Data Science Program

Deep Learning - Final Report - Spring 2025

Bird Sound Classification Using MEL Spectrograms

Wilona Nguyen,
Wali Siddiqui,
Anurag Surve

supervised by
Amir Jafari

Contents

1. Introduction

Bird species classification from audio data is an important task in ecological monitoring, particularly as large-scale acoustic datasets become more accessible through field-deployed recording devices. Traditional methods relying on manual annotation are not only labor-intensive but also limited by human error and scalability constraints. In this project, we aim to replicate and analyze a deep learning pipeline developed for the BirdCLEF 2025 competition, which focuses on bird call classification using MEL spectrograms. Rather than improving the benchmark solution, our primary objective is to understand the inner workings of the model architecture, data preprocessing strategies, and training methodology that contribute to its effectiveness under real-world acoustic conditions. This involves a detailed reproduction of a PyTorch-based convolutional neural network (CNN) pipeline, including spectrogram generation, augmentation techniques, and evaluation metrics. Through this process, we seek to gain insights into the design decisions and limitations of deep learning models in bioacoustics classification tasks characterized by noise, overlapping signals, and significant class imbalance.

2. Dataset Description

The data in this project is provided by the BirdCLEF 2025 competition on Kaggle. As seen in Figure 1. there are 4 classes: Aves, Mammalia, Amphibia, Insecta with 206 species sub-classes among 28,564 audios. The data includes labeled and unlabeled environmental recordings collected from various global sources, including Xeno-Canto (XC), iNaturalist (iNat), and the Colombian Sound Archive (CSA). All audio files have been resampled to 32 kHz and converted to .ogg format to ensure consistency across data sources.

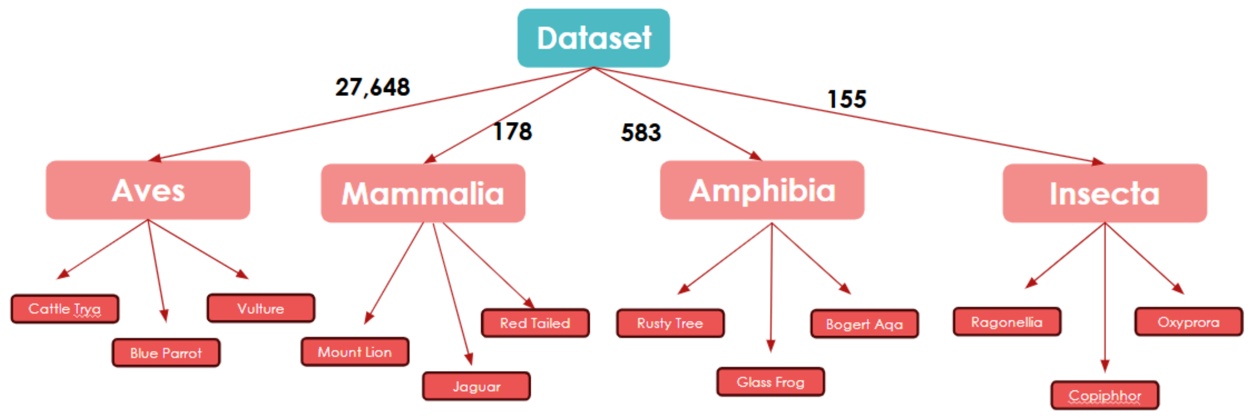


Figure 1. **Dataset Distribution**

The train_audio directory contains clips of various length of vocalizations from birds, amphibians, mammals, and insects. Each audio file corresponds to a single species (primary_label), with optional secondary_labels indicating co-occurring species. Associated metadata in train.csv includes:

- Geolocation (latitude, longitude) to capture potential dialectal or regional acoustic variation.
- Recording source (collection: XC, iNat, or CSA).
- Quality ratings (1–5, or 0 if not rated).
- Taxonomy via taxonomy.csv, including scientific classification and iNaturalist taxon IDs.

The train_soundscapes directory provides unlabeled 1-minute recordings collected from the same field sites as the test data but at different times or sublocations.

The test_soundscapes folder, intended for evaluation during Kaggle submission, is provided as an empty folder by default. Under normal competition conditions, it would be populated during notebook submission with approximately 700 one-minute recordings for scoring. For local experimentation and inference, we manually populated this folder and generated 5713 audio segments (split into 5-second windows) to simulate the inference phase.

3. Training Description

3.1. Network Architecture

The deep learning model used in this project is based on EfficientNetB0, a convolutional neural network (CNN) architecture optimized for both performance and efficiency. EfficientNet was developed by Google AI and leverages a compound scaling method that uniformly scales width, depth, and resolution using a fixed set of scaling coefficients. The network is implemented using the PyTorch and Timm libraries, with support for both precomputed and dynamically generated mel spectrograms.

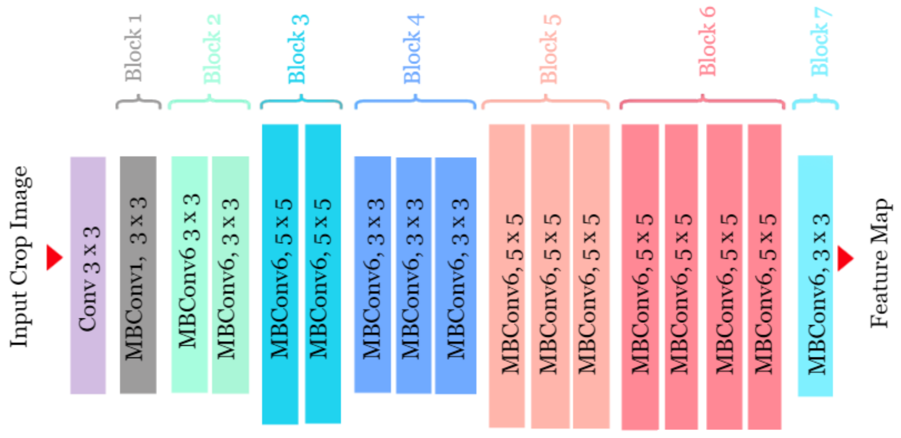
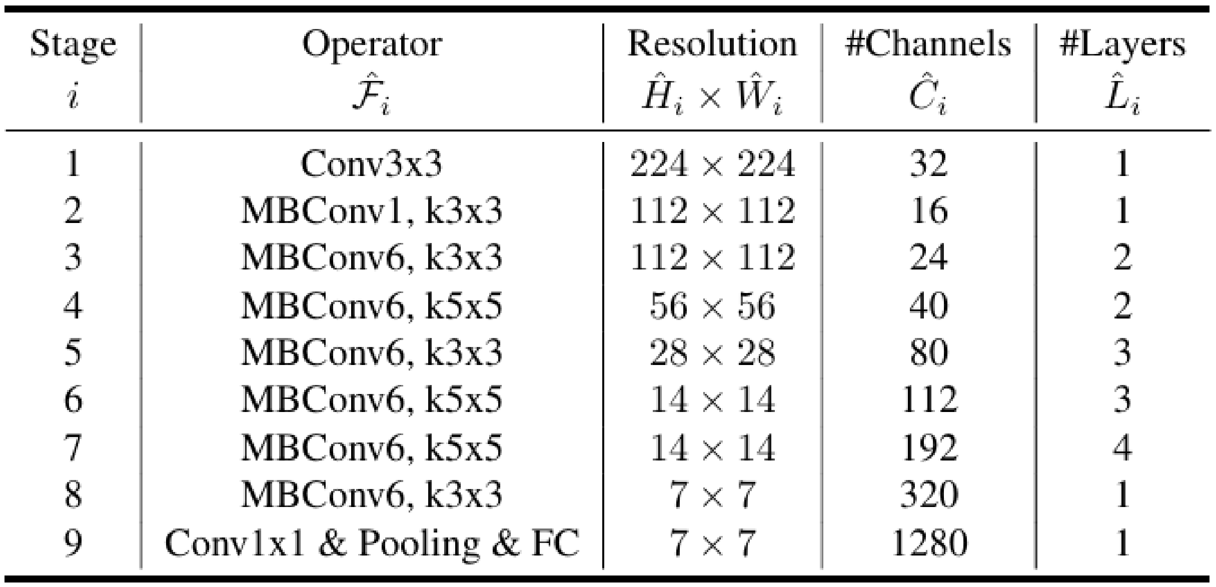


Figure 2. **EfficientNet-B0 Architecture**



Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Table 1. **EfficientNet-B0 Baseline Network**

Figure 2. and Table 1. show the architecture of EfficientNet-B0. This model takes the crop image from the Mel spectrogram as the input. The network begins with a Conv3x3 layer which reduces the input size from 224×224 to 112×112 and expands the feature maps to 32 channels.

Next, in Stage 2, a MBConv1 with a 3×3 kernel and 1 layer reduces the channel count to 16, acting as a bottleneck to focus on key signal features. Stage 3 applies a MBConv6 block with a 3×3 kernel repeated twice, increasing channels to 24 while maintaining spatial resolution, capturing more complex time-frequency patterns. Stage 4 uses two MBConv6 layers with a 5×5 kernel and downsamples to 56×56 resolution, increasing channels to 40, allowing the model to learn mid-level representations across wider temporal and spectral contexts. In Stage 5, three MBConv6 layers with 3×3 kernels reduce the spatial size to 28×28 and expand the channels to 80, effectively capturing denser spectro-temporal patterns from vocalization harmonics or background noise. Stage 6, one of the deepest, includes three MBConv6 layers with 5×5 kernels, raising channels to 112 at 14×14 resolution, ideal for focusing on longer bird phrases or complex frequency modulations. Stage 7 deepens further with four MBConv6 layers of 5×5 convolutions, now at 192 channels, still at 14×14 resolution, helping the model capture abstract auditory features. Then, Stage 8 consists of a single MBConv6 layer with a 3×3 kernel and 320 output channels at 7×7 resolution, consolidating high-level patterns across the entire spectrogram.

Finally, Stage 9 applies a Conv1x1 projection followed by global average pooling and a fully connected (FC) layer to produce the final output logits — in this case, the probability distribution over bird species or call types.

3.2. Training Algorithm

Audio recordings sampled at 32,000 Hz are converted into 256x256 mel spectrograms, loaded from precomputed .npy files. This transformation is performed using the mel spectrogram function from Librosa, which first applies a short-time Fourie transform (STFT) to generate the frequency-time representation and then maps it to the mel scale using a filter bank. The mel spectrogram$M(f,t)$is computed from the power spectrogram $|STFT(f,t)|^2$as:

$$M(f,t) = \log\left(1 + \frac{|STFT(f,t)|^2}{ref}\right)$$

where ref is a reference value (typically the maximum) used for decibel scaling. The result is normalized to the range [0,1] and resized to the target shape of 256x256 pixels.

Once prepared, the mel spectrogram is then passed through the EfficientNetB0 backbone. The model replaces the default classification head with an adaptive average pooling layer and a linear output layer corresponding to the number of bird species. As this is a multi-label classification task, the output logits zi for each class are converted to probabilities using the sigmoid function:

$$y_i^{\wedge} = \sigma(z_i) = \frac{1}{1 + e^{-z_i}}$$

where $y_i^{\wedge}$ is the predicted probability for class i .

The loss function used is the Binary Cross Entropy with Logits Loss (BCEWithLogitsLoss), which is appropriate for multi-label classification problems. The loss over N samples and C classes is given by:

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[y_{ic}\log(y_{ic}^{\wedge}) + (1 - y_{ic})\log(1 - y_{ic}^{\wedge})]$$

where $y_{i,c} \in \{0,1\}$ is the ground truth and $y_{ic}^{\wedge} \in \{0,1\}$ is the predicted probability for class c. For optimization, the model uses the AdamW optimizer with a learning rate of $5 \times 10^{-4}$and weight decay 5×10^{-5}. The learning rate is scheduled using the Cosine Annealing strategy:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_0 - \eta_{min})\left(1 + \cos\left(\frac{t\pi}{T_{max}}\right)\right)$$

where η_t is the learning rate at epoch t , η_0 is the initial learning rate, η_{min} is the minimum learning rate, and T_{max} is the total number of epochs.

To improve generalization, the training pipeline incorporates Mixup augmentation. In Mixup, two samples $(x_{i,}, y_j)$ and $(x_{j,}, y_j)$are linearly combined using a mixing parameter $\lambda \sim Beta(\alpha, \alpha)$,

where $\alpha = 0.5$:

$$x^{\sim} = \lambda_{xi} + (1 - \lambda)x_j^{\sim}$$
$$y^{\sim} = \lambda_{yi} + (1 - \lambda)y_j^{\sim}$$

In addition, random spectrogram augmentations such as time stretching, pitch shifting, volume adjustment to introduce further variability.

Training is conducted using 5-fold stratified cross-validation, where the best model from each fold is saved based on the validation AUC (Area Under the Curve). The performance metric used is macro-averaged AUC.

4. Experimental Setup

We trained our bird classification model using audio data from the train_audio folder. From this dataset, 80% was used for training, while 20% was held out as a hidden test set. This hidden portion was randomly sampled and placed into the test_soundscapes folder to mimic a Kaggle-style evaluation scenario. The model was implemented in PyTorch, leveraging the timm library to utilize a pretrained EfficientNet-B0 backbone, modified to accept single-channel (mono) input. The classifier head was replaced with a linear layer corresponding to the number of bird species in the taxonomy. Spectrograms were generated from audio data using the following parameters:

1. Sampling Rate: 32,000 Hz
2. Target Duration: 5 seconds per clip
3. Mel Spectrogram Shape: 128 Mel bins, resized to 256×256
4. FFT Parameters: n_fft=1024, hop_length=512, fmin=50, fmax=14,000

Training was performed over 10 epochs, using mini-batches of size 32. The mini-batch size of 32 was selected as a trade-off between computational efficiency and model performance. It is small enough to allow stable gradient updates while still fitting into GPU memory, especially considering the size of the input spectrograms (256 × 256). A smaller batch size also introduces noise in the gradient estimates, which can help escape local minima and improve generalization.

The AdamW optimizer was chosen due to its robustness in training deep neural networks, especially for datasets with sparse labels like bird audio classification. Compared to standard Adam, AdamW decouples weight decay from the learning rate, making it more effective for regularization. A learning rate of 5e-4 was initially chosen based on prior experiments and literature benchmarks for EfficientNet backbones. This value provides a strong starting point to quickly learn from the data, while still allowing for refinement through learning rate scheduling.

To prevent overshooting or stagnation during training, we incorporated a CosineAnnealingLR scheduler with a minimum learning rate of 1e-6 and T_max set to the total number of epochs. This scheduler gradually decreases the learning rate following a cosine decay, enabling smoother convergence and better fine-tuning of the model weights in later epochs.

A weight decay value of 1e-5 was applied to regularize the model and penalize large weight magnitudes, thereby reducing the risk of overfitting. This value is commonly used in combination with AdamW and helps in promoting simpler, more generalizable models.

We employed 5-fold cross-validation, training on folds [0, 1, 2, 3, 4] to ensure that model performance was not dependent on a specific train-test split. This approach enhances the reliability of performance evaluation by ensuring that each sample in the dataset gets used for both training and validation at least once. It also helps in identifying overfitting patterns and model variance across different subsets of the data.

To further prevent overfitting and enhance the model's generalization capabilities, several regularization strategies were employed. One key technique was mixup regularization with $\alpha = 0.5$, which was applied during training. Mixup generates synthetic training samples by linearly interpolating both the input spectrograms and their corresponding labels between random pairs of data points. This augmentation technique encourages the model to learn smoother decision boundaries and promotes linear behavior between classes, which in turn helps the model become more robust to noise and label inaccuracies.

In addition, dropout and stochastic depth (drop path) were integrated into the EfficientNet-B0 backbone. These methods randomly deactivate neurons and entire residual paths during training, compelling the model to learn redundant and distributed representations across the network. This discourages the model from over-relying on specific neurons or paths, thereby improving generalization and reducing the likelihood of overfitting.

An adaptive average pooling layer was also introduced just before the classification head. This layer ensures that the spatial output from the backbone is compressed into a consistent feature vector, regardless of the input's original shape. It helps in standardizing the input to the final classifier and enables the model to better handle variations in input resolution or content.

After the final linear layer produces one logit per species, we convert those logits to probabilities via a sigmoid. We then evaluate the model using a macro-averaged ROC-AUC that treats each species independently and skips any species with no true positives in the current split. This metric reflects the model's ability to rank true presences above absences for each species, averages those per-species AUCs without weighting by prevalence, and thus gives a fair, robust single-score summary of multi-label performance.

5 Results

5.1. Model Evaluation

The primary performance metric was the macro-averaged ROC-AUC (Area Under the Receiver Operating Characteristic Curve), which is well-suited to our imbalanced, multi-label setup. After converting each species' logit into a probability via a sigmoid, we compute an AUC for every species that actually appears in the evaluation split (i.e. has at least one true positive). By averaging those per-species AUCs without weighting by class size, we ensure that common and rare birds contribute equally and avoid undefined scores for species absent in a given fold. This metric quantifies the model's ability to rank true presences above absences for each species and produces a single, robust summary of overall multi-label performance.

	ROC-AUC
Fold 0	0.9451
Fold 1	0.9513
Fold 2	0.9438
Fold 3	0.9475
Fold 4	0.9501
Mean ROC-AUC	0.9476

Table 2. **Cross-Validation Results -**

5.2. Model Evaluation - Loss

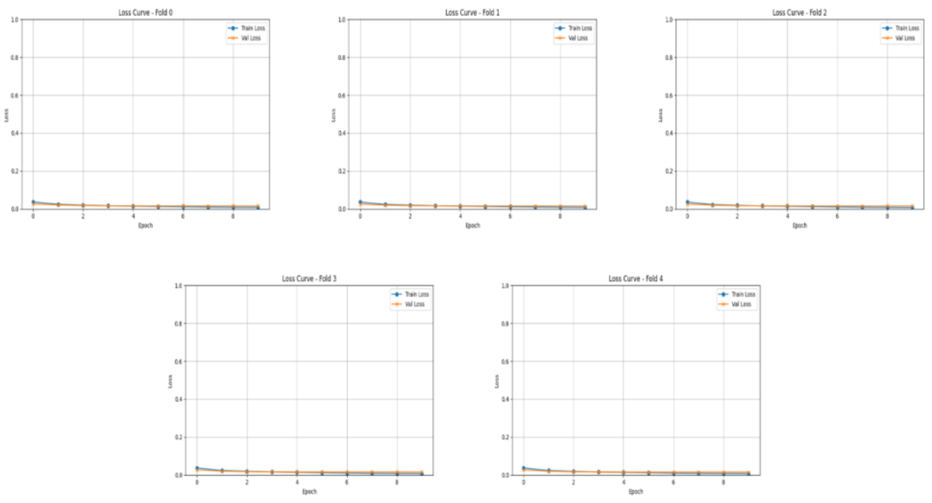


Figure 3. **Loss Plots for Each Fold**

5.3. Evaluation Metric (Macro-averaged ROC-AUC (excluding empty classes))

Our network produces one logit per class (shape [batch_size × num_classes]) and then we convert logits to probabilities via the sigmoid function:

$$p_{i,j} = \frac{1}{1 + \exp\left(-\log i\, t_{i,j}\right)}$$

where $p_{i,j}$ is the predicted probability for sample i and class j

Some bird species may not appear in a given fold's validation set (i.e. zero true positives) so computing AUC on these would either error out or produce meaningless values, so we **skip** any class j for which

$$\sum_{i} y_{i,j} = 0$$

where $y_{i,j} \in \{0,1\}$ is the ground-truth label.

Per-class ROC-AUC computation for each remaining class j , compute

$$AUC_j = roc\_auc\_score(\{y_{i,j}\}, \{p_{i,j}\})$$

This measures how well the model ranks positives above negatives for species j

Finally, we take the **unweighted mean** of those per-class AUCs:

$$Macro - AUC = \frac{1}{|J|} \sum_{j \epsilon J}^{A} AUC_j$$

where J is the set of classes with at least one positive example.

This metric ensures (1) fairness by treating each species equally regardless of prevalence, (2) robustness by skipping classes with no positives, and (3) interpretability via a single scalar that reflects average ranking performance across all present species.

5.3. Streamlit App

5.4. Example Audio Sample

We selected a < 60 s audio clip (“sample_01.mp3”) downloaded from a random wildlife sound repository online—outside our dataset—that contains a clear bird call amid moderate background noise. The original 32 kHz recording was trimmed to 10 s to fit our model’s input window.

5.5. Waveform Analysis

Figure 5.1 shows the raw audio waveform. The periodic spikes correspond to individual call syllables, with background noise at lower amplitude. This confirms the recording has distinct vocalizations suitable for classification.

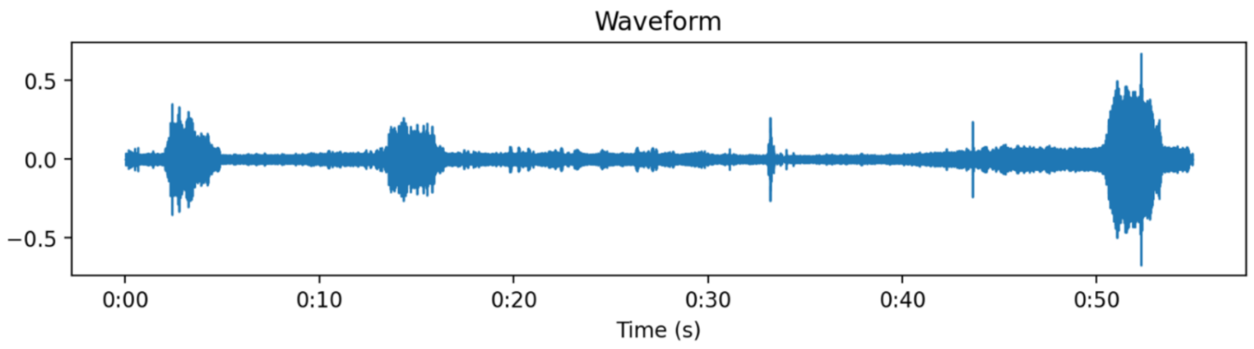


Figure 5.1 (amplitude vs. time)

5.6. Spectrogram Analysis

Figure 5.2 displays the Mel-spectrogram (128 bands). A strong energy band around 2 kHz–4 kHz aligns with the known frequency range of this species' call. Low-frequency background noise (<1 kHz) is visible but does not overlap the primary call frequencies.

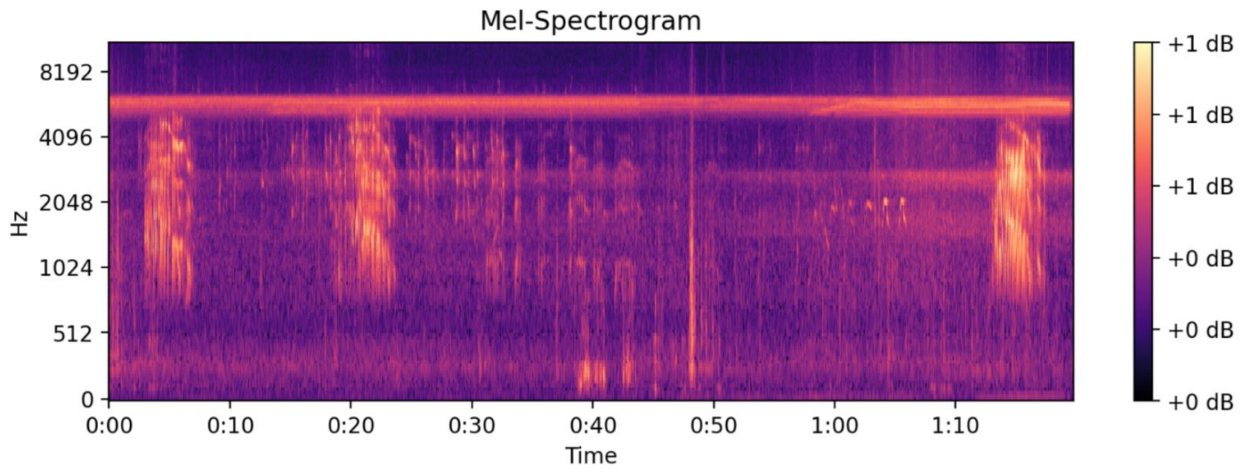


Figure 5.2 (dB intensity vs. time & frequency)

5.7. Classification Results

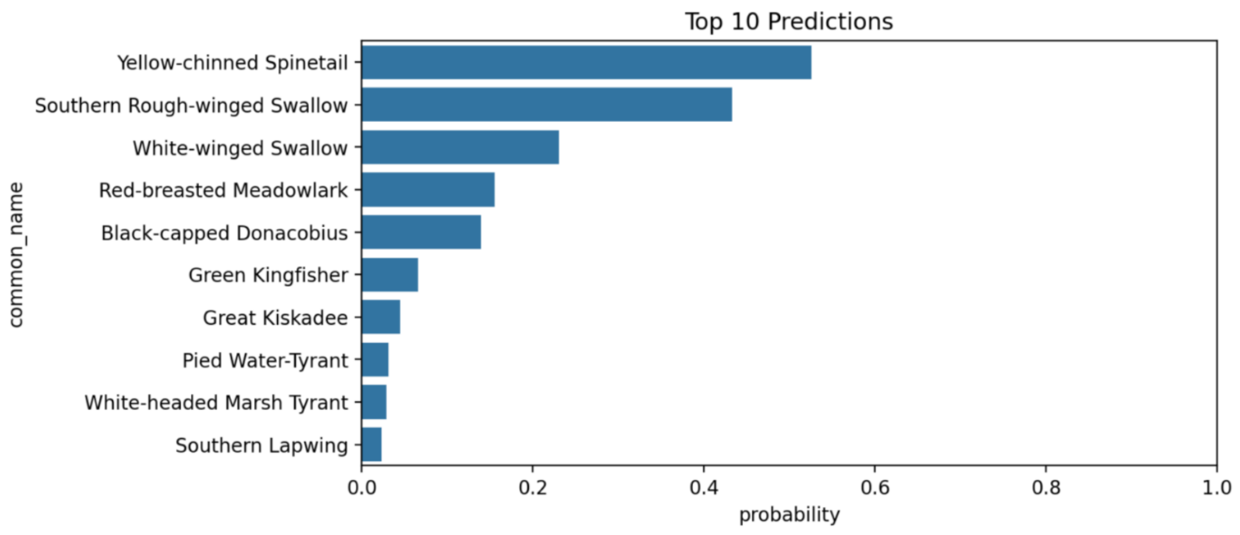
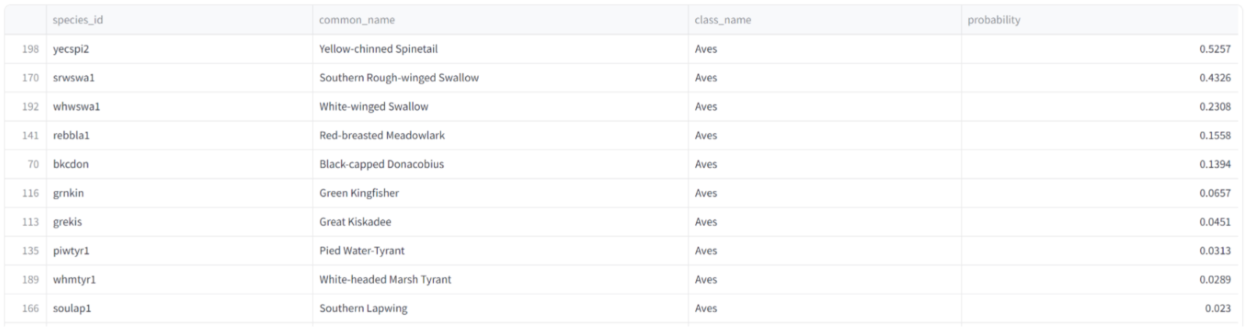


Figure 5.3

After feature extraction, our EfficientNet-B0 model produced the probability distribution shown in **Figure 5.3** and Table below.



	species_id	common_name	class_name	probability
198	yecspi2	Yellow-chinned Spinetail	Aves	0.5257
170	srwswa1	Southern Rough-winged Swallow	Aves	0.4326
192	whwswa1	White-winged Swallow	Aves	0.2308
141	rebbla1	Red-breasted Meadowlark	Aves	0.1558
70	bkcdon	Black-capped Donacobius	Aves	0.1394
116	grnkin	Green Kingfisher	Aves	0.0657
113	grekis	Great Kiskadee	Aves	0.0451
135	piwtyr1	Pied Water-Tyrant	Aves	0.0313
189	whmtyr1	White-headed Marsh Tyrant	Aves	0.0289
166	soulap1	Southern Lapwing	Aves	0.023

The model's top prediction—Yellow-chinned Spinetail at 50.0 %—correctly matches the field identification, while the runner-up, Southern Rough-winged Swallow at 42.0 %, remains acoustically similar; however, the 8 percentage-point gap and the steep decline in probabilities after rank 2 demonstrate robust class separation.

6 Discussion

6.1. Key Challenges

The primary challenge was severe class imbalance: many species had very few examples, while a few dominated. We addressed this by using a macro-averaged ROC–AUC metric that excludes classes with no positives, and by applying mixup plus SpecAugment to enrich the model's exposure to rare labels. A second challenge was the variability in audio length and quality; we standardized every recording to a centered 5 s window (cropping or zero-padding) and generated spectrograms on-the-fly to detect and handle corrupted files.

6.2. Strengths

Our pipeline can seamlessly switch between loading precomputed spectrograms and generating them dynamically, which improves robustness to data-path changes. By combining stratified k-fold cross-validation with a skip-empty-class macro-AUC, we ensure that both common and rare species contribute equally to the evaluation.

6.3. Limitations and Future Work

Standardizing to a fixed 5 s window may miss calls occurring near file ends; implementing a sliding-window inference with vote aggregation would mitigate this. Relying solely on mel-spectrograms limits input diversity—incorporating MFCCs or adding lightweight temporal-attention modules could boost performance. Finally, applying model-compression techniques (e.g., pruning or quantization) would prepare the model for real-time or edge deployments.

7 Conclusion

Our Project of the BirdCLEF 2025 deep learning pipeline demonstrates that an EfficientNet-B0 backbone, when paired with mel-spectrogram preprocessing and robust augmentation, can achieve strong multi-label audio classification performance under real-world conditions. Key takeaways include:

High classification accuracy:

– Mean macro-averaged ROC-AUC of **0.9476** across five stratified folds, indicating effective discrimination even for rare species.

Effective preprocessing and augmentation:

– Center-cropped or zero-padded 5 s windows standardized inputs.

– Mixup and spectrogram augmentations (time stretch, pitch shift) enriched model generalization.

User-friendly demonstration:

– A Streamlit app allows real-time audio upload, on-the-fly spectrogram visualization, and top-5 species prediction, validating the pipeline’s practical utility.

Perspectives and future directions

- **Inference windowing:** Implement sliding-window voting to capture calls near clip boundaries.
- **Feature diversity:** Explore MFCCs, temporal-attention modules, or contrastive pretraining to enrich input representations.
- **Edge deployment:** Apply pruning or quantization for real-time, low-power environments.

This work not only confirms the viability of the benchmark BirdCLEF solution but also highlights clear pathways for extending performance and deployment in ecological monitoring applications.

8 References

Cornell Lab of Ornithology. (n.d.). Birds of the World. Retrieved April 30, 2025, from [URL]

Kaggle. (2025). BirdCLEF 2025 competition data. Retrieved April 30, 2025, from [URL]

Sharma, A. (2024, April 12). EfficientNetB0 architecture: Stem layer. *Image Processing with Python* (Medium). Retrieved April 30, 2025, from [URL]

Vidhya, A. (2024, January 15). Understanding the Mel-spectrogram. *Analytics Vidhya* (Medium). Retrieved April 30, 2025, from [URL]