

## SYDE 577 Project: Bird Vocalization Classifier (BirdClef 2023)

Anna Yang (20935486), Jessica Zhu (20962439), Sebastian Mendoza (20970266), Serena Li (20930882)

### Abstract

With the rise of autonomous recording units (ARUs) as a tool for monitoring the populations of bird species around the world, an increasingly salient challenge is how to extract meaning from tens of thousands of hours of recorded data. This project aims to develop a neural network for bird vocalization identification, inspired by BirdClef 2023 – a competition challenging developers to construct models to identify the presence of 264 East African bird species in soundscape recordings. Techniques such as data pre-processing, chunking, and augmentation are used alongside architectural experimentation with CNNs, RNNs and pre-trained model backbones. The outcome of this development process is two models able to achieve significantly better than baseline performance when trained and tested on the identification of 10 species classes.

## 1. Introduction

### 1.1 Competition Background and Motivation

Passive acoustic monitoring (PAM) is a commonly used, low-cost technique to monitor and observe wildlife populations, which can help inform conservation efforts and provide information about ecological health. However, gleaning valuable insights from this recorded data is less straightforward – research into how to best extract accurate information from this data is active and ongoing. The main challenge lies in a lack of available data for rare species, which necessitates the development of robust, innovative models and algorithms to identify and track these species of interest [1]. Other challenges associated with bioacoustics include the presence of background noise, variability in individual vocalizations, and the extraction of multiple species in overlapping soundscapes.

BirdClef is an annual competition that challenges developers to develop accurate, compact, well-generalized audio classifiers trained on the task of avian species identification. The 2023 iteration, which focused on Eastern African bird species, challenged developers to build deep learning models that recognized and differentiated 264 species by call [2]. The aim of this competition is to aid researchers in tracking species of interest, so they can better understand the impacts of conservation efforts and help protect avian biodiversity in Eastern Africa.

Training call audio was provided for each species on Kaggle (varying numbers of recordings for each, up to 500), while models were evaluated with cmAP (derivative of the macro-averaged average precision score) on test soundscapes which could contain multiple simultaneous calls of different species [2].

### 1.2 Competition Scope and Constraints

For the purposes of this project, some aspects of the official BirdClef 2023 competition were minimized or reduced in scope. First, while the official competition provides data for 264 species, the classifier described in this project is limited to 10 species. This decision to limit the number of involved species was made to reduce training and inference time, while also accounting for storage limits on collaborative platforms like GitHub and Google Colab for training audio samples. In the competition, a macro-averaged average precision score (cmAP) was used in evaluation to account for rare birds with few samples [2]. Given the now reduced number of species, the competition evaluation metric cmAP was no longer appropriate to gauge model results and accuracy was used instead.

Notably, the competition also has a stated limit of 2 hours for CPU inference time – as a side effect of scoping the field down to 10 species, this limit was easily cleared. As a result, runtime will be discussed

in the context of the scoped problem (i.e. on a relative scale), and not the original competition's requirements.

### 1.3 Project Goal

Following updates to the original project description, the project goal can be reframed as the creation of a classifier capable of accurately detecting and identifying which of 10 species' vocalizations appear in a given audio recording.

## 2. Related Work and Competition Solutions

As was previously mentioned, bird classification models are an active research focus because of their utility to avian conservation efforts. Approaches and architectures are diverse, and continue to be refined in the interest of reducing inference runtimes and improved accuracy and generalization across regional contexts.

BirdNET is a popular, ResNet-based model that was developed at Cornell University and released in 2021 [3][4]. BirdNET was trained on large amounts of labelled avian acoustic data, with pre-processing steps including normalization, augmentation and mixup [4]. Leveraging spectrogram representations of the audio data, BirdNET was capable of identifying 984 North American and European bird species with good accuracy (at time of release) [3]. However, specialized regional models often outperformed it on local tasks. Recent research has demonstrated BirdNET's ability to transfer to novel tasks – many regionally-focused, specialized models use BirdNET or its embeddings as a backbone [5][6].

Another generalist model, Perch, was developed by Google researchers and first released in 2023. While version 1.0 was focused on birds, version 2.0 (2025) expands Perch's classification focus to multiple taxa [7]. Perch 2.0 uses self-distillation with a prototype-learning classifier and a new source-prediction training criterion to achieve state-of-the-art performance on BirdSet and the BEncmark of Animal Sounds (BEANS) [7]. BirdVoxDetect (2024), yet another model, targets nocturnal migratory calls and uses a combination of random forest, deep CNN and hierarchical classifiers on spectrograms [8].

Notably, these kinds of large, generalized models lack the specificity and fine-tuning needed for region-specific, rare species challenges like BirdClef 2023 – these models, or their embeddings are often used as a backbone or foundation instead [9][10]. As a result, it may be more valuable to consider the approaches taken by successful entrants to the competition to inform model development in this project. Considering the top 3 solutions on Kaggle, some noteworthy practices included data augmentation [11][12][13], chunking [13], supplementing competition data with external datasets [13], use of SED models [11][13], ensembling [11], and use of CNNs (particularly EfficientNet, as a backbone) [11][13]. Not all techniques may transfer well to the objectives of the reduced scope version of the competition.

## 3. Data

### 3.1 Data Source

All data used in this project was obtained from the BirdClef 2023 Kaggle competition, hosted by the Cornell Lab of Ornithology. Training data is composed of short clips from xeno-canto.org [14], which are submitted by members of the public. Clips have associated metadata which includes a primary\_label (i.e. the primary species identified in the clip), author, latitude/longitude and filename. Competition test data is in the form of unlabelled 10-minute long clips (soundscapes) – as was previously mentioned, these soundscapes were not used in this project, and training/validation/test data was split out of the provided training dataset to facilitate efficient local testing.

### 3.2 Exploratory Data Analysis

Prior to processing, exploratory data analysis (EDA) was performed on the provided training data to understand class distributions, gaps and select 10 species of focus. EDA revealed ~16000 rows of metadata (i.e. ~16000 total recordings), with latitude/longitude missing from 1% of these metadata rows. 86.3% of records were designated with a quality of rating of 3 or more (out of 5). The top 10 species were extracted by identifying the 10 species with the most primary label occurrences in the training audio metadata file. Choosing the most plentiful species meant having more data to train on, which was intended to produce higher accuracy scores later on. The balance of these classes in both the original dataset and the reduced scope dataset are summarized in Table 1.

**Table 1:** Summary of top 10 species metadata and class distribution.

Primary label	Common name	Family	Primary label occurrences	% of original dataset	% of reduced dataset
barswa	Barn Swallow	Hirundinidae (Swallows)	500	2.95	10.5
wlwwar	Willow Warbler	Phylloscopidae (Leaf Warblers)	500	2.95	10.5
thrnig1	Thrush Nightingale	Muscicapidae (Old World Flycatchers)	500	2.95	10.5
eaywag1	Western Yellow Wagtail	Motacillidae (Wagtails and Pipits)	500	2.95	10.5
comsan	Common Sandpiper	Scolopacidae (Sandpipers and Allies)	500	2.95	10.5
woosan	Wood Sandpiper	Scolopacidae (Sandpipers and Allies)	486	2.87	10.2
combuz1	Common Buzzard	Accipitridae (Hawks, Eagles, and Kites)	477	2.82	10.0
eubeat1	European Bee-eater	Meropidae (Bee-eaters)	437	2.58	9.18
hoopoe	Eurasian Hoopoe	Upupidae (Hoopoes)	436	2.57	9.16
cohmar1	Common House-Martin	Hirundinidae (Swallows)	425	2.51	8.93

This set of the top 10 species is well-balanced overall, reducing the need for downsampling, and has a combination of unique and similar species (i.e. belonging to the same family). All together, these samples represent roughly a third of the original dataset (4761 records).

### 3.3 Data Processing

The first stage of processing the data was cleaning. Durations of each clip were extracted using librosa, and appended to the metadata of each record. The dataset was also deduped, according to a strategy used by the 1st place solution [12], wherein duplicates are defined as records with matching duration, author and primary labels. Deduplication removed 66 samples.

Further cleaning was performed, roughly following the steps detailed in [15]. Unused columns (secondary\_labels, scientific\_name, common\_name, author, license, url) were dropped from the metadata to reduce dimensionality. Records with NaN values were not removed – with only 63 occurrences in the latitude/longitude columns, doing so was deemed to have minimal impact. The “type” column was

cleaned, reconciling one-off inclusions of lifestage and sex so the column only included four possible values: call, song, blank or both. Longitude and latitude were mapped to countries and continents using batched geopy API calls [15][16]; country and continent were appended, however they did not end up being used in model training. Finally, recordings were trimmed to remove leading or trailing silences, chunked into 15 second clips, and padded where necessary to maintain consistency of scale and to improve ease of augmentation.

### 3.4 Data Augmentation

Generally, the goal of data augmentation in BirdClef challenges is to handle the domain shift between training (single clips) and test data (long soundscapes). While this motivation is less applicable to this reduced-scope version of the competition, augmentation remains beneficial as it produces many additional samples for more robust training, and helps the model generalize more effectively.

A variety of augmentation techniques were selected from competition solution writeups, based on which techniques appeared most often. These were: adding background noise (2021 2nd place [17], [15], 4th [18]), adding Gaussian/pink/brown noise (2nd [11], 3rd [13], 6th [10]), pitch and time shifting (2nd [11], 3rd [13], [15], 4th [18]), time and frequency masking (2nd [11]) and colour jitter. Mixup, a technique that combines multiple samples using an overlay [19] was also used at the waveform level to produce additional samples, and is featured in 1st [12], 3rd [13] and 4th place [18] solutions, though implemented differently across these solutions. Noise augmentations were defined using numpy, all other techniques were applied using a combination of numpy and librosa. Parameters and constants in the implementation of these augmentations were standard and not adjusted between model runs. All Kaggle files had augmentations applied and these augmented files are appended to the provided training data.

## 4. Methodology

### 4.1 Feature Extraction

After cleaning and applying augmentations, audio features were manually extracted from each recording to provide learnable, dimensionally-reduced representations to the models. As was described in the literature review, audio classification models often use Mel spectrograms for input. In this case, features were directly extracted from these spectrograms. Features included the spectrograms themselves, Mel-frequency cepstrum coefficients (MFCC), log-Mel spectrogram means (vector of the means of each of 64 Mel bands over time), chroma (a feature that summarizes pitch classes) and root mean square energy (RMS) [15].

For baseline testing with linear regression and random forest models, RMS and log-Mel spectrogram means were used. For the recurrent neural networks (RNNs) and convolutional neural networks (CNNs) tested, Mel spectrograms were used directly, instead of taking band means. This is because RNNs and CNNs are better suited to learn from temporal patterns and sequential data [20]. The 1D CNN used MFCC(20) and Chroma(12) based on the CNN classifier described in [15]. MFCCs capture the timbral envelope of bird calls while Chroma features encode harmonic and pitch-class information, providing a complementary representation that helps the 1D CNN distinguish species with both spectral-shape and tonal differences. Loading audio files and re-extracting the features on each run was time-consuming and computationally costly, so feature vectors were stored in .pkl files for repeated use.

### 4.2 Training, Test and Validation Data

Data was split into test, training, and validation using the `train_test_split` from scikit-learn. The percentage splits used were 70% training, 15% validation and 15% test. Data was stratified to ensure class balances within the training set, and within the validation and test sets altogether.

### 4.3 Model Selection and Initial Experiments

Since 10 species are being identified, a random guess should yield a 10% accuracy. This accuracy is used as a baseline metric throughout experimentation.

Experimentation of different models progressed with increasing complexity, beginning with linear regression and random forest baselines, to a CNN and RNN. BirdNET, a pre-trained model, was also tried. Model accuracies on the test set were then compared.

## 5. Results and Discussion

### 5.1a Baseline Model (Logistic Regression, Random Forest) Results

Baseline performance was evaluated using two standard classifier architectures: logistic regression (linear) and random forest (nonlinear). Each model was trained independently on RMS energy features and log-Mel spectrogram means to assess feature effectiveness. Table 2 summarizes the results.

**Table 2:** Baseline model accuracies

Model	Feature	Accuracy
Logistic Regression	RMS	0.175331
	Log-Mel spectrogram means	0.487258
Random Forest	RMS	0.15800
	Log-Mel spectrogram means	0.394495

Table 2 shows that RMS-based features yielded accuracies near random guessing, indicating inadequate information capture for classification. In contrast, log-Mel spectrogram features achieved substantially higher performance, indicating better information capture. Logistic regression performed poorly as it is generally better suited for simple classification tasks (linearly separable data) [15].

### 5.1b 1D CNN Results

For the CNN experiments, audio features were no longer average-pooled or collapsed across frequency bands. Instead, MFCC (20) and Chroma (12) features were kept in their full 2D time–frequency form so the model could convolve along the time axis, preserving temporal structure. This allowed the network to learn short and mid-range temporal patterns in the calls.

Model development proceeded iteratively through three versions of the 1D CNN architecture. Version 1 served as a lightweight baseline with limited depth and minimal regularization. Although it established a starting point, the architecture lacked sufficient capacity to capture the complexity of the inputs and therefore produced modest performance with 37.04% test accuracy. This motivated an expansion of the model, where deeper convolutional layers, larger kernels, and regularization mechanisms such as dropout and batch normalization were introduced. These changes significantly improved stability during training and increased test accuracy to 65.07%, demonstrating that the CNN required a richer hierarchical structure to extract meaningful temporal patterns from the features.

Building on these insights, version 3 of the architecture expanded the model’s capacity through additional convolutional blocks, increased filter widths, and stronger regularization. These changes allowed the CNN to more effectively model the multi-scale temporal patterns present in the MFCC–Chroma features. This version also introduced audio-based data augmentation, including noise injection and temporal warping, which marginally improved performance by 2.25%. Together, these architectural refinements

and training enhancements produced a substantial performance improvement, with the augmented model achieving 87.04% test accuracy.

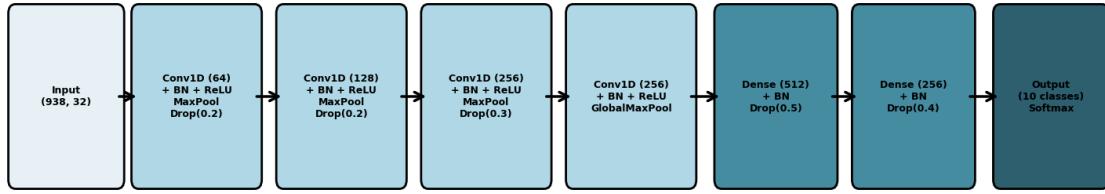


Fig. 1. Final 1D CNN model architecture.

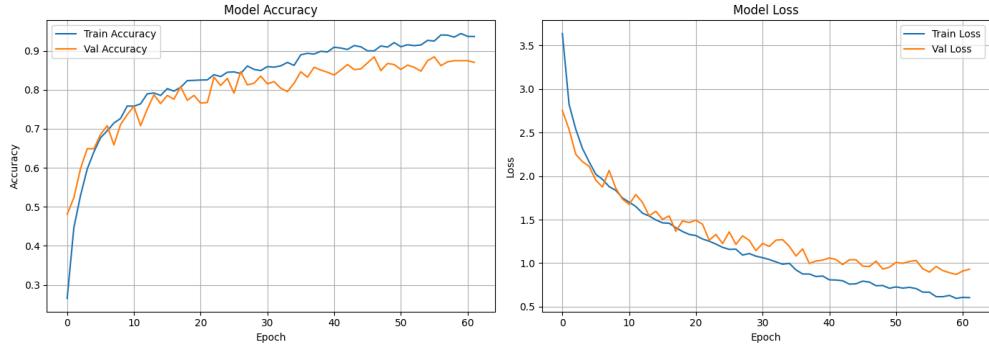


Fig. 2. 1D CNN final model accuracy and loss over epochs.

### 5.1c RNN Results

Due to the nature of audio being time dependent, a gated recurrent unit (GRU) model was also tried. GRU RNNs are known for their ability to process sequential data [20] and are commonly used in speech recognition contexts [21]. Initially, the model was trained with average pooled Mel spectrogram features (i.e. band means). This, however, yielded a test accuracy of 49.01%. Early stopping was used throughout this experimentation to prevent overfitting and stop the model once it started to degrade, hence the variation in epochs performed across the models.

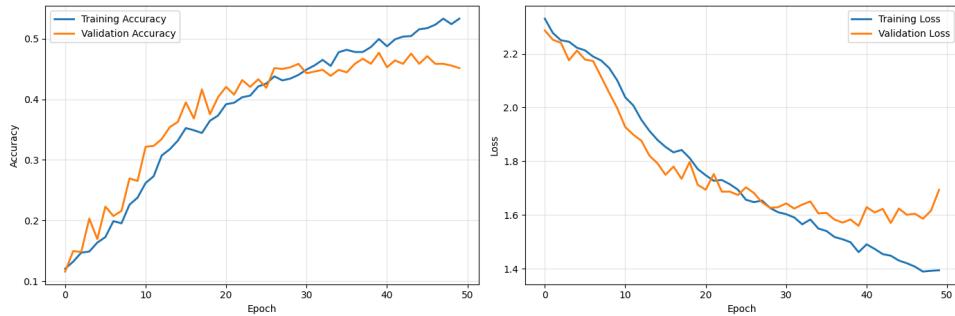


Fig. 3. GRU model accuracy and loss over epochs using average pooled features.

As was mentioned in section 4.1, it was realized that because of the importance of temporal sequences in RNNs, average pooling the features would significantly impact the results of the RNN model as it is designed specifically to learn temporal patterns. The model was thus restructured and trained with the combination of the Mel spectrogram and RMS features without average pooling which significantly improved the model performance to a test accuracy of 80.13%.

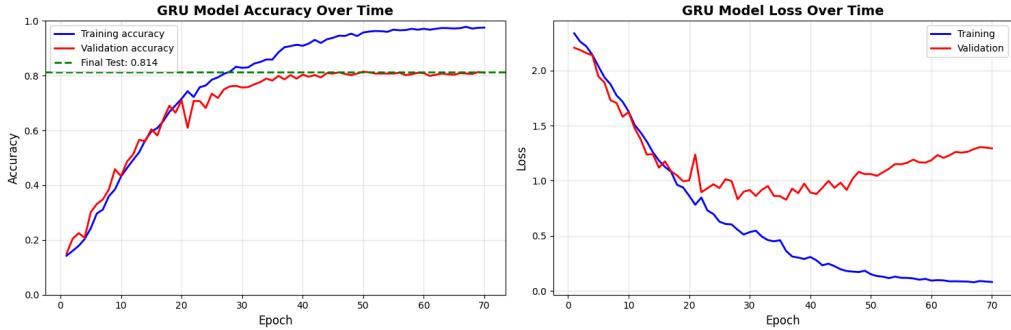


Fig. 4. GRU model accuracy and loss over epochs without average pooled features.

Additionally, `reduceLROnPlateau` was used to dynamically lower the learning rate when a monitored metric stops improving.

### 5.1d BirdNET Results

BirdNET, the aforementioned pre-trained model, was also tried out-of-the-box on the Kaggle competition data. This model is trained on significantly more species than are considered in this project, and as expected, immediately produced strong accuracy results without fine-tuning.

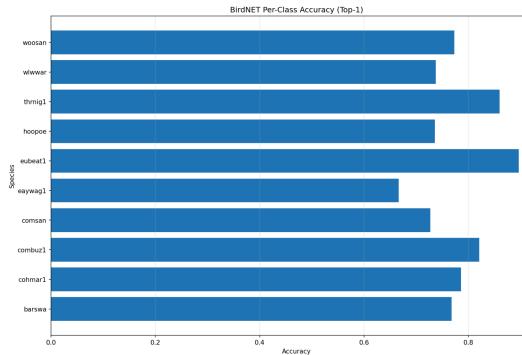


Fig. 5. BirdNET top-1 per-class accuracy, visualized.

### 5.2 Discussion

Across all model experiments, a common approach that tended to work well was the use of features geared to the strengths of each model.

One issue that was faced often in the experimental process was the risk of overfitting. Initial RNN and CNN models both showed high accuracy on the validation set, but produced substantially lower test accuracies. Increasing the complexity of each model was important to be able to capture and learn from salient features in both cases.

Data augmentation was found to be mostly inconsequential across model runs, resulting in marginal accuracy gains only. A speculated reason for this result is the lack of a need for increased generalization to achieve the end goal of this project. Since training, validation and test datasets were fundamentally similar, it was less important to bridge domain shifts between the sets – the model gained no benefit from the augmented samples.

Because of the reduced dataset sizes involved in this project, and the use of pre-computed features, runtime was trivial for both training and inference. Models were run on Google Colab's GPU, taking a range of 2-10 minutes to run 60-100 epochs (epochs depended on the use of early stopping). However, if

these models were to be included in the actual competition, they would likely be incapable of fulfilling the time limit of running in 2 hours on CPU due to inefficiencies that are imperceptible with this reduced dataset. Runtime was not considered across the development process due to the trivial runtime associated with completing the reduced scope task on GPU.

Overall, while both models achieve reasonable accuracy when compared to the baseline metric, and achieve the initially stated project goal, these models would not extend well to the official BirdClef competition as they are likely not robust enough to account for 200+ species.

## 6. Conclusions and Next Steps

The models described in this report are able to achieve accuracies significantly higher than the “guessing” baseline metric in the classification of 10 rare East African bird species. These models run quickly (on GPU) and are based on deep networks with relatively simple architectures. Data processing techniques, such as the production of audio spectrograms and extraction of their features, were necessary for the model to learn effectively from the audio recordings.

Crucially, this project involved many assumptions and reductions of scope from the original competition brief. Thus, most suggested next steps are related to broadening the model developed in this project to the original scope of the brief. Some techniques from competition solutions to try include ensembling, knowledge distillation, pre-training on past competitions’ data, and including different augmentation techniques.

Additionally, the accuracy of the model described in this project could benefit from additional hyperparameter tuning. Due to the limited time constraints of the course, efforts to tune model hyperparameters were not extensive, and could be expanded upon to achieve optimal accuracy on the selected architecture. Ablation studies were not conducted on the models tried.

Another possible next step is to extract different features than the ones described in section 4.1. For example, quality ratings, country and continent were available in competition metadata, but never used. These features could have helped the models learn associations between geographic locations and species, or be able to discount lower quality samples. A different approach that could have been used is to use another simple model to learn embeddings, or leverage the embeddings of a model like BirdNET [10]. Pre-trained models could be leveraged more overall (e.g. as a backbone), as is evidenced from the strong initial performance of BirdNET without fine-tuning.

## References

- [1] S. Kahl, T. Denton, H. Klinck, H. Reers, F. Cherutich, et al., "Overview of BirdCLEF 2023: Automated bird species identification in Eastern Africa," in *Proc. 24th Conf. Labs Eval. Forum (CLEF-WN 2023)*, Thessaloniki, Greece, Sep. 2023, pp. 1934-1942.
- [2] "Birdclef 2023," Kaggle, <https://www.kaggle.com/competitions/birdclef-2023/overview> (accessed Dec. 1, 2025).
- [3] S. Kahl, C. M. Wood, M. Eibl, and H. Klinck, "BirdNET: A deep learning solution for avian diversity monitoring," *Ecological Informatics*, vol. 61, p. 101236, Mar. 2021. doi:10.1016/j.ecoinf.2021.101236
- [4] "About BirdNET," BirdNET, <https://birdnet.cornell.edu/about/> (accessed Dec. 1, 2025).
- [5] G. Schiavo, A. Portaccio, and A. Testolin, "Fine-tuning birdnet for the automatic ecoacoustic monitoring of bird species in the Italian alpine forests," *Information*, vol. 16, no. 8, p. 628, Jul. 2025. doi:10.3390/info16080628
- [6] H. Dogan, Improved detection of bird vocalisations using BirdNET embeddings and machine learning, May 2025. doi:10.31224/4466
- [7] B. van Merriënboer et al., "Perch 2.0: The bittern lesson for Bioacoustics," *Perch 2.0: The Bittern Lesson for Bioacoustics*, <https://arxiv.org/html/2508.04665v1> (accessed Dec. 1, 2025).
- [8] V. Lostanlen et al., "BIRDVOXDETECT: Large-scale detection and classification of flight calls for Bird Migration Monitoring," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 4134–4145, 2024. doi:10.1109/taslp.2024.3444486
- [9] A. Miyaguchi, A. Cheung, M. Gustineli, and A. Kim, "Transfer Learning with Pseudo Multi-Label Birdcall Classification for DS@GT BirdCLEF 2024," *Transfer learning with pseudo multi-label BIRDCALL classification for DS@GT BirdClef 2024*, <https://arxiv.org/html/2407.06291v1> (accessed Dec. 1, 2025).
- [10] anonamename, "6th place solution: Birdnet embedding + CNN," Kaggle, <https://www.kaggle.com/competitions/birdclef-2023/writeups/anonamename-6th-place-solution-birdnet-embedding-c> (accessed Dec. 1, 2025).
- [11] honglihang, "2nd place solution: SED + CNN with 7 models ensemble," Kaggle, <https://www.kaggle.com/competitions/birdclef-2023/writeups/griffith-2nd-place-solution-sed-cnn-with-7-models-> (accessed Dec. 1, 2025).
- [12] vladimirsydor, "1st Place Solution: Correct Data is all you need," Kaggle, <https://www.kaggle.com/competitions/birdclef-2023/writeups/volodymyr-1st-place-solution-correct-data-is-all-y> (accessed Dec. 1, 2025).
- [13] mariotsaberlin, "3rd Place Solution: SED with attention on Mel frequency bands," Kaggle, <https://www.kaggle.com/competitions/birdclef-2023/writeups/adsr-3rd-place-solution-sed-with-attention-on-mel-> (accessed Dec. 1, 2025).
- [14] xeno-canto, <https://xeno-canto.org/> (accessed Dec. 1, 2025).

- [15] R. Gao, “Bird Song Classifier with Machine Learning,” Bird Song Classifier with Machine Learning | Rachel Gao, [https://rachlllg.github.io/project/2023-Bird\\_Song\\_Classifier\\_with\\_Machine\\_Learning/](https://rachlllg.github.io/project/2023-Bird_Song_Classifier_with_Machine_Learning/) (accessed Dec. 1, 2025).
- [16] “geopy,” PyPI, <https://pypi.org/project/geopy/> (accessed Dec. 1, 2025).
- [17] christofhenkel, P. Pfeiffer, and philippsinger, “2nd place solution,” Kaggle, <https://www.kaggle.com/competitions/birdclef-2021/writeups/new-baseline-2nd-place-solution> (accessed Dec. 1, 2025).
- [18] atsunorifujita, “4th place solution: Knowledge distillation is all you need,” Kaggle, <https://www.kaggle.com/competitions/birdclef-2023/writeups/atfujita-4th-place-solution-knowledge-distillation> (accessed Dec. 1, 2025).
- [19] L. Monigatti, “Data augmentation techniques for audio data in python,” Towards Data Science, [https://towardsdatascience.com/data-augmentation-techniques-for-audio-data-in-python-15505483c63c/](https://towardsdatascience.com/data-augmentation-techniques-for-audio-data-in-python-15505483c63c) (accessed Dec. 1, 2025).
- [20] I. D. Mienye, T. G. Swart, and G. Obaido, “Recurrent neural networks: A comprehensive review of architectures, variants, and applications,” *Information*, vol. 15, no. 9, p. 517, Aug. 2024. doi:10.3390/info15090517
- [21] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, “Light gated recurrent units for speech recognition,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 92–102, Apr. 2018. doi:10.1109/tetci.2017.2762739