Bird Sound Classification Using Deep Learning

Abstract

This report delves deep into the application of neural networks for classifying bird species based on the sounds it makes. The data used are the spectrograms generated from 3-seconds audio clips across twelve bird species which are taken from the Xen-Canto a crowd-sourced birds sounds archive. Using this data, I have developed binary and multi-class convolutional neural network models and compared variation in architecture and hyperparameters to get the most effective architecture. Model performance is evaluated using accuracy and confusion matrix. The report also demonstrates the potential and limitations of neural networks for this specific bird species identification application.

Introduction

In this project, Neural networks are used to classify bird species based on their sounds. The primary objective is to explore how deep learning models, mainly Convolutional Neural Networks(CNN) , can be applied to spectrograms (Time-Frequency representation) for effective classification of birds.

The data is collected from the Xeno-Canto bird recordings which contains recordings of 264 species. Out of all, 12 bird species are chosen for this project and high quality sounds are selected for each species. Spectrograms were then generated from the first 3 seconds of each sound clip which are fed as an input to the neural network.

The project explores two major classification tasks:

- Binary Classification: Distinguish between American robin (amerob) and red-winged blackbird (rewbla).
- Multiclass Classification: Distinguish between all 12 bird species [American crow (amerco), American Robin (amerob), Bewicks Wren (bewwre), Black-capped chickadee bkcchi), dark-eyed junco (daejun), house finch (houfin), house sparrow (houspa), northern flicker (norfli), red-winged blackbird (rewbla), song sparrow (sonspa), spotted towhee (spotow), white-crowned sparrow (whcspa)].

For the above classification tasks, I implemented and trained CNN with different hyperparameters tuning.

Theoretical Background

Deep learning is a rapidly evolving field in machine learning and artificial intelligence, with **neural networks** being the foundational block.

	Traditional Machine Learning	Deep Learning
Feature Extraction	Required	Not Required
Dataset	Deals with small datasets	Larger datasets are required
Computation complexity	Low	Require high computing

Most Common Deep Learning Techniques are Deep Neural Network (DNN) which are not very complex problems and works well with structured data, Convolutional Neural Network (CNN) used for image classification, Recurrent Neural Network (RNN) for time series, and also works well with language or audio data.

Neural networks are inspired by the human brain which typically consists of interconnected nodes (neurons) arranged in layers. The network receives an input vector of p variables X = (X1, X2, … Xp) and builds a nonlinear function f(X) to predict the response Y.

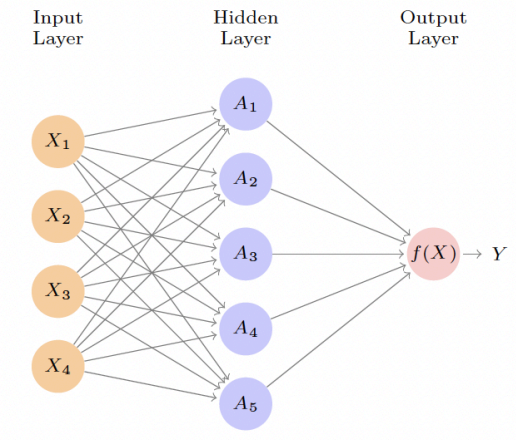


Figure 1: Neural network with single hidden layer

The hidden layer computes activations Ak = hk(X) that are nonlinear transformations of linear combinations of the inputs X1, X2,...,Xp. Hence these Ak are not directly

observed. The output layer is a linear model that uses these activations Ak as inputs, resulting in a function f(X).
There are many different types of activation functions. Some examples are:

- Linear
- ReLU: The preferred choice in modern neural networks. A ReLU activation can be computed and stored more efficiently than a sigmoid activation.

$$g(z) = max(0, z)$$; where x is the input

- Sigmoid : This is the function used in logistic regression to convert function into probabilities between zero and one.

$$g(z) = \frac{e^z}{1 + e^z}$$

There are many hyperparameters in the neural networks which determine the network structure and the variables which determine how the network is trained. These hyperparameters are set before training(before optimizing the weights and bias).

Pre-determined hyperparameters are:

Number of neurons in the input layer

- This will depend on the data that we are using for the training. For instance if we have n features in the dataset we will have n input neurons. If we have the image as an input with n*in pixels, we will have input neurons.

Number of neurons in the output layer

- This will also depend on the data that we are using for the training. For instance if we are doing binary classification or regression, single output will be sufficient. If we are having multi-class or multi-label class classification problems, we need a number of output neurons as classes/labels.

Hyperparameters that need tuning

Number of hidden layers

- The more hidden layers we have, the deeper the network will be. In general having more layers will help network more than having more number of neurons in the layer.

Number of neurons in the hidden layers

- Each hidden layer can have a different number of neurons. This will be changed based on how the network is performing.

Activation Function

- Each layer can have any type of activation function except the input layer and moreover the activation function of hidden layers cannot be linear.

Optimization Algorithms

- This algorithm determines how the network updates its weights. Most common algorithms are Stochastic Gradient Descent(SGD), Adam, RMSProp, AdaGrad.

Loss Function

- The loss (cost) function is what we're trying to minimize during training. This changes depending on the problem we have.

Batch size

- Batch size is the amount of data points we put in each batch while training the network.

Number of epochs

- This is basically how many times we run the whole dataset through the network. This helps the network learn the dataset a little bit better after each epoch.

Learning Rate

- Learning rate determines how fast/slow steps need to take towards the direction set by gradient descent. Small learning rate slows down the learning and too big might overshoot the optimum value.

Regularization

- This helps reduce overfitting. There are many ways to do regularization (L1, L2, Dropout,etc.) these often come with their own hyperparameters.

There are various types of neural networks each with their own architecture and characteristics. In this project I have used convolutional neural networks.

Convolutional Neural Network(CNN)

These are a special family of neural networks evolved for classifying images and have shown spectacular success on a wide range of problems. This combines two specialized types of hidden layers, called convolution layers and pooling layers. Convolution layers search for instances of patterns in the image, whereas pooling layers downsample these to select a subset.

Methodology

Data Preparation

The data is collected from Xeno-Canto, a crowd-sourced bird sounds archive. There are many sound clips available and out of all sound clips with high quality of 12 species are collected. The sound clip is then subsampled to 22050 Hz and the first 3 seconds of the sound clip are selected. The spectrogram is derived from an audio signal for each 2-second window, resulting in a 128(frequency) x 517 (time) image of the bird call. All bird calls for all clips in a given species are saved individually which results in an uneven number of samples in each species ranging from 37-630 samples. These spectrograms are used as an input to the neural network. The dataset is split into 80% training set and 20% testing set. For the binary classification task I used spectrograms of American Robin (amerob) and Red-winged Blackbird(rewbla). Whereas for multi-class classification I used spectrograms for all the 12 bird species and labelled each class from 0 to 11.

Binary Classification

In this classification task, two birds amerob, rewbla are being classified. Three different models are used for this classification task. The first model is the basic one with one hidden layer and ReLU activation function and one single dense layer with sigmoid activation function since this is a binary classification. The optimization algorithm used is Adam, loss function is the binary_crossentropy and metrics is the accuracy. The subsequent models are created by adding dropout rate, increasing the number of epochs and also increasing the number of hidden layers for better generalization. Early stopping is included to avoid overtraining by stopping when validation loss stops improving.

Model1

Neural Network Architecture

- **Two convolutional blocks**:
  - Each includes Conv2D, Maxpooling2D.
  - The number of filters increases progressively: 32 → 64
- **Flatten** which converts features to 1D vector.
- A dense layer with 128 neurons precedes the final classification layer.
- The final Dense(1, activation = 'sigmoid').

Training Configuration:

- **Optimizer**: Adam with a learning rate of 0.0001 for stable convergence
- **Loss Function**: binary_crossentropy
- **EarlyStopping** with patience = 10
- **epochs = 10, batch_size = 16**

Model2

- **Two convolutional blocks**:
  - Each includes Conv2D → MaxPooling2D → Dropout(rate = 0.2)
  - The number of filters increases progressively: 32 → 64
- **Flatten** which converts features to 1D vector.
- A dense layer with 128 neurons precedes the final classification layer.
- The final Dense(1, activation = 'sigmoid').
- **epochs = 20, batch_size = 16**

Model3

- **Four convolutional blocks**:
  - Each includes Conv2D → BatchNormalization() → MaxPooling2D → Dropout(rate = 0.2)
  - The number of filters increases progressively: 16 → 32 → 64 → 128
- **GlobalAveragePooling2D** ()
- A dense layer with 128 neurons precedes the final classification layer.
- The final Dense(1, activation = 'sigmoid').
- **epochs = 50, batch_size = 16**

Training Configuration:

- **Optimizer**: Adam with a learning rate of 0.0001 for stable convergence
- **Loss Function**: binary_crossentropy
- **EarlyStopping** with patience = 10
- **Validation Split**: 20% of the training data is used for internal validation.

Multi class Classification

In this classification task, 12 bird species are being classified. To prepare the spectrogram data for training a convolutional neural network in a 12-class bird species classification task, each species shape was stored as a spectrogram array of shape (128, 517, N) where N is the number of recordings for that species. Each spectrogram was transposed and reshaped to a consistent format (N, 128, 517, 1) 1 is for the single grayscale channel. Class labels were numerically encoded and one-hot encoded for multi-class classification. This pipeline ensured that all data was consistently formatted, labeled, and ready for training using categorical cross-entropy loss. Three different models were created.

Model1

The model performed well in the binary classification is used as the base model. The architecture configuration is below

Neural Network Architecture

- **Four convolutional blocks**:
    - Each includes Conv2D → BatchNormalization → Maxpolling2D -> Dropout rate(0.25)
    - The number of filters increases progressively: 16 → 32 → 64 → 128
- **GlobalAveragePooling2D** is used instead of flattening, reducing overfitting by averaging feature maps spatially
- A dense layer with 128 neurons precedes the final classification layer.
- The final Dense(12, activation = ‘softmax’) outputs probabilities over 12 classes.

Training Configuration:

- **Optimizer**: Adam with a learning rate of 0.0001 for stable convergence
- **Loss Function**: categorical_crossentropy, appropriate for multi-class classification with one-hot labels

- **EarlyStopping** with patience = 10 halts training if no improvement in validation loss is observed.
- **Epochs = 30, Batch_size = 16**

Model2

Six convolutional blocks:

Each block includes Conv2D → BatchNormalization → MaxPooling2D → Dropout(0.2)

The number of filters increases progressively 16 → 32 → 64 → 128 → 256 → 512

- GlobalAveragePooling2D is used instead of flatten to reduce the number of parameters.
- A dense layer of 128 neurons with ReLU activation function
- The final Dense(12, activation = ‘softmax’)
- Learning rate = 0.0001, epochs = 30, batch_size = 16

Model3

Six convolutional blocks:

Each block includes Conv2D → BatchNormalization → MaxPooling2D → Dropout(0.2)

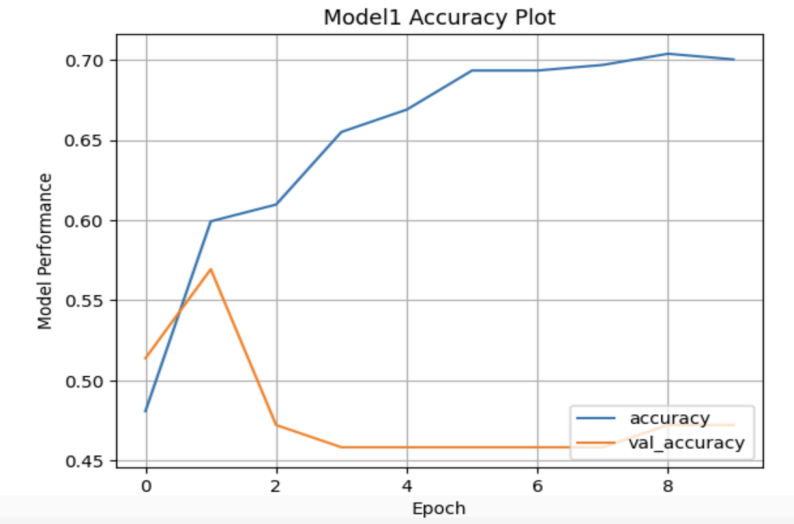
The number of filters increases progressively 16 → 32 → 64 → 128 → 256 → 512

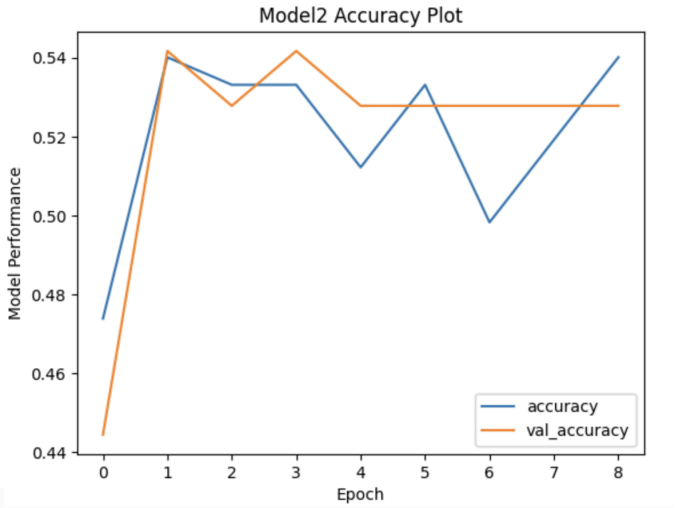
- GlobalAveragePooling2D is used instead of flatten to reduce the number of parameters.
- A dense layer of 128 neurons with ReLU activation function
- The final Dense(12, activation = ‘softmax’)
- Learning rate = 0.0005, epochs = 50, batch_size = 32

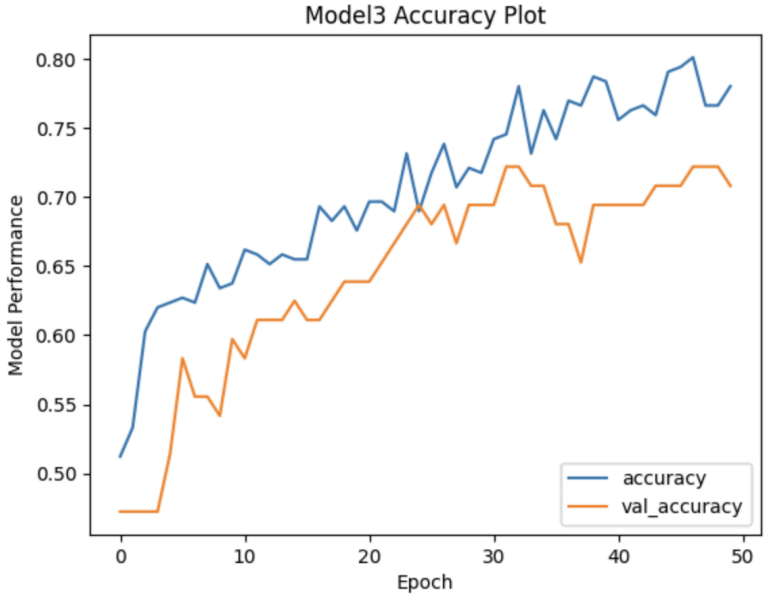
Computational Results

Binary Classification

Metric	Model1	Model2	Model3
Test Accuracy	47.22%	54.16%	72.08%
Precision	American Robin - 0.33 Red-winged Blackbird - 0.50	American Robin - 1.00 Red-winged Blackbird - 0.54	American Robin - 0.77 Red-winged Blackbird - 0.66
Recall	American Robin - 0.12 Red-winged Blackbird - 0.79	American Robin - 0.03 Red-winged Blackbird - 1.00	American Robin - 0.50 Red-winged Blackbird - 0.87
Time	38.5s	1:26 min	4.59 min





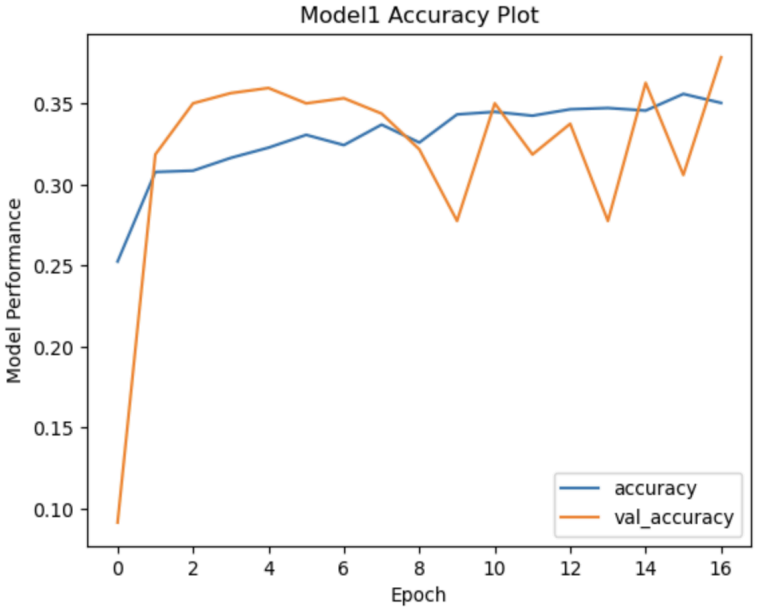


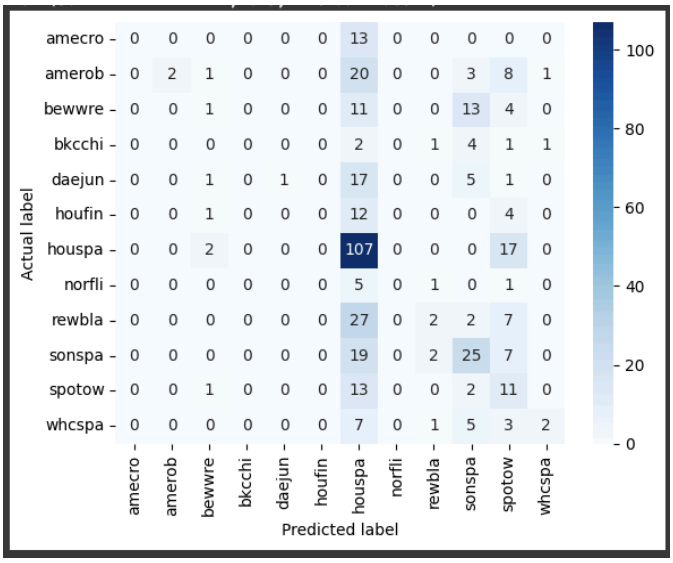
Model 1 is the basic one but shows overfitting with poor generalization training accuracy increases while validation accuracy drops early. While recall for Red-winged Blackbird is relatively high (0.79), it under-performed for American Robin with very low recall(0.12), which shows the model is biased towards the majority class.
Model2 is the improvisation of model1 which slightly performed better than model 1. Precision for American robin is 1.00, but recall is very low 0.03 which shows there are many false negatives while predicting American Robin.
Model 3 outperformed in terms of accuracy and recall. American Robin recall is improved to 0.5, precision is 0.77 and accuracy is 72.08%. This model effectively handles class imbalance.

Multi-class Classification

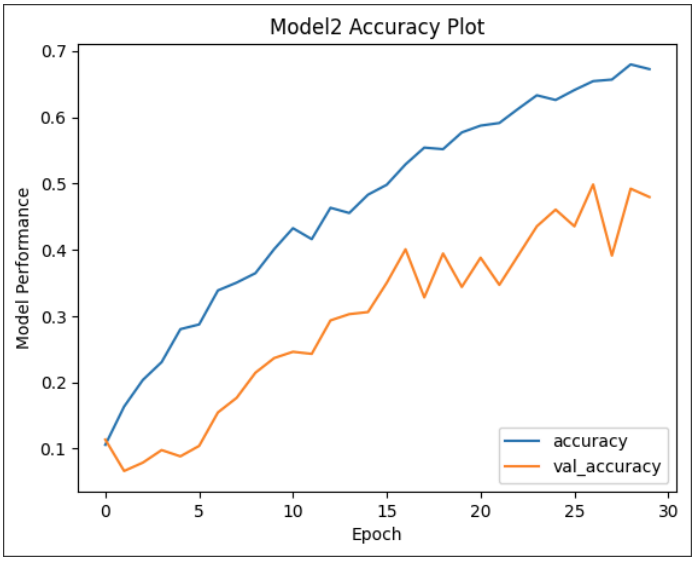
Metric	Model1	Model2	Model3
Test Accuracy	38%	49.8%	61%
val_auc	0.7865	0.8597	0.8864
Test_loss	1.9586	1.6315	1.05913
Time	1.90 min	2.49min	2.66min

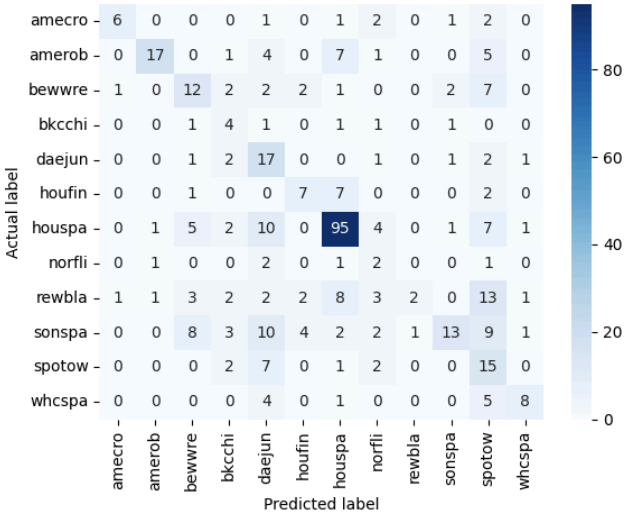
Model 1



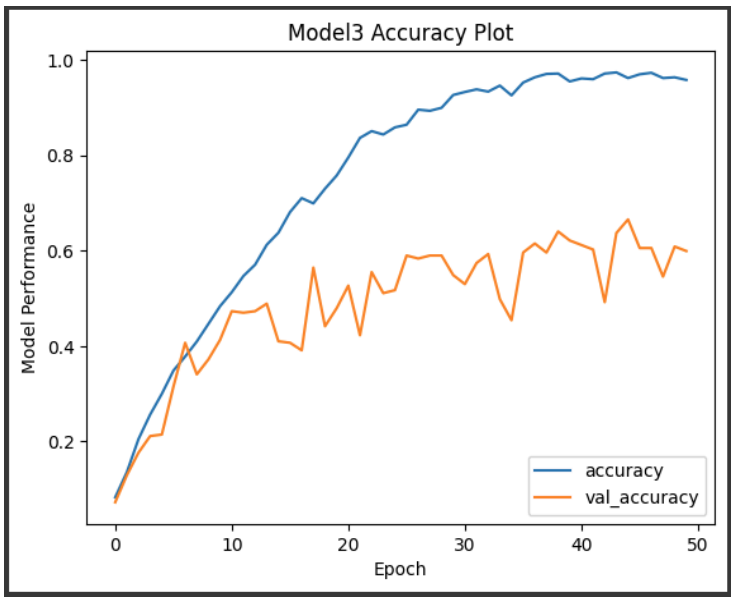


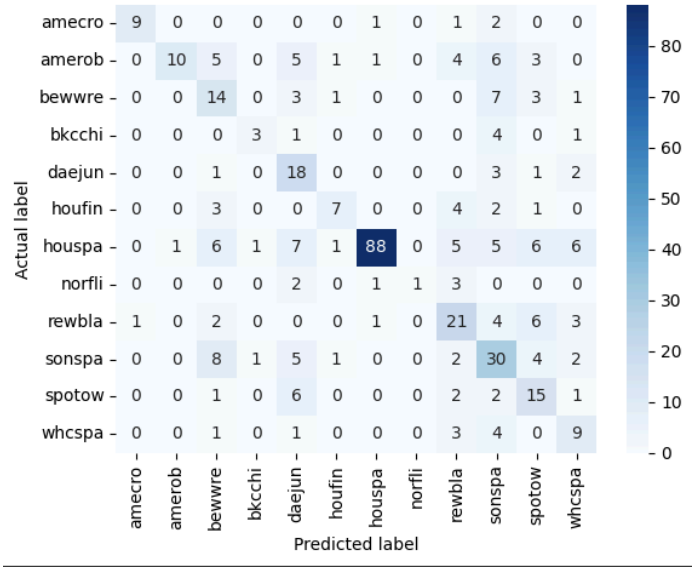
Model 2





Model 3





Model 1 is the basic model which is the final model of binary classification. This under-performed with very low accuracy but validation AUC is 0.7865

Test Data Predictions

Discussion

Challenges & Limitations

References

1. *James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). An Introduction to Statistical Learning with Applications in Python. (Original work published 2023)* https://hastie.su.domains/ISLP/ISLP_website.pdf.download.html
2. *Rao, R. (n.d.). Xeno-canto Bird Recordings Extended (A-M)https://www.kaggle.com/datasets/rohanrao/xeno-canto-bird-recordings-extended-a-m*