *recall* pengujian kelas ayam betina memanggil jantan (53.33%) dan ketika ada ancaman (86.67%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada kedua kelas tersebut.



Gambar 4.18 *Confusion matrix* pelatihan CNN percobaan 6

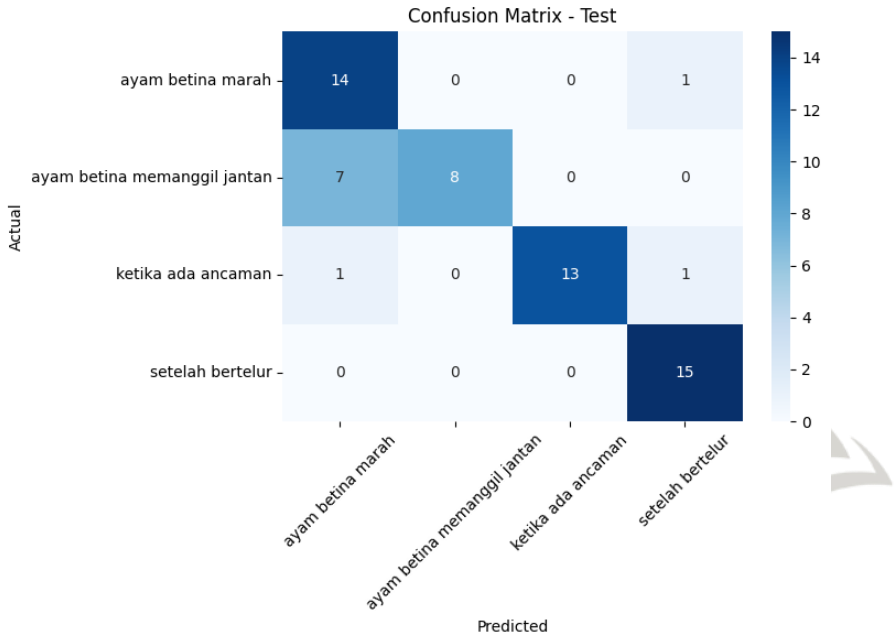
Berdasarkan *confusion matrix* hasil pelatihan, semua data vokalisasi ayam berhasil diprediksi dengan benar.



Gambar 4.19 *Confusion matrix* validasi CNN percobaan 6

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 70 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan, semua data berhasil diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 61 data benar diprediksi sebagai ketika ada ancaman, 2

data salah diprediksi sebagai ayam betina marah, 1 data salah diprediksi sebagai ayam betina memanggil jantan, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.20 *Confusion matrix* pengujian CNN percobaan 6

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, ada 14 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi ayam betina memanggil jantan, ada 8 data benar diprediksi sebagai ayam betina memanggil jantan dan 7 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 13 data benar diprediksi sebagai ketika ada ancaman, 1 data salah diprediksi sebagai ayam betina marah, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, semua data berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan keenam ini sebesar 6.22 MB, lebih besar dibanding percobaan kelima (1.75 MB) karena konfigurasi *layer* menjadi lebih kompleks (penambahan *filter* dan *dense* unit).

4.2.7. Percobaan 7

Pada percobaan ketujuh, model CNN dikonfigurasi dengan jumlah *filter* 32-64-128, 128 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0001.

Tabel 4.14 Akurasi Model pada Percobaan 7 CNN

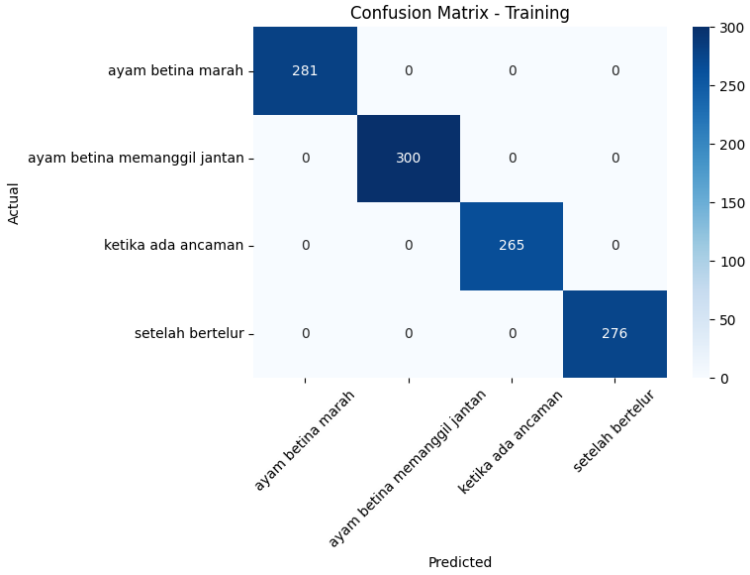
Subset	**Akurasi**
Pelatihan	100.00%
Validasi	98.58%
Pengujian	93.33%

Model berhasil mencapai akurasi yang sangat tinggi pada ketiga *subset*. Akurasi pengujian sebesar 93.33% menjadi salah satu yang terbaik sejauh ini, menunjukkan bahwa peningkatan *filter* secara bertahap berhasil meningkatkan performa tanpa menyebabkan *overfitting* yang parah.

Tabel 4.15 Metrik Evaluasi per Kelas pada Percobaan 7 CNN

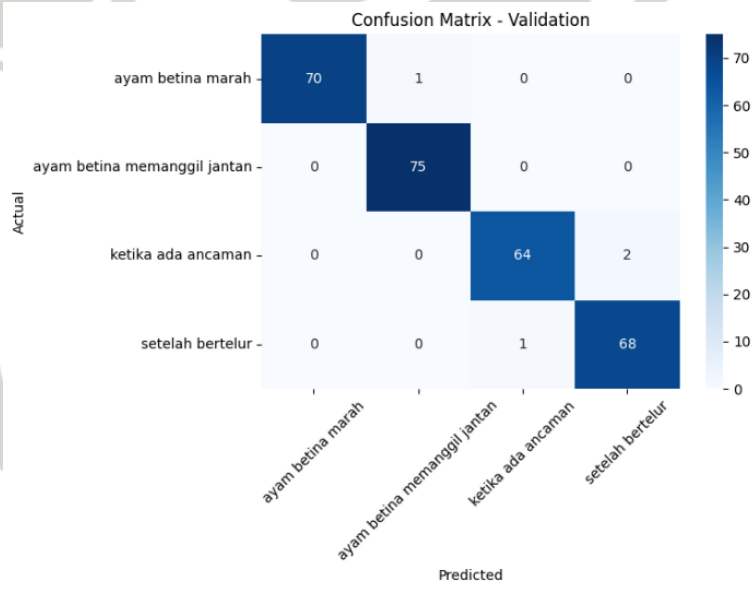
Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	98.68%	98.46%	97.14%
	Pengujian	88.24%	100.00%	93.75%	93.75%
Recall	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	98.59%	100.00%	96.97%	98.55%
	Pengujian	100.00%	73.33%	100.00%	100.00%
F1-score	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	99.29%	99.34%	97.71%	97.84%
	Pengujian	93.75%	84.62%	96.77%	96.77%

Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (88.24%) dan *recall* pengujian kelas ayam betina memanggil jantan (73.33%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada kedua kelas tersebut.



Gambar 4.21 *Confusion matrix* pelatihan CNN percobaan 7

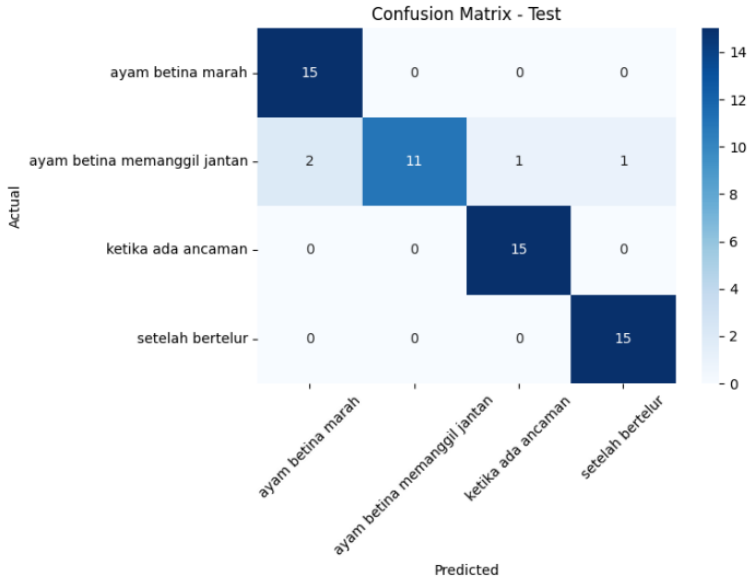
Berdasarkan *confusion matrix* hasil pelatihan, semua data vokalisasi ayam berhasil diprediksi dengan benar.



Gambar 4.22 *Confusion matrix* validasi CNN percobaan 7

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 70 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan, semua data berhasil diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 64 data benar diprediksi sebagai ketika ada ancaman dan

2 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.23 *Confusion matrix* pengujian CNN percobaan 7

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 11 data benar diprediksi sebagai ayam betina memanggil jantan, 2 data salah diprediksi sebagai ayam betina marah, 1 data salah diprediksi sebagai ketika ada ancaman, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan ketujuh ini sebesar 12.7 MB, lebih besar dibanding percobaan keenam (6.22 MB). Hal ini sejalan dengan jumlah *filter* yang lebih besar pada tiap blok, yang berkontribusi terhadap jumlah parameter yang lebih banyak.

4.2.8. Percobaan 8

Pada percobaan kedelapan, model CNN dikonfigurasi dengan jumlah *filter* 32-64-128, 128 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0005.

Tabel 4.16 Akurasi Model pada Percobaan 8 CNN

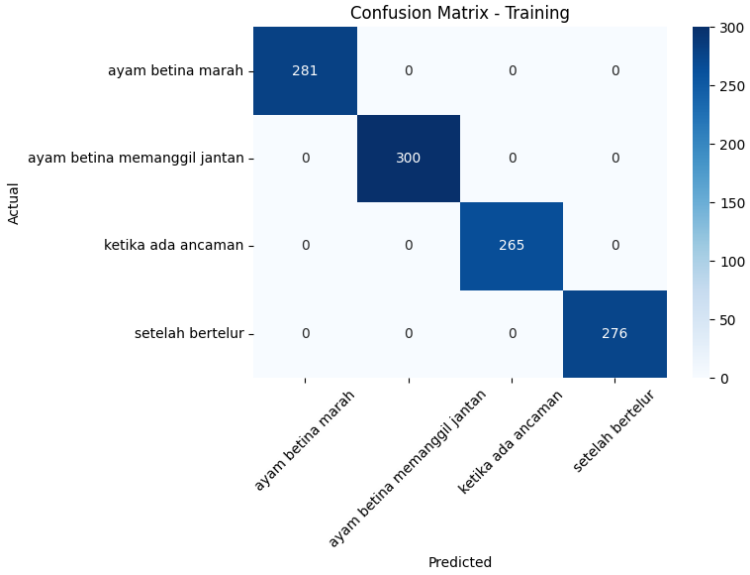
Subset	**Akurasi**
Pelatihan	100.00%
Validasi	99.29%
Pengujian	88.33%

Meskipun model mampu mencapai akurasi tinggi pada pelatihan dan validasi, terjadi penurunan akurasi pada pengujian, yang mengindikasikan kemungkinan *overfitting*. Akurasi pengujian sebesar 88.33% masih tergolong baik.

Tabel 4.17 Metrik Evaluasi per Kelas pada Percobaan 8 CNN

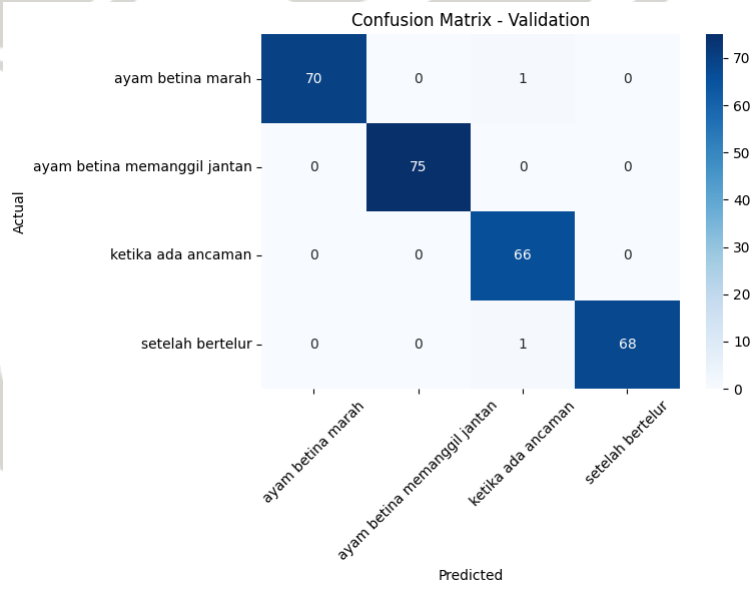
Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	100.00%	97.06%	100.00%
	Pengujian	68.18%	100.00%	100.00%	100.00%
Recall	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	98.59%	100.00%	100.00%	98.55%
	Pengujian	100.00%	66.67%	86.67%	100.00%
F1-score	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	99.29%	100.00%	98.51%	99.27%
	Pengujian	81.08%	80.00%	92.86%	100.00%

Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (68.18%). Lalu, terdapat pula penurunan pada *recall* pengujian kelas ayam betina memanggil jantan (66.67%) dan ketika ada ancaman (86.67%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada kedua kelas tersebut.



Gambar 4.24 *Confusion matrix* pelatihan CNN percobaan 8

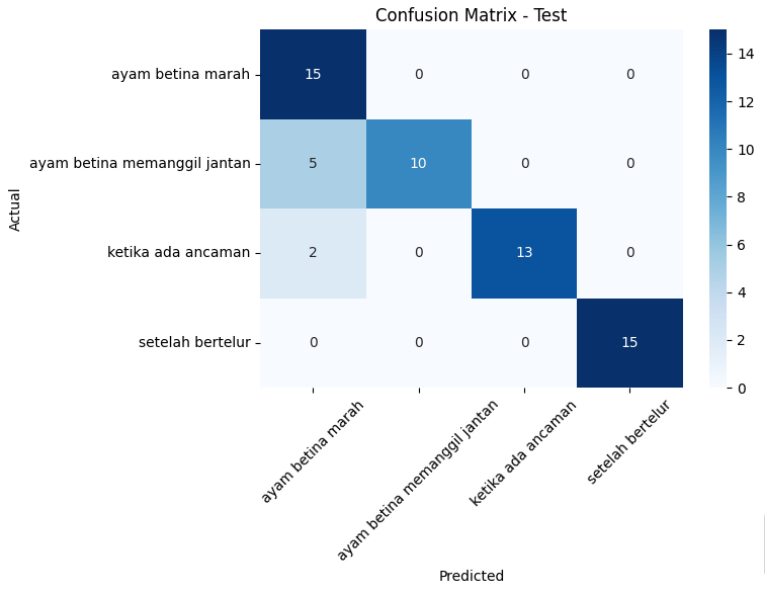
Berdasarkan *confusion matrix* hasil pelatihan, semua data vokalisasi ayam berhasil diprediksi dengan benar.



Gambar 4.25 *Confusion matrix* validasi CNN percobaan 8

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 70 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan dan ketika ada ancaman, semua data berhasil diprediksi dengan benar. Pada

vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.26 *Confusion matrix* pengujian CNN percobaan 8

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 10 data benar diprediksi sebagai ayam betina memanggil jantan dan 5 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 13 data benar diprediksi sebagai ketika ada ancaman dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi setelah bertelur, semua data berhasil diprediksi dengan benar.

Ukuran *file* sama dengan percobaan ketujuh, yaitu sebesar 12.7 MB, karena struktur arsitektur yang sama, menunjukkan banyaknya parameter yang dilatih. Dengan *learning rate* yang lebih tinggi (0.0005), mungkin model melakukan *update* parameter yang terlalu besar, sehingga meskipun *training loss* cepat turun, kemampuan adaptasi terhadap data baru berkurang.

4.2.9. Percobaan 9

Pada percobaan kesembilan, model CNN dikonfigurasi dengan jumlah *filter* 32-64-128, 256 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0005.

Tabel 4.18 Akurasi Model pada Percobaan 9 CNN

Subset	**Akurasi**
Pelatihan	100.00%
Validasi	98.93%
Pengujian	96.67%

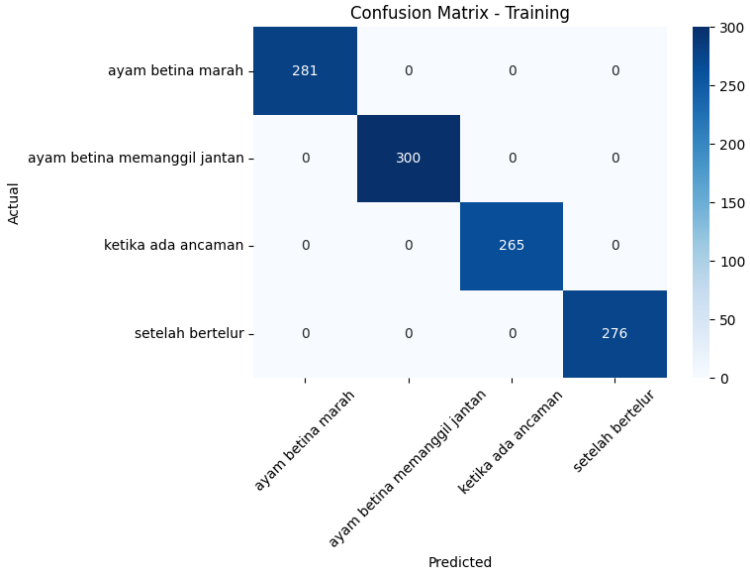
Akurasi model dalam percobaan ini menunjukkan performa yang sangat tinggi dan stabil pada ketiga *subset*, terutama pada data uji, yang merupakan indikator penting dalam mengukur generalisasi model. Nilai akurasi pengujian mencapai 96.67%, tertinggi dibanding percobaan-percobaan sebelumnya.

Tabel 4.19 Metrik Evaluasi per Kelas pada Percobaan 9 CNN

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	100.00%	97.01%	98.55%
	Pengujian	88.24%	100.00%	100.00%	100.00%
Recall	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	98.59%	100.00%	98.48%	98.55%
	Pengujian	100.00%	86.67%	100.00%	100.00%
F1-score	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	99.29%	100.00%	97.74%	98.55%
	Pengujian	93.75%	92.86%	100.00%	100.00%

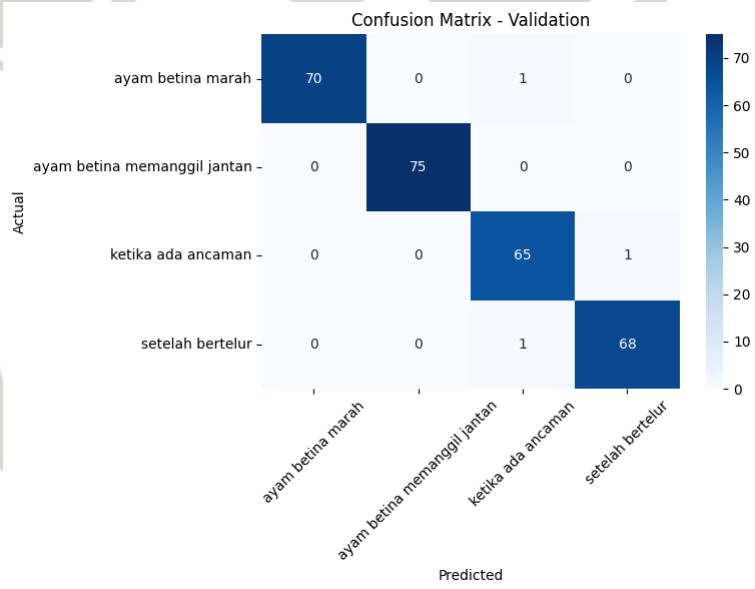
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (88.24%) dan *recall* pengujian kelas ayam betina memanggil jantan (86.67%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada kedua kelas tersebut.

Meski demikian, secara keseluruhan, nilai metrik pada semua kelas masih tergolong tinggi, model tetap menunjukkan performa yang kuat dan stabil.



Gambar 4.27 *Confusion matrix* pelatihan CNN percobaan 9

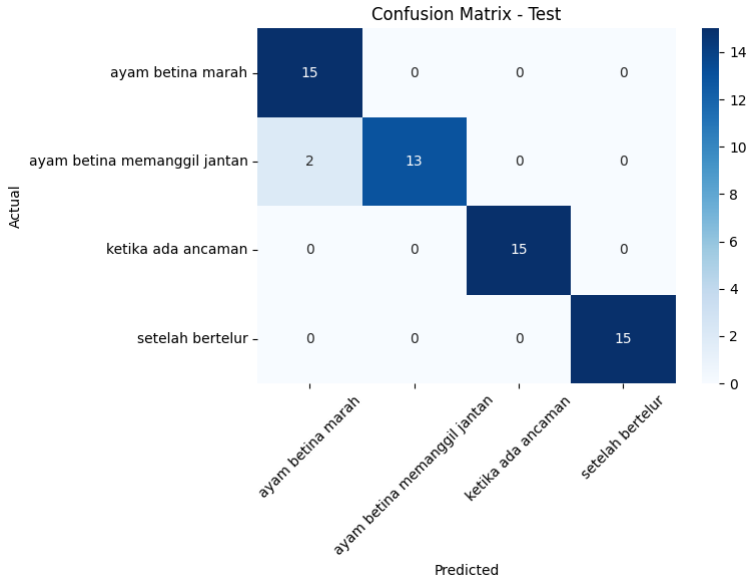
Berdasarkan *confusion matrix* hasil pelatihan, semua data vokalisasi ayam berhasil diprediksi dengan benar.



Gambar 4.28 *Confusion matrix* validasi CNN percobaan 9

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 70 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, semua data berhasil diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 65 data benar diprediksi sebagai ketika ada ancaman dan 1 data salah

diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.29 *Confusion matrix* pengujian CNN percobaan 9

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 13 data benar diprediksi sebagai ayam betina memanggil jantan dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan kesembilan ini sebesar 24.4 MB, lebih besar dibanding percobaan kedelapan (12.7 MB) karena kompleksitas jaringan meningkat, khususnya dengan penambahan unit *dense* menjadi 256.

4.2.10. Percobaan 10

Pada percobaan kesepuluh, model CNN dikonfigurasi dengan jumlah *filter* 32-64-128, 256 unit *dense*, *dropout* sebesar 0.2, dan *learning rate* 0.0005.

Tabel 4.20 Akurasi Model pada Percobaan 10 CNN

Subset	**Akurasi**
Pelatihan	100.00%
Validasi	99.64%
Pengujian	85.00%

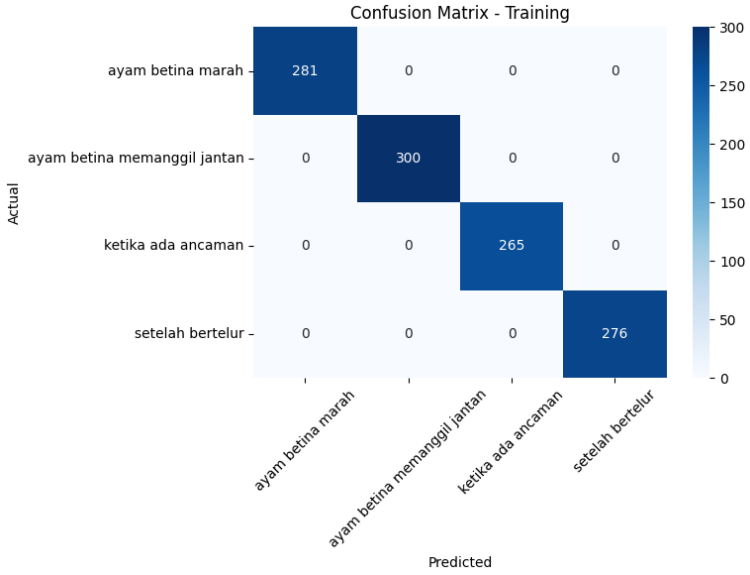
Model menunjukkan akurasi sangat tinggi pada data pelatihan dan validasi. Namun, akurasi data uji turun menjadi 85%, menunjukkan adanya *gap* performa antara pelatihan / validasi dan generalisasi ke data baru.

Tabel 4.21 Metrik Evaluasi per Kelas pada Percobaan 10 CNN

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	100.00%	98.51%	100.00%
	Pengujian	62.50%	100.00%	100.00%	100.00%
Recall	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	100.00%	100.00%	98.55%
	Pengujian	100.00%	40.00%	100.00%	100.00%
F1-score	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	100.00%	99.25%	99.27%
	Pengujian	76.92%	57.14%	100.00%	100.00%

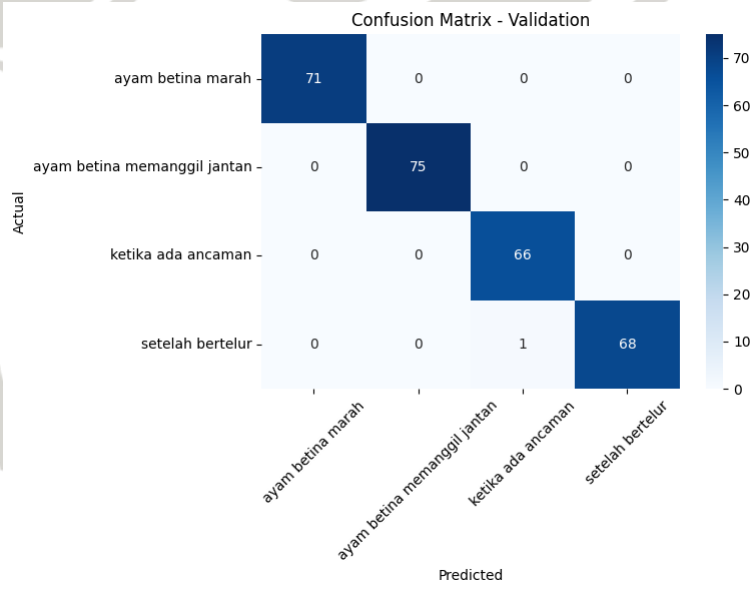
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (62.50%) dan *recall* pengujian kelas ayam betina memanggil jantan (40%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada kedua kelas tersebut.

Secara umum, performa model pada kelas ketika ada ancaman dan setelah bertelur sangat tinggi dan stabil, baik dalam pelatihan, validasi, maupun pengujian.



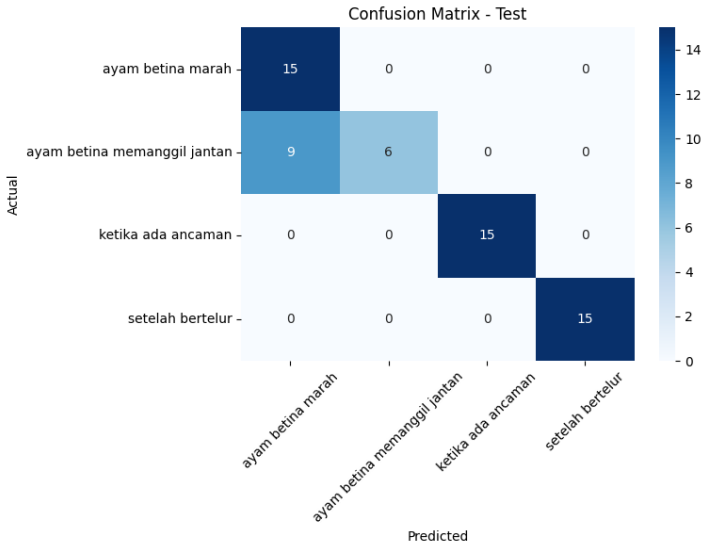
Gambar 4.30 *Confusion matrix* pelatihan CNN percobaan 10

Berdasarkan *confusion matrix* hasil pelatihan, semua data vokalisasi ayam berhasil diprediksi dengan benar.



Gambar 4.31 *Confusion matrix* validasi CNN percobaan 10

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ayam betina memanggil jantan, dan ketika ada ancaman, semua data berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.32 *Confusion matrix* pengujian CNN percobaan 10

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 6 data benar diprediksi sebagai ayam betina memanggil jantan dan 9 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data berhasil diprediksi dengan benar.

Ukuran *file* sama dengan percobaan kesembilan, yaitu sebesar 24.4 MB, sebanding dengan konfigurasi kompleks pada layer *dense* dan jumlah *filter*. Percobaan ini menunjukkan bahwa peningkatan *dropout* dari 0.1 menjadi 0.2 belum tentu memberikan efek positif terhadap performa model, terutama dalam hal generalisasi ke data uji.

4.2.11. Percobaan 11

Pada percobaan kesebelas, model CNN dikonfigurasi dengan jumlah *filter* 32-64-128, 256 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.001.

Tabel 4.22 Akurasi Model pada Percobaan 11 CNN

Subset	**Akurasi**
Pelatihan	100.00%

Tabel 4.22 (Lanjutan)

Subset	**Akurasi**
Validasi	98.93%
Pengujian	100.00%

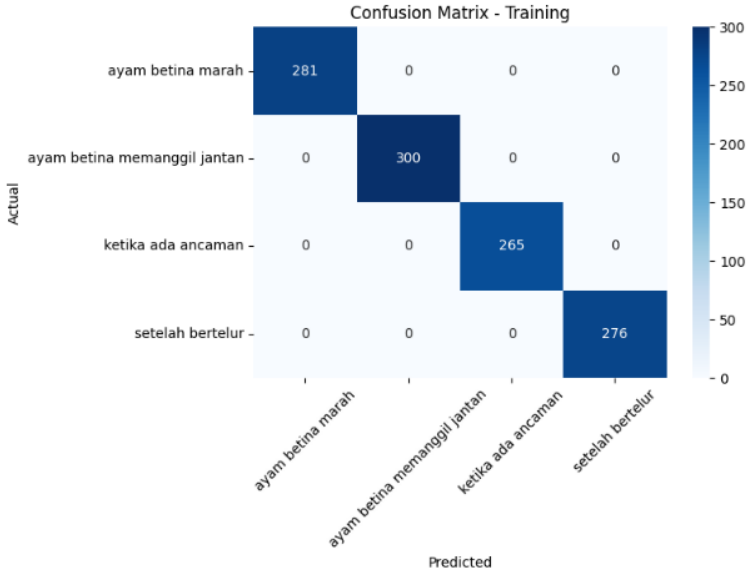
Model menunjukkan akurasi yang sangat tinggi di semua *subset*, termasuk akurasi sempurna pada data uji (100%), yang merupakan hasil terbaik sepanjang percobaan yang telah dilakukan.

Tabel 4.23 Metrik Evaluasi per Kelas pada Percobaan 11 CNN

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	100.00%	100.00%	95.65%	100.00%
	Pengujian	100.00%	100.00%	100.00%	100.00%
Recall	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	97.18%	100.00%	100.00%	98.55%
	Pengujian	100.00%	100.00%	100.00%	100.00%
F1-score	Pelatihan	100.00%	100.00%	100.00%	100.00%
	Validasi	98.57%	100.00%	97.78%	99.27%
	Pengujian	100.00%	100.00%	100.00%	100.00%

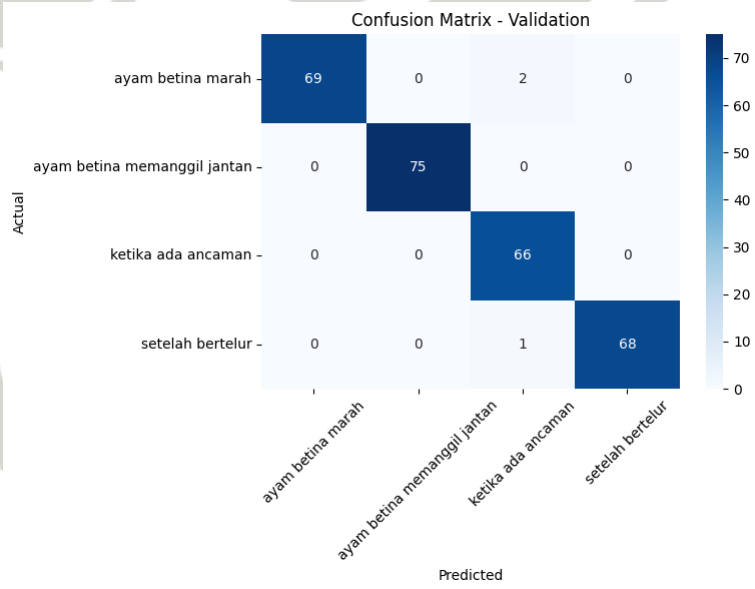
Secara umum, hasil evaluasi model menunjukkan performa yang sangat baik, meskipun terdapat beberapa nilai metrik yang sedikit menurun pada *subset* tertentu, nilai terendah tetap berada pada kisaran yang tinggi, yaitu 95.65%. Hal ini menunjukkan bahwa model mampu mengklasifikasikan data dengan konsistensi dan tingkat kesalahan yang sangat rendah.

Tidak adanya kesalahan klasifikasi pada data pengujian (100% untuk seluruh metrik dan kelas) menjadi indikator bahwa model memiliki kemampuan generalisasi yang sangat kuat terhadap data uji.



Gambar 4.33 *Confusion matrix* pelatihan CNN percobaan 11

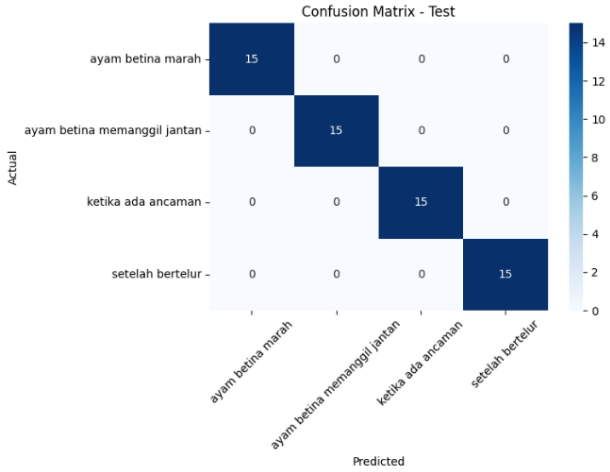
Berdasarkan *confusion matrix* hasil pelatihan, semua data vokalisasi ayam berhasil diprediksi dengan benar.



Gambar 4.34 *Confusion matrix* validasi CNN percobaan 11

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 69 data benar diprediksi sebagai ayam betina marah dan 2 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan dan ketika ada ancaman, semua data berhasil diprediksi dengan benar. Pada

vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.35 *Confusion matrix* pengujian CNN percobaan 11

Berdasarkan *confusion matrix* hasil pengujian, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* sama dengan percobaan kesepuluh, yaitu sebesar 24.4 MB, yang konsisten dengan kompleksitas arsitektur. Percobaan ini menunjukkan hasil terbaik dari seluruh eksperimen yang telah dilakukan.

4.2.12. Percobaan 12

Pada percobaan keduabelas, model CNN dikonfigurasi dengan jumlah *filter* 16-32-64, 128 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.001. Konfigurasi ini dipilih untuk menguji apakah model yang lebih sederhana dapat menghasilkan performa yang tetap tinggi, namun dengan ukuran model yang lebih kecil dan efisien.

Tabel 4.24 Akurasi Model pada Percobaan 12 CNN

Subset	**Akurasi**
Pelatihan	99.38%
Validasi	98.93%
Pengujian	100.00%

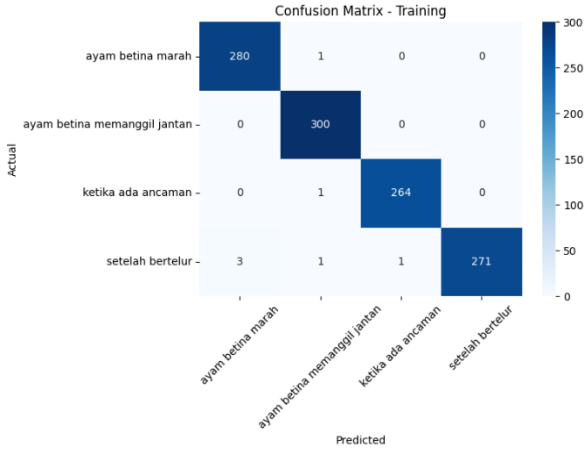
Meskipun kompleksitas model lebih rendah dibandingkan beberapa percobaan sebelumnya, hasil akurasi tetap sangat tinggi, dan bahkan menunjukkan akurasi sempurna pada data uji (100%). Namun, akurasi pada data pelatihan dan validasi sedikit lebih rendah dibandingkan akurasi pada data pengujian, yang mungkin terjadi karena data uji berjumlah kecil dan lebih mudah dikenali oleh model.

Tabel 4.25 Metrik Evaluasi per Kelas pada Percobaan 12 CNN

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	98.94%	99.01%	99.62%	100.00%
	Validasi	95.95%	100.00%	100.00%	100.00%
	Pengujian	100.00%	100.00%	100.00%	100.00%
Recall	Pelatihan	99.64%	100.00%	99.62%	98.19%
	Validasi	100.00%	100.00%	98.48%	97.10%
	Pengujian	100.00%	100.00%	100.00%	100.00%
F1-score	Pelatihan	99.29%	99.50%	99.62%	99.09%
	Validasi	97.93%	100.00%	99.24%	98.53%
	Pengujian	100.00%	100.00%	100.00%	100.00%

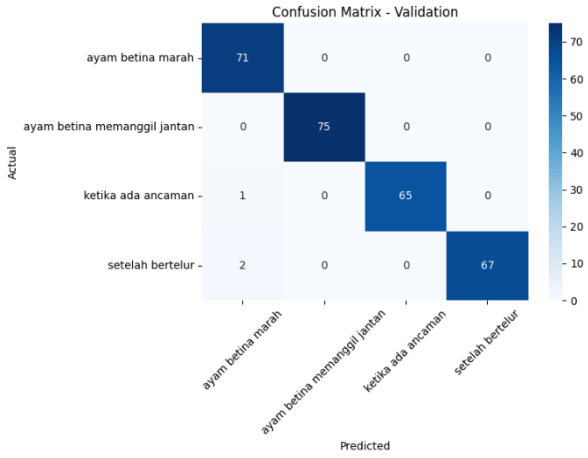
Secara umum, metrik evaluasi model menunjukkan hasil yang sangat baik dengan rata-rata nilai mendekati 100%. Penurunan nilai hanya terjadi sedikit pada data pelatihan dan validasi, dengan nilai terendah sebesar 95.65%, yang masih tergolong sangat tinggi.

Ketepatan model dalam mengklasifikasikan seluruh data pengujian tanpa kesalahan juga menunjukkan bahwa model mampu mengenali pola vokalisasi secara efektif. Selain itu, konsistensi performa pada seluruh *subset* menunjukkan bahwa model mampu belajar dengan baik selama proses pelatihan dan mempertahankan akurasi tinggi saat diuji pada data yang belum pernah dilihat sebelumnya.



Gambar 4.36 *Confusion matrix* pelatihan CNN percobaan 12

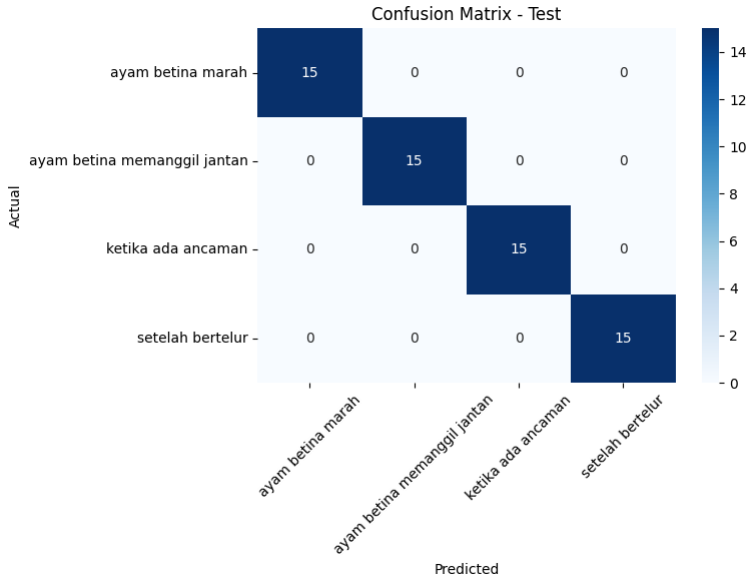
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 280 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 264 data benar diprediksi sebagai ketika ada ancaman dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi setelah bertelur, ada 271 data benar diprediksi sebagai setelah bertelur, 3 data salah diprediksi sebagai ayam betina marah, 1 data salah diprediksi sebagai ayam betina memanggil jantan, dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.37 *Confusion matrix* validasi CNN percobaan 12

Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah dan ayam betina memanggil jantan, semua data vokalisasi ayam berhasil

diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 65 data benar diprediksi sebagai ketika ada ancaman dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi setelah bertelur, ada 67 data benar diprediksi sebagai setelah bertelur dan 2 data salah diprediksi sebagai ayam betina marah.



Gambar 4.38 *Confusion matrix* pengujian CNN percobaan 12

Berdasarkan *confusion matrix* hasil pengujian, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan keduabelas ini sebesar 6.16 MB, lebih kecil dibanding percobaan kesebelas (24.4 MB). Hal ini membuktikan bahwa model yang lebih sederhana tetap dapat menghasilkan performa yang tinggi dengan ukuran model yang lebih kecil.

Hasil percobaan 12 inilah yang akan digunakan untuk perbandingan dengan metode *deep learning* lainnya.

4.3. Ringkasan Hasil Seluruh Percobaan CNN

Berdasarkan hasil tiap percobaan yang sudah dijabarkan, dilakukan analisis untuk menilai apakah suatu model mengalami *overfitting*, *underfitting*, atau

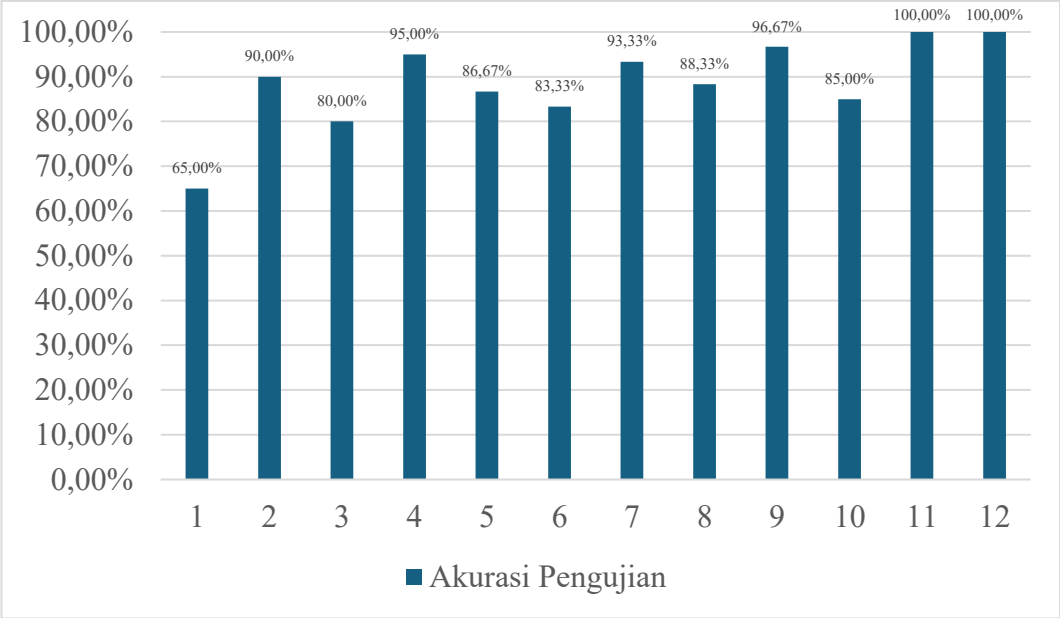
menunjukkan performa yang baik. Ringkasan kondisi dari masing-masing percobaan disajikan dalam Tabel 4.26.

Tabel 4.26 Ringkasan Kondisi Tiap Percobaan CNN Berdasarkan Akurasi

Percobaan	Akurasi Pelatihan	Akurasi Validasi	Akurasi Pengujian	Keterangan Kondisi
Percobaan 1	46.08%	41.99%	65.00%	*Underfitting*
Percobaan 2	100.00%	99.64%	90.00%	*Overfitting*
Percobaan 3	62.57%	60.14%	80.00%	*Underfitting*
Percobaan 4	99.82%	98.93%	95.00%	Bagus
Percobaan 5	100.00%	99.29%	86.67%	*Overfitting*
Percobaan 6	100.00%	97.51%	83.33%	*Overfitting*
Percobaan 7	100.00%	98.58%	93.33%	Bagus
Percobaan 8	100.00%	99.29%	88.33%	*Overfitting*
Percobaan 9	100.00%	98.93%	96.67%	Bagus
Percobaan 10	100.00%	99.64%	85.00%	*Overfitting*
Percobaan 11	100.00%	98.93%	100.00%	Bagus
Percobaan 12	99.38%	98.93%	100.00%	Bagus

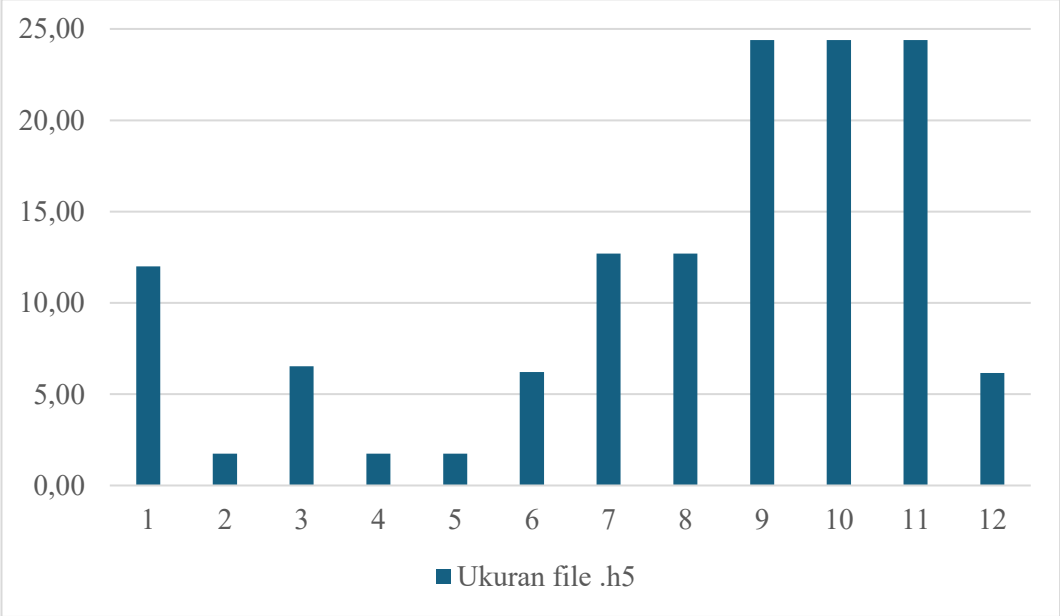
Pemberian keterangan kondisi pada setiap percobaan bertujuan untuk mengidentifikasi performa terhadap data pelatihan, validasi, dan pengujian. Dengan melihat pola akurasi, dapat diketahui apakah model mengalami *underfitting* (belum mampu menangkap pola data) atau *overfitting* (terlalu menyesuaikan dengan data pelatihan dan gagal generalisasi). Percobaan yang menunjukkan indikasi *underfitting* atau *overfitting* yang parah telah dieliminasi dari pertimbangan utama, sehingga hanya model dengan performa stabil (bagus) yang digunakan dalam analisis perbandingan antar percobaan. Langkah ini dilakukan untuk memastikan bahwa model yang dipilih memiliki generalisasi yang baik.

Berikut grafik ringkasan hasil akurasi pengujian dari seluruh percobaan CNN:



Gambar 4.39 Grafik hasil akurasi pengujian seluruh percobaan CNN

Gambar 4.39 menunjukkan pola grafik akurasi pengujian yang naik turun. Hal ini disebabkan oleh pendekatan eksploratif yang dilakukan dalam proses mengubah konfigurasi parameter di tiap percobaan.



Gambar 4.40 Grafik hasil ukuran *file* .h5 seluruh percobaan CNN

Gambar 4.40 menunjukkan pola grafik besar dan kecilnya ukuran *file* .h5 yang dihasilkan di tiap percobaan. Jika dilihat dan dibandingkan bersama pola grafik hasil akurasi pengujian, maka percobaan keduabelas menunjukkan hasil

terbaik karena akurasinya 100% dan ukuran *file* .h5 kecil. Hal ini memperkuat pengambilan keputusan untuk menggunakan hasil percobaan keduabelas sebagai hasil terbaik dari CNN.

4.4. *Transformer*

Pada bagian ini disajikan hasil pelatihan dan pengujian model *Transformer* setiap percobaan yang telah dilakukan. Setiap percobaan dilakukan dengan mengubah konfigurasi parameter tertentu yang telah dijelaskan sebelumnya pada Bab 3, tepatnya pada Tabel 3.9. Total terdapat 8 percobaan yang dilakukan, setiap percobaan dibahas dalam subbab tersendiri secara rinci.

4.4.1. Percobaan 1

Pada percobaan pertama, model *Transformer* dikonfigurasi dengan ukuran *patch* 10, dimensi *embedding* 96, jumlah *head* 8, jumlah *encode layer* 3, dan dimensi MLP 256. Percobaan ini digunakan sebagai *baseline* awal untuk membandingkan performa dengan konfigurasi pada percobaan lainnya. Tabel 4.27 menampilkan hasil akurasi model pada percobaan pertama.

Tabel 4.27 Akurasi Model pada Percobaan 1 *Transformer*

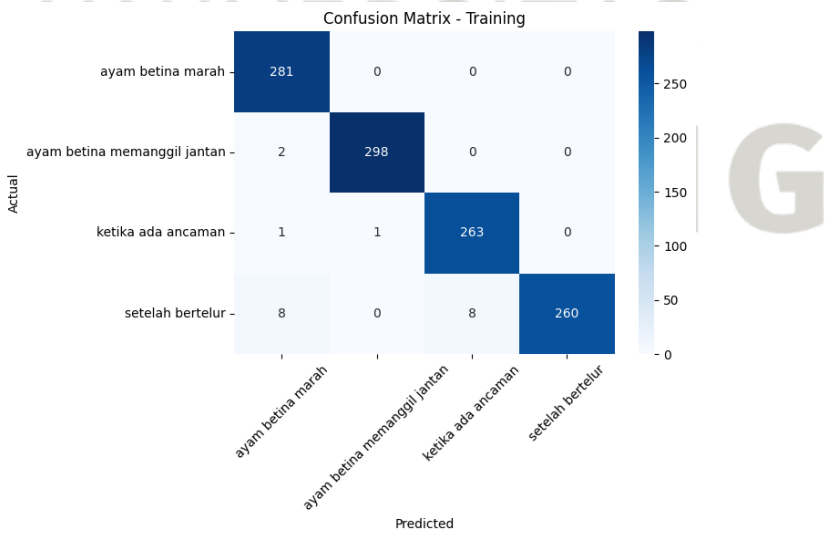
Subset	**Akurasi**
Pelatihan	98.22%
Validasi	98.58%
Pengujian	96.67%

Pada percobaan pertama menunjukkan performa yang sangat tinggi. Akurasi pelatihan mencapai 98.22%, validasi 98.58%, dan pengujian 96.67%, dengan selisih yang kecil dan tidak menunjukkan tanda *overfitting*.

Tabel 4.28 Metrik Evaluasi per Kelas pada Percobaan 1 *Transformer*

Metrik	*Subset*	Ayam Betina Marah	Ayam Betina Memanggil Jantan	Ketika Ada Ancaman	Setelah Bertelur
Precision	Pelatihan	96.23%	99.67%	97.05%	100.00%
	Validasi	95.95%	100.00%	98.51%	100.00%
	Pengujian	88.24%	100.00%	100.00%	100.00%
Recall	Pelatihan	100.00%	99.33%	99.25%	94.20%
	Validasi	100.00%	97.33%	100.00%	97.10%
	Pengujian	100.00%	86.67%	100.00%	100.00%
F1-score	Pelatihan	98.08%	99.50%	98.13%	97.01%
	Validasi	97.93%	98.65%	99.25%	98.53%
	Pengujian	93.75%	92.86%	100.00%	100.00%

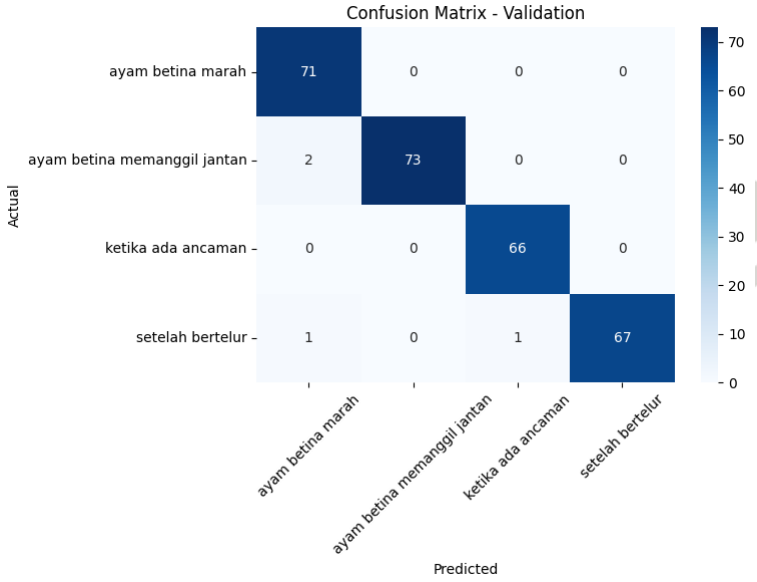
Pada seluruh *subset*, metrik evaluasi menunjukkan performa tinggi di tiap kelas, meskipun terdapat penurunan pada *recall* pengujian kelas ayam betina memanggil jantan (86.67%). Hal ini menandakan bahwa model tidak mengalami *overfitting* parah dan tetap mempertahankan generalisasi yang baik.



Gambar 4.41 *Confusion matrix* pelatihan *Transformer* percobaan 1

Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada

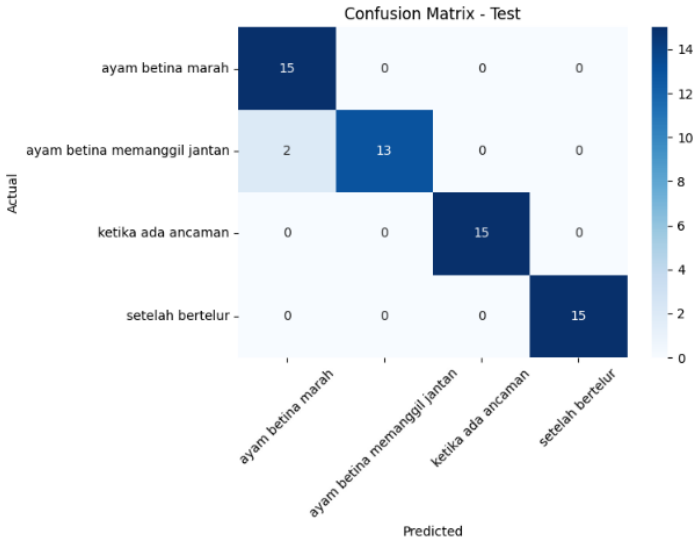
vokalisasi ayam betina memanggil jantan, ada 298 data benar diprediksi sebagai ayam betina memanggil jantan dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 263 data benar diprediksi sebagai ketika ada ancaman, 1 data salah diprediksi sebagai ayam betina marah, dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi setelah bertelur, ada 260 data benar diprediksi sebagai setelah bertelur, 8 data salah diprediksi sebagai ayam betina marah, dan 8 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.42 *Confusion matrix* validasi *Transformer* percobaan 1

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 73 data benar diprediksi sebagai ayam betina memanggil jantan dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 67 data benar diprediksi sebagai setelah bertelur, 1 data salah diprediksi sebagai ayam betina marah, dan 1 data salah diprediksi sebagai ketika ada ancaman.

Secara keseluruhan, tingkat kesalahan relatif sangat rendah dan tidak tersebar luas ke banyak kelas lain, menunjukkan bahwa model konsisten dalam mengenali pola-pola vokalisasi antar kelas selama proses validasi.



Gambar 4.43 *Confusion matrix* pengujian *Transformer* percobaan 1

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 13 data benar diprediksi sebagai ayam betina memanggil jantan dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan pertama ini sebesar 3.78 MB, tergolong kecil untuk *deployment*.

4.4.2. Percobaan 2

Pada percobaan kedua, model *Transformer* dikonfigurasi dengan ukuran *patch* 5, dimensi *embedding* 96, jumlah *head* 8, jumlah *encode layer* 3, dan dimensi MLP 256.

Tabel 4.29 Akurasi Model pada Percobaan 2 *Transformer*

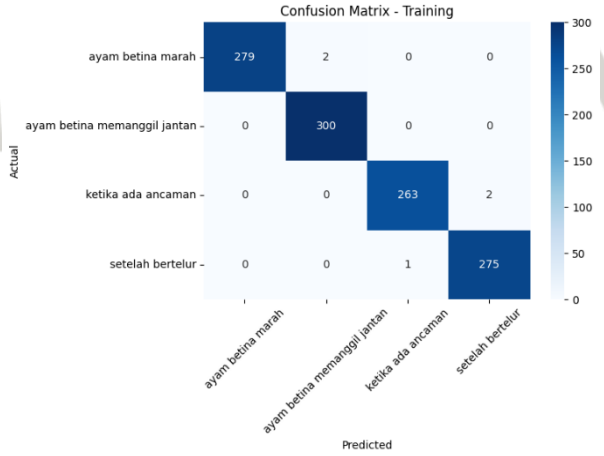
Subset	**Akurasi**
Pelatihan	99.55%
Validasi	99.29%
Pengujian	95.00%

Percobaan kedua menunjukkan performa model yang sangat baik. Akurasi pelatihan dan validasi sangat tinggi (hampir 100%), akurasi pengujian sebesar 95% mengindikasikan model mampu melakukan generalisasi dengan baik tanpa mengalami *overfitting* yang parah.

Tabel 4.30 Metrik Evaluasi per Kelas pada Percobaan 2 *Transformer*

Metrik	*Subset*	Ayam Betina Marah	Ayam Betina Memanggil Jantan	Ketika Ada Ancaman	Setelah Bertelur
Precision	Pelatihan	100.00%	99.34%	99.62%	99.28%
	Validasi	98.61%	100.00%	100.00%	98.57%
	Pengujian	88.24%	100.00%	100.00%	93.75%
Recall	Pelatihan	99.29%	100.00%	99.25%	99.64%
	Validasi	100.00%	98.67%	98.48%	100.00%
	Pengujian	100.00%	80.00%	100.00%	100.00%
F1-score	Pelatihan	99.64%	99.67%	99.43%	99.46%
	Validasi	99.30%	99.33%	99.24%	99.28%
	Pengujian	93.75%	88.89%	100.00%	96.67%

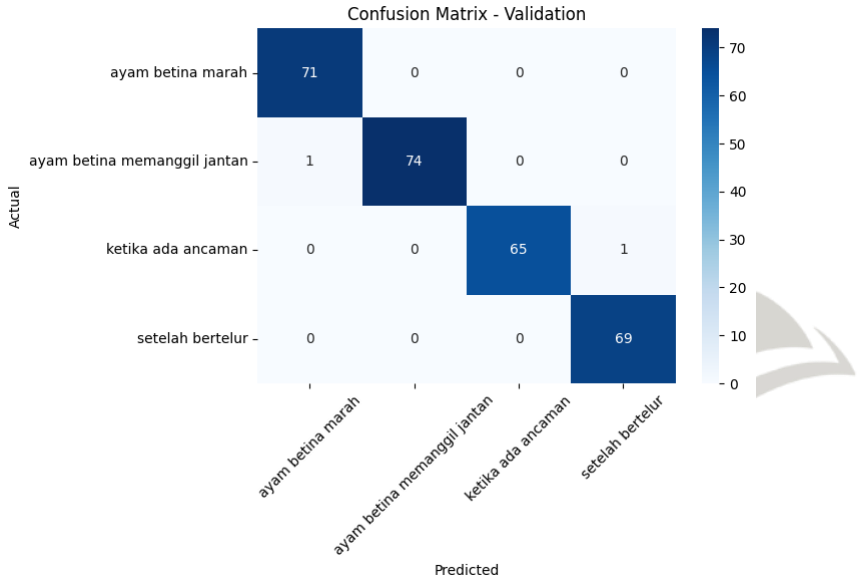
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (88.24%) dan *recall* pengujian kelas ayam betina memanggil jantan (80%).



Gambar 4.44 *Confusion matrix* pelatihan *Transformer* percobaan 2

Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 279 data benar diprediksi sebagai ayam betina marah dan 2 data salah

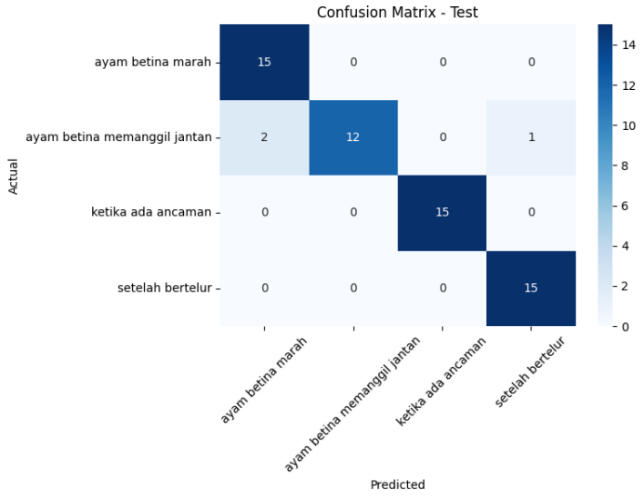
diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 263 data benar diprediksi sebagai ketika ada ancaman dan 2 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, ada 275 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.45 *Confusion matrix* validasi *Transformer* percobaan 2

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 74 data benar diprediksi sebagai ayam betina memanggil jantan dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 65 data benar diprediksi sebagai ketika ada ancaman dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Secara keseluruhan, jumlah prediksi yang benar lebih dominan dibandingkan prediksi yang salah, dan kesalahan hanya terjadi pada dua data pada masing-masing kelas ayam betina memanggil jantan dan ketika ada ancaman. Hasil ini menunjukkan bahwa model mampu memprediksi vokalisasi dengan tepat selama validasi.



Gambar 4.46 *Confusion matrix* pengujian *Transformer* percobaan 2

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 12 data benar diprediksi sebagai ayam betina memanggil jantan, 2 data salah diprediksi sebagai ayam betina marah, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan kedua ini sebesar 3.72 MB, sedikit lebih kecil dibanding percobaan pertama (3.78 MB).

4.4.3. Percobaan 3

Pada percobaan ketiga, model *Transformer* dikonfigurasi dengan ukuran *patch* 10, dimensi *embedding* 64, jumlah *head* 8, jumlah *encode layer* 3, dan dimensi MLP 256.

Tabel 4.31 Akurasi Model pada Percobaan 3 *Transformer*

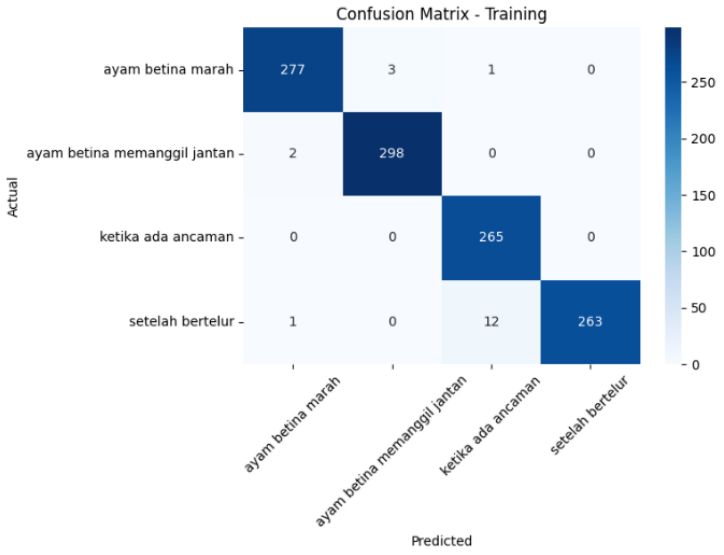
Subset	**Akurasi**
Pelatihan	98.31%
Validasi	99.29%
Pengujian	93.33%

Percobaan ketiga menunjukkan performa model yang sangat baik. Akurasi pelatihan dan validasi sangat tinggi, begitu juga akurasi pengujian sebesar 93.33%.

Tabel 4.32 Metrik Evaluasi per Kelas pada Percobaan 3 *Transformer*

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	98.93%	99.00%	95.32%	100.00%
	Validasi	98.61%	100.00%	98.51%	100.00%
	Pengujian	83.33%	100.00%	93.75%	100.00%
Recall	Pelatihan	98.58%	99.33%	100.00%	95.29%
	Validasi	100.00%	98.67%	100.00%	98.55%
	Pengujian	100.00%	73.33%	100.00%	100.00%
F1-score	Pelatihan	98.75%	99.17%	97.61%	97.59%
	Validasi	99.30%	99.33%	99.25%	99.27%
	Pengujian	90.91%	84.62%	96.67%	100.00%

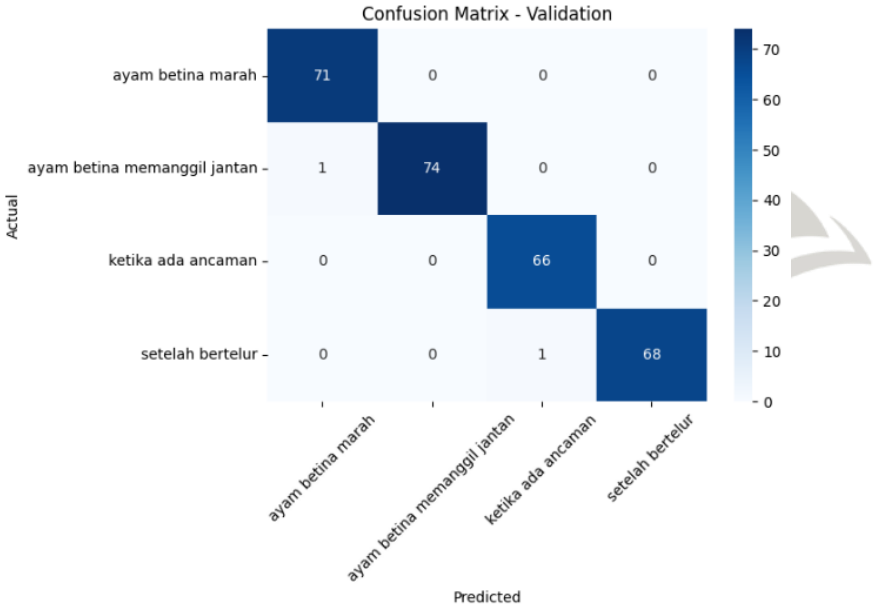
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (83.33%) dan *recall* pengujian kelas ayam betina memanggil jantan (73.33%). Hal ini menyebabkan *F1-score* juga menurun pada kedua kelas tersebut, terutama kelas ayam betina memanggil jantan (*overfitting*).



Gambar 4.47 *Confusion matrix* pelatihan *Transformer* percobaan 3

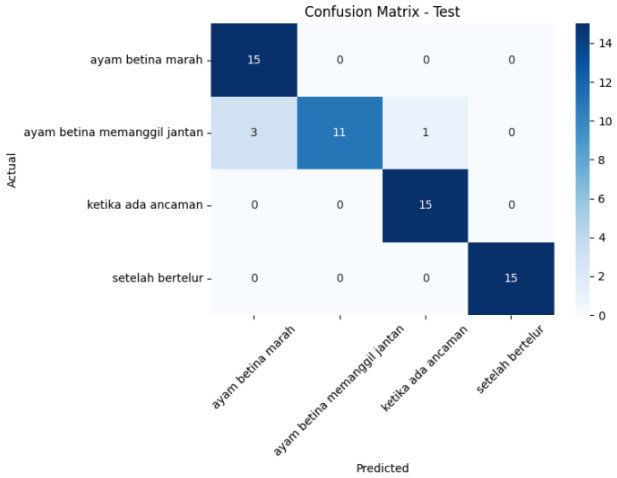
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 277 data benar diprediksi sebagai ayam betina marah, 3 data salah

diprediksi sebagai ayam betina memanggil jantan, dan 1 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, ada 298 data benar diprediksi sebagai ayam betina memanggil jantan dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 263 data benar diprediksi sebagai setelah bertelur, 1 data salah diprediksi sebagai ayam betina marah, dan 12 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.48 *Confusion matrix* validasi *Transformer* percobaan 3

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 74 data benar diprediksi sebagai ayam betina memanggil jantan dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.49 *Confusion matrix* pengujian *Transformer* percobaan 3

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 11 data benar diprediksi sebagai ayam betina memanggil jantan, 3 data salah diprediksi sebagai ayam betina marah, dan 1 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan ketiga ini sebesar 2.32 MB, lebih kecil dibanding percobaan kedua (3.72 MB).

4.4.4. Percobaan 4

Pada percobaan keempat, model *Transformer* dikonfigurasi dengan ukuran *patch* 10, dimensi *embedding* 96, jumlah *head* 6, jumlah *encode layer* 3, dan dimensi MLP 256.

Tabel 4.33 Akurasi Model pada Percobaan 4 *Transformer*

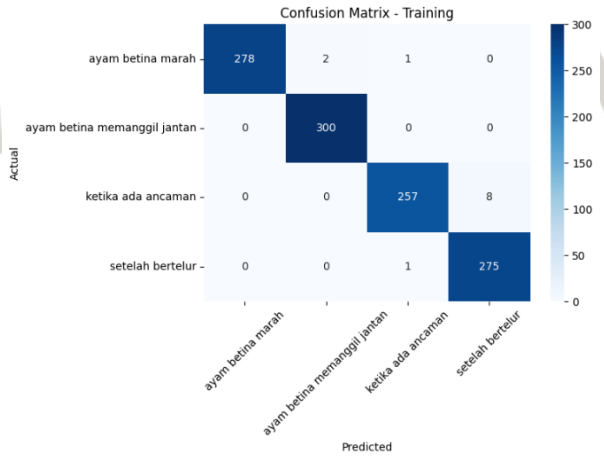
Subset	**Akurasi**
Pelatihan	98.93%
Validasi	98.93%
Pengujian	96.67%

Percobaan keempat menunjukkan performa model yang sangat baik. Akurasi pelatihan dan validasi sangat tinggi, akurasi pengujian sebesar 96.67% mengindikasikan model mampu melakukan generalisasi dengan baik tanpa mengalami *overfitting* yang parah.

Tabel 4.34 Metrik Evaluasi per Kelas pada Percobaan 4 *Transformer*

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	99.34%	99.23%	97.17%
	Validasi	100.00%	100.00%	98.46%	97.14%
	Pengujian	100.00%	100.00%	100.00%	88.24%
Recall	Pelatihan	98.93%	100.00%	96.98%	99.64%
	Validasi	100.00%	100.00%	96.97%	98.55%
	Pengujian	100.00%	100.00%	86.67%	100.00%
F1-score	Pelatihan	99.46%	99.67%	98.09%	98.39%
	Validasi	100.00%	100.00%	97.71%	97.84%
	Pengujian	100.00%	100.00%	92.86%	93.75%

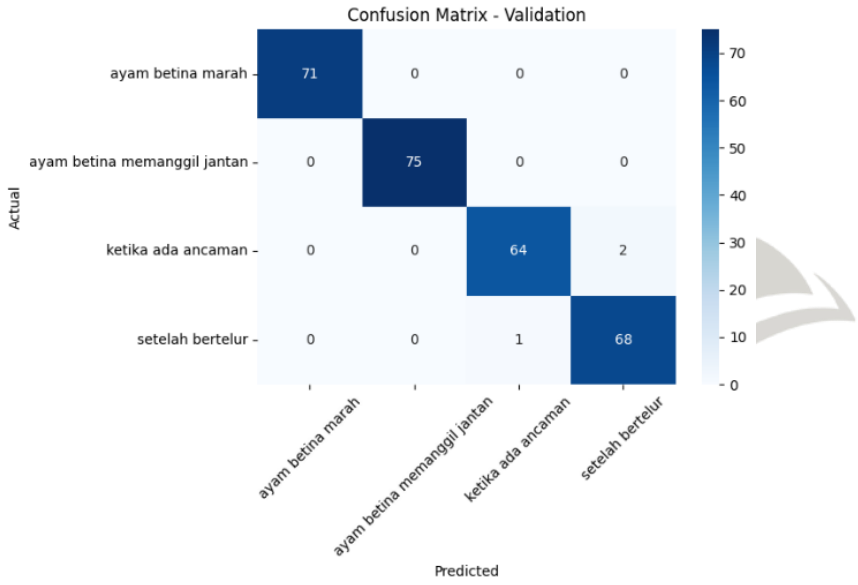
Terdapat sedikit penurunan *recall* pengujian kelas ketika ada ancaman (86.67%), namun masih dalam rentang wajar.



Gambar 4.50 *Confusion matrix* pelatihan *Transformer* percobaan 4

Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 278 data benar diprediksi sebagai ayam betina marah, 2 data salah

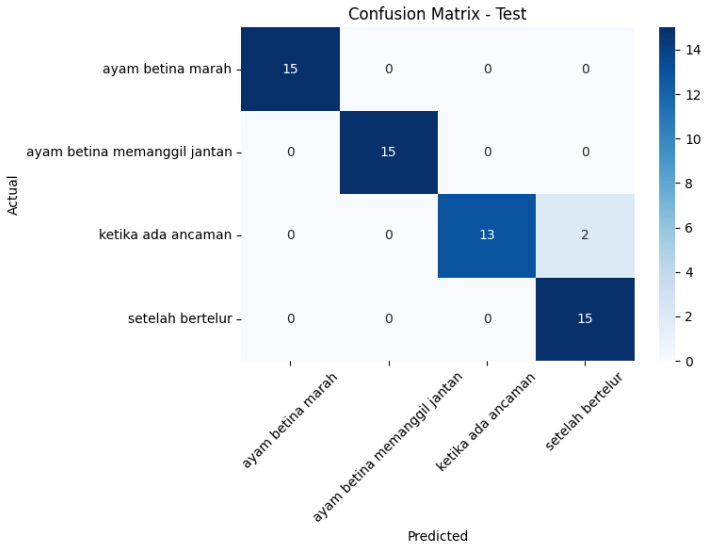
diprediksi sebagai ayam betina memanggil jantan, dan 1 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 257 data benar diprediksi sebagai ketika ada ancaman dan 8 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, ada 275 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.51 *Confusion matrix* validasi *Transformer* percobaan 4

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah dan ayam betina memanggil jantan, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 64 data benar diprediksi sebagai ketika ada ancaman dan 2 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.

Secara keseluruhan, model menunjukkan akurasi yang sangat tinggi pada data validasi, dengan sebagian besar data diklasifikasikan dengan tepat. Kesalahan prediksi hanya terjadi dalam jumlah yang sangat kecil dan terbatas pada dua kelas, yaitu kelas ketika ada ancaman dan setelah bertelur. Jumlah kesalahan yang minim ini menunjukkan bahwa model memiliki konsistensi yang baik dalam mengenali pola dari masing-masing jenis vokalisasi ayam.



Gambar 4.52 *Confusion matrix* pengujian *Transformer* percobaan 4

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah dan ayam betina memanggil jantan, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 13 data benar diprediksi sebagai ketika ada ancaman dan 2 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan keempat ini sebesar 3.78 MB, lebih besar dibanding percobaan ketiga (2.32 MB).

4.4.5. Percobaan 5

Pada percobaan kelima, model *Transformer* dikonfigurasi dengan ukuran *patch* 10, dimensi *embedding* 96, jumlah *head* 4, jumlah *encode layer* 3, dan dimensi MLP 256.

Tabel 4.35 Akurasi Model pada Percobaan 5 *Transformer*

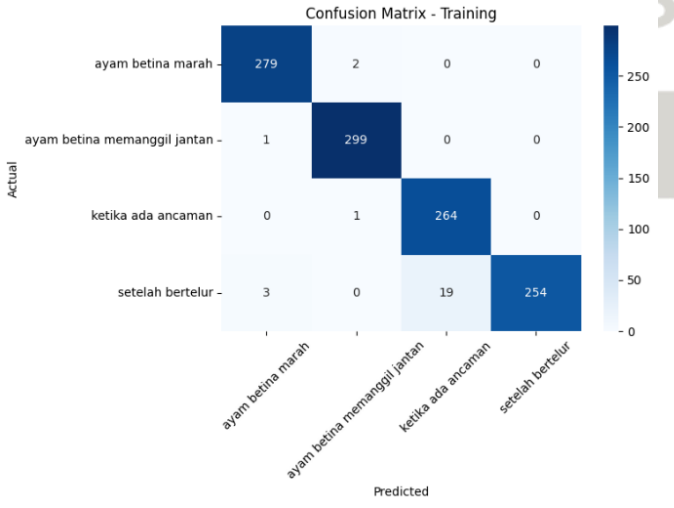
Subset	**Akurasi**
Pelatihan	97.68%
Validasi	98.58%
Pengujian	93.33%

Percobaan kelima menunjukkan performa model yang sangat baik. Akurasi pelatihan dan validasi sangat tinggi, akurasi pengujian sebesar 93.33% mengindikasikan model mampu melakukan generalisasi dengan baik tanpa mengalami *overfitting* yang parah.

Tabel 4.36 Metrik Evaluasi per Kelas pada Percobaan 5 *Transformer*

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	98.59%	99.01%	93.29%	100.00%
	Validasi	100.00%	97.40%	97.06%	100.00%
	Pengujian	78.95%	100.00%	100.00%	100.00%
Recall	Pelatihan	99.29%	99.67%	99.62%	92.03%
	Validasi	97.18%	100.00%	100.00%	97.10%
	Pengujian	100.00%	93.33%	80.00%	100.00%
F1-score	Pelatihan	98.94%	99.34%	96.35%	95.85%
	Validasi	98.57%	98.68%	98.51%	98.53%
	Pengujian	88.24%	96.55%	88.89%	100.00%

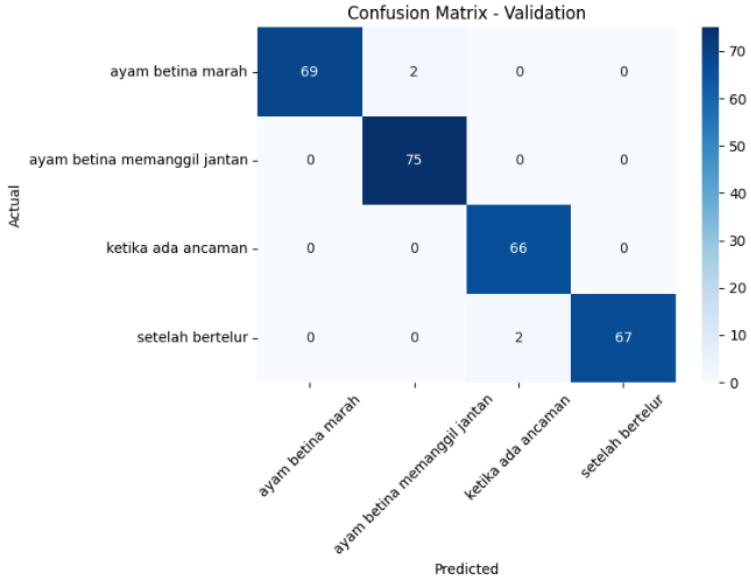
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (78.95%) dan *recall* pengujian kelas ketika ada ancaman (80%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada kedua kelas tersebut.



Gambar 4.53 *Confusion matrix* pelatihan *Transformer* percobaan 5

Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 279 data benar diprediksi sebagai ayam betina marah dan 2 data salah

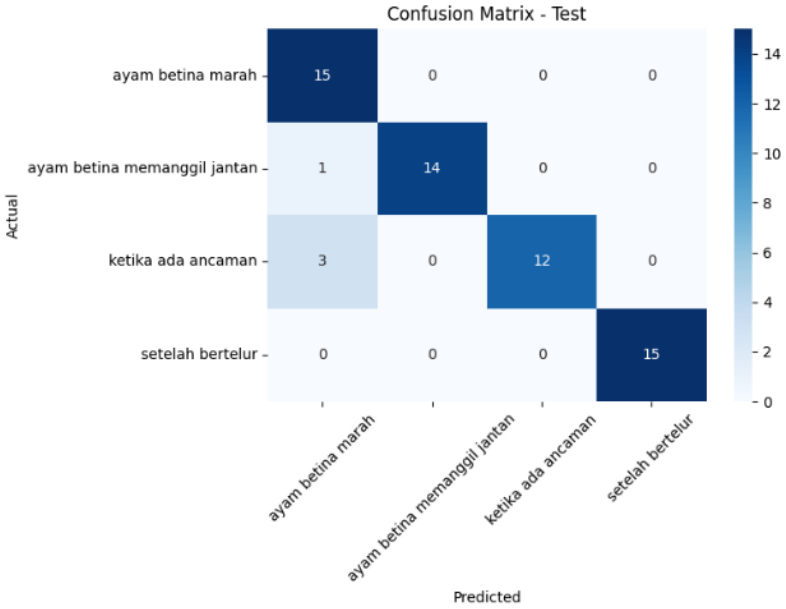
diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan, ada 299 data benar diprediksi sebagai ayam betina memanggil jantan dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 264 data benar diprediksi sebagai ketika ada ancaman dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi setelah bertelur, ada 254 data benar diprediksi sebagai setelah bertelur, 3 data salah diprediksi sebagai ayam betina marah, dan 19 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.54 *Confusion matrix* validasi *Transformer* percobaan 5

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 69 data benar diprediksi sebagai ayam betina marah dan 2 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan dan ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 67 data benar diprediksi sebagai setelah bertelur dan 2 data salah diprediksi sebagai ketika ada ancaman.

Secara keseluruhan, hasil validasi sudah menunjukkan bahwa model mampu memprediksi hampir semua data vokalisasi dengan benar. Terdapat sedikit kesalahan prediksi pada kelas ayam betina marah dan setelah bertelur.



Gambar 4.55 *Confusion matrix* pengujian *Transformer* percobaan 5

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 14 data benar diprediksi sebagai ayam betina memanggil jantan dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 12 data benar diprediksi sebagai ketika ada ancaman dan 3 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan kelima ini sebesar 3.78 MB, sama dengan percobaan keempat.

4.4.6. Percobaan 6

Pada percobaan keenam, model *Transformer* dikonfigurasi dengan ukuran *patch* 10, dimensi *embedding* 96, jumlah *head* 6, jumlah *encode layer* 2, dan dimensi MLP 256.

Tabel 4.37 Akurasi Model pada Percobaan 6 *Transformer*

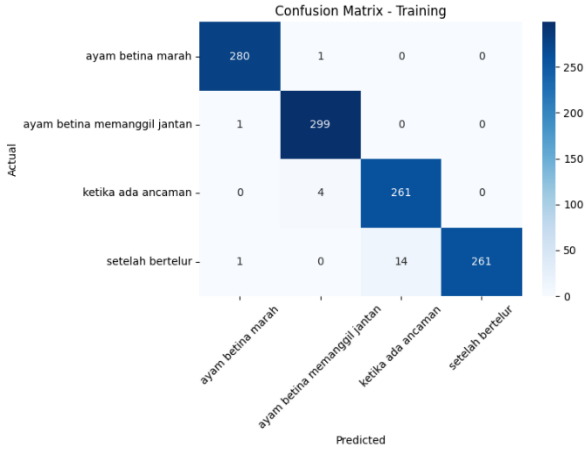
Subset	**Akurasi**
Pelatihan	98.13%
Validasi	98.93%
Pengujian	95.00%

Percobaan keenam menunjukkan performa model yang sangat baik. Akurasi pelatihan dan validasi sangat tinggi, akurasi pengujian sebesar 95% mengindikasikan model mampu melakukan generalisasi dengan baik tanpa mengalami *overfitting* yang parah.

Tabel 4.38 Metrik Evaluasi per Kelas pada Percobaan 6 *Transformer*

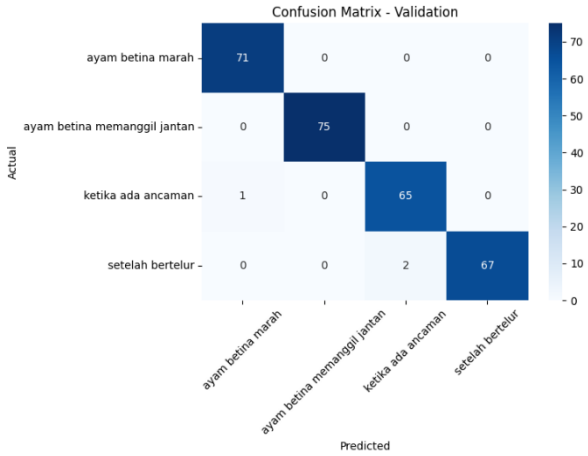
Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	99.29%	98.36%	94.91%	100.00%
	Validasi	98.61%	100.00%	97.01%	100.00%
	Pengujian	83.33%	100.00%	100.00%	100.00%
Recall	Pelatihan	99.64%	99.67%	98.49%	94.57%
	Validasi	100.00%	100.00%	98.48%	97.10%
	Pengujian	100.00%	100.00%	80.00%	100.00%
F1-score	Pelatihan	99.47%	99.01%	96.67%	97.21%
	Validasi	99.30%	100.00%	97.74%	98.53%
	Pengujian	90.91%	100.00%	88.89%	100.00%

Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (83.33%) dan *recall* pengujian kelas ketika ada ancaman (80%). Hal ini menyebabkan *F1-score* juga menurun pada kedua kelas tersebut, tapi tidak sampai *overfitting*.



Gambar 4.56 *Confusion matrix* pelatihan *Transformer* percobaan 6

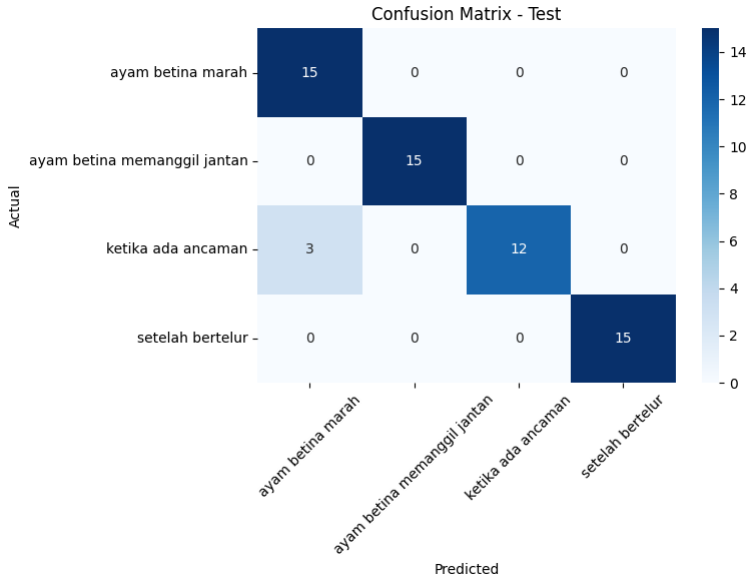
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 280 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan, ada 299 data benar diprediksi sebagai ayam betina memanggil jantan dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 261 data benar diprediksi sebagai ketika ada ancaman dan 4 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi setelah bertelur, ada 261 data benar diprediksi sebagai setelah bertelur, 1 data salah diprediksi sebagai ayam betina marah, dan 14 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.57 *Confusion matrix* validasi *Transformer* percobaan 6

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah dan ayam betina memanggil jantan, semua data vokalisasi ayam berhasil

diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 65 data benar diprediksi sebagai ketika ada ancaman dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi setelah bertelur, ada 67 data benar diprediksi sebagai setelah bertelur dan 2 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.58 *Confusion matrix* pengujian *Transformer* percobaan 6

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah dan ayam betina memanggil jantan, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 12 data benar diprediksi sebagai ketika ada ancaman dan 3 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan keenam ini sebesar 2.72 MB, lebih kecil dibanding percobaan kelima (3.78 MB).

4.4.7. Percobaan 7

Pada percobaan ketujuh, model *Transformer* dikonfigurasi dengan ukuran *patch* 10, dimensi *embedding* 96, jumlah *head* 6, jumlah *encode layer* 3, dan dimensi MLP 2 kali dimensi *embedding*.

Tabel 4.39 Model pada Percobaan 7 *Transformer*

Subset	**Akurasi**
Pelatihan	99.55%
Validasi	98.93%
Pengujian	95.00%

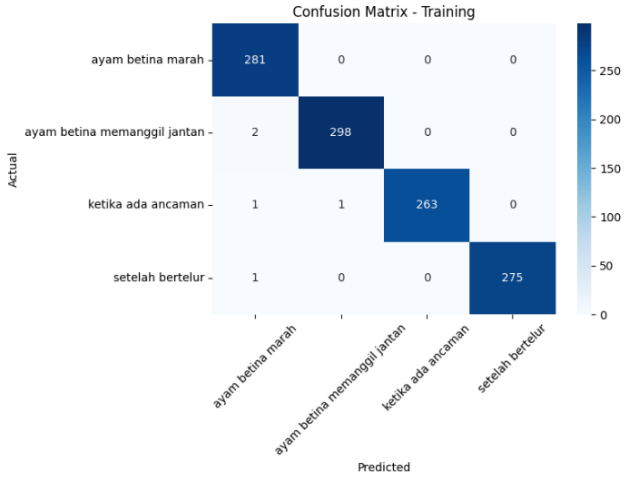
Pada percobaan ketujuh menunjukkan performa yang sangat tinggi, dengan selisih kecil pada akurasi pengujian tapi tidak menunjukkan tanda *overfitting*.

Tabel 4.40 Metrik Evaluasi per Kelas pada Percobaan 7 *Transformer*

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	98.60%	99.67%	100.00%	100.00%
	Validasi	97.26%	100.00%	98.51%	100.00%
	Pengujian	83.33%	100.00%	100.00%	100.00%
Recall	Pelatihan	100.00%	99.33%	99.25%	99.64%
	Validasi	100.00%	97.33%	100.00%	98.55%
	Pengujian	100.00%	80.00%	100.00%	100.00%
F1-score	Pelatihan	99.29%	99.50%	99.62%	99.82%
	Validasi	98.61%	98.65%	99.25%	99.27%
	Pengujian	90.91%	88.89%	100.00%	100.00%

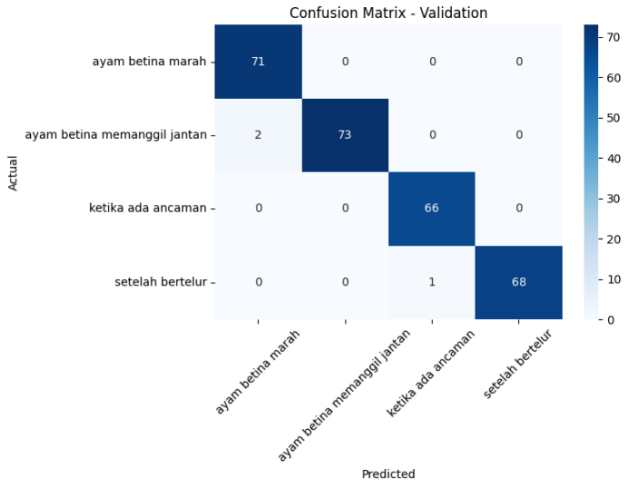
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (83.33%) dan *recall* pengujian kelas ayam betina memanggil jantan (80%). Hal ini menyebabkan *F1-score* juga menurun pada kedua kelas tersebut, mendekati *overfitting*.

Secara keseluruhan, performa model sudah cukup baik dan stabil. Hal ini dapat dilihat dari rata-rata nilainya di atas 90%.



Gambar 4.59 *Confusion matrix* pelatihan *Transformer* percobaan 7

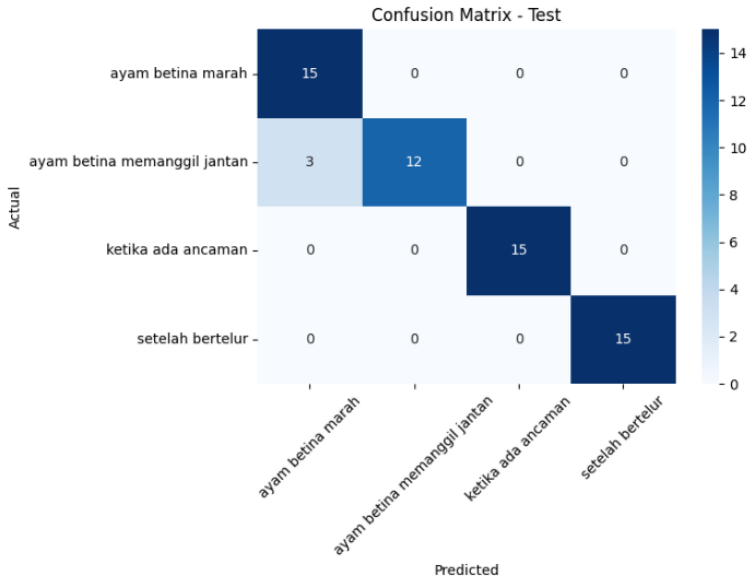
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, semua data vokaliasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 298 data benar diprediksi sebagai ayam betina memanggil jantan dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 263 data benar diprediksi sebagai ketika ada ancaman, 1 data salah diprediksi sebagai ayam betina marah, dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi setelah bertelur, ada 275 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ayam betina marah.



Gambar 4.60 *Confusion matrix* validasi *Transformer* percobaan 7

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, semua data vokaliasi ayam berhasil diprediksi dengan benar. Pada vokalisasi

ayam betina memanggil jantan, ada 73 data benar diprediksi sebagai ayam betina memanggil jantan dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, semua data vokaliasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.61 *Confusion matrix* pengujian *Transformer* percobaan 7

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data vokaliasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ayam betina memanggil jantan, ada 12 data benar diprediksi sebagai ayam betina memanggil jantan dan 3 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data vokaliasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan ketujuh ini sebesar 3.35 MB, lebih besar dibanding percobaan keenam (2.72 MB).

4.4.8. Percobaan 8

Pada percobaan kedelapan, model *Transformer* dikonfigurasi dengan ukuran *patch* 10, dimensi *embedding* 96, jumlah *head* 8, jumlah *encode layer* 3, dan dimensi MLP 2 kali dimensi *embedding*.

Tabel 4.41 Akurasi Model pada Percobaan 8 *Transformer*

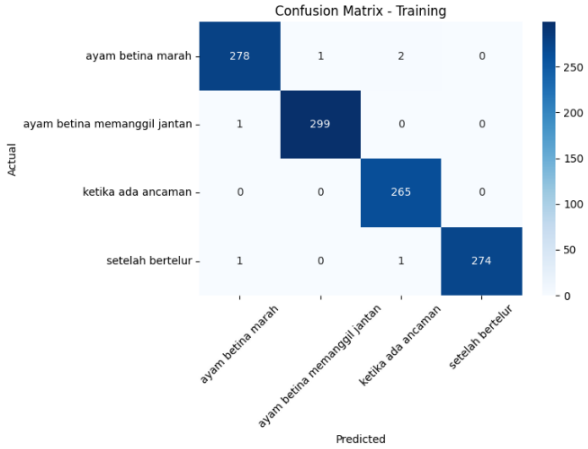
Subset	**Akurasi**
Pelatihan	99.47%
Validasi	98.93%
Pengujian	100.00%

Percobaan kedelapan menunjukkan performa model yang sangat baik. Akurasi pelatihan dan validasi sangat tinggi, begitu juga akurasi pengujian sebesar 100%.

Tabel 4.42 Metrik Evaluasi per Kelas pada Percobaan 8 *Transformer*

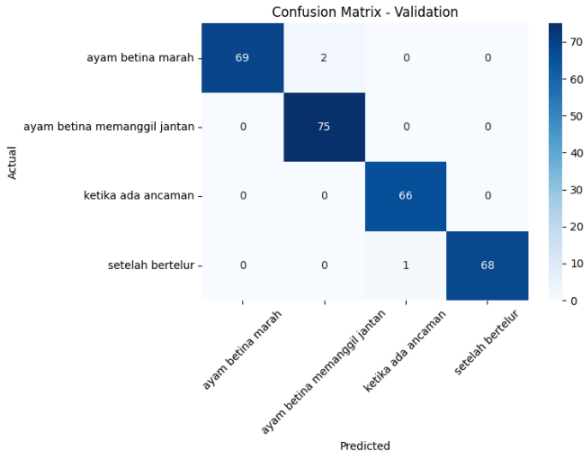
Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	99.29%	99.67%	98.88%	100.00%
	Validasi	100.00%	97.40%	98.51%	100.00%
	Pengujian	100.00%	100.00%	100.00%	100.00%
Recall	Pelatihan	98.93%	99.67%	100.00%	99.28%
	Validasi	97.18%	100.00%	100.00%	98.55%
	Pengujian	100.00%	100.00%	100.00%	100.00%
F1-score	Pelatihan	99.11%	99.67%	99.44%	99.64%
	Validasi	98.57%	98.68%	99.25%	99.27%
	Pengujian	100.00%	100.00%	100.00%	100.00%

Secara umum, metrik evaluasi model menunjukkan hasil yang sangat baik dengan rata-rata nilai mendekati 100%. Penurunan nilai hanya terjadi sedikit pada data pelatihan dan validasi, dengan nilai terendah sebesar 97.18%, yang masih tergolong sangat tinggi.



Gambar 4.62 *Confusion matrix* pelatihan *Transformer* percobaan 8

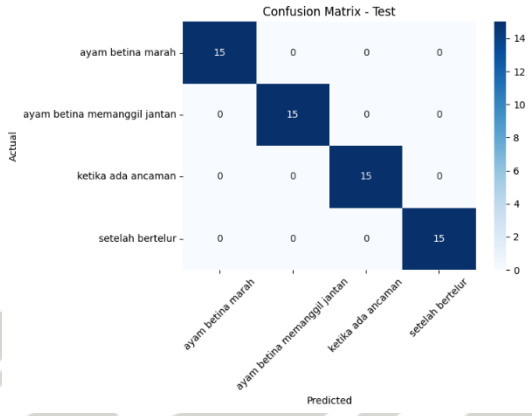
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 278 data benar diprediksi sebagai ayam betina marah, 1 data salah diprediksi sebagai ayam betina memanggil jantan, dan 2 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, ada 299 data benar diprediksi sebagai ayam betina memanggil jantan dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 274 data benar diprediksi sebagai setelah bertelur, 1 data salah diprediksi sebagai ayam betina marah, dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.63 *Confusion matrix* validasi *Transformer* percobaan 8

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 69 data benar diprediksi sebagai ayam betina marah dan 2 data salah

diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan dan ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.64 *Confusion matrix* pengujian *Transformer* percobaan 8

Berdasarkan *confusion matrix* hasil pengujian, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan kedelapan ini sebesar 3.35 MB, sama dengan percobaan ketujuh. Hasil percobaan 8 inilah yang akan digunakan untuk perbandingan dengan metode *deep learning* lainnya.

4.5. Ringkasan Hasil Seluruh Percobaan *Transformer*

Berdasarkan hasil tiap percobaan yang sudah dijabarkan, dilakukan analisis untuk menilai apakah suatu model mengalami *overfitting*, *underfitting*, atau menunjukkan performa yang baik. Ringkasan kondisi dari masing-masing percobaan disajikan dalam Tabel 4.43.

Tabel 4.43 Ringkasan Kondisi Tiap Percobaan *Transformer* Berdasarkan Akurasi

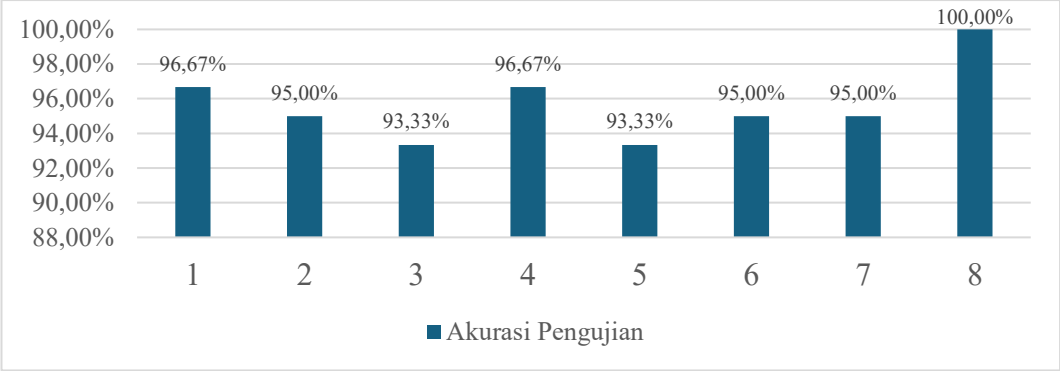
Percobaan	Akurasi Pelatihan	Akurasi Validasi	Akurasi Pengujian	Keterangan Kondisi
Percobaan 1	98.22%	98.58%	96.67%	Bagus

Tabel 4.43 (Lanjutan)

Percobaan	Akurasi Pelatihan	Akurasi Validasi	Akurasi Pengujian	Keterangan Kondisi
Percobaan 2	99.55%	99.29%	95.00%	Bagus
Percobaan 3	98.31%	99.29%	93.33%	Bagus
Percobaan 4	98.93%	98.93%	96.67%	Bagus
Percobaan 5	97.68%	98.58%	93.33%	Bagus
Percobaan 6	98.13%	98.93%	95.00%	Bagus
Percobaan 7	99.55%	98.93%	95.00%	Bagus
Percobaan 8	99.47%	98.93%	100.00%	Bagus

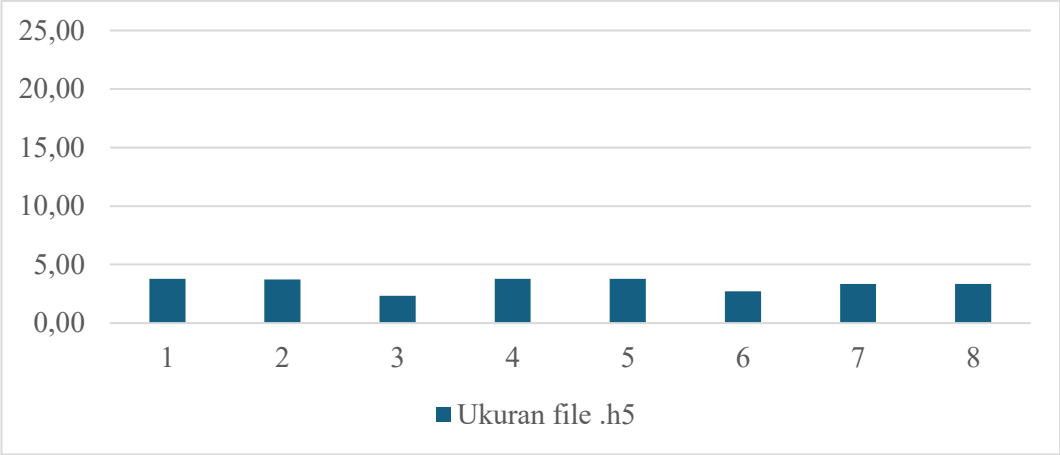
Berbeda dengan metode CNN yang menunjukkan variasi performa antar percobaan, seluruh percobaan pada model *Transformer* menunjukkan hasil yang konsisten dan baik. Hal ini ditunjukkan oleh nilai akurasi pelatihan, validasi, dan pengujian yang tinggi dan seimbang, tanpa indikasi *underfitting* maupun *overfitting* yang signifikan. Dengan demikian, seluruh percobaan dinilai memenuhi kriteria model yang layak dibandingkan.

Berikut grafik ringkasan hasil akurasi pengujian dari seluruh percobaan *Transformer*:



Gambar 4.65 Grafik hasil akurasi pengujian seluruh percobaan *Transformer*

Gambar 4.65 menunjukkan pola grafik akurasi pengujian yang naik turun, mirip seperti CNN. Hal ini disebabkan oleh pendekatan eksploratif yang dilakukan dalam proses mengubah konfigurasi parameter di tiap percobaan.



Gambar 4.66 Grafik hasil ukuran *file* .h5 seluruh percobaan *Transformer*

Gambar 4.66 menunjukkan pola grafik besar dan kecilnya ukuran *file* .h5 yang dihasilkan di tiap percobaan. Jika dilihat dan dibandingkan bersama pola grafik hasil akurasi pengujian, maka percobaan kedelapan menunjukkan hasil terbaik karena akurasinya 100% dan ukuran *file* .h5 kecil. Hal ini memperkuat pengambilan keputusan untuk menggunakan hasil percobaan kedelapan sebagai hasil terbaik dari *Transformer*.

4.6. *Deepspeech*

Pada bagian ini disajikan hasil pelatihan dan pengujian model *Deepspeech* setiap percobaan yang telah dilakukan. Setiap percobaan dilakukan dengan mengubah konfigurasi parameter tertentu yang telah dijelaskan sebelumnya pada Bab 3, tepatnya pada Tabel 3.11. Total terdapat 6 percobaan yang dilakukan, setiap percobaan dibahas dalam subbab tersendiri secara rinci.

4.6.1. Percobaan 1

Pada percobaan pertama, model *Deepspeech* dikonfigurasi dengan jumlah *filter* 128, jumlah unit 128-128-128, 64 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0005. Percobaan ini digunakan sebagai *baseline* awal untuk membandingkan performa dengan konfigurasi pada percobaan lainnya.

Tabel 4.44 Akurasi Model pada Percobaan 1 *Deepspeech*

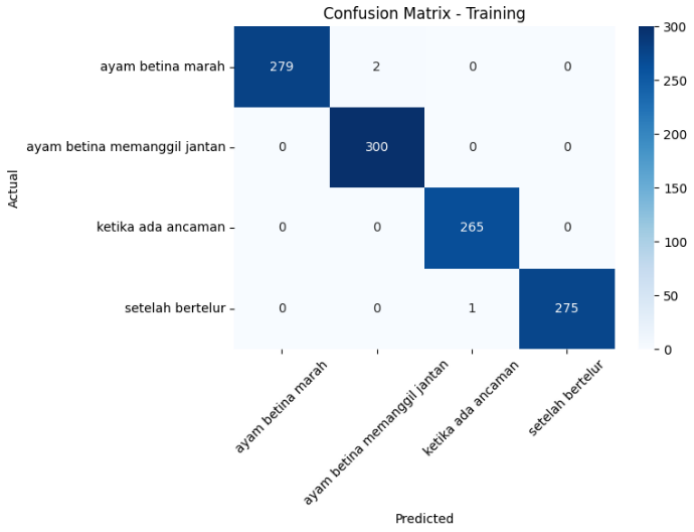
Subset	**Akurasi**
Pelatihan	99.73%
Validasi	98.93%
Pengujian	91.67%

Pada percobaan pertama menunjukkan performa yang sangat tinggi, dengan selisih cukup kecil pada akurasi pengujian tapi tidak menunjukkan tanda *overfitting*.

Tabel 4.45 Metrik Evaluasi per Kelas pada Percobaan 1 *Deepspeech*

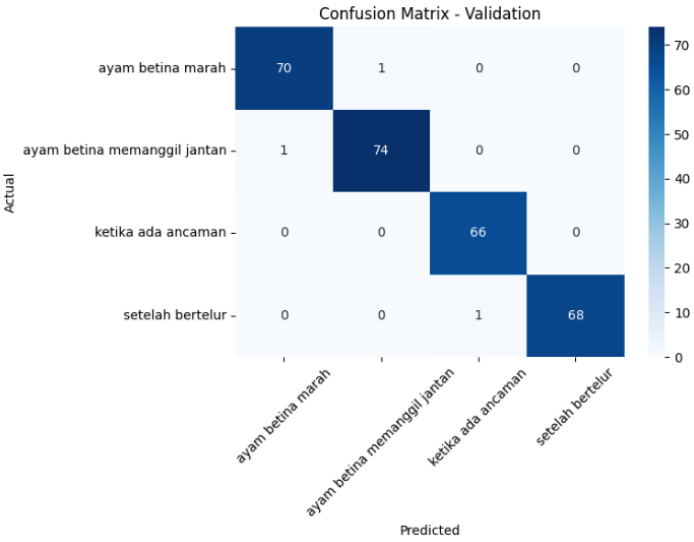
Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	100.00%	99.34%	99.62%	100.00%
	Validasi	98.59%	98.67%	98.51%	100.00%
	Pengujian	82.35%	100.00%	93.75%	93.75%
Recall	Pelatihan	99.29%	100.00%	100.00%	99.64%
	Validasi	98.59%	98.67%	100.00%	98.55%
	Pengujian	93.33%	73.33%	100.00%	100.00%
F1-score	Pelatihan	99.64%	99.67%	99.81%	99.82%
	Validasi	98.59%	98.67%	99.25%	99.27%
	Pengujian	87.50%	84.62%	96.77%	96.77%

Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (82.35%) dan *recall* pengujian kelas ayam betina memanggil jantan (73.33%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada kedua kelas tersebut.



Gambar 4.67 *Confusion matrix* pelatihan *Deepspeech* percobaan 1

Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 279 data benar diprediksi sebagai ayam betina marah dan 2 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan dan ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 275 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.

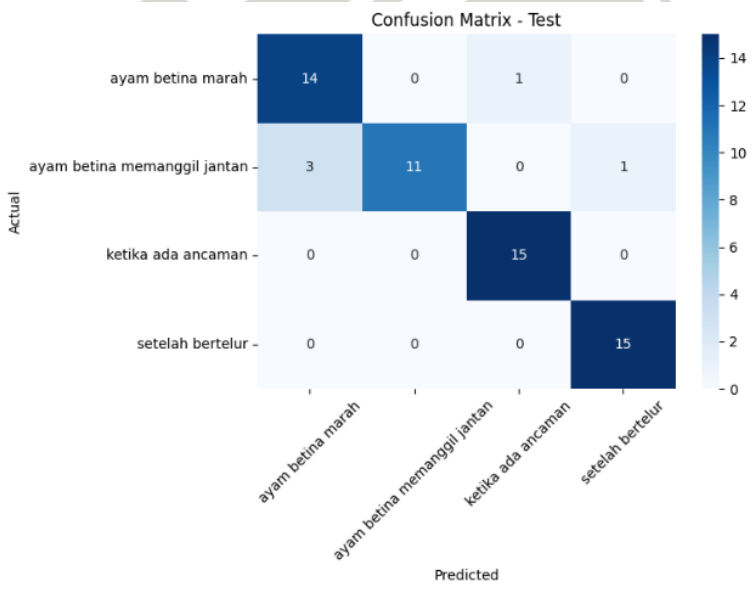


Gambar 4.68 *Confusion matrix* validasi *Deepspeech* percobaan 1

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 70 data benar diprediksi sebagai ayam betina marah dan 1 data salah

diprediksi sebagai ayam betina memanggil jantan. Pada vokalisasi ayam betina memanggil jantan, ada 74 data benar diprediksi sebagai ayam betina memanggil jantan dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.

Secara keseluruhan, jumlah prediksi yang benar lebih dominan dibandingkan prediksi yang salah. Kesalahan hanya terjadi pada satu atau dua data pada masing-masing kelas. Masing-masing pada kelas ayam betina marah, ayam betina memanggil jantan, dan setelah bertelur hanya terdapat 1 prediksi yang salah.



Gambar 4.69 *Confusion matrix* pengujian *Deepspeech* percobaan 1

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, ada 14 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ketika ada ancaman. Pada vokalisasi ayam betina memanggil jantan, ada 11 data benar diprediksi sebagai ayam betina memanggil jantan, 3 data salah diprediksi sebagai ayam betina marah, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan pertama ini sebesar 12.4 MB, tergolong sedang untuk *deployment*.

4.6.2. Percobaan 2

Pada percobaan kedua, model *Deepspeech* dikonfigurasi dengan jumlah *filter* 64, jumlah unit 64-64-128, 32 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0005.

Tabel 4.46 Akurasi Model pada Percobaan 2 *Deepspeech*

Subset	**Akurasi**
Pelatihan	99.91%
Validasi	99.29%
Pengujian	85.00%

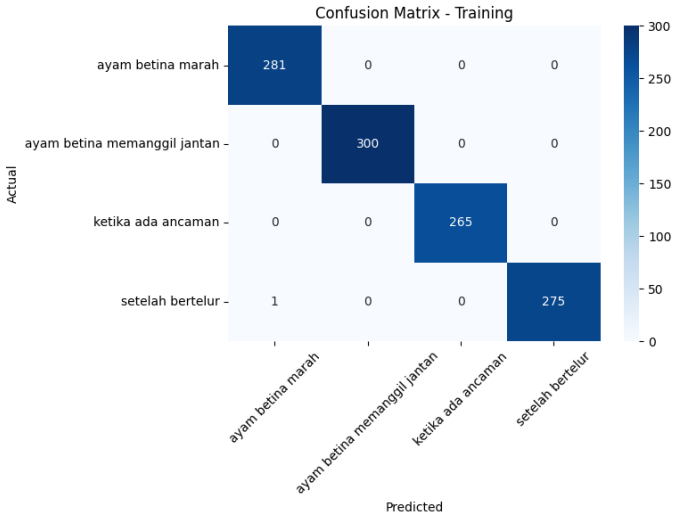
Model menunjukkan akurat sangat tinggi pada data pelatihan dan validasi, namun mengalami penurunan drastis pada data pengujian dengan akurasi 85.00%.

Tabel 4.47 Metrik Evaluasi per Kelas pada Percobaan 2 *Deepspeech*

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	99.65%	100.00%	100.00%	100.00%
	Validasi	100.00%	100.00%	98.51%	98.55%
	Pengujian	80.00%	90.00%	78.95%	93.75%
Recall	Pelatihan	100.00%	100.00%	100.00%	99.64%
	Validasi	98.59%	100.00%	100.00%	98.55%
	Pengujian	80.00%	60.00%	100.00%	100.00%
F1-score	Pelatihan	99.82%	100.00%	100.00%	99.82%
	Validasi	99.29%	100.00%	99.25%	98.55%
	Pengujian	80.00%	72.00%	88.24%	96.77%

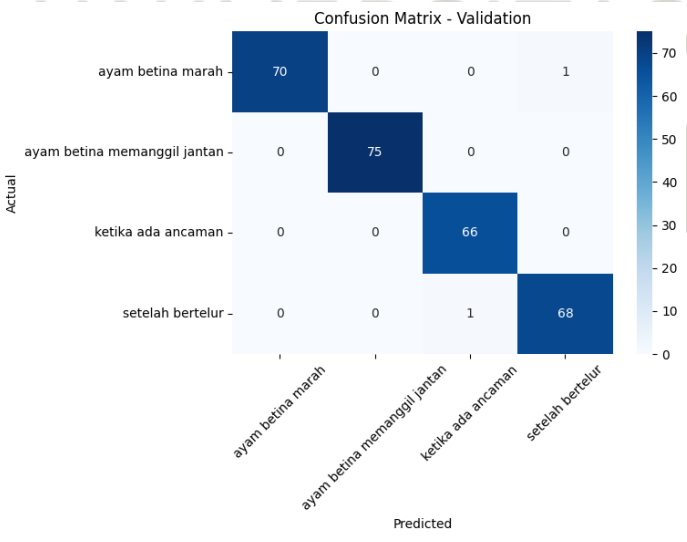
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (80%), ayam betina memanggil jantan (90%), dan ketika ada ancaman (78.95%). Terdapat pula penurunan *recall* pengujian kelas ayam betina marah (80%) dan ayam

betina memanggil jantan (60%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada ketiga kelas tersebut.



Gambar 4.70 *Confusion matrix* pelatihan *Deepspeech* percobaan 2

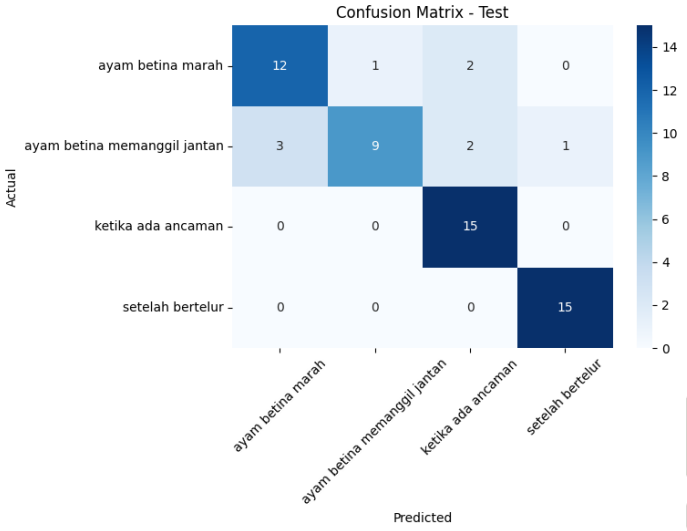
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ayam betina memanggil jantan, dan ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 275 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ayam betina marah.



Gambar 4.71 *Confusion matrix* validasi *Deepspeech* percobaan 2

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 70 data benar diprediksi sebagai ayam betina marah dan 1 data salah

diprediksi sebagai setelah bertelur. Pada vokaliasi ayam betina memanggil jantan dan ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.72 *Confusion matrix* pengujian *Deepspeech* percobaan 2

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, ada 12 data benar diprediksi sebagai ayam betina marah, 1 data salah diprediksi sebagai ayam betina memanggil jantan, dan 2 data salah diprediksi sebagai setelah bertelur. Pada vokaliasi ayam betina memanggil jantan, ada 9 data benar diprediksi sebagai ayam betina memanggil jantan, 3 data salah diprediksi sebagai ayam betina marah, 2 data salah diprediksi sebagai ketika ada ancaman, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan kedua ini sebesar 5.17 MB, lebih kecil dibanding percobaan pertama (12.4 MB).

4.6.3. Percobaan 3

Pada percobaan ketiga, model *Deepspeech* dikonfigurasi dengan jumlah *filter* 128, jumlah unit 128-128-128, 64 unit *dense*, *dropout* sebesar 0.2, dan *learning rate* 0.0005.

Tabel 4.48 Akurasi Model pada Percobaan 3 *Deepspeech*

Subset	**Akurasi**
Pelatihan	99.20%
Validasi	98.93%
Pengujian	90.00%

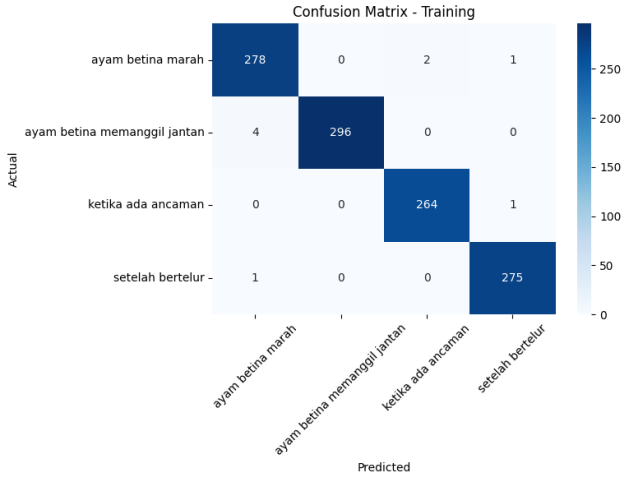
Pada percobaan ketiga menunjukkan performa yang sangat tinggi, dengan selisih cukup kecil pada akurasi pengujian tapi tidak menunjukkan tanda *overfitting*.

Tabel 4.49 Metrik Evaluasi per Kelas pada Percobaan 3 *Deepspeech*

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	98.23%	100.00%	99.25%	99.28%
	Validasi	98.59%	100.00%	97.06%	100.00%
	Pengujian	78.95%	100.00%	93.75%	93.75%
Recall	Pelatihan	98.93%	98.67%	99.62%	99.64%
	Validasi	98.59%	98.67%	100.00%	98.55%
	Pengujian	100.00%	60.00%	100.00%	100.00%
F1-score	Pelatihan	98.58%	99.33%	99.44%	99.46%
	Validasi	98.59%	99.33%	98.51%	99.27%
	Pengujian	88.24%	75.00%	96.77%	96.77%

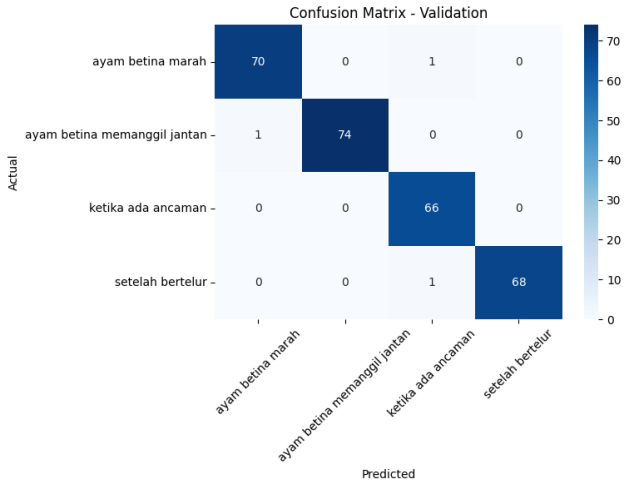
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (78.95%) dan *recall* pengujian kelas ayam betina memanggil jantan (60%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada kedua kelas tersebut.

Meskipun kelas lainnya menunjukkan hasil yang baik pada tahap pengujian, penurunan ini mengindikasikan bahwa model belum mampu mempertahankan performa yang merata di semua kelas, khususnya ketika menghadapi variasi data yang tidak terdapat pada saat pelatihan.



Gambar 4.73 *Confusion matrix* pelatihan *Deepspeech* percobaan 3

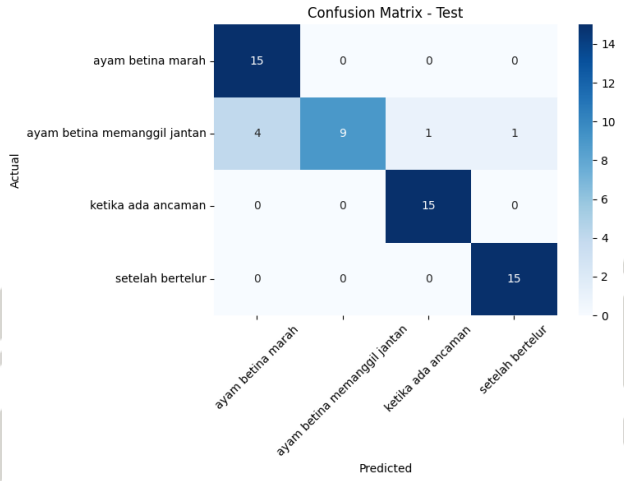
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 278 data benar diprediksi sebagai ayam betina marah, 2 data salah diprediksi sebagai ketika ada ancaman, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokaliasi ayam betina memanggil jantan, ada 296 data benar diprediksi sebagai ayam betina memanggil jantan dan 4 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, ada 264 data benar diprediksi sebagai ketika ada ancaman dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, ada 275 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ayam betina marah.



Gambar 4.74 *Confusion matrix* validasi *Deepspeech* percobaan 3

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 70 data benar diprediksi sebagai ayam betina marah dan 1 data salah

diprediksi sebagai ketika ada ancaman. Pada vokaliasi ayam betina memanggil jantan, ada 74 data benar diprediksi sebagai ayam betina memanggil jantan dan 1 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.75 *Confusion matrix* pengujian *Deepspeech* percobaan 3

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokaliasi ayam betina memanggil jantan, ada 9 data benar diprediksi sebagai ayam betina memanggil jantan, 4 data salah diprediksi sebagai ayam betina marah, 1 data salah diprediksi sebagai ketika ada ancaman, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan ketiga ini sebesar 12.4 MB, lebih besar dibanding percobaan kedua (5.17 MB).

4.6.4. Percobaan 4

Pada percobaan keempat, model *Deepspeech* dikonfigurasi dengan jumlah *filter* 128, jumlah unit 128-128-128, 64 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.0001.

Tabel 4.50 Akurasi Model pada Percobaan 4 *Deepspeech*

Subset	**Akurasi**
Pelatihan	99.55%
Validasi	98.58%
Pengujian	95.00%

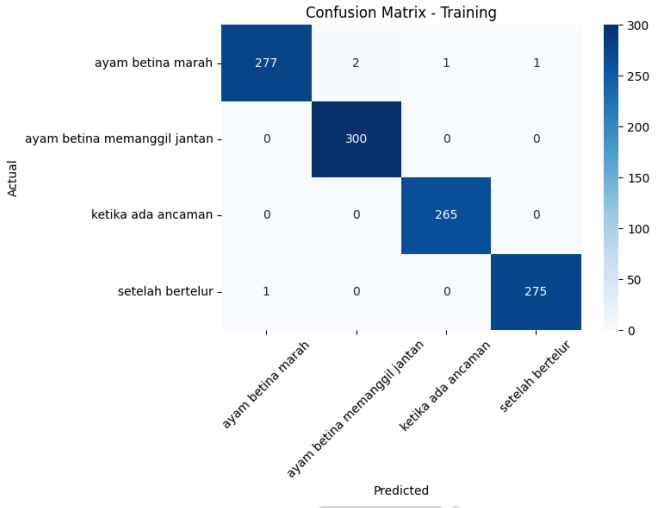
Pada percobaan keempat menunjukkan performa yang sangat tinggi, dengan selisih kecil pada akurasi pengujian.

Tabel 4.51 Metrik Evaluasi per Kelas pada Percobaan 4 *Deepspeech*

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	99.64%	99.34%	99.62%	99.64%
	Validasi	100.00%	98.68%	95.65%	100.00%
	Pengujian	87.50%	92.86%	100.00%	100.00%
Recall	Pelatihan	98.58%	100.00%	100.00%	99.64%
	Validasi	95.77%	100.00%	100.00%	98.55%
	Pengujian	93.33%	86.67%	100.00%	100.00%
F1-score	Pelatihan	99.11%	99.67%	99.81%	99.64%
	Validasi	97.84%	99.34%	97.78%	99.27%
	Pengujian	90.32%	89.66%	100.00%	100.00%

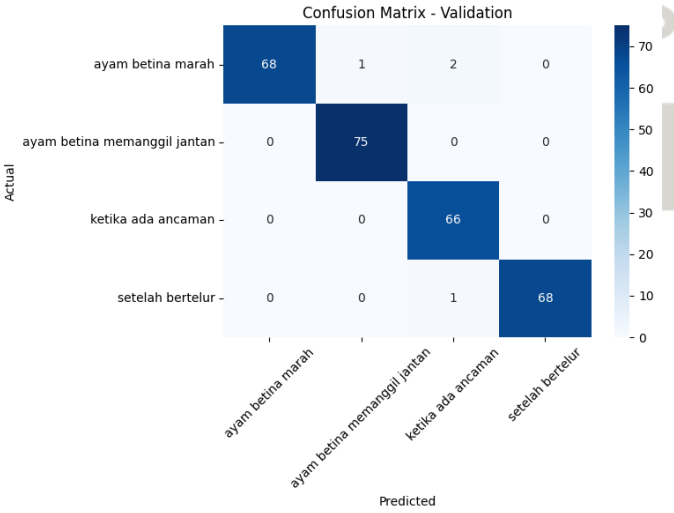
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (87.50%) dan *recall* pengujian kelas ayam betina memanggil jantan (86.67%). Hal ini menyebabkan *F1-score* juga menurun pada kedua kelas tersebut, mendekati *overfitting*.

Secara keseluruhan, performa model dapat dikatakan sudah cukup baik. Nilai *precision*, *recall*, dan *F1-score* pada sebagian besar kelas menunjukkan angka yang tinggi, terutama pada kelas ketika ada ancaman dan setelah bertelur yang mencapai 100% di data pengujian.



Gambar 4.76 *Confusion matrix* pelatihan *Deepspeech* percobaan 4

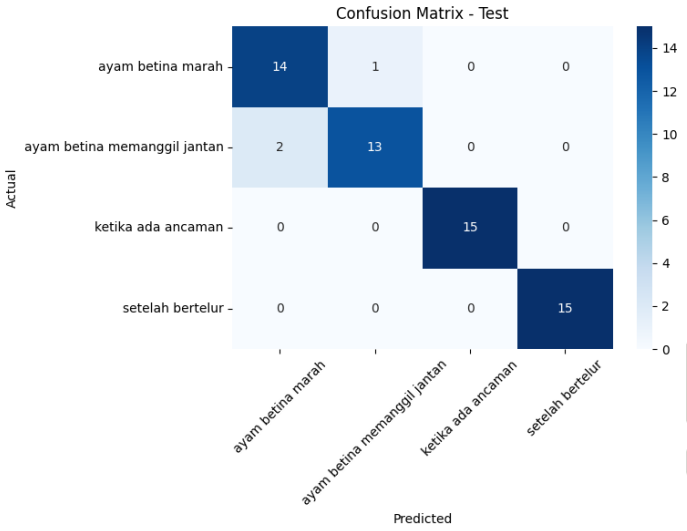
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 277 data benar diprediksi sebagai ayam betina marah, 2 data salah diprediksi sebagai ayam betina memanggil jantan, 1 data salah diprediksi sebagai ketika ada ancaman, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokaliasi ayam betina memanggil jantan dan ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 275 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ayam betina marah.



Gambar 4.77 *Confusion matrix* validasi *Deepspeech* percobaan 4

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 68 data benar diprediksi sebagai ayam betina marah, 1 data salah

diprediksi sebagai ayam betina memanggil jantan, dan 2 data salah diprediksi sebagai ketika ada ancaman. Pada vokaliasi ayam betina memanggil jantan dan ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.78 *Confusion matrix* pengujian *Deepspeech* percobaan 4

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, ada 14 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokaliasi ayam betina memanggil jantan, ada 13 data benar diprediksi sebagai ayam betina memanggil jantan dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan keempat ini sebesar 12.4 MB, sama dengan percobaan ketiga.

4.6.5. Percobaan 5

Pada percobaan kelima, model *Deepspeech* dikonfigurasi dengan jumlah *filter* 128, jumlah unit 128-128-128, 64 unit *dense*, *dropout* sebesar 0.1, dan *learning rate* 0.001.

Tabel 4.52 Akurasi Model pada Percobaan 5 *Deepspeech*

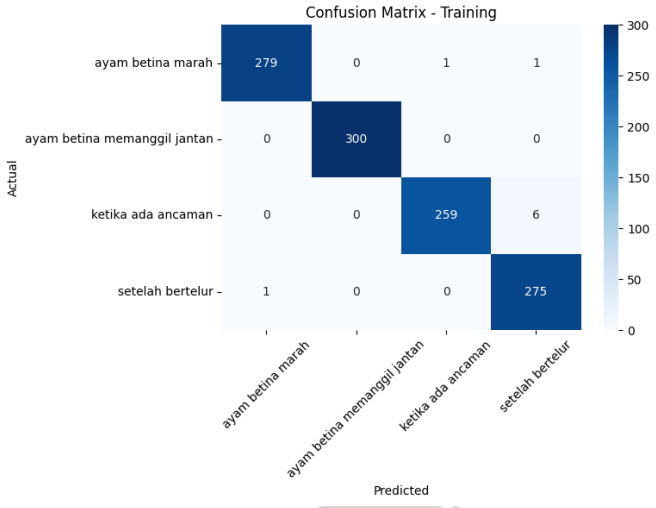
Subset	**Akurasi**
Pelatihan	99.20%
Validasi	99.29%
Pengujian	91.67%

Pada percobaan kelima menunjukkan performa yang sangat tinggi, dengan selisih cukup kecil pada akurasi pengujian.

Tabel 4.53 Metrik Evaluasi per Kelas pada Percobaan 5 *Deepspeech*

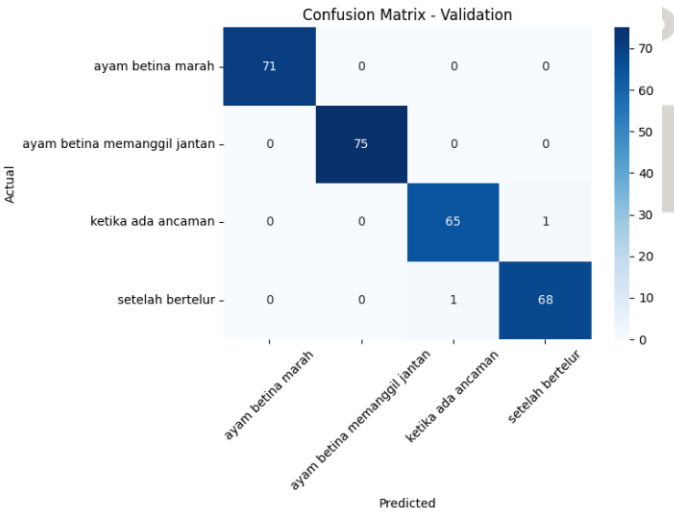
Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	99.64%	100.00%	99.62%	97.52%
	Validasi	100.00%	100.00%	98.48%	98.55%
	Pengujian	81.25%	86.67%	100.00%	100.00%
Recall	Pelatihan	99.29%	100.00%	97.74%	99.64%
	Validasi	100.00%	100.00%	98.48%	98.55%
	Pengujian	86.67%	86.67%	100.00%	93.33%
F1-score	Pelatihan	99.47%	100.00%	98.67%	98.57%
	Validasi	100.00%	100.00%	98.48%	98.55%
	Pengujian	83.87%	86.67%	100.00%	96.55%

Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (81.25%) dan ayam betina memanggil jantan (86.67%). Terdapat pula penurunan *recall* pengujian kelas ayam betina marah (86.67%) dan ayam betina memanggil jantan (86.67%). Hal ini menyebabkan *F1-score* juga menurun (*overfitting*) pada kedua kelas tersebut.



Gambar 4.79 *Confusion matrix* pelatihan *Deepspeech* percobaan 5

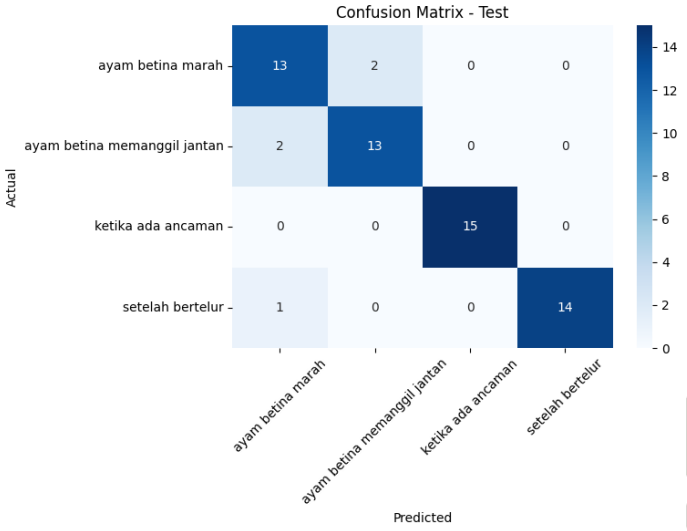
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 279 data benar diprediksi sebagai ayam betina marah, 1 data salah diprediksi sebagai ketika ada ancaman, dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokaliasi ayam betina memanggil jantan, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 259 data benar diprediksi sebagai ketika ada ancaman dan 6 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, ada 275 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ayam betina marah.



Gambar 4.80 *Confusion matrix* validasi *Deepspeech* percobaan 5

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah dan ayam betina memanggil jantan, semua data vokalisasi ayam berhasil

diprediksi dengan benar. Pada vokalisasi ketika ada ancaman, ada 65 data benar diprediksi sebagai ketika ada ancaman dan 1 data salah diprediksi sebagai setelah bertelur. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.81 *Confusion matrix* pengujian *Deepspeech* percobaan 5

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, ada 13 data benar diprediksi sebagai ayam betina marah dan 2 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokaliasi ayam betina memanggil jantan, ada 13 data benar diprediksi sebagai ayam betina memanggil jantan dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 14 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ayam betina marah.

Ukuran *file* model hasil percobaan kelima ini sebesar 12.4 MB, sama dengan percobaan keempat.

4.6.6. Percobaan 6

Pada percobaan keenam, model *Deepspeech* dikonfigurasi dengan jumlah *filter* 128, jumlah unit 128-128-128, 64 unit *dense*, *dropout* sebesar 0.2, dan *learning rate* 0.0001.

Tabel 4.54 Akurasi Model pada Percobaan 6 *Deepspeech*

Subset	**Akurasi**
Pelatihan	99.73%
Validasi	98.58%
Pengujian	95.00%

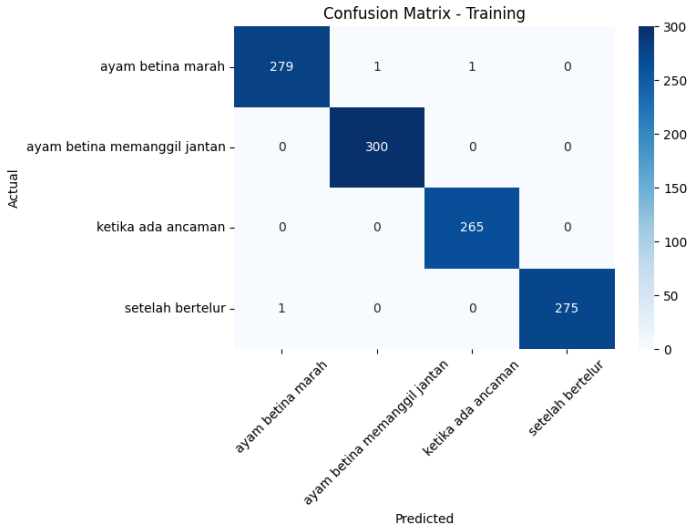
Pada percobaan keenam menunjukkan performa yang sangat tinggi, dengan selisih kecil pada akurasi pengujian.

Tabel 4.55 Metrik Evaluasi per Kelas pada Percobaan 6 *Deepspeech*

Metrik	***Subset***	**Ayam Betina Marah**	**Ayam Betina Memanggil Jantan**	**Ketika Ada Ancaman**	**Setelah Bertelur**
Precision	Pelatihan	99.64%	99.67%	99.62%	100.00%
	Validasi	100.00%	96.15%	98.51%	100.00%
	Pengujian	87.50%	92.86%	100.00%	100.00%
Recall	Pelatihan	99.29%	100.00%	100.00%	99.64%
	Validasi	95.77%	100.00%	100.00%	98.55%
	Pengujian	93.33%	86.67%	100.00%	100.00%
F1-score	Pelatihan	99.47%	99.83%	99.81%	99.82%
	Validasi	97.84%	98.04%	99.25%	99.27%
	Pengujian	90.32%	89.66%	100.00%	100.00%

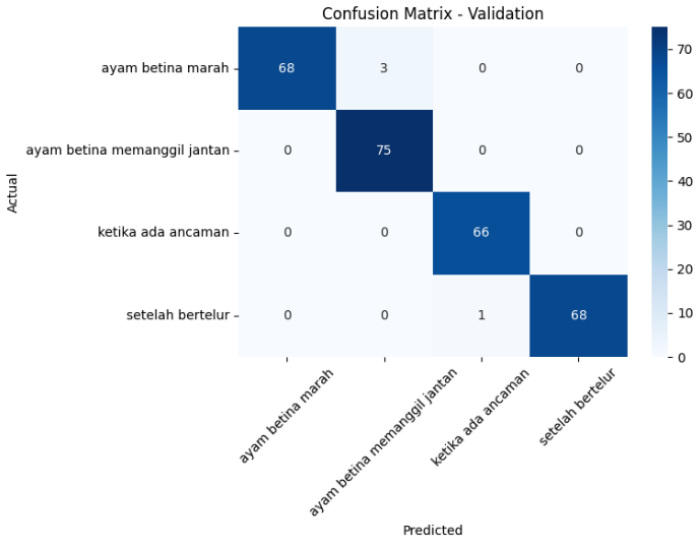
Terdapat penurunan pada *precision* pengujian kelas ayam betina marah (87.50%) dan *recall* pengujian kelas ayam betina memanggil jantan (86.67%). Hal ini menyebabkan *F1-score* juga menurun pada kedua kelas tersebut.

Secara keseluruhan, performa model dapat dikatakan sudah cukup baik. Nilai *precision*, *recall*, dan *F1-score* pada sebagian besar kelas menunjukkan angka yang tinggi, terutama pada kelas ketika ada ancaman dan setelah bertelur yang mencapai 100% di data pengujian.



Gambar 4.82 *Confusion matrix* pelatihan *Deepspeech* percobaan 6

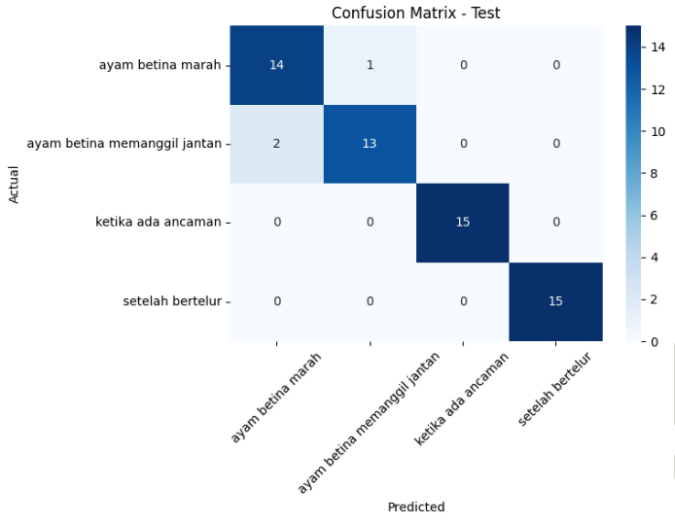
Berdasarkan *confusion matrix* hasil pelatihan, pada vokalisasi ayam betina marah, ada 279 data benar diprediksi sebagai ayam betina marah, 1 data salah diprediksi sebagai ayam betina memanggil jantan, dan 1 data salah diprediksi sebagai ketika ada ancaman. Pada vokaliasi ayam betina memanggil jantan dan ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 275 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ayam betina marah.



Gambar 4.83 *Confusion matrix* validasi *Deepspeech* percobaan 6

Berdasarkan *confusion matrix* hasil validasi, pada vokalisasi ayam betina marah, ada 68 data benar diprediksi sebagai ayam betina marah dan 3 data salah

diprediksi sebagai ayam betina memanggil jantan. Pada vokaliasi ayam betina memanggil jantan dan ketika ada ancaman, semua data vokalisasi ayam berhasil diprediksi dengan benar. Pada vokalisasi setelah bertelur, ada 68 data benar diprediksi sebagai setelah bertelur dan 1 data salah diprediksi sebagai ketika ada ancaman.



Gambar 4.84 *Confusion matrix* pengujian *Deepspeech* percobaan 6

Berdasarkan *confusion matrix* hasil pengujian, pada vokalisasi ayam betina marah, ada 14 data benar diprediksi sebagai ayam betina marah dan 1 data salah diprediksi sebagai ayam betina memanggil jantan. Pada vokaliasi ayam betina memanggil jantan, ada 13 data benar diprediksi sebagai ayam betina memanggil jantan dan 2 data salah diprediksi sebagai ayam betina marah. Pada vokalisasi ketika ada ancaman dan setelah bertelur, semua data vokalisasi ayam berhasil diprediksi dengan benar.

Ukuran *file* model hasil percobaan keenam ini sebesar 12.4 MB, sama dengan percobaan kelima. Hasil percobaan 6 inilah yang akan digunakan untuk perbandingan dengan metode *deep learning* lainnya.

4.7. Ringkasan Hasil Seluruh Percobaan *DeepSpeech*

Berdasarkan hasil tiap percobaan yang sudah dijabarkan, dilakukan analisis untuk menilai apakah suatu model mengalami *overfitting*, *underfitting*, atau

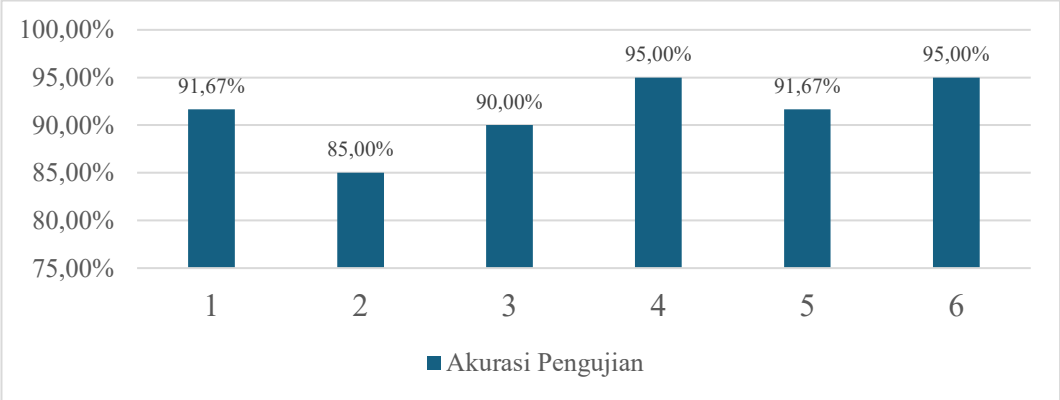
menunjukkan performa yang baik. Ringkasan kondisi dari masing-masing percobaan disajikan dalam Tabel 4.56.

Tabel 4.56 Ringkasan Kondisi Tiap Percobaan *Transformer* Berdasarkan Akurasi

Percobaan	Akurasi Pelatihan	Akurasi Validasi	Akurasi Pengujian	Keterangan Kondisi
Percobaan 1	98.22%	98.58%	96.67%	Bagus
Percobaan 2	99.55%	99.29%	95.00%	Bagus
Percobaan 3	98.31%	99.29%	93.33%	Bagus
Percobaan 4	98.93%	98.93%	96.67%	Bagus
Percobaan 5	97.68%	98.58%	93.33%	Bagus
Percobaan 6	98.13%	98.93%	95.00%	Bagus

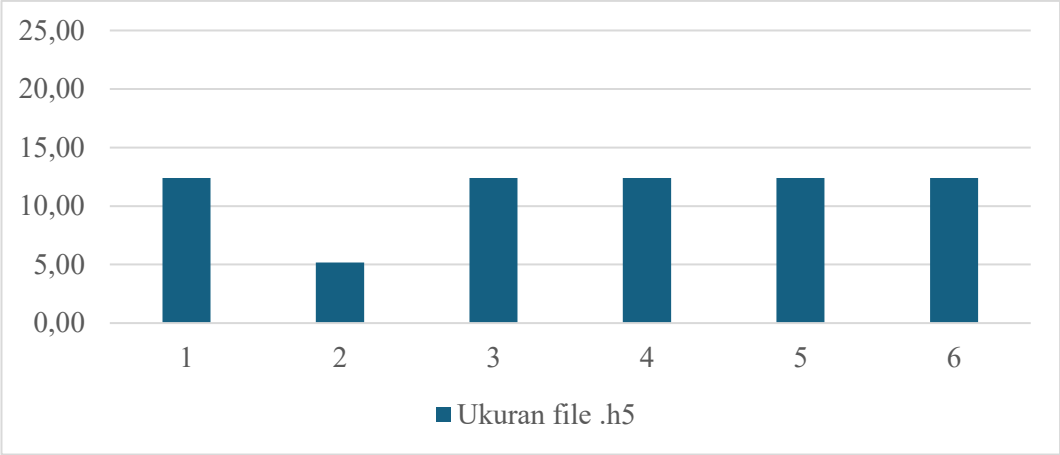
Seluruh percobaan yang dilakukan pada *DeepSpeech* menunjukkan hasil yang konsisten dan baik, dengan akurasi pelatihan, validasi, dan pengujian yang tinggi serta seimbang. Tidak ditemukan indikasi *overfitting* maupun *underfitting* yang signifikan. Oleh karena itu, semua model dianggap layak untuk dianalisis dan dibandingkan.

Berikut grafik ringkasan hasil akurasi pengujian dari seluruh percobaan *DeepSpeech*:



Gambar 4.85 Grafik hasil akurasi pengujian seluruh percobaan *DeepSpeech*

Gambar 4.85 menunjukkan pola grafik akurasi pengujian yang naik turun. Hal ini disebabkan oleh pendekatan eksploratif yang dilakukan dalam proses mengubah konfigurasi parameter di tiap percobaan.



Gambar 4.86 Grafik hasil ukuran *file* .h5 seluruh percobaan *DeepSpeech*

Gambar 4.86 menunjukkan pola grafik besar dan kecilnya ukuran *file* .h5 yang dihasilkan di tiap percobaan. Jika dilihat dan dibandingkan bersama pola grafik hasil akurasi pengujian, maka percobaan keenam menunjukkan hasil terbaik karena akurasinya 95% dan ukuran *file* .h5 kecil. Hal ini memperkuat pengambilan keputusan untuk menggunakan hasil percobaan keenam sebagai hasil terbaik dari *DeepSpeech*.

4.8. Perbandingan dan Evaluasi Model

Setelah dilakukan pelatihan dan pengujian terhadap ketiga metode *deep learning*, dilakukan perbandingan performa untuk menilai model mana yang paling optimal untuk diimplementasikan ke aplikasi *mobile*. Evaluasi ini mempertimbangkan dua aspek utama, yaitu akurasi klasifikasi pada data uji dan ukuran *file* .h5.

Masing-masing metode diambil percobaan terbaik berdasarkan hasil yang sudah dijabarkan pada subbab sebelumnya. CNN diambil dari hasil percobaan 12, *Transformer* dari percobaan 8, dan *Deepspeech* dari percobaan 6.

Tabel 4.57 menunjukkan hasil perbandingan akurasi masing-masing metode pada data pelatihan, validasi, dan pengujian, serta ukuran *file* model:

Tabel 4.57 Perbandingan Akurasi dan Ukuran *File* Ketiga Metode

Metode	Akurasi Pelatihan	Akurasi Validasi	Akurasi Pengujian	Ukuran *File* .h5
CNN	99.38%	98.93%	100.00%	6.16 MB
Transformer	99.47%	98.93%	100.00%	3.35 MB
Deepspeech	99.73%	98.58%	95.00%	12.4 MB

Berdasarkan dua aspek evaluasi utama, yaitu akurasi dan efisiensi ukuran *file* model, maka *Transformer* dipilih sebagai model paling optimal dalam penelitian ini. Keunggulann *Transformer* dalam efisiensi ukuran *file* menjadikannya lebih sesuai untuk digunakan dalam aplikasi *mobile* yang memiliki keterbatasan ruang penyimpanan dan daya komputasi.

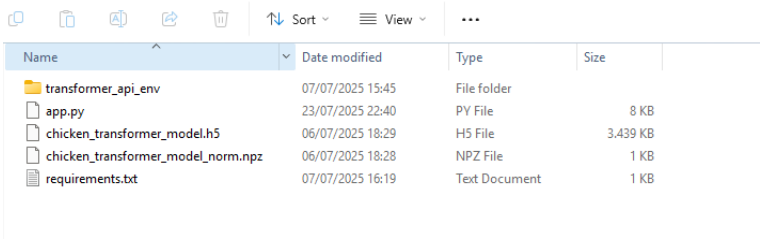
4.9. Implementasi Aplikasi *Mobile*

Pendekatan yang digunakan berupa arsitektur *client-server*, di mana proses klasifikasi tidak dilakukan langsung di perangkat pengguna, melainkan di sisi *server* menggunakan *Flask* API, aplikasi *Android* hanya bertugas untuk merekam suara ayam dan menampilkan hasil prediksinya.

Aplikasi dikembangkan menggunakan *Android Studio* dengan bahasa pemrograman *Java*. *Server* dibangun menggunakan *framework Flask* dari Python untuk menjalankan *file* model (.h5).

4.9.1. Implementasi *Backend*

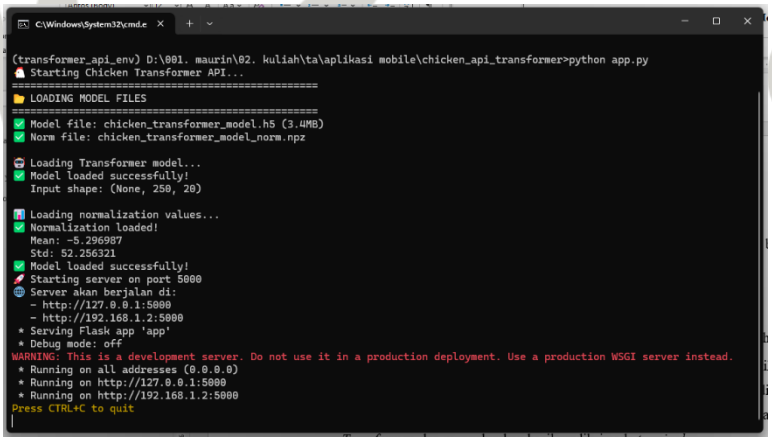
Untuk mengelola sistem *backend* secara terstruktur, dibuat sebuah *folder* khusus bernama ‘chicken_api_transformer’ yang berisi seluruh *file* yang dibutuhkan untuk menjalankan API klasifikasi suara ayam. Struktur *folder* ini ditampilkan pada Gambar 4.87.



Gambar 4.87 Struktur *folder* 'chicken_api_transformer'

Folder ini memuat *file* utama 'app.py' yang merupakan inti dari *server Flask*, *file* model hasil pelatihan 'chicken_transformer_model.h5', serta *file* 'chicken_transformer_model_norm.npz' yang menyimpan parameter normalisasi (rata-rata dan standar deviasi) dari data pelatihan. Selain itu, disertakan pula *file* 'requirements.txt' yang berisi daftar pustaka Python yang diperlukan untuk menjalankan *server*, seperti *Flask*, *TensorFlow*, dan *librosa*.

Setelah seluruh *file* disiapkan, langkah berikutnya adalah menjalankan *server backend* menggunakan *Command Prompt* (CMD). *Terminal* diarahkan terlebih dahulu ke direktori *server*, kemudian membuat *environment* Python khusus yang bernama 'transformer_api_env'. Setelah *environment* aktif, *file* 'app.py' dijalankan menggunakan perintah 'python app.py'. Jika berhasil, akan muncul *output* pada *terminal* yang menunjukkan bahwa *server Flask* telah berjalan dan siap menerima permintaan (*request*). Berikut tampilan *backend* API yang telah berhasil dijalankan secara lokal:



Gambar 4.88 Tampilan respon dari *backend* API yang telah berhasil

Gambar 4.88 menunjukkan bahwa proses pemuatan model berhasil dilakukan. Setelah pemuatan selesai, *server Flask* berjalan pada *port* 5000 dan

dapat diakses melalui alamat IP lokal ‘http://192.168.1.2:5000’, selama perangkat klien berada dalam jaringan yang sama. Alamat ini kemudian akan dihubungkan ke aplikasi *Android*.

4.9.2. Implementasi *Android*

Langkah selanjutnya adalah menghubungkan alamat *server backend* yang telah dijalankan sebelumnya ke dalam aplikasi di *Android Studio*. Gambar 4.89 menunjukkan konfigurasi alamat *server* yang digunakan dalam aplikasi:

```
1 usage
private static final String SERVER_IP = "192.168.1.2";
1 usage
private static final String SERVER_PORT = "5000";
5 usages
private static final String BASE_URL = "http://" + SERVER_IP + ":" + SERVER_PORT;
// ====================================
```

Gambar 4.89 Konfigurasi alamat *server* pada aplikasi

Potongan kode program di atas merupakan deklarasi awal untuk menentukan alamat *server* yang akan dihubungi.

Selanjutnya, untuk menghubungkan alamat *server*, dibuat fungsi `uploadAudioFile()` yang bertugas untuk mengirimkan *file* audio hasil rekaman ke *server*. Berikut potongan kodenya:

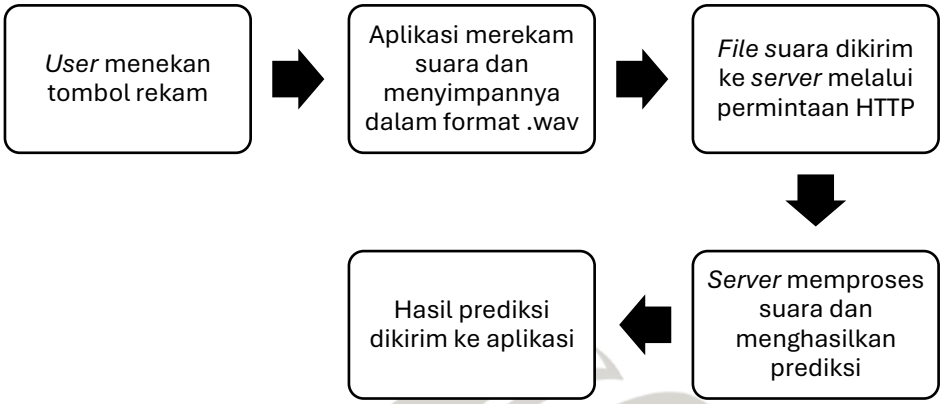
```
String url = BASE_URL + "/predict";
Log.d(TAG, msg: "Sending request to: " + url);
// ====================================

RequestBody fileBody = RequestBody.create(audioFile, MediaType.parse( $this$parse: "audio/wav"));
RequestBody requestBody = new MultipartBody.Builder()
        .setType(MultipartBody.FORM)
        .addFormDataPart( name: "audio",  filename: "recorded_audio.wav", fileBody)
        .build();

Request request = new Request.Builder()
        .url(url)
        .post(requestBody)
        .build();
```

Gambar 4.90 Fungsi untuk mengirim suara ke *server*

Fungsi ini mengatur pengiriman *file* dalam format .wav ke *server*. Setelah dikirim, *server* akan memproses *file* tersebut dan mengembalikan hasil klasifikasi suara ayam, yang nantinya ditampilkan di aplikasi.

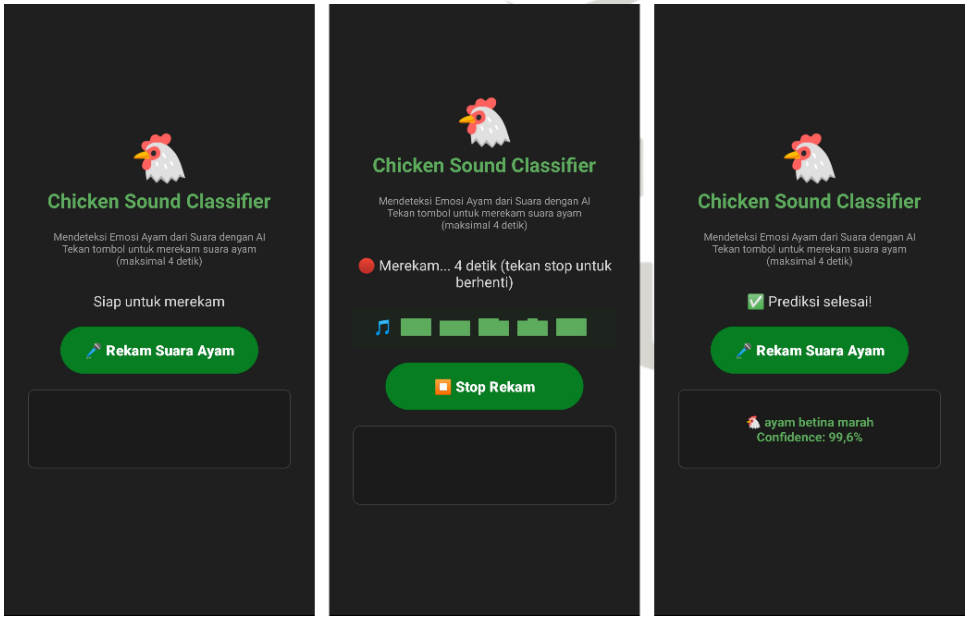
4.10. Aplikasi *Android*

Secara umum, alur kerja aplikasi ini sebagai berikut:



Gambar 4.91 Alur kerja aplikasi

Alur kerja aplikasi dimulai dari proses perekaman suara oleh pengguna, dilanjutkan dengan pengiriman data ke *server Flask* melalui API, pemrosesan suara menggunakan model klasifikasi yang telah dilatih, hingga pengembalian hasil prediksi ke aplikasi dan penampilannya dalam antarmuka pengguna.



Gambar 4.92 Tampilan aplikasi

Di paling atas, terdapat nama aplikasi, diikuti keterangan singkat mengenai aplikasi. Lalu, terdapat tombol untuk merekam suara, dan di paling bawah terdapat kotak untuk tempat hasil klasifikasi.

Ketika *user* menekan tombol rekam, akan ada tampilan gelombang suara, di mana gelombang suara ini akan bergerak naik turun sesuai dengan keras kecilnya suara yang masuk. *User* dapat menghentikan perekaman suara dengan menekan kembali tombol rekam. Jika tidak dihentikan, perekaman suara akan dibatasi sampai 4 detik, setelah 4 detik suara tersebut akan otomatis diproses untuk diklasifikasi.

Setelah suara yang masuk diproses *server*, hasil prediksi dengan tingkat *confidence* tertinggi akan ditampilkan sebagai hasil klasifikasi. *User* dapat melakukan klasifikasi lagi dengan menekan tombol rekam.

Dengan sistem ini, proses klasifikasi suara ayam dapat dilakukan secara otomatis dan cepat hanya melalui satu kali tekan tombol. Keunggulan dari pendekatan ini adalah model tidak perlu dimuat langsung di perangkat, sehingga aplikasi tetap ringan dan tidak membutuhkan sumber daya perangkat yang besar.

4.11. Uji Coba Aplikasi

Uji coba aplikasi dilakukan terhadap seluruh data uji yang terdiri dari 60 *file* audio, masing-masing mewakili empat kategori vokalisasi ayam. Semua *file* audio tersebut diputar menggunakan perangkat lain, sedangkan aplikasi dijalankan di perangkat *Android* untuk menangkap suara melalui mikrofon secara langsung. Hasil uji coba ditampilkan dalam bentuk tangkapan layar dari aplikasi yang dapat dilihat pada Lampiran.

Berikut rangkuman hasil uji coba aplikasi yang dibandingkan dengan hasil uji saat pemodelan (sebelum diintegrasikan ke aplikasi).

Tabel 4.58 Ringkasan Perbandingan Hasil Uji Coba Aplikasi dengan Pemodelan

Kategori	Jumlah *File* Uji	Prediksi Benar (Pemodelan)	Akurasi Pemodelan	Prediksi Benar (Aplikasi)	Akurasi Aplikasi
Ayam betina marah	15	15	100.00%	1	6.67%

Tabel 4.58 (Lanjutan)

Kategori	Jumlah *File* Uji	Prediksi Benar (Pemodelan)	Akurasi Pemodelan	Prediksi Benar (Aplikasi)	Akurasi Aplikasi
Ayam betina memanggil jantan	15	15	100.00%	15	100.00%
Ketika ada ancaman	15	15	100.00%	14	93.33%
Setelah bertelur	15	15	100.00%	15	100.00%

Berdasarkan hasil uji coba aplikasi terhadap 60 data uji, diperoleh penurunan drastis pada akurasi kelas ayam betina marah jika dibandingkan dengan hasil uji pemodelan sebelum diintegrasikan ke aplikasi, di mana model berhasil memprediksi semua *file* ayam betina marah dengan benar.

Untuk memahami lebih lanjut penyebab penurunan akurasi pada saat pengujian aplikasi (khususnya ayam betina marah), dilakukan analisis hasil prediksi aplikasi terhadap setiap *file* uji per kategori. Analisis ini mencakup identifikasi *file-file* yang diprediksi salah serta kategori mana yang menjadi hasil prediksinya.

Tabel 4.59 menampilkan hasil prediksi aplikasi ketika diuji dengan data uji kategori ayam betina marah.

Tabel 4.59 Hasil Prediksi Aplikasi Kategori Ayam Betina Marah

File Uji	Prediksi Aplikasi	Tingkat *Confidence*
File 1	Ayam betina memanggil jantan	99.9%
File 2	Ayam betina memanggil jantan	99.9%
File 3	Ayam betina memanggil jantan	90.5%
File 4	Ayam betina memanggil jantan	99.9%
File 5	Ayam betina memanggil jantan	72.5%
File 6	Ayam betina memanggil jantan	98.4%

Tabel 4.59 (Lanjutan)

File Uji	Prediksi Aplikasi	Tingkat *Confidence*
File 7	Ayam betina memanggil jantan	75.4%
File 8	Ketika ada ancaman	99.0%
File 9	Ketika ada ancaman	98.2%
File 10	Ketika ada ancaman	99.5%
File 11	Ayam betina memanggil jantan	60.1%
File 12	Ayam betina marah	76.2%
File 13	Setelah bertelur	98.7%
File 14	Ketika ada ancaman	50.4%
File 15	Ayam betina memanggil jantan	98.9%

Hal ini mengindikasikan adanya penurunan performa setelah model diimplementasikan ke dalam aplikasi. Penurunan tersebut kemungkinan disebabkan oleh perbedaan kondisi lingkungan saat perekaman suara melalui perangkat *Android*, seperti kualitas mikrofon atau karakteristik vokalisasi ayam betina marah yang melengking.

Tabel 4.60 menampilkan hasil prediksi aplikasi ketika diuji dengan data uji kategori ayam betina memanggil jantan.

Tabel 4.60 Hasil Prediksi Aplikasi Kategori Ayam Betina Memanggil Jantan

File Uji	Prediksi Aplikasi	Tingkat *Confidence*
File 1	Ayam betina memanggil jantan	98.8%
File 2	Ayam betina memanggil jantan	97.9%
File 3	Ayam betina memanggil jantan	99.7%
File 4	Ayam betina memanggil jantan	99.9%
File 5	Ayam betina memanggil jantan	100.0%
File 6	Ayam betina memanggil jantan	99.8%
File 7	Ayam betina memanggil jantan	100.0%
File 8	Ayam betina memanggil jantan	100.0%
File 9	Ayam betina memanggil jantan	100.0%

Tabel 4.60 (Lanjutan)

File Uji	Prediksi Aplikasi	Tingkat *Confidence*
File 10	Ayam betina memanggil jantan	55.5%
File 11	Ayam betina memanggil jantan	100.0%
File 12	Ayam betina memanggil jantan	100.0%
File 13	Ayam betina memanggil jantan	96.6%
File 14	Ayam betina memanggil jantan	97.8%
File 15	Ayam betina memanggil jantan	95.2%

Hasil prediksi aplikasi pada kategori ayam betina memanggil jantan sudah sangat bagus, dapat dilihat pada hasil labelnya yang benar semua, serta tingkat *confidence* yang tinggi. Hal ini menunjukkan bahwa akurasi pengujian saat pemodelan dan akurasi saat diuji pada aplikasi sesuai, membuktikan bahwa model berhasil melakukan prediksi dengan akurat.

Tabel 4.61 menampilkan hasil prediksi aplikasi ketika diuji dengan data uji kategori ketika ada ancaman.

Tabel 4.61 Hasil Prediksi Aplikasi Kategori Ketika Ada Ancaman

File Uji	Prediksi Aplikasi	Tingkat *Confidence*
File 1	Ketika ada ancaman	51.9%
File 2	Ketika ada ancaman	97.8%
File 3	Ketika ada ancaman	93.6%
File 4	Ketika ada ancaman	58.6%
File 5	Ketika ada ancaman	99.4%
File 6	Ketika ada ancaman	95.0%
File 7	Ketika ada ancaman	99.9%
File 8	Ketika ada ancaman	99.6%
File 9	Ketika ada ancaman	94.9%
File 10	Ketika ada ancaman	84.4%
File 11	Setelah bertelur	84.2%
File 12	Ketika ada ancaman	99.9%

Tabel 4.61 (Lanjutan)

File Uji	Prediksi Aplikasi	Tingkat *Confidence*
File 13	Ketika ada ancaman	99.7%
File 14	Ketika ada ancaman	99.5%
File 15	Ketika ada ancaman	93.2%

Hasil prediksi aplikasi pada kategori ketika ada ancaman menunjukkan performa yang sangat baik, dengan 14 dari 15 *file* berhasil diprediksi dengan benar. Tingkat *confidence* yang dihasilkan juga tinggi, sebagian besar di atas 90%. Secara umum, hasil ini menunjukkan bahwa akurasi pengujian saat pemodelan dan akurasi saat diuji pada aplikasi sesuai, membuktikan bahwa model berhasil melakukan prediksi dengan akurat.

Tabel 4.62 menampilkan hasil prediksi aplikasi ketika diuji dengan data uji kategori setelah bertelur.

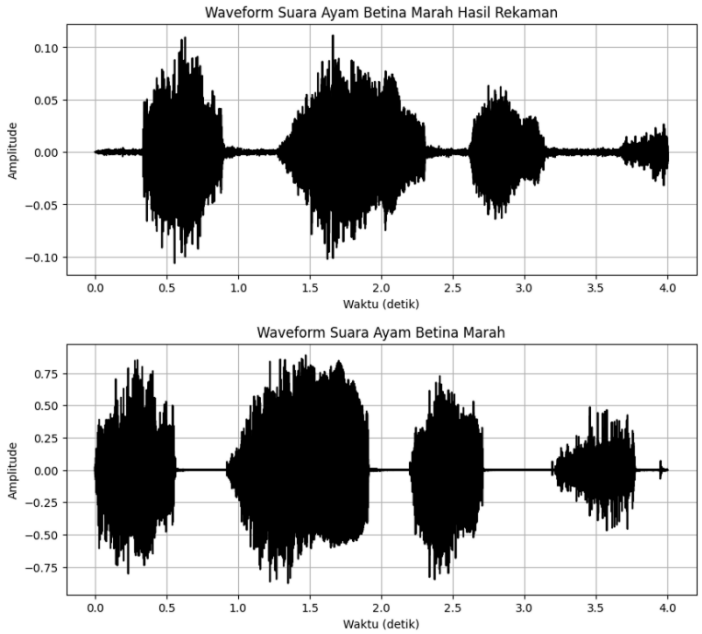
Tabel 4.62 Hasil Prediksi Aplikasi Kategori Setelah Bertelur

File Uji	Prediksi Aplikasi	Tingkat *Confidence*
File 1	Setelah bertelur	99.8%
File 2	Setelah bertelur	100.0%
File 3	Setelah bertelur	99.9%
File 4	Setelah bertelur	100.0%
File 5	Setelah bertelur	99.9%
File 6	Setelah bertelur	99.9%
File 7	Setelah bertelur	99.9%
File 8	Setelah bertelur	97.4%
File 9	Setelah bertelur	97.0%
File 10	Setelah bertelur	90.9%
File 11	Setelah bertelur	98.6%
File 12	Setelah bertelur	97.6%
File 13	Setelah bertelur	99.6%
File 14	Setelah bertelur	98.7%
File 15	Setelah bertelur	99.9%

Hasil prediksi aplikasi pada kategori setelah bertelur sudah sangat bagus, dapat dilihat pada hasil labelnya yang benar semua, serta tingkat *confidence* yang tinggi. Hal ini menunjukkan bahwa akurasi pengujian saat pemodelan dan akurasi saat diuji pada aplikasi sesuai, membuktikan bahwa model berhasil melakukan prediksi dengan akurat.

4.11.1. Evaluasi Hasil Uji Coba Aplikasi

Sebagai langkah lanjutan untuk menganalisis penyebab rendahnya akurasi prediksi aplikasi khususnya pada kategori ayam betina marah, dilakukan perbandingan antara bentuk gelombang (*waveform*) dari *file* audio uji dengan hasil rekamannya melalui aplikasi. Tujuan dari perbandingan ini adalah untuk mengetahui sejauh mana kualitas suara yang berhasil ditangkap oleh mikrofon aplikasi, serta apakah perbedaan karakteristik sinyal tersebut memengaruhi kemampuan model dalam melakukan klasifikasi. Gambar 4.93 menyajikan visualisasi bentuk gelombang dari kedua jenis audio.



Gambar 4.93 Perbandingan *waveform* suara asli dengan hasil rekaman

Dapat dilihat bahwa amplitudo dari hasil rekaman suara untuk kategori ayam betina marah jauh lebih kecil dibandingkan dengan *file* uji aslinya.

Gelombang suara hasil rekaman hanya memiliki puncak amplitudo sekitar 0.10, sedangkan *file* aslinya mencapai lebih dari 0.75. Hal ini mengindikasikan bahwa aplikasi hanya mampu menangkap sinyal audio dengan intensitas rendah. Kualitas rekaman yang rendah ini kemungkinan menyebabkan model kesulitan mengenali pola khas dari suara ayam betina marah, sehingga menghasilkan prediksi yang salah secara konsisten. Temuan ini menunjukkan bahwa model kemungkinan belum cukup *robust* terhadap variasi sinyal suara khusus untuk kategori ini.

Bab V
Simpulan dan Saran

5.1. Simpulan

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan pada bab-bab sebelumnya, dapat disimpulkan beberapa hal sebagai berikut:

1. Sistem klasifikasi vokalisasi ayam berhasil dikembangkan menggunakan pendekatan *deep learning* untuk mendeteksi kondisi ayam secara otomatis. Sistem ini mampu mengidentifikasi 4 jenis vokalisasi, yaitu ayam betina marah, ayam betina memanggil jantan, ketika ada ancaman, dan setelah bertelur, dengan menggunakan maksimal suara yang diterima 4 detik.
2. Ketiga metode *deep learning* (CNN, *Transformer*, dan *DeepSpeech*) diuji dengan pengaturan konfigurasi yang berbeda-beda untuk memperoleh hasil terbaik. Model *Transformer* memberikan performa paling optimal dengan akurasi pengujian sebesar 100% dan ukuran *file* model sebesar 3.35 MB.
3. Model *Transformer* yang telah dilatih berhasil diimplementasikan ke dalam aplikasi *mobile Android* menggunakan pendekatan *client-server*. Aplikasi dapat merekam suara ayam, mengirimkan audio ke *server Flask* untuk diklasifikasikan oleh model, lalu menampilkan hasil klasifikasi secara *real-time*. Uji coba aplikasi menggunakan 60 data uji menunjukkan bahwa sistem mampu berjalan dengan baik meskipun terjadi penurunan akurasi pada kategori ayam betina marah dibandingkan hasil pengujian model.

5.2. Saran

Berdasarkan hasil yang telah dicapai, terdapat beberapa saran yang dapat diberikan untuk pengembangan lebih lanjut:

1. Peningkatan jumlah dan variasi *dataset*: Agar model menjadi lebih *robust* dan mampu menangani variasi lingkungan nyata, jumlah data sebaiknya

ditambah dan mencakup berbagai kondisi seperti kebisingan latar belakang, kualitas mikrofon, dan lokasi perekaman yang berbeda.

2. Pengujian di lapangan: Aplikasi sebaiknya diuji langsung di kandang ayam dengan suara secara *real-time* untuk melihat seberapa baik sistem bekerja dalam kondisi sebenarnya dan apakah model mampu beradaptasi terhadap suara yang tidak sempurna.
3. Eksplorasi arsitektur lain: Di masa depan, arsitektur lain seperti *EfficientNet*, *MobileNetV3*, atau model lain berbasis audio dapat dijadikan alternatif untuk mengembangkan model yang lebih ringan dan cepat.

Daftar Pustaka

Adebayo, S., Aworinde, H.O., Akinwunmi, A.O., Alabi, O.M., Ayandiji, A., Sakpere, A.B. *et al.,* 2023, *Enhancing Poultry Health Management Through Machine Learning-Based Analysis of Vocalization Signals Dataset, Poultry Science,* <https://www.sciencedirect.com/science/article/pii/S2352340923006285/pdfft?md5=1cc589fccfe2dfac2c446a416a3214b8&pid=1-s2.0-S2352340923006285-main.pdf> [Accessed 21 April 2025]

Adhinata, F.D., Rakhmadani, D.P. dan Segara, A.J.T., 2021, *Pengenalan Jenis Kelamin Manusia Berbasis Suara Menggunakan MFCC dan GMM, JURNAL DINDA,* 1(1):1-6 <https://journal.ittelkom-pwt.ac.id/index.php/dinda/article/view/198/92> [diakses pada 22 Maret 2025]

Afida, A.M., 2020, *Klasifikasi Jenis Burung Berdasarkan Suara Menggunakan Algoritme Support Vector Machine, Jurnal Ilmiah Matematika,* 8(1):1-6 <https://ejournal.unesa.ac.id/index.php/mathunesa/article/view/31702> [diakses pada 7 Maret 2025]

Ajinurseto, G., Bakrim, L.O. dan Islamuddin, N., 2023, *Penerapan Metode Mel Frequency Cepstral Coefficients pada Sistem Pengenalan Suara Berbasis Desktop, INFOMATEK: Jurnal Informatika, Manajemen dan Teknologi,* 25(2):11-20 <https://journal.unpas.ac.id/index.php/infomatek/article/view/6109/3385> [diakses pada 22 Maret 2025]

Alberto, J. dan Hermanto, D., 2023, *Klasifikasi Jenis Burung Menggunakan Metode CNN Dan Arsitektur ResNet-50, Jurnal Teknik Informatika dan Sistem Informasi,* 10(3):34-46 <https://www.researchgate.net/publication/374556086_Klasifikasi_Jenis_Burung_Menggunakan_Metode_CNN_Dan_Arsitektur_ResNet-50> [diakses pada 7 Maret 2025]

Apalowo, O.O., Ekunseitan, D.A., and Fasina, Y.O., 2024, *Impact of Heat Stress on Broiler Chicken Production, Poultry,* 3(2):107–128 <https://www.mdpi.com/2674-1164/3/2/10> [Accessed 15 April 2025]

Caroline, Shabrina, N.H., Ao, M.R., Laurencya, N.L. dan Lee, V., 2020, *Analisis Aplikasi Filter FIR dan Filter IIR dalam Pra-pemrosesan Sinyal Elektroensefalografi, ULTIMA Computing,* 12(1):40-48 <https://ejournals.umn.ac.id/index.php/SK/article/download/1621/946> [diakses pada 29 Maret 2025]

Cheok, A., Cai, J. and Yan, Y., 2023, *Deciphering Avian Emotions: A Novel AI and Machine Learning Approach to Understanding Chicken Vocalizations. Research Square, Preprint* <https://www.researchsquare.com/article/rs-3034567/v1> [Accessed 22 April 2025]

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T. *et. al,* 2021, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR* <https://arxiv.org/abs/2010.11929> [Accessed 7 March 2025]

Fauziah, F., Tritoasmoro, I.I. dan Rizal, S., 2021, *Sistem Keamanan Berbasis Pengenalan Suara sebagai Pengakses Pintu Menggunakan Metode Mel Frequency Cepstral Coefficient (MFCC), e-Proceeding of Engineering,* 8(6):11839-11848 <https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/17304> [diakses pada 28 Maret 2025]

Ferdiawan, F. dan Hartono, B., 2022, *Deteksi Suara Chord Piano Menggunakan Metode Convolutional Neural Network, JIRE (Jurnal Informatika & Rekayasa Elektronika),* 5(1) <https://www.researchgate.net/publication/367057039_DETEKSI_SUARA_CHOCH_PIANO_MENGGUNAKAN_METODE_CONVOLUTIONAL_NEURAL_NNETWOR> [diakses pada 7 Maret 2025]

GeeksforGeeks, *Speech Recognition with DeepSpeech using Mozilla's DeepSpeech*, *GeeksforGeeks*, 25 September 2024 <https://www.geeksforgeeks.org/speech-recognition-with-deepspeech-using-mozilla-s-deepspeech/> [Accessed pada 21 March 2025]

Gultom, A.Y.R., Finatih, R.A., Issaputra, I.K., Martiansyah, M.G., Lia, S.R., Nabilla, N. *et al.,* 2025, *Klasifikasi Suara Nyamuk Berbasis CNN (Convolutional Neural Network) untuk Inovasi Pengendalian Hama dan Penyakit, Prosiding Seminar Nasional Sains dan Teknologi Seri III,* 2(1):615-626 <https://conference.ut.ac.id/index.php/saintek/article/view/5063> [diakses pada 23 Maret 2025]

Gumilang, K., Nugraha, A. F., Savitri, I., Yani, N.F., Yunita Puspitasari, A., Sekartaji, J.G. *et. al,* 2025, *Klasifikasi Suara Katak Menggunakan Model Deep Learning Modified Densenet-121 dan Densenet-169 dengan Fitur Ekstraksi MFCC, Prosiding Seminar Nasional Sains dan Teknologi Seri III,* 2(1):627-637 <https://conference.ut.ac.id/index.php/saintek/article/view/5065> [diakses pada 7 Maret 2025]

Halim, S.S., Kanata, B. dan Akbar, L.A.S.I., 2024, *Klasifikasi Suara Paru-Paru Menggunakan Mel Frequency Cepstral Coefficient dan Convolutional Neural Network, Jurnal Bumigora Information Technology (BITe),*

6(2):163-176
<https://journal.universitasbumigora.ac.id/bite/article/view/4487/1842>
[diakses pada 22 Maret 2025]

Hazmar, L.O.A. dan Prasetio, B.H., 2023, *Sistem Pengenalan Tingkatan Emosi Ketakutan melalui Ucapan Menggunakan Ekstraksi Gammatone-Frequency Cepstral Coefficients dan Klasifikasi Random Forest Classifier berbasis Raspberry Pi 4, Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 7(2):1003-1011 <https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/12364/5620> [diakses pada 29 Maret 2025]

Hendry, J., Sumanto, B., Mileniandi, Y. dan Wening, P.M., 2022, *Implementation of Pre-Emphasis Analog Filter on Raspberry Pi Using Zero-Order Hold Discretization as a Pre-Processing to Humanoids' Commands Recognition, Jurnal Listrik, Instrumentasi, dan Elektronika Terapan (JuLIET),* 3(2):40-44 <https://journal.ugm.ac.id/juliet/article/view/71226/35352> [diakses pada 28 Maret 2025]

Howdy. (n.d.). *Mozilla DeepSpeech*. *Howdy*. Retrieved March 21, 2025, from https://www.howdy.com/glossary/mozilla-deepspeech

Karina, F. F., 2020, *Penerapan Metode MFCC dan Backpropagation untuk Pengenalan Kicauan Kakatua dalam Penentuan Jantan dan Betina, Itenas Repository* <https://eprints.itenas.ac.id/1271/> [diakses 7 Maret 2025]

Latif, S., Rana, R., Khalifa, S., Jurdak, R., Qadir, J. and Schuller, B.W., 2021, *Deep Representation Learning in Speech Processing: Challenges, Recent Advances, and Future Trends, IEEE Transactions on Affective Computing* <https://arxiv.org/pdf/2001.00378> [Accessed 7 March 2025]

Manikandan, V. and Neethirajan, S., 2024, *Decoding Poultry Vocalizations - Natural Language Processing and Transformer Models for Semantic and Emotional Analysis, The Preprint Server For Biology* <Decoding Poultry Vocalizations - Natural Language Processing and Transformer Models for Semantic and Emotional Analysis | bioRxiv> [Accessed 7 March 2025]

Mao, A., Giraudet, C.S.E., Liu, K., Nolasco, I.D.A., Xie, Z., Gao, Y. *et al.,* 2022, *Automated Identification of Chicken Distress Vocalizations Using Deep Learning Models, Journal of the Royal Society Interface,* 19(191): 20210921
<https://royalsocietypublishing.org/doi/10.1098/rsif.2021.0921>
[Accessed 15 April 2025]

Mayasari, N., Hiroyuki, A., Budinuryanto, D.C., Firmansyah, I. dan Ismiraj, M.R., 2023, *Penerapan Prinsip Kesejahteraan Hewan pada Pemeliharaan Ternak, Jurnal Aplikasi Ipteks untuk Masyarakat,* 12(3):360-373 <https://www.bing.com/ck/a?!&&p=580354770d402bcc6480983c542c76deb0b731df3a3ff14204c2c963cf7ed40eJmltdHM9MTc1MzIyODgwMA&ptn=3&ver=2&hsh=4&fclid=0ff0b13a-c6fb-6b00-20af-

a413c7fb6ae4&psq=PENERAPAN+PRINSIP+KESEJAHTERAAN+HEWAN+PADA+PEMELIHARAAN+TERNAK&u=a1aHR0cHM6Ly9qdXJuYWwudW5wYWQuYWMuaWQvZGhhcm1ha2FyeWEvYXJ0aWNsZS9kb3dubG9hZC80MjY4Ni8yMTM5MA&ntb=1> [diakses pada 7 Maret 2025]

Merdeka, *Pakai AI, Ilmuwan Jepang Ini Mengaku Bisa Pahami Bahasa Ayam, Merdeka.com*, 25 September 2023 <https://www.merdeka.com/teknologi/pakai-ai-ilmuwan-jepang-ini-mengaku-bisa-pahami-bahasa-ayam-28101-mvk.html> [Accessed 22 April 2025]

Mulyana, D.I. dan Nurrohman, A.T., 2025, *Analisis Efektivitas GAN dalam Meningkatkan Akurasi Deteksi Tuna Rungu dengan Menggunakan Data Audio dan Visual, Institut Riset dan Publikasi Indonesia (IRPI),* 5(1):101-110 <https://journal.irpi.or.id/index.php/malcom/article/view/1635/849> [diakses pada 29 Mar 2025]

Ranny, Suwardi, I.S., Rajab, T.L.E. dan Lestari, D.P., 2019, *Kajian Penelitian Pemrosesan Bunyi dan Aplikasinya pada Teknologi Informasi (Study of Sound Processing and Application on Information Technology), JUITA: Jurnal Informatika,* 7(1):1-10 <https://jurnalnasional.ump.ac.id/index.php/JUITA/article/view/3491/2559> [diakses pada 28 Maret 2025]

Safitra, M.I., Putranto, H.D. dan Brata, B., 2022, *Karakteristik Suara Kokok Ayam Burgo Jantan Kota Bengkulu, Jurnal Peternakan,* 19(1): 64-70 <https://www.bing.com/ck/a?!&&p=998d9f8d964fa5263eb4df5eb01b5d8c8e9f1a4b9fc2fe605883cb9704fb1822JmltdHM9MTc1MzIyODgwMA&ptn=3&ver=2&hsh=4&fclid=0ff0b13a-c6fb-6b00-20af-a413c7fb6ae4&psq=Karakteristik+Suara+Kokok+Ayam+Burgo+Jantan+Kota+Bengkulu%2c+Jurnal+Peternakan&u=a1aHR0cHM6Ly9lam91cm5hbC51aW4tc3Vza2EuYWMuaWQvaW5kZXgucGhwL3BldGVybmFrYW4vYXJ0aWNsZS9kb3dubG9hZC8xNTA0Ny83Mzg0&ntb=1> [diakses pada 7 Maret 2025]

Soster, P.D.C., Grzywalski, T., Hou, Y., Thomas, P., Dedeurwaerder, A., Gussem, M.D. *et al.,* 2025, *Automated Detection of Broiler Vocalizations a Machine Learning Approach for Broiler Chicken Vocalization Monitoring, Poultry Science* <https://www.sciencedirect.com/science/article/pii/S0032579125002019> [Accessed 22 April 2025]

Subitmele, S.E., 2023, *Bikin Takjub, Peneliti Jepang Gunakan AI untuk Terjemahkan Suara Ayam*, *Liputan6.com*, 27 September 2023 <https://www.liputan6.com/hot/read/5408645/bikin-takjub-peneliti-jepang-gunakan-ai-untuk-terjemahkan-suara-ayam> [diakses pada 21 Maret 2025]

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N. *et. al,* 2017, *Attention Is All You Need, Cornell University* <https://arxiv.org/pdf/1706.03762> [Accessed 7 March 2025]

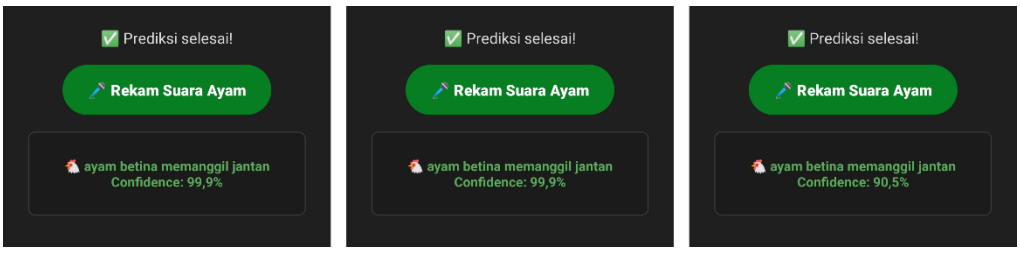
Wijaya, C.K., 2024. 'Pengembangan Aplikasi Berbasis *Android* untuk Klasifikasi Vokalisasi Ayam dengan Perbandingan Model *Deep Learning'*, *Skripsi*, Sarjana Komputer, Universitas Ma Chung, Malang.

Zaharo, L., Damayanti, E.P., Julianti, H. dan Basriwijaya, K.M.Z., 2024, *Manajemen Pemeliharaan Ayam Kampung di Kecamatan Perbaungan Kabupaten Serdang Bedagai, Sumatera Utara, Botani: Publikasi Ilmu Tanaman dan Agribisnis*, 2(1):153–161 <https://journal.asritani.or.id/index.php/Botani/article/view/169/271> [diakses pada 7 Maret 2025]

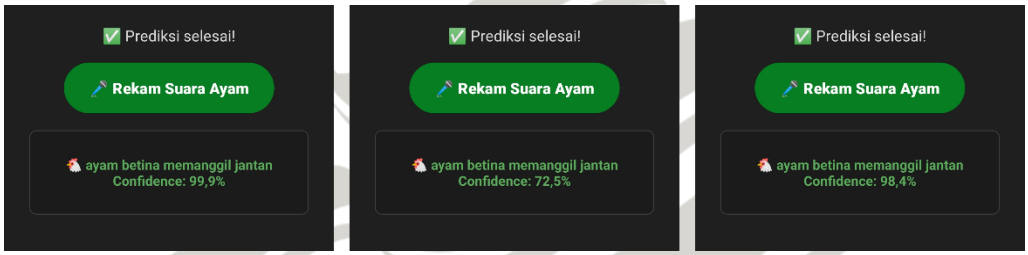
Zampiga, M., Calini, F. and Sirri, F., 2021, *Importance of Feed Efficiency for Sustainable Intensification of Chicken Meat Production: Implications and Role for Aminoacids, Feed Enzymes and Organic Trace Minerals, World's Poultry Science Journal,* 77(3): 639–659 <https://www.tandfonline.com/doi/epdf/10.1080/00439339.2021.1959277?needAccess=true> [Accessed 22 April 2025]

Lampiran

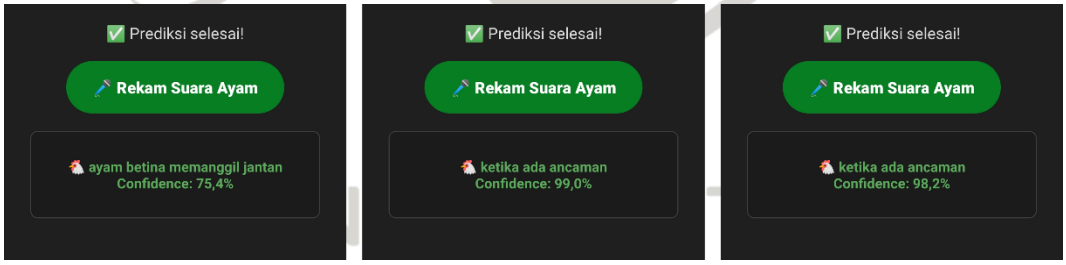
A. Hasil Uji Coba Aplikasi Kategori Ayam Betina Marah



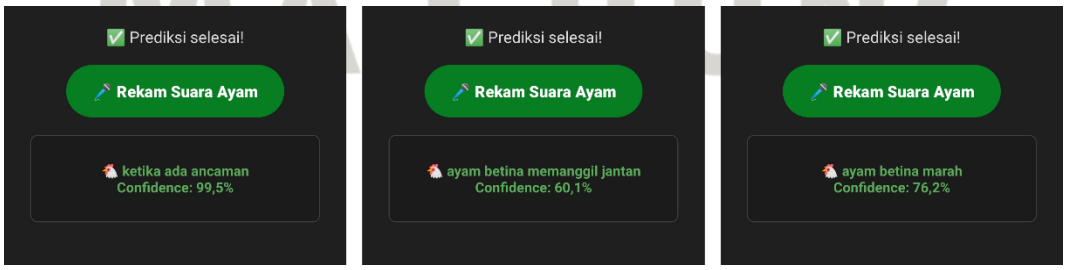
Gambar A.1 Hasil prediksi *file* 1-3



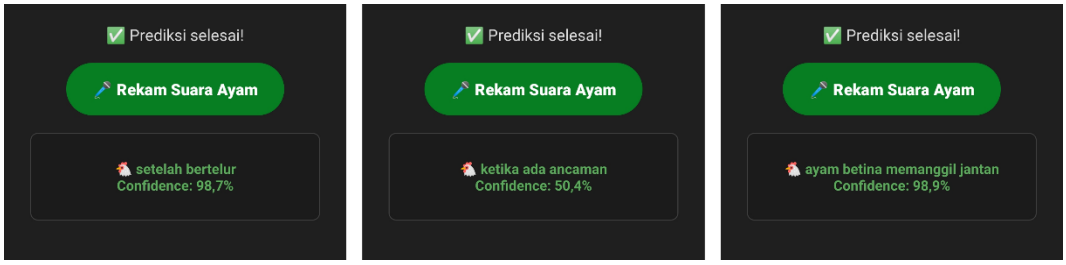
Gambar A.2 Hasil prediksi *file* 4-6



Gambar A.3 Hasil prediksi *file* 7-9

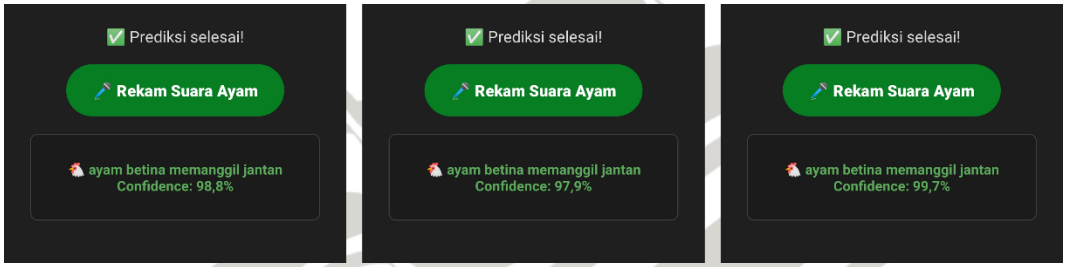


Gambar A.4 Hasil prediksi *file* 10-12

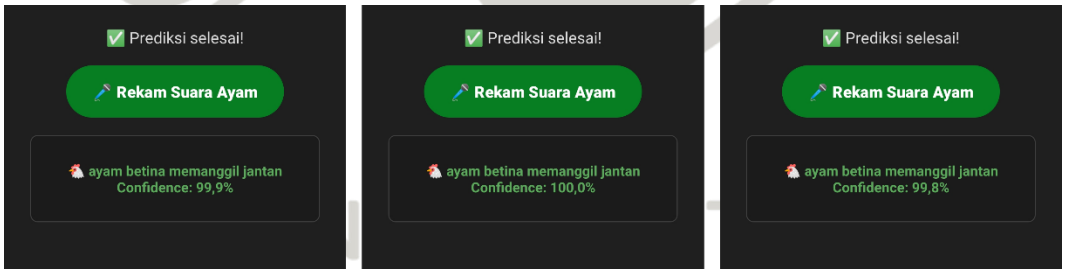


Gambar A.5 Hasil prediksi *file* 13-15

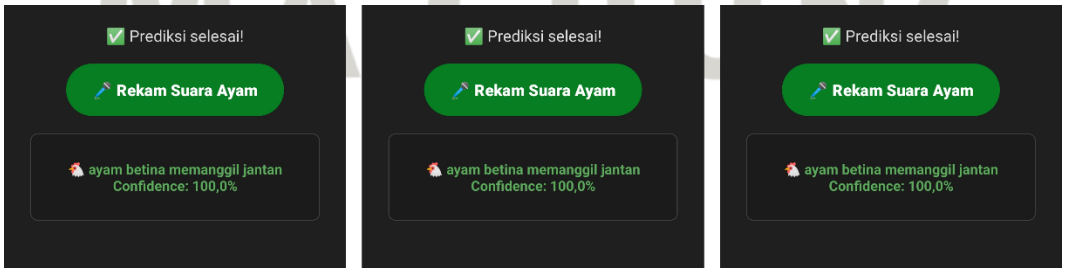
B. Hasil Uji Coba Aplikasi Kategori Ayam Betina Memanggil Jantan



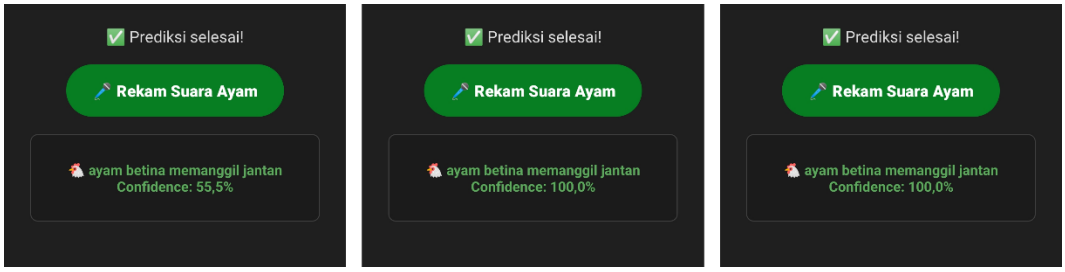
Gambar B.1 Hasil prediksi *file* 1-3



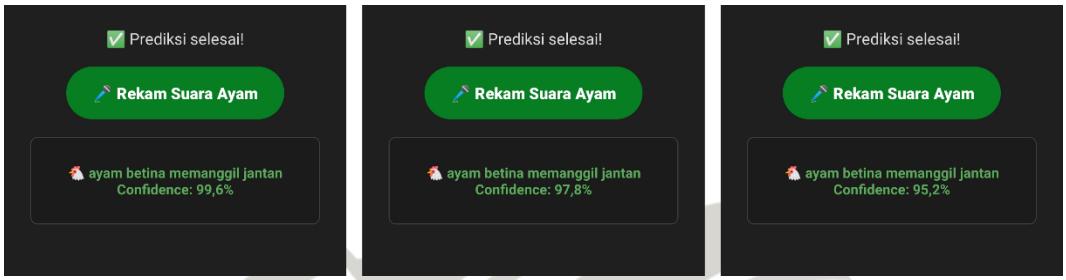
Gambar B.2 Hasil prediksi *file* 4-6



Gambar B.3 Hasil prediksi *file* 7-9

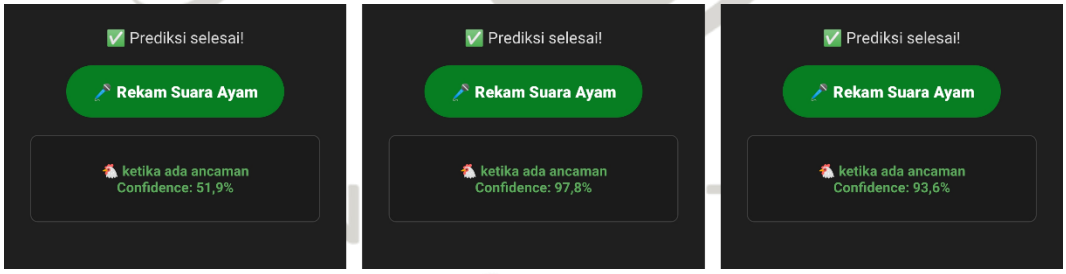


Gambar B.4 Hasil prediksi *file* 10-12

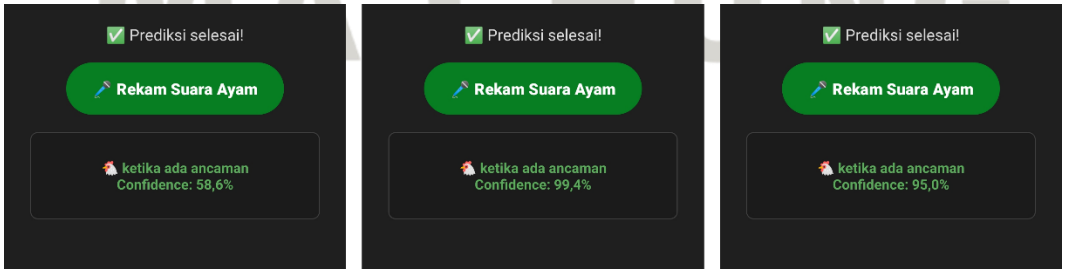


Gambar B.5 Hasil prediksi *file* 13-15

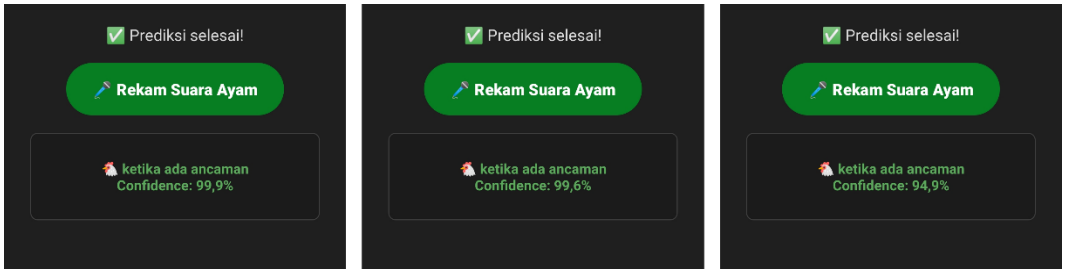
C. Hasil Uji Coba Aplikasi Kategori Ketika Ada Ancaman



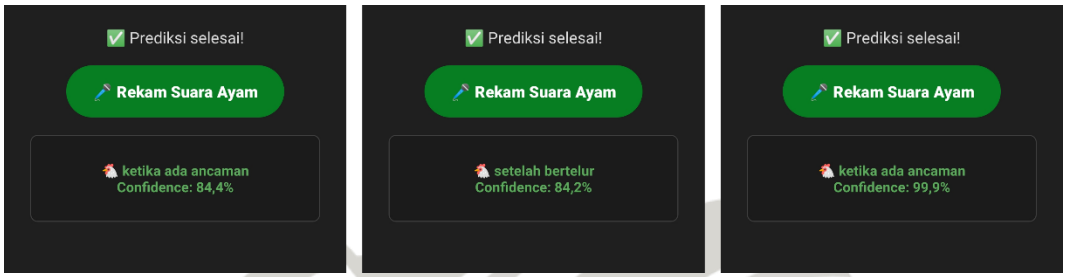
Gambar C.1 Hasil prediksi *file* 1-3



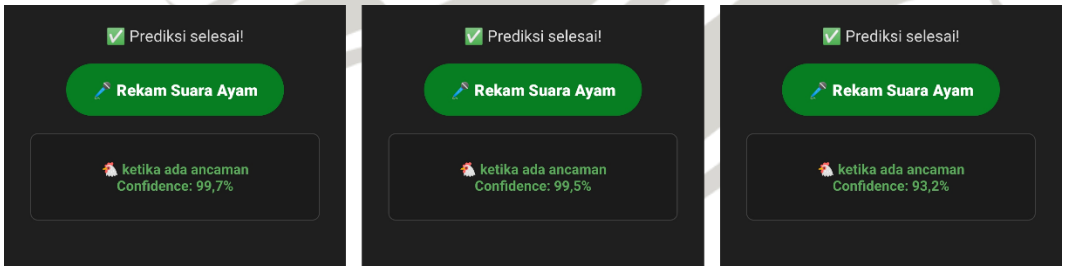
Gambar C.2 Hasil prediksi *file* 4-6



Gambar C.3 Hasil prediksi *file* 7-9

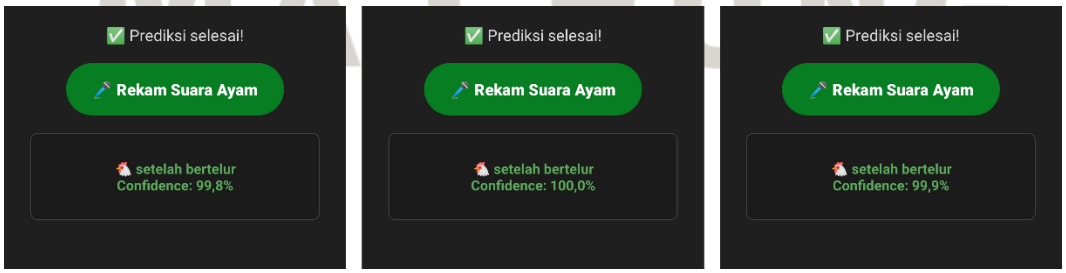


Gambar C.4 Hasil prediksi *file* 10-12

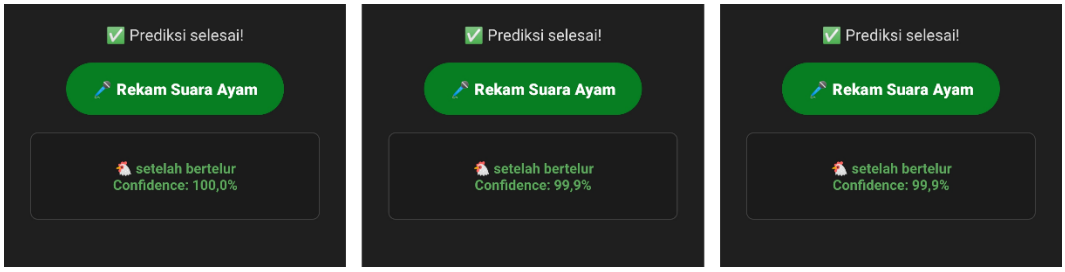


Gambar C.5 Hasil prediksi *file* 13-15

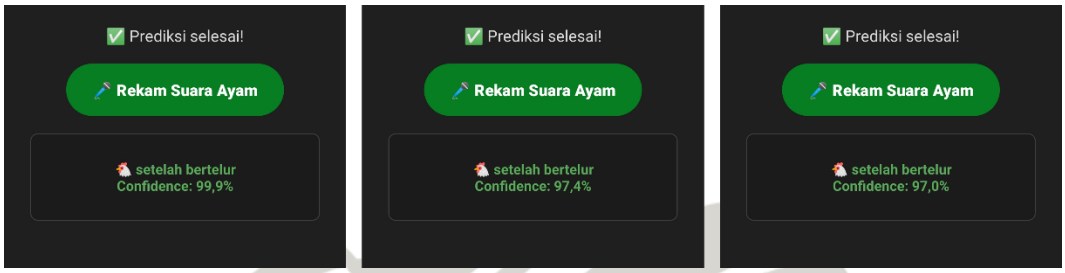
D. Hasil Uji Coba Aplikasi Kategori Setelah Bertelur



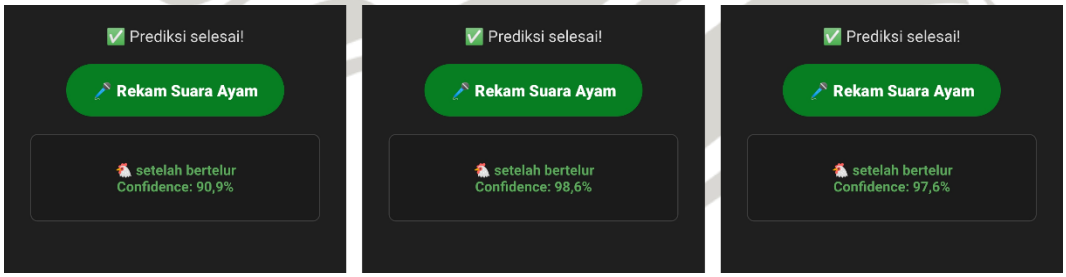
Gambar D.1 Hasil prediksi *file* 1-3



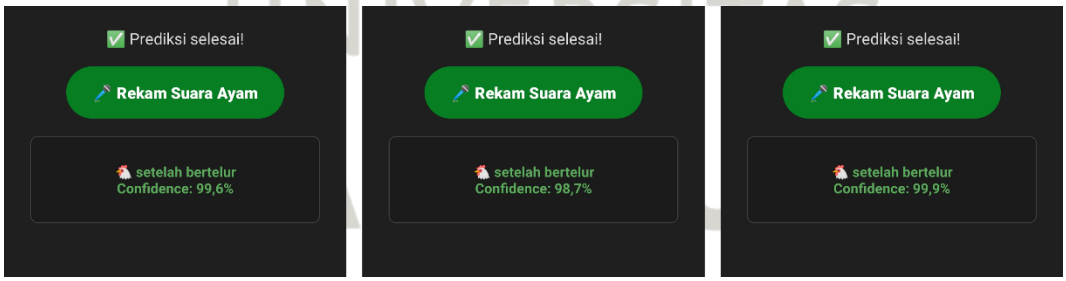
Gambar D.2 Hasil prediksi *file* 4-6



Gambar D.3 Hasil prediksi *file* 7-9



Gambar D.4 Hasil prediksi *file* 10-12



Gambar D.5 Hasil prediksi *file* 13-15