EEA

BIRD SOUND CLASSIFICATION

SOUND
EXPLORERS

EEA

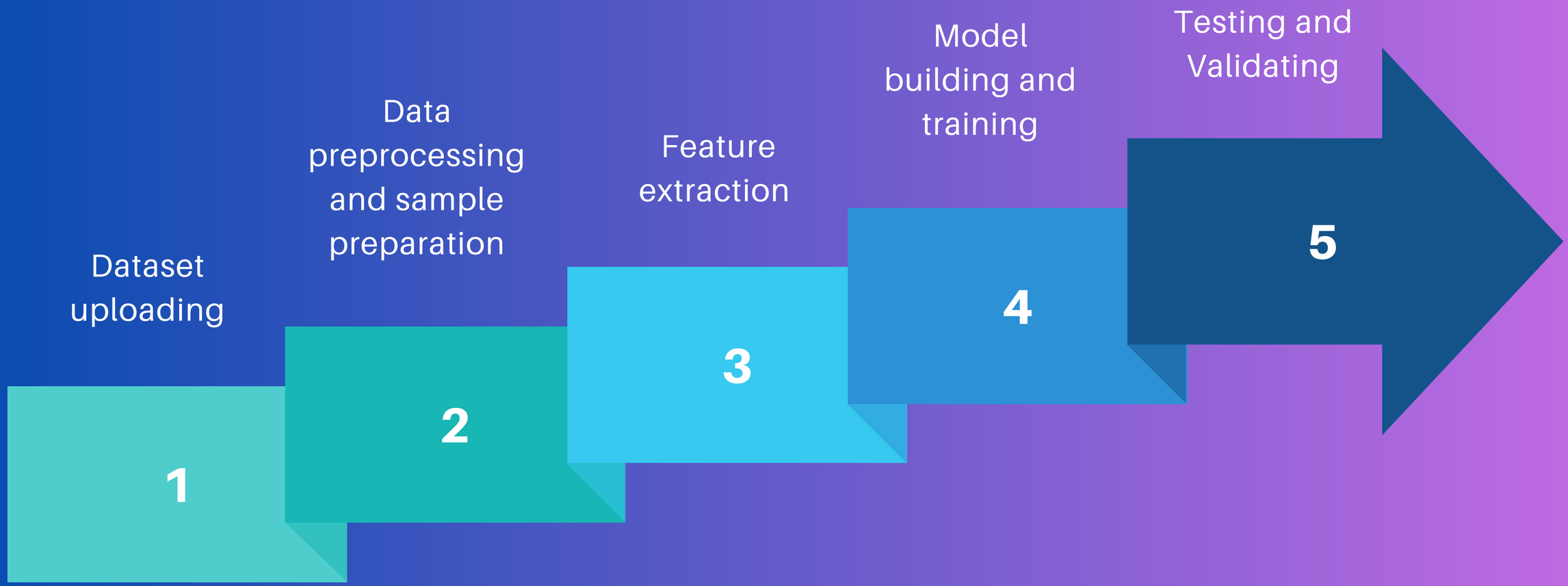
INTRODUCTION

- Objective: The project aims to accurately identify bird species through their unique sounds, leveraging Convolutional Neural Networks (CNNs) implemented with TensorFlow. This approach addresses challenges in environmental conservation and wildlife research by automating the classification of bird calls, thereby aiding ecological monitoring and biodiversity studies..
- Dataset: The dataset comprises 2,161 audio files in MP3 format, representing vocalizations from 114 distinct bird species. Each audio file is annotated with the corresponding species label, facilitating supervised learning. The dataset is sourced from Kaggle.

E E A

STEPS

Overview



DataSet Uploading

Title : Sound of 114 Species of Birds Till 2022

Source: Kaggle

Dataset URL: https://www.kaggle.com/datasets/soumendraprasad/sound-of-114-species-of-birds-till-2022

Sample Audio Checking :

Mono Channel Conversion: Audio is automatically converted to a mono channel, resulting in a 1-dimensional signal suitable for processing.

Resampling: The audio is resampled to a standardized rate of **22050 Hz** for consistency across all audio inputs.

PREPROCESSING AND SAMPLE PREPARATION

Feature Extraction

- we extracted Mel-Frequency Cepstral Coefficients (MFCCs), which capture the essential frequency characteristics of bird sounds.
- To obtain a consistent feature representation, we extracted 40 MFCC coefficients per audio file and took their average over time.
- These extracted features are then reshaped into a structured format suitable for convolutional neural networks (CNNs).
- To ensure an efficient and scalable data pipeline, we used TensorFlow’s Dataset API to automate feature extraction, transformation, and batching before feeding the data into the model.

This approach significantly improves computational efficiency and enhances the model’s ability to distinguish between various bird species based on their vocal patterns.

Converting Audio to Tensors

PREPROCESSING AND SAMPLE PREPARATION

Converting Audio to Tensors

Why Did We Do This?

- *Transformed raw MP3 audio into structured format for deep learning.*
- *Extracted MFCC, spectral centroid, and chroma features.*
- *Converted extracted features into TensorFlow tensors for dataset creation*

```python
# Convert the mp3 Audio into Tensors

def audio_to_tensors(audio_file):
    # Extract the Signal and Sample_Rate from Audio
    audio, sample_rate =librosa.load(audio_file)

    # Extract the MFCC Features and Aggrigate
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
    mfccs_features = np.mean(mfccs_features, axis=1)

    # Convert into Tensors
    mfccs_tensors = tf.convert_to_tensor(mfccs_features, dtype=tf.float32)

    return mfccs_tensors
```

What We Achieved?

- *A Tensor representation of each audio sample, ready for dataset creation.*

TensorFlow Dataset Creation

How Did We Organize the Data?

- *Extracted features from all audio files and stored tensors with labels.*
- *Encoded labels into numerical values for TensorFlow processing.*
- *Created a TensorFlow dataset and applied batching, shuffling, and prefetching for efficient training.*

PREPROCESSING AND SAMPLE PREPARATION

```
[ ]  # Converting Features and Targets to TensorFlow Tensors
     features_tensor = tf.convert_to_tensor(features)
     target_tensor = tf.convert_to_tensor(target)
```

```
# Create TensorFlow datasets
train_dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train))
test_dataset = tf.data.Dataset.from_tensor_slices((X_test, y_test))
```

What We Achieved?

A TensorFlow-ready dataset, optimized for training deep learning models.

PREPROCESSING AND SAMPLE PREPARATION

Data Preparation

Convert Audio to Tensors: Prepare audio data in a tensor format suitable for machine learning models.
Feature Extraction from Audio Files: Processes all audio files to extract necessary features.

DataFrame Creation

Creating DataFrame: Build a DataFrame from the extracted features, including class labels to associate audio features with specific bird species.

Label Encoding

Class labels are encoded using LabelEncoder to convert categorical data into numerical format for model compatibility. Adding Encoded Labels: The encoded class labels are added to the DataFrame, enhancing its usability for model training.

Dictionary for Predictions

Creating Prediction Dictionary: Establish a mapping dictionary from class labels to corresponding audio features for easy lookup during predictions.

PREPROCESSING AND SAMPLE PREPARATION

- ***TensorFlow Preparation***

Convert to TensorFlow Tensors: Transform features and targets into TensorFlow tensors for model training.
Creating TensorFlow Dataset: Construct a TensorFlow dataset from the tensors for enhanced performance in training.

- ***Batching and Splitting Data***

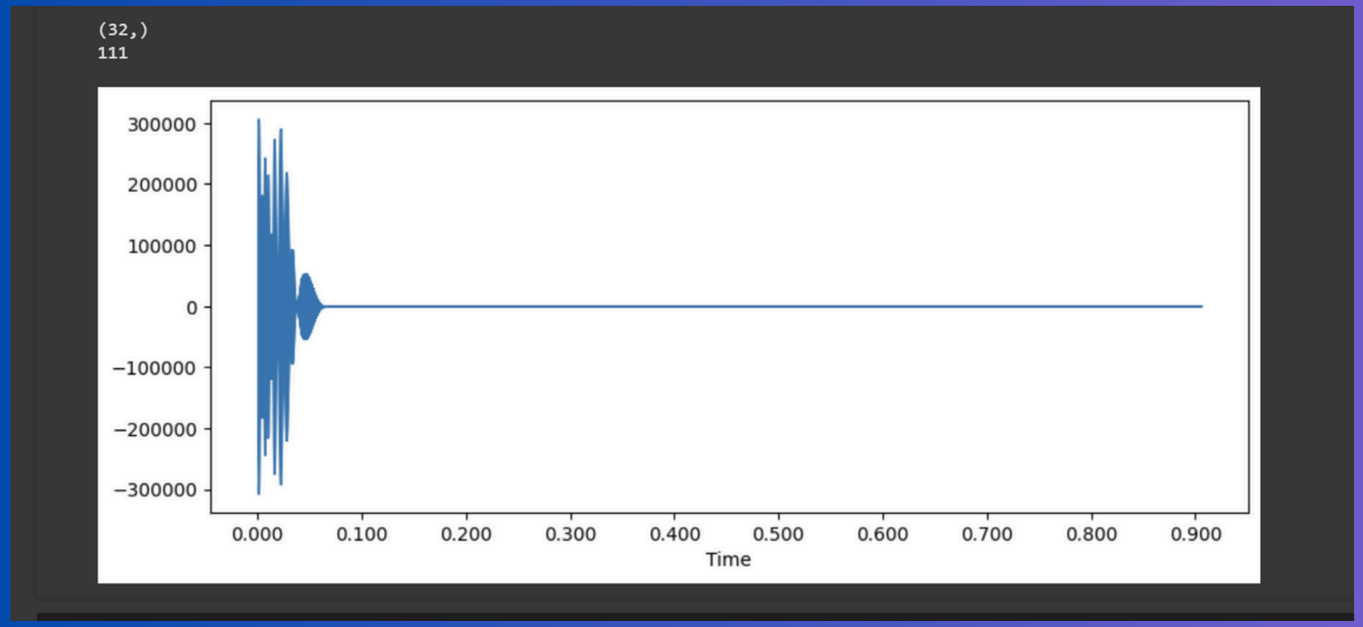
Batch Splitting: Divided the dataset into manageable batches for efficient training.

Dataset Splitting Function:

Define a function to split the dataset into training (80%), validation (10%), and testing (10%) sets.

- ***Data Pipeline Optimization***

Optimizing Data Pipeline: Build an optimized data pipeline to enhance model performance and training efficiency.



Waveform Visualization of the first audio batch in the dataset.

The waveform above shows you a visual representation of the reconstructed audio signal, allowing you to see how the sound's loudness changes over time.

The line *audio = librosa.feature.inverse.mfcc_to_audio(audio_batch.numpy())* in the code performs this reconstruction, essentially converting the MFCCs back into a waveform that can be visualized and heard.

MFCC’s don't directly represent the raw audio waveform. This visualization helps understand the structure and content of the training data.

CNN: MODEL FOR CLASSIFACTION

E E A

We employed a 1D Convolutional Neural Network (CNN) designed for audio classification, specifically tailored for bird sound identification.

- Input layer: Shape defined by input_shape
- 3 Convolutional layers:
  -Filters: 128, 256, 256
  -Each followed by batch normalization and max pooling
- 2 Dense layers:
  -512 units each
  -ReLU activation, L2 regularization(applied to mitigate overfitting), 30% dropout rate between dense layers for further regularization
- Output layer:
  -Softmax activation for multi-class prediction(Number of units corresponds to the number of bird species to be classified)

0s

```
# Display the Model Summary
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 38, 128)	512
batch_normalization (BatchNormalization)	(None, 38, 128)	512
max_pooling1d (MaxPooling1D)	(None, 19, 128)	0
conv1d_1 (Conv1D)	(None, 17, 256)	98,560
batch_normalization_1 (BatchNormalization)	(None, 17, 256)	1,024
max_pooling1d_1 (MaxPooling1D)	(None, 9, 256)	0
conv1d_2 (Conv1D)	(None, 7, 256)	196,864
batch_normalization_2 (BatchNormalization)	(None, 7, 256)	1,024
max_pooling1d_2 (MaxPooling1D)	(None, 4, 256)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 512)	524,800
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262,656
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 114)	58,482

Sequential architecture

Layers :

Convolutional Layers (Conv1D) :

The model starts with three Conv1D layers that apply 1D convolutions to extract features from the input data. The first layer has 128 filters, with subsequent layers increasing in depth.

Batch Normalization :

This layer normalizes outputs from the Conv1D layers to stabilize and accelerate training, reducing internal covariate shifts.

Max Pooling Layers:

Three max pooling layers downsample the data, lowering dimensionality while preserving important features and reducing computational load.

Flatten Layer:

Converts the multi-dimensional output from the last pooling layer into a single vector for the fully connected dense layers.

Dense Layers:

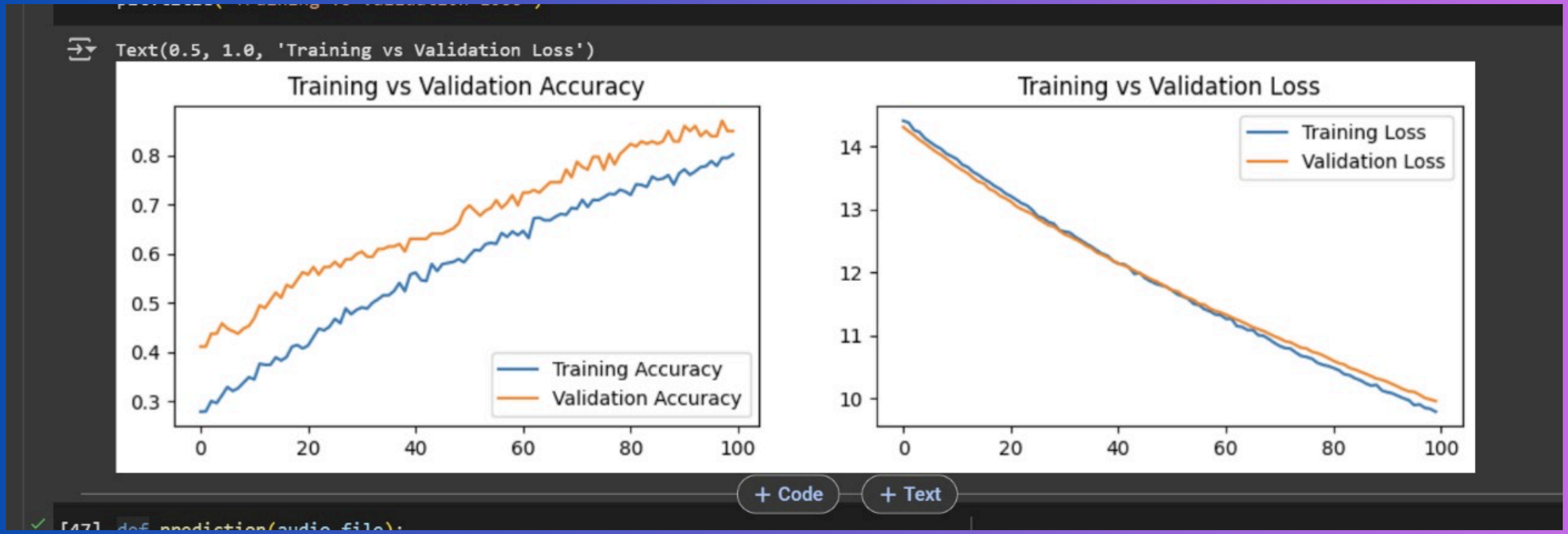
Several fully connected layers are included. The first two have 512 units each, while the final layer outputs 114 units for classification

Dropout Layers:

Dropout Layers are used between dense layers to prevent overfitting by randomly setting a fraction of input units to zero during input training.

1. The right graph illustrates training and validation loss trends over the same number of epochs. The x-axis denotes the number of epochs, while the y-axis represents the loss value.

Training Loss



2. A steady decrease in both loss curves is a positive sign of learning and improvement.
If validation loss starts to rise while training loss continues to drop, it might indicate that the model is beginning to memorize the training data rather than learning general patterns.

EEA

Hyper-parameter tuning

1

Optimizer

Adam was chosen because it adapts the learning rate dynamically for better convergence.

2

Loss Function

Sparse Categorical Crossentropy was used because the output labels are categorical.

3

Batch Size & Epochs

These were adjusted iteratively to balance training speed and model performance.We used Batch size=32 and 100 epochs

4

Regularization (Dropout)

Introduced in dense layers to reduce overfitting.

5

Data Augmentation

Methods like time shifting and pitch shifting were likely used to enhance generalization.

MODEL TRAINING

EEA

Training Approach:

- TensorFlow Dataset API was used to create an efficient data pipeline.Instead of manually loading and preprocessing audio files one by one, the tf.data pipeline automates data handling.
- The dataset is divided into training, validation, and testing subsets to ensure robust model evaluation and to prevent overfitting.Techniques such as caching, shuffling, and prefetching are employed to enhance training efficiency and reduce computational overhead.
- The training loss and validation accuracy were monitored after each epoch to ensure the model was learning effectively.

Final Accuracy:

- The model achieved 93.4% accuracy, indicating a well-optimized training process.

OBSERAVATIONS AND RESULTS

```
[76] audio_file_path = '/content/Voice of Birds/Voice of Birds/Rufous-bellied Chachalaca_sound/Rufous-bellied Chachalaca19.mp3'
     prediction(audio_file_path)

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be emp
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 202ms/step
Predicted Class : Rufous-bellied Chachalaca_sound
Confidence : 87.45%
```

```
[77] audio_file_path = '/content/Voice of Birds/Voice of Birds/Puna Tinamou_sound/Puna Tinamou8.mp3'
     prediction(audio_file_path)

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be emp
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 205ms/step
Predicted Class : Puna Tinamou_sound
Confidence : 92.21%
```

```
[83] audio_file_path = '/content/Voice of Birds/Voice of Birds/Grey Tinamou_sound/Grey Tinamou26.mp3'
     prediction(audio_file_path)

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be emp
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 304ms/step
Predicted Class : Grey Tinamou_sound
Confidence : 54.35%
```

```
[72] audio_file_path = '/content/Voice of Birds/Voice of Birds/Puna Tinamou_sound/Puna Tinamou7.mp3'
     prediction(audio_file_path)

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be emp
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 210ms/step
Predicted Class : Puna Tinamou_sound
```

This shows audio files from a dataset and utilizes the pre-trained model to predict the bird species present in each file.

Obtained Predictions :

Rufous-bellied Chachalaca (87.45% confidence)
Puna Tinamou (92.21% & 91.46% confidence)
Grey Tinamou (54.35% confidence)

~ Group SOUND EXPLORERS
Members-
Kavita Kumari
Kavya Palla
Shachhi

Thank You !