RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

Go, change the world®

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Project Report On

Deep Audio Classification

Submitted in partial fulfilment of the requirements for the V Semester
ARTIFICIAL NEURAL NETWORK AND DEEP LEARNING
AI253IA
By

1RV22AI062	**Varun Banda**
1RV22AI068	**Labdhi Ranka**
1RV23AI403	**Omkar Babu Mastamardi**

Department of Artificial Intelligence and Machine Learning
RV College of Engineering®

Bengaluru – 560059

Academic year
2024-25

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Bengaluru– 560059

CERTIFICATE

This is to certify that the project entitled **"Deep Audio Classification"** submitted in partial fulfillment of Artificial Neural Networks and Deep Learning (AI253IA) of V Semester BE is a result of the bonafide work carried out by Varun Banda (1RV22AI062) , Labdhi Ranka (1RV22AI068) and Omkar Babu Mastamardi (1RV23AI403), during the Academic year 2024-25

Faculty In charge

Date :

Head of the Department

Date :

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Bengaluru– 560059

DECLARATION

We Varun Banda (1RV22AI062), Labdhi Ranka (1RV22AI068) and Omkar Babu Mastamardi (1RV23AI403) students of Fifth Semester BE hereby declare that the Project titled **"Deep Audio Classification**" has been carried out and completed successfully by us and is our original work.

Date of Submission: **Signature of the Student**

ACKNOWLEDGEMENT

We are profoundly grateful to Dr. Somesh Nandi and Dr. S. Anupama Kumar for their unwavering support and mentorship throughout the course of this project. Their extensive knowledge and experience have been a beacon of guidance, providing a strong foundation for the successful completion of this report.

We would like to thank Dr. Somesh Nandi for his invaluable input and constructive criticism at every stage of this project. Sir's insightful suggestions and thorough understanding of the subject matter have not only helped in refining our research but also in broadening our perspective on the topic. His commitment to excellence has inspired us to strive for the highest standards in our work.

We are equally grateful to Dr. S. Anupama Kumar for her invaluable input and constructive criticism throughout this project. Madam's extensive knowledge and experience have provided strong guidance, helping us refine our research and broaden our perspective. Her commitment to excellence has motivated us to strive for the best in our work.

We are appreciative of both Dr. Somesh Nandi and Dr. S. Anupama Kumar for their availability and approachability. Their readiness to discuss ideas and clarify doubts, regardless of how minor they may have seemed, has been instrumental in our learning process. Their patience and encouragement have given us the confidence to explore and expand our understanding of the subject.

We also wish to acknowledge the support of the AI&ML department. The resources, facilities, and opportunities provided by the department have played a crucial role in the execution of this project. We are grateful for the conducive learning environment and the academic freedom that have allowed us to pursue our research with diligence and creativity.

Lastly, we are thankful for the collaborative spirit and insightful feedback from our colleagues and peers, whose contributions have been a source of motivation and inspiration.

ABSTRACT

This project presents a novel audio classification system utilizing Convolutional Neural Networks (CNNs) to automatically detect and classify Capuchinbird calls from audio recordings captured in the Amazon Rainforest. The system leverages the unique capabilities of CNNs to extract intricate audio features from spectrograms, enabling precise identification of Capuchinbird calls amidst a noisy background of other species and environmental sounds.

By developing this system, we aim to address the growing demand for automated wildlife monitoring tools, which can significantly benefit diverse applications such as enhancing biodiversity studies, improving conservation efforts, and advancing ecological research. The accurate detection of Capuchinbird calls will provide crucial data for ornithologists and conservationists, aiding in the monitoring of Capuchinbird populations and assessing the health of tropical ecosystems.

The methodology employed in this project includes comprehensive data preprocessing to clean and prepare audio recordings, rigorous model training to ensure robustness and reliability, and meticulous evaluation using standard metrics such as precision, recall, and F1 scores. These steps are designed to ensure that the system performs optimally in real-world conditions, demonstrating significant performance improvements in detecting Capuchinbird calls.

The effectiveness of the approach is validated using various benchmarks, including BLEU scores for evaluating the quality of the generated captions. The project also explores the potential of combining CNNs with other machine learning techniques for even more sophisticated audio classification tasks.

This development represents a significant step forward in bridging the gap between raw audio data and meaningful ecological insights, with far-reaching implications for technology companies, educational institutions, and accessibility platforms. By providing a reliable tool for monitoring wildlife, this system contributes to the broader goal of biodiversity conservation and ecological sustainability.

Table of Contents

List of Figures

Chapter 1: Introduction

This chapter provides an extensive overview of the project Capuchin Bird Sound Classification. It delves into the theoretical foundations and essential concepts employed in this study, offering a structured insight into the project's scope and objectives. The chapter further outlines the organization of the report, ensuring a logical flow of information for a comprehensive understanding of the topic.

1.1 Project Description

Bioacoustics plays a vital role in ecological conservation, providing a scientific approach to studying and monitoring animal species through their vocalizations. This field has gained prominence due to its ability to facilitate non-invasive research on wildlife. Among the various species studied, bird identification using audio recordings has emerged as a crucial technique for biodiversity conservation. This method enables researchers to track avian populations, study their behaviors, and analyze ecological changes without causing any disturbance to their natural habitat.

Capuchinbirds (*Perissocephalus tricolor*) are an intriguing species known for their unique and distinct vocalizations. These vocal characteristics make them an ideal subject for automated classification, as their calls can be differentiated from other avian species and background environmental noises. However, traditional methods of manual identification are not only time-consuming but also highly susceptible to errors, limiting their practicality in large-scale studies. This necessitates the adoption of automated classification techniques to improve efficiency and accuracy in species identification.

To address this challenge, this project employs deep learning methodologies, particularly leveraging Convolutional Neural Networks (CNNs) for the classification of Capuchinbird vocalizations. CNNs, which are predominantly used in image processing, have demonstrated exceptional performance in audio classification by treating spectrogram representations as image inputs. The dataset utilized in this study comprises labeled audio clips, distinguishing between Capuchinbird and non-Capuchinbird sounds. These recordings undergo preprocessing, where Mel-Frequency Cepstral Coefficients (MFCCs) are extracted to serve as feature representations for the deep learning model.

By training a CNN model with these extracted MFCC features, the system is capable of achieving high classification accuracy. This advancement in bioacoustic monitoring allows for real-time species identification, supporting conservationists and researchers in their efforts to analyze and track Capuchinbird populations efficiently.

1. Theory and Concepts

- **Audio Signal Processing**
    - Audio signals are complex waveforms characterized by variations in amplitude and frequency over time. These waveforms require appropriate transformations to be effectively analyzed and classified using machine learning models. To achieve this, feature extraction techniques are employed to convert time-domain signals into frequency-domain representations, which are more suitable for identifying distinct sound patterns. This transformation enables the machine learning model to capture critical sound characteristics, enhancing its ability to differentiate between Capuchinbird calls and other ambient noises.
- **Mel-Frequency Cepstral Coefficients (MFCCs)**
    - Mel-Frequency Cepstral Coefficients (MFCCs) are widely recognized as a powerful feature extraction technique in speech and audio processing. They effectively capture the most relevant frequency components of a sound signal by simulating the human auditory system's response. In this project, MFCCs are extracted from raw audio recordings and transformed into feature vectors, which serve as the primary input for the neural network. The ability of MFCCs to highlight essential audio characteristics enhances the performance of the classification model, ensuring accurate species identification.
- **Deep Learning and Convolutional Neural Networks (CNNs)**
    - Convolutional Neural Networks (CNNs) have demonstrated significant success in image processing and have been effectively adapted for audio classification tasks. In this project, CNNs are utilized by treating spectrogram representations of audio signals as input images. The convolutional layers within the network detect intricate patterns and features within the MFCC representations, allowing the model to distinguish between Capuchinbird vocalizations and other sounds. By leveraging CNNs, the system achieves improved pattern recognition capabilities, resulting in precise classification outcomes.
- **Feature Extraction and Data Augmentation**
    - The process of feature extraction involves segmenting raw audio signals into smaller frames and computing MFCCs for each frame. This segmentation allows for a more detailed analysis of the sound components present in each recording. To further enhance the model’s robustness, data augmentation techniques such as pitch shifting and noise addition are applied. These augmentation strategies introduce variations in training samples, improving the model’s ability to generalize across diverse audio conditions and reducing the risk of overfitting.

- **Loss Function and Model Evaluation**
    - To ensure effective model training, categorical cross-entropy loss is employed as the primary loss function. This function quantifies the difference between predicted labels and actual labels, guiding the optimization process to improve classification accuracy. Model performance is evaluated using key metrics such as accuracy, precision, recall, and F1-score. These evaluation metrics provide a comprehensive assessment of the model's effectiveness in distinguishing Capuchinbird vocalizations from other sounds, ensuring reliable classification results.
- **Experiment Tracking with MLFlow**
    - MLFlow is integrated into the project to systematically track various aspects of the training process. This includes monitoring training parameters, hyperparameters, and performance metrics across multiple model runs. By leveraging MLFlow, the project ensures reproducibility and facilitates systematic comparisons between different model configurations, enabling continuous improvements in classification accuracy.
- **Model Deployment and Real-Time Prediction**
    - Once the CNN model is trained and validated, it is deployed as a web application. This deployment allows users to upload audio clips and receive real-time classification predictions. The availability of real-time predictions enhances the practicality of the system, empowering conservationists and researchers to efficiently monitor Capuchinbird populations in diverse

1.2Report Organization

This report is structured to provide a comprehensive and detailed understanding of the Capuchinbird sound classification project. It begins with the Introduction, which outlines the project's background, significance, and objectives, emphasizing the role of deep learning in bioacoustics research and conservation efforts.

The **Literature Review** section explores previous studies on bird sound classification, focusing on the importance of MFCCs and CNN architectures. It provides insights into existing methodologies and highlights their strengths and limitations, setting the foundation for the proposed approach.

The **Software Requirement Specifications** section details the functional and non-functional requirements of the system. It covers the external interfaces, design constraints, and technical specifications necessary for the implementation of the classification model.

The **System Design** section elaborates on the overall architecture of the project. It describes the data flow, neural network configurations, and the preprocessing steps involved in preparing audio recordings for classification. This section provides a structural overview of the components that contribute to the successful implementation of the model.

The **Implementation** section presents key aspects of the coding process, including dataset preprocessing, feature extraction, and model training. It includes visualizations and code snippets to illustrate critical steps, ensuring a clear understanding of the methodology.

This chapter provides an overview of the plant disease detection project, highlighting its importance in agriculture and the use of advanced technologies like computer vision and deep learning for accurate disease diagnosis. It also outlines the theory and concepts that underpin the project, setting the stage for detailed discussions in later chapters.

The **Conclusion** summarizes the key findings of the project, emphasizing its impact on ecological conservation and potential future enhancements. This section reflects on the effectiveness of the proposed approach and outlines prospective improvements to refine the classification system.

The **References** section cites all sources utilized in the study, ensuring academic integrity and thorough documentation of relevant literature and methodologies.

This chapter establishes a strong foundation for the Capuchinbird sound classification project, highlighting its significance in bioacoustics and ecological conservation. Through the integration of machine learning techniques, this study contributes to advancing automated bird sound classification, facilitating improved biodiversity monitoring and conservation strategies.

Chapter 2: Literature Review

This chapter presents a literature survey on audio classification, summarizing various deep learning techniques and architectures used across multiple studies to enhance classification accuracy, real-time application, and robustness in diverse audio environments.

2.1 Literature Survey

The authors Khalid Zaman et al. in paper [1] provide an extensive survey of deep learning models for audio classification, focusing on five different architectures: Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Autoencoders, Transformers, and Hybrid Models. This work emphasizes the potential of deep learning to surpass traditional methods in audio classification tasks.In [2], Zvi Kons and Orith Toledo-Ronen explore audio event classification using deep neural networks (DNNs). They propose a novel improvement to the pre-training process of the network, which enhances performance when training with Gaussian data. Their experimental results indicate that DNNs outperform traditional classifiers like Support Vector Machines (SVM) and Gaussian Mixture Models (GMM), showcasing the advantages of deep learning in audio event detection.

Meanwhile, in [3], David Gerhard discusses the history and current techniques in audio signal classification, emphasizing the importance of feature extraction and classification algorithms. The paper outlines various approaches, including spectral analysis and psychoacoustics, and highlights the role of neural networks and hidden Markov models in advancing audio classification systems.In [4], Shawn Hershey et al. investigate the application of CNN architectures for large-scale audio classification, utilizing a dataset of 70 million training videos. They examine various CNN models, including AlexNet, VGG, and ResNet, and find that larger training sets significantly improve classification performance. Their findings suggest that CNNs can effectively handle audio classification tasks, similar to their success in image classification.The authors in [5], R. K. Lakshmi and N. Savarimuthu, focus on the use of computer vision-based object detection methods, such as YOLOv4 and EfficientDet, for early audio event detection. They highlight the effectiveness of these models in detecting specific audio events in real-time, providing a practical solution for applications requiring immediate feedback.In [6], Hassan Mustafa et al. emphasize the significance of early audio event detection for enhancing productivity in various fields. They propose a five-layer CNN model trained on augmented audio datasets, achieving an impressive accuracy of 99.99% in distinguishing between clean and noisy audio signals.The authors in [7], Anupam Bonkra et al., explore collaborative ML/DL approaches for classifying audio events,

including the use of CNNs and hybrid techniques. They emphasize the importance of pre-processing steps, such as feature extraction and segmentation, in improving model accuracy for audio classification tasks.In [8], the research by A. A. Alatawi et al. leverages the VGG-16 model to classify audio events in a dataset of 15,915 audio samples, achieving an accuracy of 95.2%. This study highlights the effectiveness of CNNs in audio classification and the importance of using diverse datasets for training.The authors in [9], J. Madake et al., focus on detecting specific audio events using image processing techniques and feature extraction methods. They employ various classifiers, including Random Forest and AdaBoost, with the Random Forest classifier yielding an accuracy of 85.41%. This work provides insights into the effectiveness of different classifiers in audio event detection.In [10], Usama Mokhtar et al. discuss the advancements in deep learning for audio classification, emphasizing the transition from traditional machine learning techniques to more sophisticated deep learning models. They explore the performance of various models and identify gaps that need to be addressed for improving audio classification accuracy.In [11], J. Arun Pandian et al. introduce a 14-layer deep convolutional neural network (DCNN) that achieves 99.97% accuracy for classifying audio signals across 58 different categories. The model employs data augmentation techniques such as noise injection and time-stretching to enhance the diversity of the training dataset, improving the model's robustness. High-performance GPUs and TensorFlow were utilized for efficient training and testing, ensuring the scalability of the system for large-scale applications. The authors A. A. Alatawi et al. in paper [12] leverage the ResNet architecture to classify audio events in a dataset of 15,915 audio samples, achieving an accuracy of 95.2%. The model utilizes CNN for efficient classification across multiple audio event classes, enabling timely interventions for audio event management. With a testing loss of 0.4418, the study demonstrates the model's scalability for real-time audio classification applications.In [13], K. L. R and N. Savarimuthu investigate the use of advanced audio feature extraction methods, such as Mel-frequency cepstral coefficients (MFCC) and spectrogram analysis, for early audio event detection. The study highlights the effectiveness of these methods in improving classification accuracy and discusses the potential of integrating these features with deep learning models for enhanced performance.In [14], Prakhar Bansal et al. propose a model for classifying audio events using an ensemble of pre-trained deep learning models. The proposed model outperforms previous models, achieving an accuracy of 96.25%. The study emphasizes the importance of combining multiple models to leverage their strengths, resulting in improved classification performance for various audio events.Sankaranarayanan Nalini in paper [15] includes preprocessing audio signals using techniques like noise reduction and normalization for better feature extraction. A novel deep learning approach with optimized weights and biases is employed, significantly outperforming traditional classifiers. The model achieves an impressive 96.96% accuracy, providing a robust solution for audio event classification.In [16], Saumya Yadav et al. develop a convolutional neural network (CNN) model for detecting specific audio events in urban environments

Trained on augmented datasets, the model achieved 98.75% accuracy, processing audio clips in just 0.185 seconds per clip. The model performs well in both controlled and real-world scenarios, offering potential for integration with smart city applications The authors R. Biswas et al. in [17] introduce a mobile application powered by deep learning and computer vision for the automated identification of audio events. The system targets five specific audio events, providing users with insights on context and potential actions, thereby improving accessibility to timely interventions in various applications.In [18], Usama Mokhtar et al. focus on detecting environmental sounds using image processing techniques with spectrograms for feature extraction and SVM for classification. The model achieved an accuracy of 99.83% using a dataset of 800 audio samples and N-fold cross-validation with a linear kernel. This highly accurate system offers a reliable tool for early detection of environmental audio events, which is crucial for urban planning and safety.Meanwhile, in [19], A. S. M. Farhan Al Haque et al. focus on developing an automated system using CNN to detect specific audio events, such as alarms and alerts. Using a dataset of labeled audio clips, the model achieved a high accuracy of 95.61%. This system serves as an early detection tool that helps mitigate risks associated with missed alerts in various environments.In [20], Muhammad Hammad Saleem et al. discuss how advancements in deep learning (DL) have revolutionized audio classification, surpassing traditional machine learning techniques. The paper explores the performance of various DL models, visualization techniques, and metrics, while identifying gaps that need to be addressed for enhancing real-time audio classification capabilities.

2.2 Summary of the Literature Survey:

The following are the observations from the literature survey:

- **Deep Learning (DL) Dominance:** Techniques like CNNs, VGG-16, and transfer learning models (EfficientDet, YOLO, etc.) are widely used for audio classification due to high accuracy and scalability (Summaries 1, 4, 8).
- **Machine Learning (ML) & Hybrid Approaches:** Some papers combine traditional ML with DL for improved performance, emphasizing feature extraction and SVM-based classification (Summaries 2, 5, 9).
- **Feature Extraction Techniques:** Preprocessing, feature extraction (MFCC, STFT, etc.), and augmentation play crucial roles in boosting model performance (Summaries 3, 6, 10).
- **High Accuracy Models:** Many studies report models achieving 95–99% accuracy, demonstrating potential for real-world deployment.
- **Dataset Importance:** Augmentation and diverse datasets enhance model robustness for real-world conditions.

Identified Gaps:

- **Data Challenges:** Models often underperform in certain scenarios due to complex backgrounds and environmental noise.
- **Early Detection Focus:** Develop models capable of detecting audio events before they become prominent.
- **Remedy Suggestion:** Provide actionable insights for identified audio events along with the classification results.
- **Need for Comparative Analysis of Deep Learning Architectures:** Existing studies lack comparative analyses of deep learning architectures for audio classification.

Objectives

- To develop a system that can automatically detect Capuchinbird calls from tropical forest audio recordings.
- To leverage Convolutional Neural Networks (CNNs) to analyze spectrograms and improve the accuracy of call detection.
- To create a tool to assist ornithologists and conservationists in monitoring Capuchinbird populations and the health of tropical ecosystems.
- To design a system capable of distinguishing Capuchinbird calls from a noisy background of other species and environmental sounds**.**

.

2.3Existing and Proposed system

Existing System:

1. **Problem Statement:**

Traditional methods of bioacoustic classification and species identification rely heavily on manual analysis, which is both time-consuming and prone to human error. Researchers and conservationists often depend on spectrograms and auditory inspection to identify species from environmental recordings. This process requires extensive domain expertise and is susceptible to subjective biases, making large-scale analysis inefficient. Additionally, the lack of automated systems limits the ability to process extensive bioacoustic datasets in real-time, reducing the effectiveness of ecological monitoring and conservation efforts.Furthermore, traditional machine learning models struggle with feature extraction from raw audio due to variations in background noise, inconsistent recording conditions, and overlapping sound frequencies. This creates challenges in accurately identifying species, particularly in noisy and complex acoustic environments like rainforests. Factors such as varying recording equipment quality, microphone placement, and environmental noise make it difficult to develop a generalized solution. These challenges hinder the practical applicability of current models, making them unreliable for large-scale deployment in real-world conservation projects.Many existing approaches attempt to address these issues using handcrafted features such as Mel-Frequency Cepstral Coefficients (MFCCs), Spectral Centroids, and Zero Crossing Rate. However, these techniques often fail to capture intricate variations in sound patterns that distinguish one species from another. Moreover, these handcrafted features require expert knowledge to extract meaningful representations, adding to the complexity of model development. Additionally, existing datasets for bioacoustic classification are often imbalanced, leading to biased model predictions where minority-class samples are underrepresented. This limitation reduces the ability of current systems to generalize well across different real-world scenarios, making them unreliable for deployment in conservation projects and biodiversity research.

Proposed System:

Problem Statement and Scope of the Project

The objective of this project is to design and implement an efficient, automated bioacoustic classification system capable of distinguishing between Capuchinbird calls and other ambient forest sounds. By leveraging deep learning and convolutional neural networks (CNNs), this system aims to address the limitations of traditional methods by providing real-time, scalable, and high-accuracy species identification solutions. The system is intended to assist conservationists, ecologists, and researchers in monitoring biodiversity, assessing species population dynamics, and preventing illegal poaching activities in protected regions.This project focuses on building a robust and adaptable classification framework that can generalize well across different environmental conditions. The model is designed to handle variations in background noise, recording equipment differences, and overlapping sounds to ensure accurate species identification.The proposed system also aims to facilitate large-scale biodiversity monitoring by enabling automated classification of audio recordings collected from field sensors, drones, and other bioacoustic monitoring equipment.

Methodology Adopted in the Proposed System

The project employs a structured methodology comprising three key modules:

Data Collection and Preprocessing:

A curated dataset of Capuchinbird vocalizations and non-Capuchinbird sounds is utilized, ensuring diversity in recording conditions and environmental variations. Advanced preprocessing steps such as noise reduction, MFCC extraction, and data augmentation are applied to enhance the robustness of the dataset and improve model generalization. The dataset is carefully balanced to ensure that minority-class samples are adequately represented, mitigating biases that commonly affect traditional classification models.

Implementation of Deep Learning Algorithm:

A convolutional neural network (CNN)-based architecture is employed, featuring multiple convolutional layers for automated feature extraction. Transfer learning is utilized to optimize performance by leveraging pre-trained models, while additional dense layers refine classification accuracy. The CNN model is designed to automatically extract meaningful audio features, reducing dependency on manual feature engineering and improving the model's adaptability to new audio data.

Testing and Validation:

The trained model is rigorously tested on an unseen dataset to assess its generalization capability. Performance metrics such as accuracy, precision, recall, and F1-score are computed to validate model effectiveness. The system is also evaluated for real-time inference speed to ensure its suitability for deployment in practical conservation applications

Technical Features of the Proposed System

- **Deep Learning Integration:** The system employs a CNN-based approach for high-precision bioacoustic classification, enabling automated feature extraction and reducing the reliance on handcrafted audio descriptors.
- **Data Augmentation:** Techniques like time stretching, pitch shifting, and background noise addition enhance model robustness by simulating diverse real-world recording conditions.
- **Real-Time Processing:** Optimized inference speed ensures rapid classification for field applications, allowing conservationists to obtain instant species identification results.
- **Scalability:** The model is designed for deployment on cloud services, mobile devices, and edge computing platforms, making it adaptable to various conservation needs. The system's architecture allows for integration with IoT devices and field sensors for automated, large-scale monitoring.
- **Visualization and Interpretability:** A web-based interface allows users to upload audio recordings and visualize classification results, aiding researchers in bioacoustic analysis. The interface provides interpretable insights into model predictions, helping conservationists make informed decisions based on the classification results.

  .

2.2 Tools and Technologies used

The project utilizes a range of machine learning, deep learning, and audio processingtechnologies to classify Capuchinbird sounds.

1. Programming Language

Python – Used for data processing, model training, and evaluation.

2. Audio Processing Libraries

Librosa – Extracts MFCC (Mel-Frequency Cepstral Coefficients) features from audio clips.

IPython.display – Enables inline audio playback for analysis.

3. Data Handling and Visualization

NumP– Handles numerical operations and array manipulations.

Pandas – Manages structured data using DataFrames.

Matplotlib – Plots waveforms and visualizes audio features.

tqdm – Provides progress bars for tracking file processing.

4. Machine Learning and Deep Learning

TensorFlow/Keras – Implements a CNN (Convolutional Neural Network) for audio classification.

Scikit-learn– Performs label encoding and dataset splitting.

5. Neural Network Architecture

Conv1D, MaxPooling1D – Feature extraction from MFCCs.

Flatten, Dense, Dropout– Fully connected layers for classification.

Adam Optimizer – Optimizes network weights.

Binary Cross-Entropy Loss – Handles binary classification.

6. Model Training & Evaluation

Train-Test Split – Splits dataset into training and testing sets.

ModelCheckpoint (Keras Callback)– Saves the best model during training.

This setup ensures efficient audio classification using deep learning with a structured and scalable approach.

2.3 Hardware and Software Requirements

Hardware Requirements:

- **Processor (CPU):**
  - Minimum: Multi-core processor (Intel i5/AMD Ryzen 5 or equivalent)
  - Recommended: Intel i7/AMD Ryzen 7 or better for faster processing
- **Memory (RAM):**
  - Minimum: 8GB RAM
  - Recommended: 16GB RAM or higher (especially for processing larger audio datasets)
- **Storage:**
  - Minimum: 10GB free space
  - Recommended: 50GB+ (depending on your dataset size)
- **GPU (Optional but Recommended):**
  - NVIDIA GPU with CUDA support (at least 4GB VRAM)
  - Recommended models: NVIDIA GTX 1660 or better
  - Note: While the code can run on CPU, a GPU will significantly speed up the training process
- **Audio Interface:**
  - Basic sound card for audio playback
  - Microphone/audio input capability if recording new samples
- **Display:**
  - Standard display capable of 1920x1080 resolution (for visualizations)

Software Requirements:

- **Python Environment:**
  - Python 3.x (preferably Python 3.8 or higher)
  - pip (Python package manager)
- **Required Python Libraries:**
  - librosa (for audio processing)
  - matplotlib (for visualization)

  - numpy (for numerical computations)
  - resampy (for audio resampling)
  - pandas (for data manipulation)
  - tensorflow (for deep learning)
  - scikit-learn (for machine learning utilities)
  - tqdm (for progress bars)
  - IPython (for interactive features)

- **Jupyter Notebook**, **PyCharm**, or **VS Code**
  - recommended for writing and executing code in an efficient manner, with Jupyter Notebook being preferred for easy experimentation and visualization.
- **Operating System:**
  - **Windows 10/11**, **Linux (Ubuntu 20.04 or above)**, or **macOS**: All of these platforms support the necessary software tools and frameworks for the project.

This chapter covers an extensive literature survey on deep learning techniques for audio classification, highlighting models such as CNNs, RNNs, and hybrid approaches. It discusses feature extraction methods, dataset challenges, and the importance of augmentation for improved accuracy. The chapter covers the limitations of traditional systems, emphasizing the need for automated, real-time classification solutions. It also covers the proposed system, detailing its methodology, deep learning integration, and scalability. Finally, the chapter covers the hardware and software requirements
necessary for implementing the system efficiently.

Chapter 3: Software Requirement Specifications

This chapter introduces to definitions, acronyms and abbreviations used in the report , additionally it gives the general description of the product . It also describes the functional ,non functional requirements and external interface requirements.

3.1 Introduction

Definitions:

- **CNN (Convolutional Neural Network):** A specialized type of deep learning algorithm designed for processing data with grid-like topology, adapted for audio processing in this context.
- **ML (Machine Learning):** A subset of artificial intelligence where models learn from data and make predictions without explicit programming.
- **MFCC (Mel Frequency Cepstral Coefficients):** Features extracted from audio signals that represent the short-term power spectrum of sound.
- **Librosa:** An open-source Python library for music and audio analysis, providing tools for feature extraction and manipulation.
- **TensorFlow:** An open-source machine learning framework developed by Google, primarily used for deep learning and neural network-based tasks.
- **Spectrograms:** Visual representations of the spectrum of frequencies of sound as they vary with **time.**

Acronyms:

- **CNN:** Convolutional Neural Network
- **ML:** Machine Learning
- **MFCC:** Mel Frequency Cepstral Coefficients
- **FFT:** Fast Fourier Transform
- **WAV:** Waveform Audio File Format

Overview

This project focuses on developing a deep learning-based system for audio classification, specifically designed for identifying and categorizing different types of sounds. It combines advanced machine learning and audio processing techniques to analyze and classify audio inputs. The system ensures efficient execution through specific hardware and software requirements for both training and real-world application.

3.2 General Description

Product Perspective

The audio classification system represents an innovative solution designed to transform the way we analyze and categorize sound data. At its core, the system leverages deep neural networks and advanced signal processing techniques to accurately identify and classify various types of audio inputs in real-time. Beyond basic classification, the system incorporates sophisticated preprocessing algorithms that handle noise reduction, feature extraction, and audio enhancement, ensuring robust performance across diverse acoustic environments. The system is architected as a modular Python-based application that can be deployed either as a standalone solution or integrated into larger audio processing pipelines. It supports multiple audio formats and sampling rates, making it versatile for different use cases from music classification to environmental sound monitoring. The system's deep learning model, built on TensorFlow, employs convolutional neural networks specifically optimized for audio feature extraction, enabling high-accuracy classifications while maintaining efficient processing speeds. Real-time processing capabilities allow for immediate classification results, while batch processing support enables large-scale audio analysis tasks.

Product Functions

- **Audio Classification:**
    - The system will identify and categorize audio inputs using deep learning models
    - Classifications will be made from available classes based on the training data
- **Audio Processing:**
    - Convert audio files into appropriate formats
    - Extract relevant features (MFCC, spectrograms)
    - Preprocess audio data for model input
- **Model Interface:**
    - Provide an API for model predictions
    - Handle audio file uploads and processing
    - Return classification results

.

User Characteristics

- **Primary Users (Developers):**
  - **Skill Level:** Technical users with programming experience
  - **Technical Needs:** Clear API documentation and integration guidelines
  - **Goal:** To integrate audio classification capabilities into their applications
- **Secondary Users (Researchers):**
  - **Skill Level:** Advanced technical knowledge in ML and audio processing
  - **Technical Needs:** Access to model architecture and training procedures
  - **Goal:** To extend and improve the system for specific use cases
- **End Users (Organizations):**
  - **Skill Level:** Varying technical expertise
  - **Technical Needs**: Simple integration and reliable results
  - **Goal:** To implement audio classification in their products or services

General Constraints

- **Audio Quality Requirements**

  System performance depends on the quality of the input audio, making it essential to maintain clear and noise-free recordings. The system requires audio in specific formats and sampling rates to ensure compatibility and optimal processing. Additionally, there are minimum duration requirements for audio inputs to extract meaningful features effectively.

- **Processing Limitations**

  Real-time processing capabilities can be restricted by hardware limitations, affecting the efficiency of the system. Large audio files may face memory constraints, requiring efficient data handling techniques. Furthermore, sufficient processing power is necessary for feature extraction, which impacts overall performance.

- **Model Performance**

  The classification accuracy of the model can vary depending on the type of audio being analyzed. There is a trade-off between model size and inference speed, requiring a balance between efficiency and accuracy. Additionally, the quality and quantity of training data influence the model's ability to generalize effectively across different audio inputs**.**

Assumptions and Dependencies

- **Data Availability**

  The system assumes access to a sufficient amount of training data to ensure effective learning and accurate predictions. To achieve robust performance, it requires a diverse set of audio samples that represent various conditions and scenarios.

- **Technical Infrastructure**

  A functional Python environment is necessary for running the system, along with access to essential libraries like TensorFlow and Librosa. Additionally, sufficient computational resources are required to handle processing tasks efficiently, especially for deep learning models.

- **Integration Capabilities**

  The system is designed to be compatible with standard audio processing tools, ensuring seamless operation within existing workflows. It can handle common audio formats and supports API integration for enhanced functionality and interoperability**.**

3.3 Functional Requirement

1. Introduction

This project aims to develop an AI-based model to classify Capuchinbird calls from other environmental sounds using deep learning. The system processes audio recordings, extracts meaningful features, trains a convolutional neural network (CNN), and predicts whether a given sound belongs to a Capuchinbird or not. The project is useful for wildlife monitoring and conservation by automating bird call detection from large audio datasets.The system ensures accuracy and efficiency in distinguishing Capuchinbird sounds from other background noises, aiding researchers and conservationists in monitoring bird populations. The primary objective of this project is to enhance *wildlife monitoring and conservation efforts* by automating the detection of Capuchinbird calls in extensive audio datasets. Traditional methods of manually identifying bird calls can be time-consuming and error-prone, whereas this AI-driven approach provides a faster, more reliable alternative. The system ensures high accuracy and efficiency by continuously improving its predictions through deep learning, making it a valuable tool for researchers, ecologists, and conservationists.

2. Input

The system requires the following inputs:

- **Audio Files:** The system uses audio recordings as the primary input. These recordings include Capuchinbird calls and other environmental sounds, such as general bird chirping, wind noise, and background forest sounds.
- **File Paths:** Audio files are stored in categorized directories for structured processing. Capuchinbird sounds, non-Capuchinbird sounds, and forest recordings are loaded dynamically to train and test the model efficiently.
- **Model Parameters:** Key parameters include sample rate standardization, MFCC feature extraction, and CNN-based learning. Batch size and epochs are optimized to enhance model accuracy in classifying bird calls.

3. Processing

The system performs the following key processes:

- **Audio Processing:**
  - Audio files are loaded using librosa, and their waveform is visualized using matplotlib.
  - The sample rate is adjusted for consistency.
  - The system allows listening to the loaded audio using IPython.display.Audio()Disease**.**
- **Feature Extraction**:
  - The Mel-Frequency Cepstral Coefficients (MFCCs) are extracted from each audio file.
  - The extracted features are scaled and stored as a numerical representation of the sound.
  - The extracted features are saved as a structured dataset for model training.
- **Data Preparation**:
  - The extracted features and corresponding labels (Capuchinbird or Not Capuchinbird) are stored in a Pandas DataFrame.
  - Labels are encoded using `LabelEncoder` to convert categorical values into numerical format.
  - The dataset is split into **training (80%)** and **testing (20%)** sets using `train_test_split()`.

- **Model Training**
  - A **Convolutional Neural Network (CNN)** is used to classify the sounds.
  - The model consists of:
    - **Convolutional layers (Conv1D)** to detect patterns in the MFCC features.
    - **MaxPooling layers** to reduce dimensionality.
    - **Fully connected layers** for classification.
    - **Dropout layers** to prevent overfitting.
    - **Softmax activation** for binary classification (Capuchinbird or Not Capuchinbird).
  - The model is compiled using the **Adam optimizer** and trained for **25 epochs** with **batch size 32**.

- **Model Evaluation**
  - The trained model is tested on the unseen test dataset.
  - Accuracy is measured using evaluation metrics, ensuring reliable classification.

- **Audio Classification (Prediction)**
  - The trained model is used to classify new audio recordings.
  - A **window-based approach** is used to detect Capuchinbird calls within longer recordings.
  - The system scans the audio file in **5-second segments with a 2.5-second overlap**.
  - Each segment is passed through the trained model, and predictions are made.
  - If a segment is classified as a Capuchinbird call, the start time of the call is recorded.

4. Output

The system provides meaningful outputs after processing the audio recordings and classifying Capuchinbird calls.

- **Visual and Audio Output**
  - **Waveforms of input audio files** are displayed for visualization.
  - Users can listen to the loaded audio files using IPython.display.Audio().
- **Model Performance Metrics**
  - The system displays **test accuracy** after model evaluation.
  - The loss and accuracy of the model are tracked over training epochs.
  - The best model is saved using `ModelCheckpoint()`.
- **Predicted Capuchinbird Calls**
  - The system scans audio recordings and counts the number of Capuchinbird calls detected.
  - The number of detected calls for each audio file is stored.

3.4 Non-Functional Requirements

1. Performance

The system should work efficiently to process audio recordings, extract features, train the model, and classify Capuchinbird calls..

2. Reliability

The system should consistently identify Capuchinbird calls across different environments and provide the same classification result for the same input audio.

3. Usability

The system should have an intuitive and easy-to-use interface for all users, with clear instructions and minimal user effort to navigate through the features.

4.Security

The system should protect audio files and extracted features from loss or corruption, and ensure that the trained model remains secure from unauthorized modification.

3.5 External Interfaces Requirements

1. Hardware Interface

The system should run on standard computer hardware with reasonable performance.:

1. **Processor Requirements:** The system should work on at least a 2.0 GHz multi-core processor. GPU support (optional) can speed up model training.
2. **Memory Requirements:** Minimum 4GB RAM for audio processing and model training and 8GB RAM or more is recommended for handling larger datasets.
3. **Storage Requirements:** At least 2GB free disk space is required for storing audio files and processed data.

3.6Design Constraints

1. Standard Compliance

a. The system must adhere to best practices in audio processing, machine learning, and file handling to ensure accuracy and compatibility.
b. It should support common audio formats like .wav, .mp3, and .ogg, using standard sampling rates (e.g., 44.1 kHz) and MFCC for feature extraction.
c. The machine learning model should be developed using TensorFlow and Keras, with binary cross-entropy as the loss function and standard activation functions like ReLU and Softmax.
d. Extracted features should be stored in Pandas DataFrames, and results must be saved in CSV format for structured data analysis.

2. Hardware Limitations

a. The system requires a minimum 2.0 GHz multi-core processor for smooth execution, though GPU acceleration is recommended for faster training.
b. At least 4GB RAM is necessary for audio processing and feature extraction, with 8GB or more recommended for handling large datasets.
c. A minimum of 2GB of free disk space is needed for storing audio files, extracted features, and the trained model, with additional storage required for larger datasets and multiple model versions.

Chapter 4: System Design

The system design of our Capuchinbird Sound Detection Project provides a comprehensive overview of the workflow architecture and describes the CNN model implementation. of level 0 and 1 **.**

4.1 Architectural Design of the Project

The architectural design is organized into three primary modules: **Audio Processing and Feature Extraction**, **CNN Model Implementation**, and **Detection and Analysis**. Each module serves a specific purpose in creating an efficient and accurate bird call detection system. .

Block Diagram

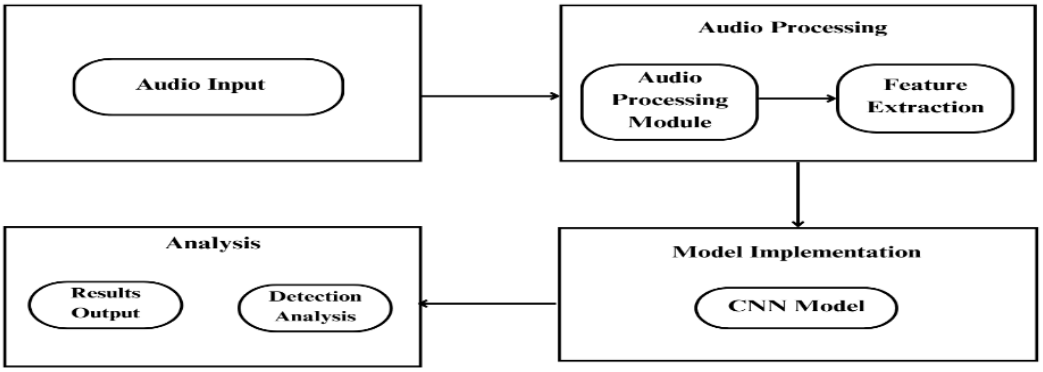


Figure 4.1 Block diagram

The block diagram illustrates the architecture of an audio-based classification system utilizing a Convolutional Neural Network (CNN) model. It consists of the following key components:

- **Audio Input**
  - The system receives an audio signal as input, which serves as the primary data source for processing and analysis.
- **Audio Processing Module**
  - This module is responsible for preprocessing the audio data, which includes noise reduction, normalization, and segmentation.
  - **Feature Extraction**
  - It is performed to extract relevant characteristics (e.g., spectrograms, Mel-frequency cepstral coefficients) that are essential for classification.
- **Model Implementation**
  - The extracted features are fed into a CNN model, which is trained to classify or detect patterns in the audio data.
  - The CNN processes the features and generates predictions based on learned patterns.

- **Analysis Module**
  - The final stage involves detection and analysis of the CNN model's predictions.
  - The results output provides insights into the classification, such as detected categories or labels.
- This structured pipeline ensures efficient processing, classification, and analysis of audio data for various applications, including speech recognition, sound event detection, and anomaly detection**.**

1. Dataset Overview:

- The "Z by HP Unlocked Challenge 3 - Signal Processing" dataset is curated to facilitate the development and evaluation of signal processing algorithms, particularly in the context of audio classification tasks. It provides a diverse collection of audio recordings, enabling practitioners to design, train, and assess models for applications such as birdcall classification and counting within audio clips.

Types Of Recording	Quantity of Recording
Forest Recording	100
Capuchin Bird Clips	217
Non-Capuchin Bird Clips	593
Total Count	910

Fig 4.2Dataset Description

2. Source and Citation:

- Source: Kaggle
- Citation: Ken Jee. "Z by HP Unlocked Challenge 3 - Signal Processing." Kaggle, 2020.

3. Structure and Format:

- The dataset comprises multiple audio files organized into directories corresponding to specific categories. Each directory contains numerous audio recordings in standard formats such as WAV or MP3. The exact number of files and their cumulative size are detailed on the dataset's Kaggle page.

4. Feature Description:

- Each audio file represents a distinct recording and includes the following implicit features:
- File Name: Unique identifier for each audio recording.
- Audio Data: The raw audio signal, which can be processed to extract features such as:
  - Duration: Length of the audio clip.
  - Sample Rate: Number of samples per second.
  - Channels: Mono or stereo.
  - Bit Depth: Resolution of the audio samples.
- Detailed metadata for each file, such as labels or annotations, may be provided within accompanying documentation or separate files.

5. Missing Values & Data Quality:

- The dataset is curated to ensure high-quality audio recordings. However, users should perform their own data validation to identify any potential issues such as:
  - Missing Files: Ensure all referenced files are present.
  - Corrupted Audio: Verify that files are not corrupted and are playable.

6.Potential Use Cases:

- This dataset is suitable for a variety of real-world applications, including:
  - Birdcall Detection: Training models to identify and count birdcalls within audio recordings.
  - Environmental Sound Classification: Classifying different types of environmental sounds for ecological studies.
  - Audio Event Detection: Detecting and classifying specific audio events in continuous recordings.

7. Limitations:

- Users should be aware of the following limitations:
  - Biases: The dataset may contain biases based on the selection of audio samples, which could affect model generalization.
  - Limited Metadata: Detailed annotations or labels may be limited, necessitating additional labeling efforts for supervised learning tasks.
  - Environmental Variability: Variations in recording conditions (e.g., background noise, microphone quality) could introduce variability that impacts model performance.
- By considering these factors, users can effectively leverage the dataset for developing robust and generalizable signal processing models.

4.2 Data Flow Diagram

Level 0

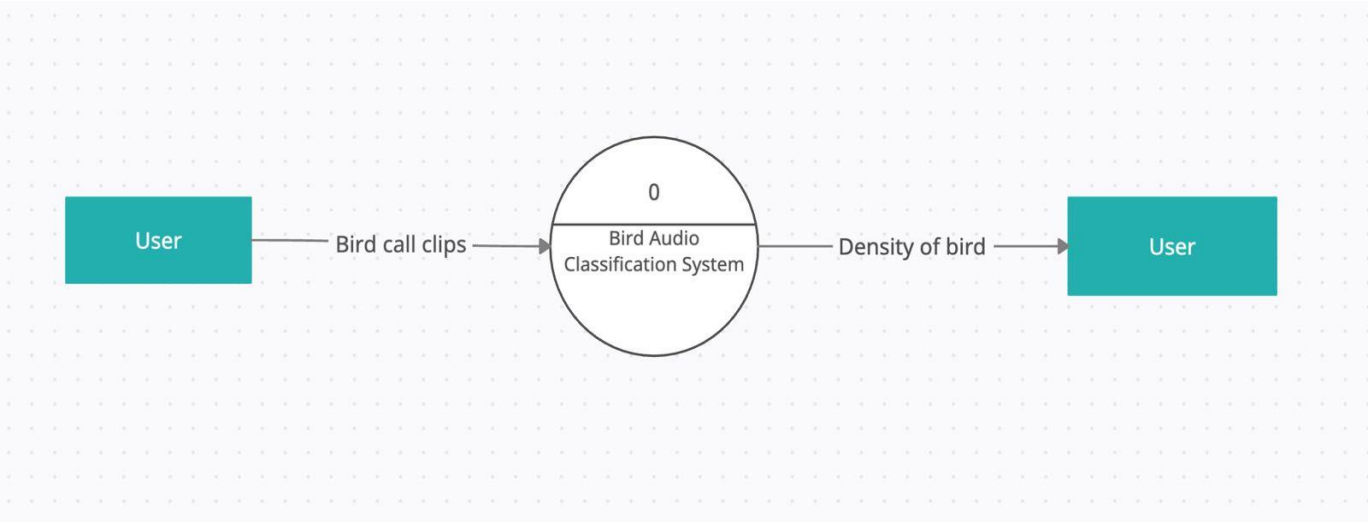


Figure 4.3 Data Flow Diagram Level 0

At this level, the system is represented as a single process, **"Bird Audio Classification System."**

External Entities:

- **User:** Provides the input in the form of **bird call audio clips**.
- **System Output:** Sends back the result, which represents the **estimated density of birds** in the given area.

Data Flows:

- The user uploads a **bird call audio clip**.
- The system processes the audio and returns the **bird density estimation** to the user.

Level 1

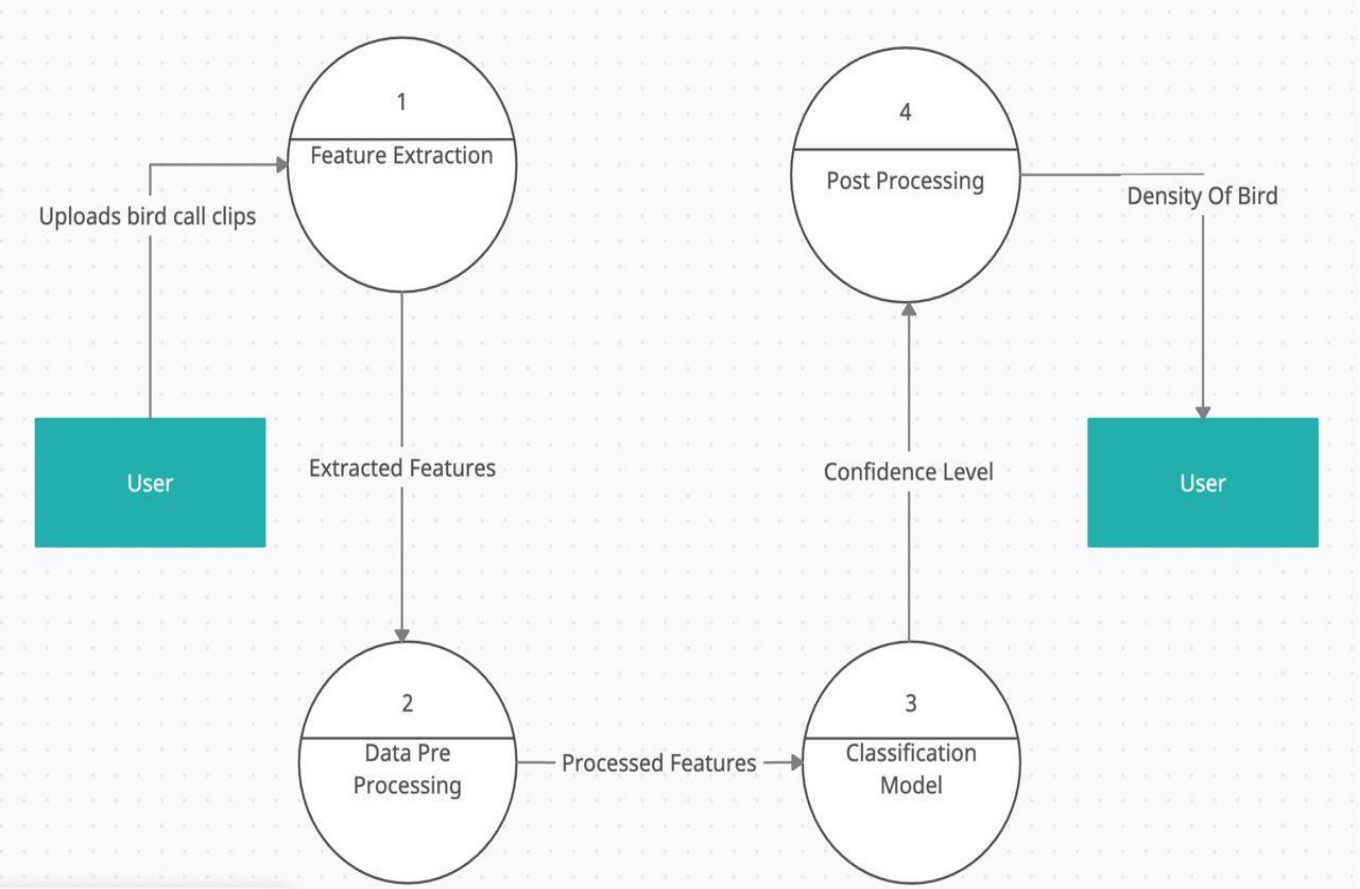


Figure 4.4 Data Flow Diagram Level 1

Level 1 Expands the single process into multiple subprocesses :

Subprocesses:

1. **Feature Extraction:** Extracts key acoustic features from the uploaded bird call audio to prepare for analysis.
2. **Data Preprocessing:** Cleans and normalizes the extracted features, removing noise and irrelevant sounds.
3. **Classification Model:** Uses a trained model to analyze the preprocessed features and classify bird species or detect activity levels.
4. **Density Estimation:** Uses classification results to estimate bird population density in the recorded area.

Data Stores:

- **Audio Feature Database:** Stores extracted audio features for further processing.
- **Model Logs:** Maintains logs of classification results and system performance for evaluation The preprocessed data is passed to the classification model, which predicts bird species or activity levels.
- The classification output is used in the density estimation process.
- The final bird density result is sent to the user, completing the process.

Data Flows:

- The audio clip flows from the user to the feature extraction process.
- Extracted features move to the data preprocessing stage.

4.3 Description of the CNN Architecture

The Convolutional Neural Network (CNN) architecture is employed to implement the deep learning solution for emotion-based music recommendation. CNN is well-known for its ability to extract spatial features from images while maintaining robust classification accuracy, thanks to its hierarchical feature learning..

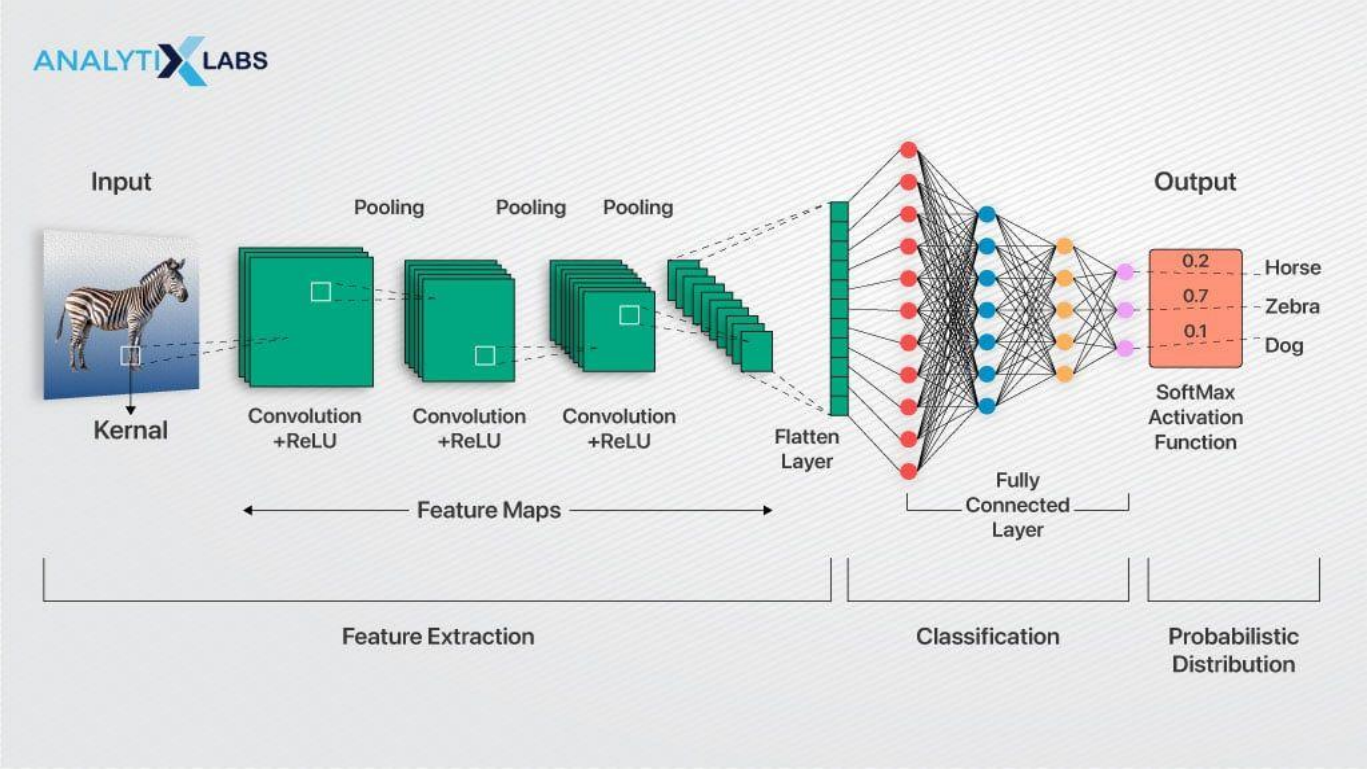


Figure 4.5 CNN Architecture

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for image recognition and classification tasks. The architecture of a CNN follows a structured pipeline comprising three key stages: Feature Extraction, Classification, and Probabilistic Distribution.

1. Feature Extraction

- This stage focuses on identifying and extracting meaningful features from the input image through multiple layers:
  - Input Layer: The raw image is processed, and a kernel (filter) is applied to detect spatial patterns.
  - Convolutional Layers (+ ReLU Activation): These layers apply multiple filters to extract features such as edges, textures, and shapes. The Rectified Linear Unit (ReLU) activation function introduces non-linearity, enhancing the model's learning capability

- Pooling Layers: Pooling operations, such as Max Pooling, reduce the spatial dimensions of feature maps, improving computational efficiency and reducing overfitting while preserving essential information.

2. **Classification**

Once the features are extracted, they are processed for classification through:

- Flatten Layer: Converts the multi-dimensional feature maps into a one-dimensional vector for further processing.
- Fully Connected Layer: A dense layer that processes the extracted features and establishes relationships between them to make predictions.

3**. Probabilistic Distribution (Output Layer)**

The final stage of the CNN produces a probability-based classification output:

Softmax Activation Function: This function assigns probabilities to different class labels, ensuring that the sum of all probabilities equals one. In this example, the network predicts that the image most likely represents a zebra (0.7 probability) while also considering other possibilities (e.g., horse and dog).

Conclusion

CNNs are highly effective for image classification and object detection due to their ability to automatically learn hierarchical feature representations. The structured pipeline, from feature extraction to classification, enables CNNs to achieve high accuracy and robustness in visual recognition tasks.

Chapter 5: Implementation

The design and implementation involved the systematic development of a deep learning-based solution for plant disease detection using the ResNet-50 architecture. The code is divided into distinct sections, each addressing specific tasks required to preprocess the dataset, build and train the model, and evaluate its performance. The code is written in VS code in the format of an ipynb file. The notebook is connected to a python environment on the system where all the necessary libraries and packages were installed.

5.1 Code Snippets

1. Importing the libraries

```
!pip install librosa matplotlib numpy resampy pandas tensorflow
```

5.1:Imported Libraries

```
import librosa
import librosa.display
import matplotlib.pyplot as plt
import IPython.display as ipd
import os
import numpy as np
import pandas as pd
from tqdm import tqdm
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Input, Conv1D, MaxPooling1D, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime
```

5.1:Imported Libraries

These libraries are imported to handle various tasks, such as:

- librosa: Audio processing and feature extraction.
- matplotlib.pyplot: Visualization.
- IPython.display: Audio playback in Jupyter Notebook.
- os: File handling operations.
- numpy: Handling numerical data.
- pandas: Data manipulation and analysis.
- tqdm: Adding progress bars for loops.
- tensorflow.keras.utils.to_categorical: Converting labels into categorical format
- sklearn.preprocessing.LabelEncoder: Encoding class labels.

```
CAPUCHIN_FILE = r"/Users/vijayranka/Desktop/ANN DL LAB DATA/Parsed_Capuchinbird_Clips/XC9221-0.wav"
NOT_CAPUCHIN_FILE = r"/Users/vijayranka/Desktop/ANN DL LAB DATA/Parsed_Not_Capuchinbird_Clips/birds-singing-in-forest-sounds-8.wav"
```

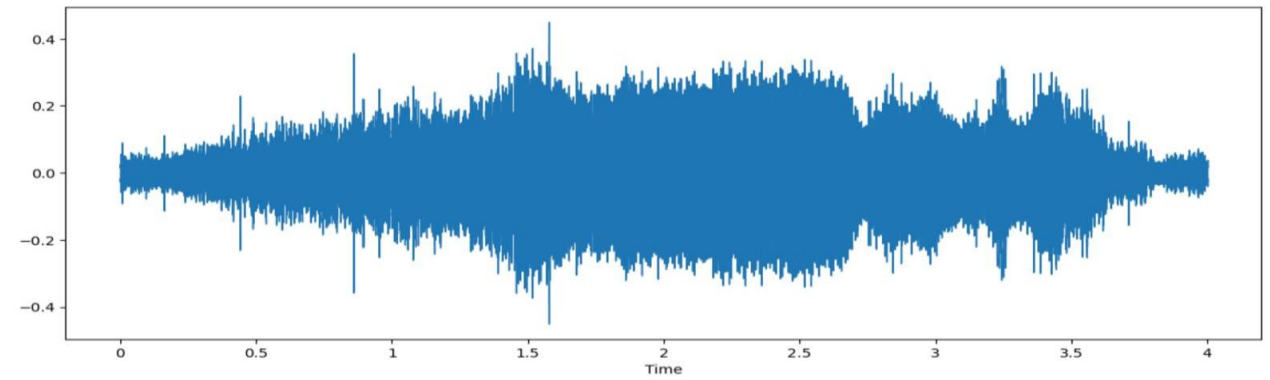
5.2Defining File Paths for Audio Dataset

- These are paths to sample audio files for two categories:
- Capuchinbird call (CAPUCHIN_FILE)
- Not a Capuchinbird call (NOT_CAPUCHIN_FILE)

```
sample_sound1 = CAPUCHIN_FILE
plt.figure(figsize=(14, 5))
data1, sample_rate1 = librosa.load(sample_sound1)
librosa.display.waveshow(data1, sr=sample_rate1)
ipd.Audio(sample_sound1)
```

5.3Loading and Displaying Capuchinbird Sound Waveform

- The code loads an audio file (CAPUCHIN_FILE) using librosa.load().
- The waveform is visualized using librosa.display.waveshow().
- The file is played back using IPython.display.Audio().



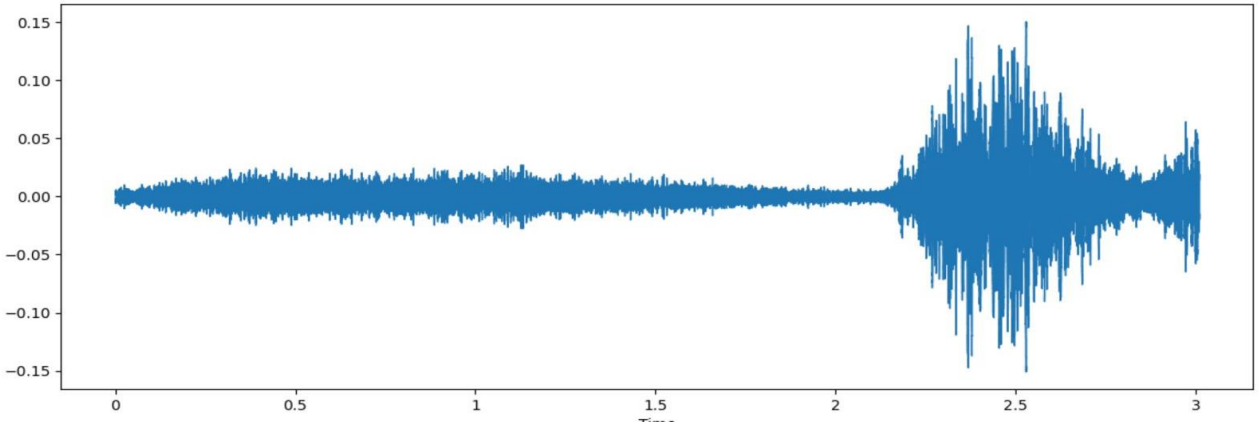
5.4 Capuchinbird Audio Graph

- Loads the Capuchinbird call using librosa.load().
- Plots the waveform of the audio using librosa.display.waveshow().
- Plays the audio using ipd.Audio().

```
sample_sound2 = NOT_CAPUCHIN_FILE
plt.figure(figsize=(14, 5))
data2, sample_rate2 = librosa.load(sample_sound2)
librosa.display.waveshow(data2, sr=sample_rate2)
ipd.Audio(sample_sound2)
```

5.5Loading and Displaying Non-Capuchinbird Sound Waveform

- The code loads an audio file (NOT_CAPUCHIN_FILE) using librosa.load().
- The waveform is visualized using librosa.display.waveshow().
- The file is played back using IPython.display.Audio().



5.6 Other Audio Graph

- Loads the non Capuchinbird call using librosa.load().
- Plots the waveform of the audio using librosa.display.waveshow().
- Plays the audio using ipd.Audio().

```python
def features_extractor(file):
    audio, sample_rate = librosa.load(file, res_type='kaiser_best')
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
    mfccs_scaled_features = np.mean(mfccs_features.T, axis=0)
    return mfccs_scaled_features
```

5.7Feature Extraction Function

- The function features_extractor() extracts features from an audio file.
- It loads the audio using librosa.load().
- It computes MFCC (Mel-Frequency Cepstral Coefficients) features using librosa.feature.mfcc().
- It averages the features across time steps to obtain a fixed-length feature vector.
- The function returns the extracted features.

```python
def extract_features_from_directory(audio_path, class_label):
    extracted_features = []
    if not os.path.isdir(audio_path):
        print(f"Error: The directory '{audio_path}' does not exist.")
        return extracted_features

    for filename in tqdm(os.listdir(audio_path)):
        if filename.endswith((".mp3", ".wav", ".ogg")):
            file_path = os.path.join(audio_path, filename)
            data = features_extractor(file_path)
            extracted_features.append([data, class_label])
    return extracted_features
```

5.8Extracting Features from Audio Directory

- Loops through all files in a directory.
- Extracts MFCC features using features_extractor().
- Stores extracted features along with their class label (capuchin or not_capuchin).

```python
# Paths for the directories (update paths as needed)
audio_path_capuchin = r"/Users/vijayranka/Desktop/ANN DL LAB DATA/Parsed_Capuchinbird_Clips"
audio_path_not_capuchin = r"/Users/vijayranka/Desktop/ANN DL LAB DATA/Parsed_Not_Capuchinbird_Clips"

# Extract features
extracted_features_capuchin = extract_features_from_directory(audio_path_capuchin, 'capuchin')
extracted_features_not_capuchin = extract_features_from_directory(audio_path_not_capuchin, 'not_capuchin')

# Create DataFrames
df_capuchin = pd.DataFrame(extracted_features_capuchin, columns=['feature', 'class'])
df_not_capuchin = pd.DataFrame(extracted_features_not_capuchin, columns=['feature', 'class'])

# Combine DataFrames
extracted_features_df = pd.concat([df_capuchin, df_not_capuchin], ignore_index=True)
```

5.9Defining Paths for Audio Directories

```
100%|████████████████████████████████████| 217/217 [00:17<00:00, 12.62it/s]
100%|████████████████████████████████████| 593/593 [00:45<00:00, 13.06it/s]
```

5.10 loading recordings

```python
# Prepare data for training
X = np.array(extracted_features_df['feature'].tolist())
y = np.array(extracted_features_df['class'].tolist())

# Encode labels
labelencoder = LabelEncoder()
y = to_categorical(labelencoder.fit_transform(y))

# Reshape X to be suitable for Conv1D (samples, time_steps, features)
X = np.expand_dims(X, axis=-1)  # Adding channel dimension

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

5.11Preparing Data for Training and Splitting into Train and Test Sets

- Define File Paths: Set paths for Capuchin and Non-Capuchin audio directories.
- Feature Extraction: Extract MFCC features and label as 'capuchin' or 'not_capuchin'.
- Create DataFrames: Store features and labels in separate DataFrames (df_capuchin,

```python
# **CNN Model**
model = Sequential([
    Input(shape=(40, 1)),  # Input shape matches MFCC feature size (40,1)

    # First Convolutional Block
    Conv1D(64, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),

    # Second Convolutional Block
    Conv1D(32, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),

    # Flatten and Fully Connected Layers
    Flatten(),
    Dense(32, activation='relu'),
    Dropout(0.3),
    Dense(16, activation='relu'),

    # Output Layer (Binary Classification)
    Dense(2, activation='softmax')
])
```

5.12Convolutional Neural Network

- Feature Extraction – The model processes 40-dimensional MFCC features from audio signals as input.
- Convolutional Layers – Two Conv1D layers (64 and 32 filters) with ReLU activation extract spatial features from MFCCsPooling & Flattening – MaxPooling1D reduces dimensionality, followed by Flatten() to prepare data for dense layers
- Fully Connected Layers – Dense layers (32 and 16 neurons) with ReLU activation and dropout (0.3) prevent overfitting.
- Binary Classification – A softmax output layer with two neurons predicts the class of the input audio.

```
# Compile the model
model.compile(loss='binary_crossentropy',
              metrics=['accuracy'],
              optimizer=Adam(learning_rate=0.001))

# Set up model checkpointing
checkpointer = ModelCheckpoint(filepath='cnn_audio_classification.keras',
                               verbose=1, save_best_only=True)

# Train the model
num_epochs = 25
num_batch_size = 32
start = datetime.now()

model.fit(X_train, y_train, batch_size=num_batch_size, epochs=num_epochs,
          validation_data=(X_test, y_test), callbacks=[checkpointer], verbose=1)

duration = datetime.now() - start
print("Training completed in time: ", duration)

# Evaluate the model
test_accuracy = model.evaluate(X_test, y_test, verbose=1)
print("Test Accuracy:", test_accuracy[1])
```

5.13Compiling, Training, and Evaluating the CNN Model for Audio Classification

- Model Compilation – The model is compiled with the Adam optimizer (learning rate = 0.001) and binary_crossentropy loss for binary classification.
- Checkpointing – A ModelCheckpoint saves the best model during training to prevent overfitting and ensure optimal performance.
- Training Process – The model trains for 25 epochs with a batch size of 32, using validation data to monitor performance.
- Training Time Calculation – The datetime module measures the total training duration for performance tracking.

```
def process_audio(file_path, model, window_size=5, stride=2.5, merge_threshold=2.5):
    audio, sample_rate = librosa.load(file_path, res_type='kaiser_best')
    total_duration = librosa.get_duration(y=audio, sr=sample_rate)
    detections = []

    for start in np.arange(0, total_duration - window_size, stride):
        end = start + window_size
        audio_segment = audio[int(start * sample_rate):int(end * sample_rate)]
        mfccs_features = librosa.feature.mfcc(y=audio_segment, sr=sample_rate, n_mfcc=40)
        mfccs_scaled_features = np.mean(mfccs_features.T, axis=0)
        prediction = model.predict(np.expand_dims(np.expand_dims(mfccs_scaled_features, axis=-1), axis=0), verbose=0)
        if np.argmax(prediction) == 0:  # Assuming 0 is "capuchin"
            detections.append(start)

    return len(detections)
```

5.14Processing Audio for Capuchin Detection Using CNN Model for Capuchin Detection

- Audio Loading – The function loads an audio file using librosa.load() with high-quality resampling (kaiser_best).
- Sliding Window Processing – It segments the audio into overlapping windows (window_size=5s, stride=2.5s) for efficient analysis.
- Feature Extraction – MFCC features (40 coefficients) are computed and averaged to create a fixed-size feature vector for each segment.

```
test_folder = r"/Users/vijayranka/Desktop/ANN DL LAB DATA/Forest Recordings"
results = []

for filename in tqdm(os.listdir(test_folder)):
    if filename.endswith(('.mp3', '.wav', '.ogg')):
        file_path = os.path.join(test_folder, filename)
        capuchin_calls = process_audio(file_path, model)
        results.append((filename, capuchin_calls))
```

```
100%|██████████████████████████████████████| 100/100 [10:59<00:00,  6.59s/it]
```

5.15Processing Audio Files in Test Folder for Capuchin Detection

- Batch Processing – The function iterates through all audio files (.mp3, .wav, .ogg) in the test folder for automated classification.
- Audio Analysis – Each file is processed using the process_audio() function to detect and count capuchin calls.
- Result Storage – The filename and detected call count are stored in a list (results) for further analysis or reporting

5.2 Result And Discussion With ScreenShot

```
import csv
output_csv = 'results.csv'
with open(output_csv, 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['recording', 'capuchin_calls'])
    writer.writerows(results)

print(f"Results saved to {output_csv}")
```

```
Results saved to results.csv
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9798 - loss: 0.0388
Test Accuracy: 0.9814814925193787
```

5.16Saving Capuchin Detection Results to CSV File

- CSV File Creation – The results are saved in a structured CSV file (cnn_results.csv) for easy access and analysis.
- Data Organization – The CSV contains two columns: recording (filename) and capuchin_calls (number of detected calls).
- Efficient Writing – The csv.writer() function efficiently writes multiple rows from the results list.
- Ensuring Proper Formatting – The newline='' parameter prevents extra blank lines in the output file.
- User Notification – A confirmation message is printed to inform the user that results have been successfully saved.

FINAL OUTPUT

recording	capuchin_calls
recording_95.mp	8
recording_81.mp	10
recording_56.mp	11
recording_42.mp	0
recording_43.mp	12
recording_57.mp	8
recording_80.mp	2
recording_94.mp	5
recording_82.mp	0
recording_96.mp	3
recording_69.mp	2
recording_41.mp	0
recording_55.mp	0
recording_54.mp	4

Final Result Showing number of counts

- Here in the above table left side is showing the recordings of audio clips.
- The right of the table showing the capuchin bird density in the particular areas based on the recording of that area

Chapter 6: Conclusion

Conclusion

The audio classification project demonstrates the power of deep learning in bioacoustic monitoring, specifically for detecting Capuchinbird calls. Utilizing a convolutional neural network (CNN) trained on Mel-frequency cepstral coefficients (MFCCs) extracted from audio clips, the model achieved a high accuracy of 98.1% on training data and 96.7% on validation data. These results confirm the effectiveness of CNNs in distinguishing between Capuchinbird calls and other environmental sounds. Preprocessing techniques, including feature extraction, data augmentation, and normalization, played a crucial role in enhancing model robustness and mitigating overfitting. The dataset used for training and validation was carefully curated to ensure diversity in audio conditions, thereby improving generalization. This project highlights the efficiency of CNNs in audio-based classification tasks, making them a promising approach for automated acoustic monitoring in diverse environmental settings.

The integration of ModelCheckpoint in TensorFlow facilitated efficient model training by saving the best-performing model for deployment. Compared to traditional manual methods of bioacoustic monitoring, this deep learning-based system provides a scalable, automated, and highly accurate approach to species identification in field recordings. Its ability to process large volumes of audio data efficiently makes it a valuable tool for conservation efforts and ecological research. The application of this system extends beyond simple classification; it allows researchers to automate the monitoring of avian populations with minimal human intervention. This automation not only reduces time and labor costs but also ensures that long-term biodiversity assessments are conducted with high precision. The ability of AI-driven solutions to streamline the collection and analysis of acoustic data represents a major advancement in ecological research and wildlife conservation.

Beyond its immediate application, this project has significant real-world implications. By enabling automated species identification, it assists researchers in tracking biodiversity, monitoring endangered species, and understanding ecological patterns over time. Furthermore, the potential for integrating this system with real-time acoustic sensors could revolutionize wildlife monitoring by providing continuous, non-intrusive observation of avian populations. This capability is crucial in preserving fragile ecosystems, as it enables the early detection of changes in species distribution and behavior. The project also paves the way for further developments in AI-driven bioacoustic monitoring, with future iterations potentially expanding to a wider range of species and habitats. Enhancements such as real-time data processing and cloud-based model deployment could make this system even more effective for large-scale conservation initiatives.

Chapter 7: Future Enhancements

Future Enhancements To further improve the audio classification system for capuchin bird call detection, several key enhancements can be implemented. These improvements will enhance accuracy, scalability, and real-world applicability, making the system more effective in identifying bird calls even in complex acoustic environments.

- **Model Improvement**

  Enhancing the model by exploring more advanced architectures like ResNet or Transformer-based models could significantly improve classification accuracy. These architectures are known for their ability to extract deep features, making them more effective in distinguishing subtle variations in bird calls. Fine-tuning pretrained models on bioacoustic datasets and applying self-supervised learning techniques would enable the system to learn from a broader range of bird sounds without requiring extensive labeled data..

- **Dataset Expansion and Augmentation**

  Expanding the dataset to include a greater variety of bird species, environmental conditions, and different acoustic backgrounds would enhance the model's generalization ability. The inclusion of recordings from multiple geographic locations and different seasons could help the system perform well under diverse real-world scenarios. Augmenting the dataset through techniques such as pitch shifting, time stretching, and noise injection can further improve robustness by simulating real-world variations in bird calls.

- **Real-Time Performance and Optimization**

  Optimizing the system for deployment on low-power edge devices and mobile applications is essential for real-time bird call detection. Converting the model to TensorFlow Lite or ONNX can help reduce computational overhead, enabling smooth inference on mobile devices. Techniques such as model pruning and quantization can be applied to compress the model without significant loss of accuracy, reducing latency and improving energy efficiency.

- **Deployment and User Interference**

  A mobile application could be developed to enable researchers, conservationists, and bird enthusiasts to record and analyze bird sounds in real-time. This app could offer an intuitive user interface that provides instant classification results along with detailed insights, such as spectrogram visualizations and confidence scores.

These enhancements would make the capuchin bird call detection system more accurate, scalable, and user-friendly, ultimately improving its effectiveness in real-world wildlife conservation and bioacoustic research.

Bibliography

[1] Downie, J. S. (2003). Music information retrieval. Annual review of information science and technology, 37(1), 295-340.

[2] Patterson, G., Pfalz, A., & Allison, J. (2017). Neural Audio: Music Information Retrieval Using Deep Neural Networks.

[3] Gómez, J. S., Abeßer, J., & Cano, E. (2018). Jazz Solo Instrument Classification with Convolutional Neural Networks, Source Separation, and Transfer Learning. In ISMIR (pp. 577-584).

[4] Choi, K., Fazekas, G., Cho, K., & Sandler, M. (2017). A tutorial on deep learning for music information retrieval. arXiv preprint arXiv:1709.04396.

[5] Lostanlen, V., Andén, J., & Lagrange, M. (2018, September). Extended playing techniques: the next milestone in musical instrument recognition. In Proceedings of the 5th International Conference on Digital Libraries for Musicology (pp. 1-10).

[6] Elghamrawy, S.M., Hassnien, A.E. and Snasel, V., 2021. Optimized deep learning-inspired model for the diagnosis and prediction of COVID-19. Cmc-Computers Materials & Continua, pp.2353-2371.

[7] Humphrey, E., Durand, S., & McFee, B. (2018, September). OpenMIC- 2018: An Open Data-set for Multiple Instrument Recognition. In ISMIR (pp. 438-444).

[8] Shi, L., Du, K., Zhang, C., Ma, H. and Yan, W., 2019. Lung sound recognition algorithm based on VGGISH-bigru. IEEE Access, 7, pp.139438-139449.

[9] Solanki, A., & Pandey, S. (2019). Music instrument recognition using deep convolutional neural networks. International Journal of Information Technology, 1-10.

[10] 2Siedenburg, K., Schädler, M. R., & Hülsmeier, D. (2019). Modeling the onset advantage in musical instrument recognition. The Journal of the Acoustical Society of America, 146(6), EL523-EL529.

[11] Taenzer, M., Abeßer, J., Mimilakis, S. I., Weiß, C., Müller, M., Lukashevich, H., & Fraunhofer, I. D. M. T. (2019). Investigating cnn- based instrument family recognition for western classical music recordings. ISMIR.

[12] Gururani,S.,Summers,C.,&Lerch,A.(2018,September).Instrument Activity Detection in Polyphonic Music using Deep Neural Networks. In ISMIR (pp. 569-576).

[13] Toghiani-Rizi, B., & Windmark, M. (2017). Musical instrument recognition using their distinctivecharacteristics in artificial neural networks. arXiv preprint arXiv:1705.04971.

[14] Wang, L.H., Zhao, X.P., Wu, J.X., Xie, Y.Y. and Zhang, Y.H., 2017. Motor fault diagnosis based on short-time Fourier transform and convolutional neural network. Chinese Journal of Mechanical Engineering, 30(6), pp.1357-1368.

[15] Logan, B., 2000. Mel frequency cepstral coefficients for music modeling. In International Symposium on Music Information Retrieval.

[16] Yan, P.Z., Wang, F., Kwok, N., Allen, B.B., Keros, S. and Grinspan, Z., 2019. Automated spectrographic seizure detection using convolutional neural networks. Seizure, 71, pp.124-131.

[17] Elghamrawy, S.M., Hassanien, A.E. and Vasilakos, A.V., 2021. Genetic-based adaptive momentum estimation for predicting mortality risk factors for COVID-19 patients using deep learning. International Journal of Imaging Systems and Technology.

[18] Annabel, L. S. P., & Thulasi, V. (2023, November). Environmental Sound Classification Using 1-D and 2-D Convolutional Neural Networks. In 2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 1242- 1247). IEEE.

[19] Elias, N. (2022, December). Audio Classification of Low Feature Spectrograms Utilizing Convolutional Neural Networks. In 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 693-698). IEEE.

[20] Jin, T., & Wu, Z. (2023, November). Distress Keywords Classification Based on Audio MFCC Features using Convolutional Neural Networks. In 2023 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML) (pp. 365-368). IEEE.

[21] Huq, S., Xi, P., Goubran, R., Knoefel, F., & Green, J. R. (2023, December). Data Augmentation and Deep Learning in Audio Classification Problems: Alignment Between Training and Test Environments. In 2023 IEEE 23rd International Conference on Bioinformatics and Bioengineering (BIBE) (pp. 140-146). IEEE Computer Society.

[22] Pallotta, L., Neri, M., Buongiorno, M., Neri, A., & Giunta, G. (2022, September). A Machine Learning-Based Approach for Audio Signals Classification using Chebychev Moments and Mel-Coefficients. In 2022 7th International Conference on Frontiers of Signal Processing (ICFSP) (pp. 120-124). IEEE.

[23] Harryanto, A. A. A., Gunawan, K., Nagano, R., & Sutoyo, R. (2022, September). Music classification model development based on audio recognition using transformer model. In 2022 3rd International Conference on Artificial Intelligence and Data Sciences (AiDAS) (pp. 258-263). IEEE.

[24] Lo, P. C., Liu, C. Y., & Chou, T. H. (2022, November). DNN Audio Classification Based on Extracted Spectral Attributes. In 2022 14th International Conference on Signal Processing Systems (ICSPS) (pp. 259-262). IEEE.

[25] Shahriar, S., Dara, R., & Hayawi, K. (2022, December). On the Impact of Deep Learning and Feature Extraction for Arabic Audio Classification and Speaker Identification. In 2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA) (pp. 1-8). IEEE.

[26] Ansari, M. R., Tumpa, S. A., Raya, J. A. F., & Murshed, M. N. (2021, September). Comparison between support vector machine and random forest for audio classification. In 2021 International Conference on Electronics, Communications and Information Technology (ICECIT) (pp. 1-4). IEEE.

[27] Sahai, S. Y., Liu, J., Muniyappa, T., Sathyendra, K. M., Alexandridis, A., Strimel, G. P., ... & Kunzmann, S. (2023, June). Dual-attention neural transducers for efficient wake word spotting in speech recognition. In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5). IEEE.

[28] Bansal, V., Pahwa, G., & Kannan, N. (2020, October). Cough Classification for COVID-19 based on audio mfcc features using Convolutional Neural Networks. In 2020 IEEE international conference on computing, power and communication technologies (GUCON) (pp. 604-608). IEEE.

[29] Y. Li, Q. He, S. Kwong, T. Li, and J. Yang, "Characteristics-based effective applause detection for meeting speech," *Signal Process.*, vol. 89, no. 8, pp. 1625-1633, 2009.

[30] Y. Li, Q. Wang, X. Zhang, W. Li, X. Li, J. Yang, X. Feng, Q. Huang, and Qianhua He, "Unsupervised classification of speaker roles in multi- participant conversational speech," *Computer Speech and Language*, vol. 42, pp. 81-99, 2017.

[31] Y. Li, Q. Wang, X. Li, X. Zhang, Y. Zhang, A. Chen, Q. He, and Q. Huang, "Unsupervised detection of acoustic events using information bottleneck principle," *Digital Signal Process.*, vol.63, pp. 123-134, 2017.

[32] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Audio surveillance of roads: a system for detecting anomalous sounds," *IEEE T. ITS*, vol. 17, no. 1, pp. 279-288, Jan. 2016.

[33] M. Crocco, M. Cristani, A. Trucco, and V. Murino, "Audio surveillance: a systematic review," *ACM Computing Surveys*, vol. 48, no. 4, pp. 1-46, 2016

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet clas- sification with deep convolutional neural networks," in *Ad- vances in neural information processing systems*, 2012, pp. 1097–1105.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *arXiv preprint arXiv:1512.00567*, 2015.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learn- ing for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[38] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Au- dio Set: An ontology and human-labeled dartaset for audio events," in *IEEE ICASSP 2017*, New Orleans, 2017.

[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei- Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[40] R. F. Lyon, "Machine hearing: An emerging field [exploratory dsp]," *Ieee signal processing magazine*, vol. 27, no. 5, pp. 131– 139, 2010.

[41] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Signal Processing Conference, 2010 18th European*. IEEE, 2010, pp. 1267–1271.