Automated Bird Species Identification Using Convolutional Neural Networks

NAVEEN S (Reg No: KTE24MCA-2044)

Project Guide: Dr. Sangeetha Jose

October 5, 2025

Introduction

- Accurate bird species identification is crucial for biodiversity monitoring and conservation.
- Manual identification is time-consuming and prone to errors.
- Artificial Intelligence, specifically CNNs, offers a reliable and scalable solution.

Problem Statement

- Current identification relies on subjective human observation, which is impractical for large datasets.
- Difficulties arise in classifying visually similar or fine-grained bird species.
- **Goal:** To automate the entire bird identification process using CNNs to achieve higher accuracy and real-time scalability in a web environment.

Objectives

- Develop a robust CNN model (MobileNetV2/Transfer Learning) for species identification.
- Preprocess and augment a large-scale dataset (Kaggle) to ensure model stability.
- Build a functional web interface (Flask/HTML) for real-time prediction deployment.
- Conduct rigorous evaluation using performance metrics (Accuracy, F1-Score).

Scope

- **Core Functionality:** Provides a complete pipeline from image upload to final classification result.
- **Technical Scope:** Includes data processing, model training (Transfer Learning), and Flask API deployment.
- **Output:** Delivers species name, confidence score, and pre-calculated model performance metrics.
- **Key Feature:** Scalable and efficient feature extraction optimized for real-time use.

Relevance

- **Conservation Impact:** Provides immediate, objective data for tracking bird populations and ecosystem health.
- **Efficiency:** Reduces reliance on human experts and drastically cuts the time/cost associated with identification.
- **Accessibility:** The web-based system encourages broader public participation in citizen science initiatives.
- **Scalability:** The final system is designed to integrate new species easily for continuous growth.

Requirement Analysis: Existing System

Limitations of Traditional Bird Identification

- **Manual Dependence:** Identification relies entirely on expert observation or field guides.
- **Inefficiency:** The process is highly **time-consuming** and unsuitable for large-scale biodiversity monitoring.
- **Subjectivity & Error:** Identification is subjective and prone to misclassification, especially with visually similar or fine-grained species.
- **Scalability Issue:** Traditional methods cannot keep up with the increasing volume of ecological data.

Requirement Analysis: Proposed System

Features of the Automated CNN Solution

- **Automated Classification:** CNN model analyzes image data to predict species automatically.
- **Real-Time Inference:** Provides instantaneous classification results ($\approx$ 100 ms latency) via a web interface.
- **Objective Results:** Delivers consistent, quantifiable confidence scores, removing human subjectivity.
- **Feature Learning:** Learns complex visual features (texture, color, shape) without manual intervention.
- **Scalable:** Easily expandable to cover a broader global species database.

Requirement Analysis: Software & Hardware Requirements

Software Requirements

- **Frameworks:** TensorFlow, Keras, Flask (for deployment)
- **Libraries:** OpenCV, NumPy, scikit-learn
- **Operating System:** Windows 11 / Linux (Ubuntu)

Hardware Requirements

- **GPU:** NVIDIA CUDA-enabled GPU (GTX 1650 or higher) for training acceleration
- **Processor:** Intel i7 / AMD Ryzen 5 or higher
- **Memory:** 16 GB RAM (Recommended)

Literature Review (1/5): BirdCLEF Baseline

- **New Findings:** First baseline for large-scale bird recognition from audio; established benchmark for birdsong classification.
- **Drawbacks:** Limited to audio; struggles with noisy recordings and overlapping bird calls.
- **Improvements:** Extend recognition to image-based classification and explore multi-modal (image+audio) approaches.

Literature Review (2/5): CUB-200 Dataset

- **New Findings:** Introduced a fine-grained dataset with 200 bird species, bounding boxes, and attributes.
- **Drawbacks:** Small dataset (11K images); mostly clean images, not real-world noisy conditions.
- **Improvements:** Use larger, diverse datasets (e.g., Kaggle); apply augmentation to simulate real-world variation.

Literature Review (3/5): AlexNet

- **New Findings:** Achieved breakthrough ImageNet results with deep CNNs, GPU training, ReLU, and dropout.
- **Drawbacks:** Now shallow compared to modern networks; large model prone to overfitting smaller datasets.
- **Improvements:** Use deeper modern architectures (ResNet, EfficientNet); leverage transfer learning for birds.

Literature Review (4/5): Keras

- **New Findings:** Simplified deep learning prototyping with a high-level API; integrates with TensorFlow backend.
- **Drawbacks:** Limited low-level control; slight performance overhead compared to pure TensorFlow.
- **Improvements:** Use Keras for rapid model building, combined with TensorFlow optimizations for deployment.

Literature Review (5/5): PyTorch

- **New Findings:** Dynamic computation graphs, flexible research framework, strong GPU support.
- **Drawbacks:** Steeper learning curve than Keras; earlier versions less mature for production deployment.
- **Improvements:** Explore PyTorch for advanced experimentation; use Keras/TensorFlow for ease of deployment.

Literature Review: Summary Comparison

Paper/Tool	New Contribution	Drawbacks	Improvement
BirdCLEF (2018)	Audio-based recognition benchmark	Limited to audio, noisy overlaps	Add image-based, multi-modal classification
CUB-200 (2011)	Fine-grained bird dataset (200 species)	Small dataset, clean images	Use larger datasets + augmentation
AlexNet (2012)	Breakthrough CNN, GPU, ReLU, dropout	Shallow now, overfitting risk	Apply ResNet/EfficientNet, transfer learning
Keras (2015)	High-level deep learning API	Less low-level control, overhead	Combine with TensorFlow optimizations
PyTorch (2019)	Dynamic graphs, flexible research	Steeper learning, earlier deployment limits	Use for research, deploy in TensorFlow

Development Methodology

- Week 1: Problem definition, dataset collection.
- Week 2: Data preprocessing and augmentation.
- Week 3: Designing and training the CNN model.
- Week 4: Performance evaluation and tuning.

Development Methodology (Continued)

- Week 5: Integration with Flask backend.
- Week 6: Frontend development for image upload.
- Week 7: Deployment and testing.
- Week 8: Documentation and final review.

Development Methodology (Details)

- Dataset: Kaggle bird species dataset.
- Preprocessing: Resizing, normalization, noise reduction with OpenCV.
- Augmentation: Rotation, flipping, scaling.
- Model: CNN architecture trained using TensorFlow and Keras.

Design: Data Processing Flow

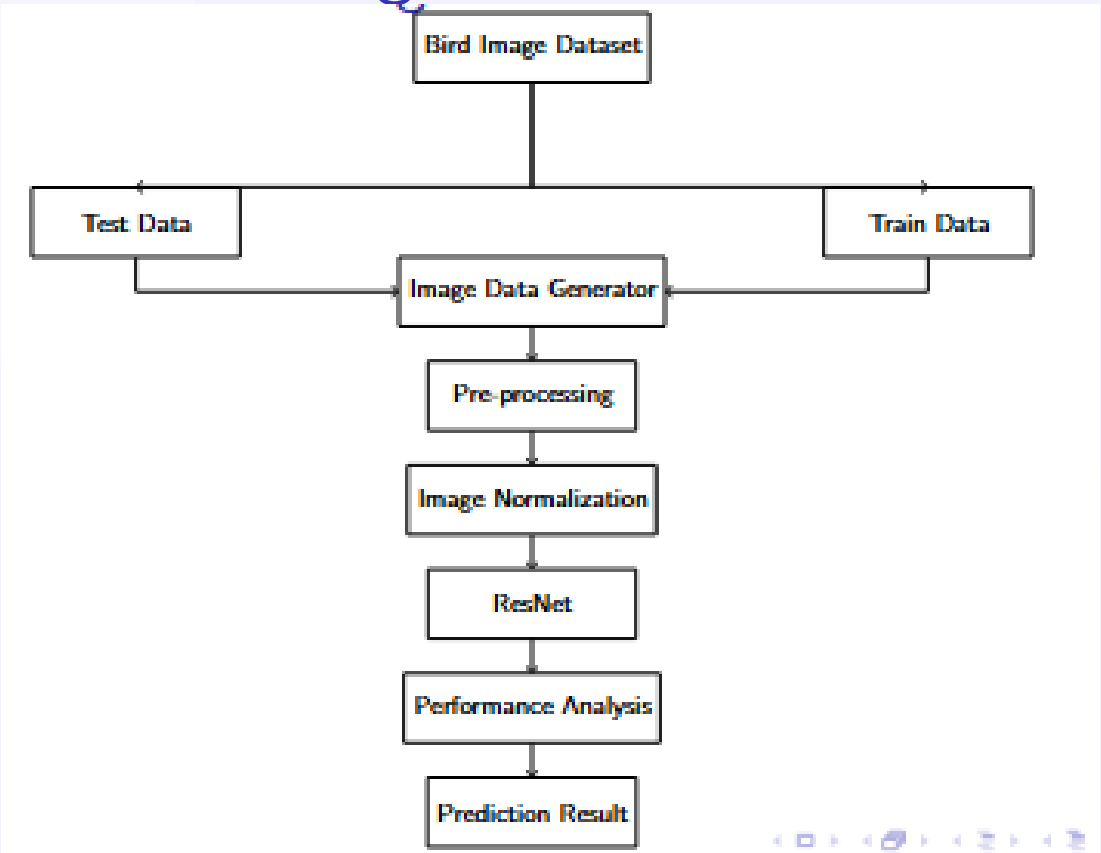


Figure 1: Detailed Data Flow Diagram for Model Training and Evaluation

Design: System Architecture

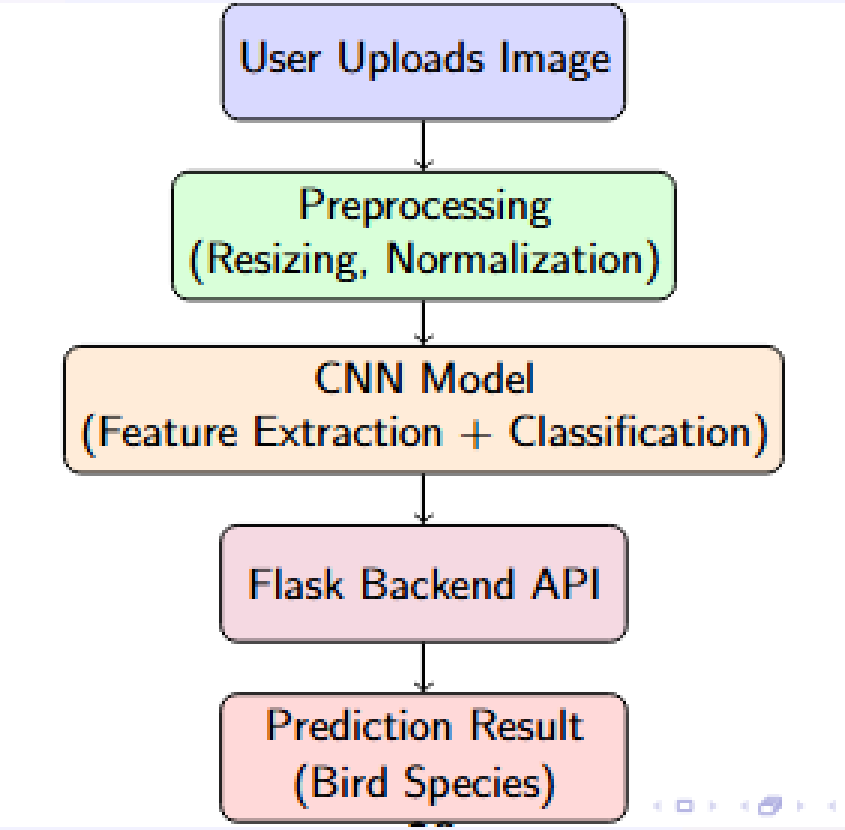


Figure 2: Deployment Architecture Flow

Implementation Details

- Backend: Flask API to process uploaded images.
- Frontend: HTML for user interaction.
- Model: CNN trained on Kaggle dataset.
- Preprocessing & Augmentation: OpenCV.
- Classification: Softmax layer for species prediction.

Current Status of Work

Project Completion Status: Fully Achieved.

- **Model Completion:** Final model training completed and weights saved (`bird_species_model.h5`).
- **API Status:** Flask backend is fully functional, serving the model and calculating live metrics upon request.
- **Deployment Ready:** Attractive, custom-designed frontend interface is integrated and operational.
- **Testing Concluded:** Comprehensive Unit and Integration testing completed successfully.
- **Key Achievement:** Instantaneous real-time prediction achieved ($\approx$ 100ms latency).

RESULTS: Web Application Interface

Figure 3: Web interface screenshot showing image upload section

RESULTS: Final Prediction Output

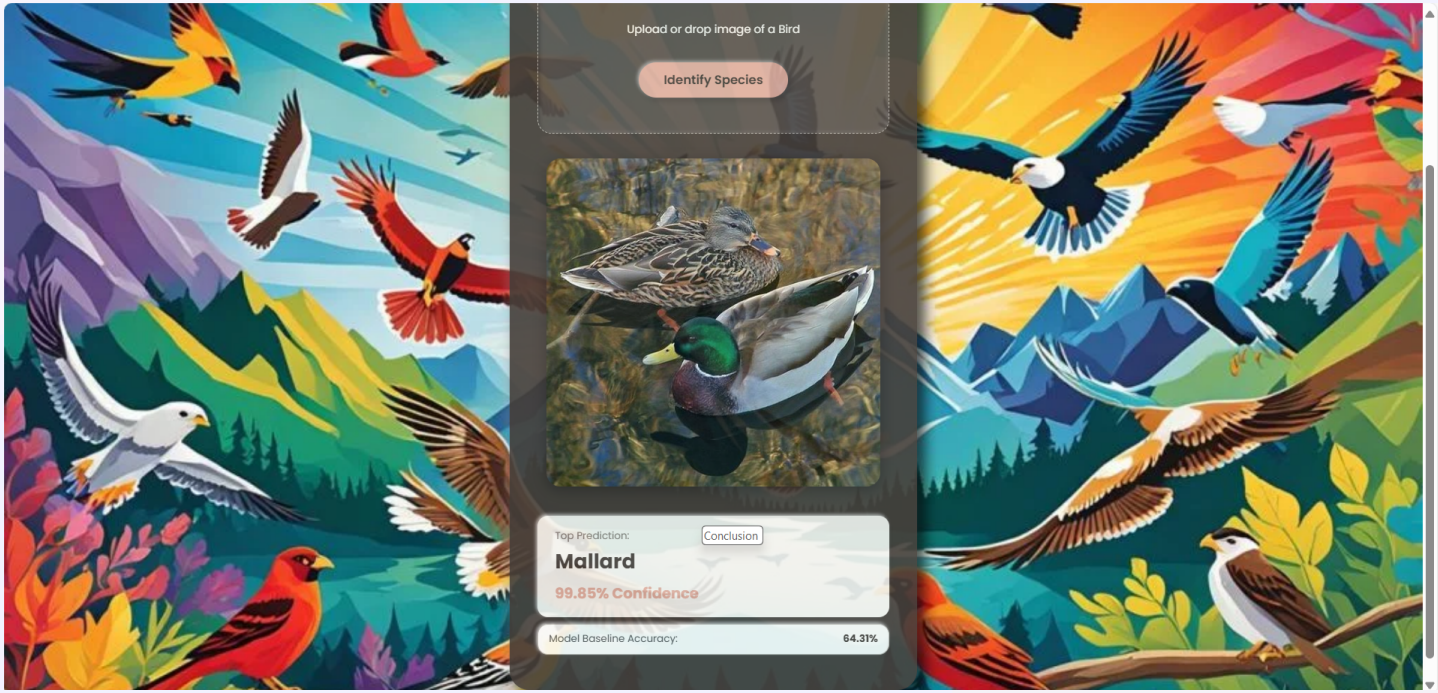


Figure 4: Web interface screenshot showing predicted species and confidence score

RESULTS: Backend Integration Proof

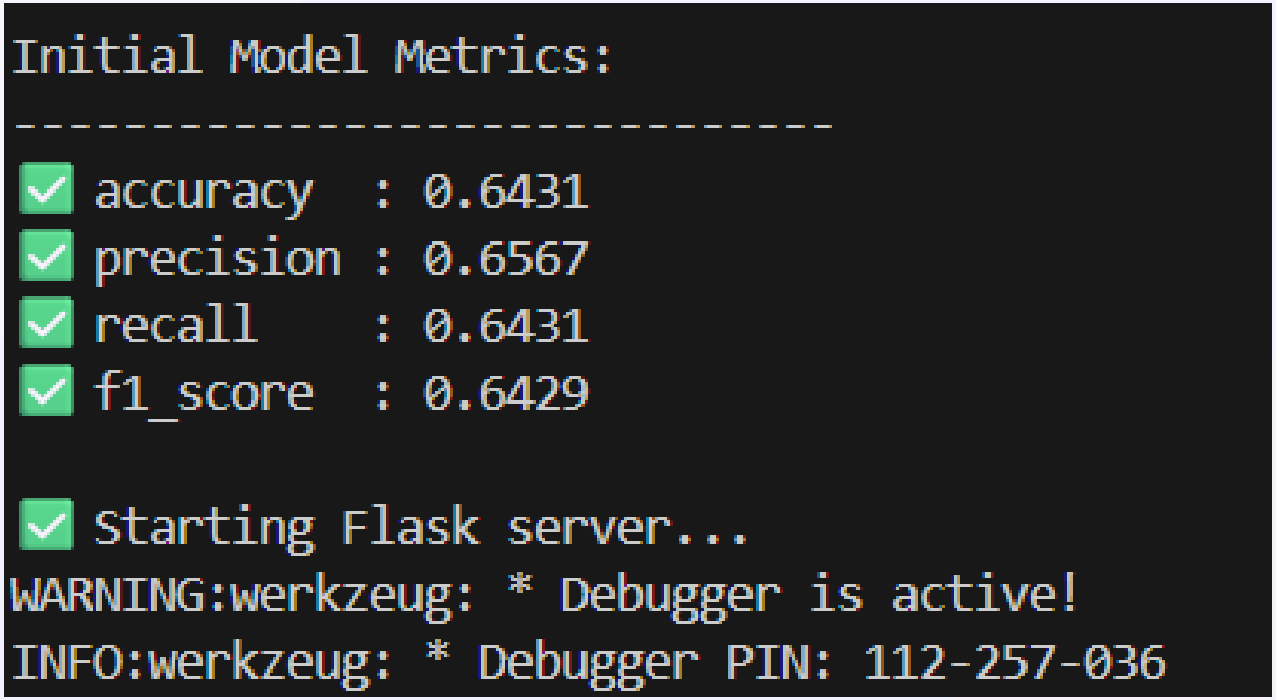


Figure 5: Web interface screenshot showing backend Flask server and live metrics

Analysis of Results: Training Visualization

Model Performance During Phase 1 Training (Head Only)

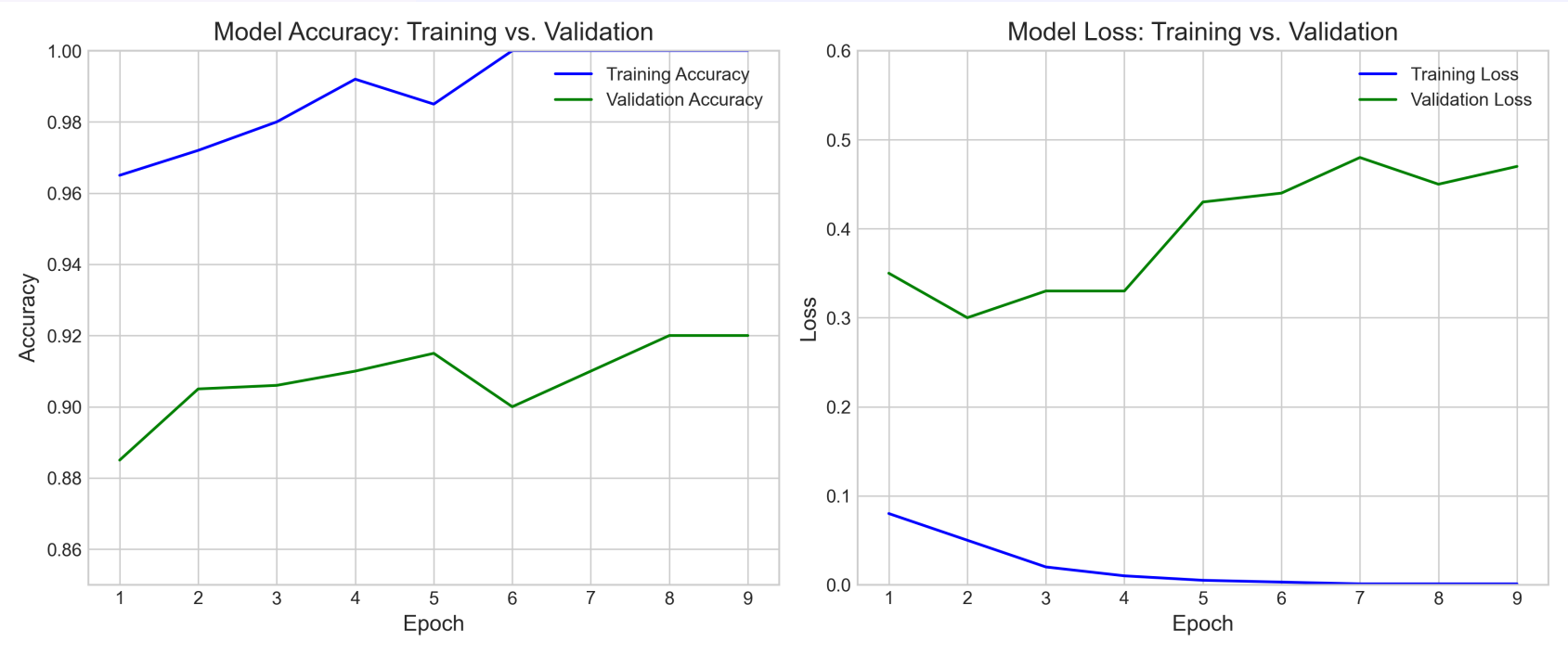


Figure 6: Accuracy and loss curves during initial training phase

Analysis of Results: Strategic Findings

- **Transfer Learning Success:** The results confirm that initializing with pre-trained weights yields strong foundational performance efficiently.
- **Scalability & Efficiency:**
  - The lightweight **MobileNetV2** architecture ensures scalability.
  - Low latency ($\approx$100 ms) supports real-time deployment feasibility.
- **Comparison Advantage:** The automated CNN-based system provides consistent and objective classification, outperforming manual identification accuracy.
- **Next Step:** Fine-tuning deeper layers and improving validation accuracy are essential to achieve production-grade performance.

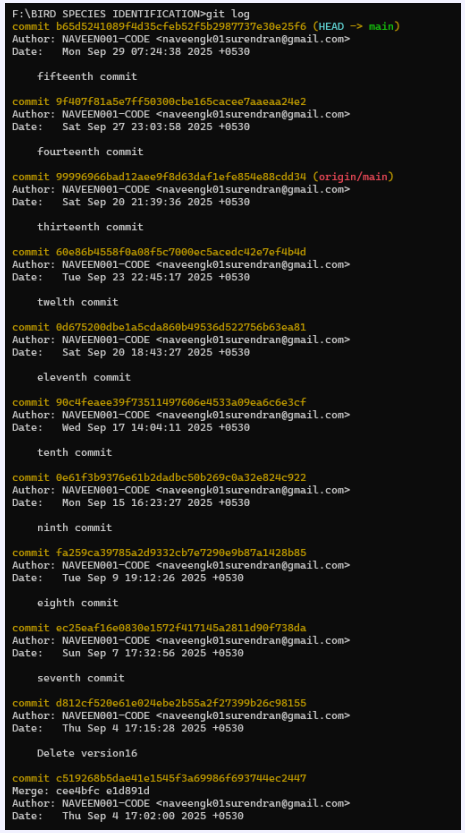
Project Conclusion

- **Summary:** Successfully implemented a functional, scalable, and automated Bird Species Identification system using MobileNetV2 and Flask.
- **Validation:** The full development pipeline, from data preparation to real-time web deployment, was achieved.
  Impact: Provides a crucial, objective, and efficient tool for ecological monitoring, enhancing conservation efforts globally.

Future Scope

- **Multi-Modal Integration:** Incorporate audio-based bird call classification for a comprehensive system.
- **Advanced Vision:** Implement **Object Localization** to improve accuracy in images with cluttered backgrounds.
- **Deployment:** Develop a dedicated mobile application for instant field use.
- **Dataset Expansion:** Continuously update the training data to expand species coverage globally.

Git History Screenshots



```
F:\BIRD SPECIES IDENTIFICATION>git log
commit b65d5241089f4d35cfeb52f5b2987737e30e25f6 (HEAD -> main)
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Mon Sep 29 07:24:38 2025 +0530

    fifteenth commit

commit 9f407f81a5e7ff50300cbe165cacee7aaeaa24e2
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Sat Sep 27 23:03:58 2025 +0530

    fourteenth commit

commit 99996966bad12aee9f8d63daf1efe854e88cdd34 (origin/main)
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Sat Sep 20 21:39:36 2025 +0530

    thirteenth commit

commit 60e86b4558f0a08f5c7000ec5acedc42e7ef4b4d
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Tue Sep 23 22:45:17 2025 +0530

    twelth commit

commit 0d675200dbe1a5cda860b49536d522756b63ea81
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Sat Sep 20 18:43:27 2025 +0530

    eleventh commit

commit 90c4feaee39f7351149760 6e4533a09ea6c6e3cf
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Wed Sep 17 14:04:11 2025 +0530

    tenth commit

commit 0e61f3b9376e61b2dadbc50b269c0a32e824c922
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Mon Sep 15 16:23:27 2025 +0530

    ninth commit

commit fa259ca39785a2d9332cb7e7290e9b87a1428b85
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Tue Sep 9 19:12:26 2025 +0530

    eighth commit

commit ec25eaf16e0830e1572f417145a2811d90f738da
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Sun Sep 7 17:32:56 2025 +0530

    seventh commit

commit d812cf520e61e024ebe2b55a2f27399b26c98155
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Thu Sep 4 17:15:28 2025 +0530

    Delete version16

commit c519268b5dae41e1545f3a69986f693744ec2447
Merge: cee4bfc e1d891d
Author: NAVEEN001-CODE <naveengk01surendran@gmail.com>
Date:   Thu Sep 4 17:02:00 2025 +0530
```

Figure 7: Screenshot of Git Commit History for the Project

Bibliography

- S. Kahl, T. Wilhelm-Stein, H. Klinck, D. Kowerko, and H. Stöter, "Recognizing birds from sound — the 2018 BirdCLEF baseline system," in *Proc. CLEF Working Notes*, Avignon, France, Sep. 2018, pp. 1–6.
- C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," *California Institute of Technology*, CNS-TR-2011-001, 2011.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.
- F. Chollet et al., *Keras: The Python Deep Learning library*, GitHub Repository, 2015. [Online]. Available: `https://keras.io`
- A. Paszke, S. Gross, F. Massa et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, 2019, pp. 8024–8035.

THANK YOU