Spectrogram-Based Bird Sound Classification

Adrià García García
266036 / u213940
adria.garcia04@estudiant.upf.edu

Daniel Ortega Barberán
241163 / u186399
daniel.ortega02@estudiant.upf.edu

Didac Raya Rodriguez
242597 / u189525
didac.raya01@estudiant.upf.edu

1 Introduction

BirdCLEF is a yearly competition focused on deep learning models for bird call recognition, a task with important implications for biodiversity monitoring and ecological research [7]. The competition provides a rich, real-world audio dataset that is a subset from the **Xeno-Canto archive**, which includes thousands of annotated bird vocalizations collected across the globe [22].

There have been multiple approaches to the problem each year, so the aim of this project was to develop our own. The best participants usually focus on **CNN models** using **spectrograms** as input data, generating multiple spectrogram segments per audio sample.

Our approach was to encode a sample as a single input, placing spectrogram segments in each input channel. The core idea was to analyze how effective it is to treat each sample as a whole.

The models used for this task were a variation ***MobileNetV2*** [18] and ***ResNet18*** [4], both strong and well-established architectures in image classification tasks. We also analyzed how a *ResNet18* model that was pre-trained on the *ImageNet* dataset [1] performed on this task to evaluate the benefits of transfer learning in this context.

By comparing these models, our goal was to measure how different input representations and pretrained weights impact **classification accuracy**. This exploration not only aims to optimize bird call recognition performance but also contributes to the broader understanding of how input encoding strategies can influence audio classification tasks.

2 State of the art

Automatic animal sound recognition has received growing attention in recent years, motivated by its applications in biodiversity monitoring, conservation, and ecological research. The *BirdCLEF* competition, part of the *LifeCLEF* challenge, has become a benchmark for evaluating state-of-the-art methods in this field. Most of the performing approaches in recent editions share a common strategy: converting raw audio into spectrograms and applying **convolutional neural networks (CNNs)** to perform the species classification [8].

2.1 Spectrogram-based CNN approaches

Spectrograms, particularly **log-mel** and **mel-spectrograms**, are the most widely used audio representations due to their ability to capture time-frequency features in a visual format.

The mel-spectrogram transforms an audio signal into a **time-frequency representation** that aligns more closely with **human auditory perception**. Unlike linear spectrograms, it applies a **mel scale filter bank**, which spaces frequency bands non-linearly (denser at lower frequencies and sparser at higher ones) to reflect how humans perceive pitch [19]. This design allows models to prioritize information in ranges that are most relevant to the human auditory system.

High-performing models such as *ResNet* [4], *DenseNet* [6], and *EfficientNet* [20] have been adapted for bird call classification, often leveraging pretrained weights from the ImageNet dataset to benefit from transfer learning.

A common technique among top *BirdCLEF* participants involves slicing long audio recordings into overlapping segments (for example, 5-second chunks) and computing spectrograms for each. These are then used as independent inputs to the CNN. Although this strategy is effective, it may overlook temporal continuity and interdependencies between segments. This is the case for the top performing model of the 2024 edition of the contest [9].

2.2 Data augmentation and meta-information

The best performing models from previous years usually use audio-specific augmentations such as ***SpecAugment*** [10], **mixup**, **background noise injection**, and **time/frequency masking** to improve results. In addition, metadata-based filtering, which uses geographical and temporal information (GPS coordinates, time of year), greatly boosts accuracy by reducing the set of candidate species [8].

2.3 Relevance to our project

Our project revolves around variation to the standard spectrogram segmentation strategy. Rather than processing each audio segment independently, we encode **multiple spectrogram segments** into the input channels of a **single image-like tensor**, preserving more global context within a single forward pass. This strategy draws inspiration from **multi-channel image processing** but has not been widely explored in audio classification literature, which is a gap our project aims to investigate.

We compare *MobileNetV2*, which is a lightweight yet effective CNN, and *ResNet18*, with and without ImageNet pretraining. These models provide a strong baseline for understanding how channel-wise encoding and transfer learning influence classification performance.

3 Methodology

With a clear understanding of our goal and the current state of the art, we move on to explaining the methods we have used to develop our project.

3.1 Data analysis

The dataset we used throughout the project is an alternative version of the official one for the 2021 edition of BirdCLEF [3]. This dataset contains every single sound from the original dataset already converted into a mel spectrogram. This was crucial for us, since our computational capacity was limited and converting all files ourselves would have been very time-consuming.

The original dataset includes three main folders, being

- `train_short_audio`, which contains **short audio files** with **labeled** birds in them.
- `train_soundscapes`, which contains **longer audio files** that may or may not include **unlabeled** bird calls along with many other sounds from the soundscape.
- `test_soundscapes`, which is the unpublished set of examples to be used for **testing**. Only contestants are allowed to access it.

We chose to focus on the `train_short_audio` folder. The main reason for this choice was that the samples in this folder were relatively short and clearly labeled. In contrast, the other two folders were designed for the main competition, which involved identifying birds in longer audio recordings. Given our time and computational constraints, we decided not to follow the full competition setup. Including large audio files with noisy soundscapes or uncertain labels would have made the task much more difficult and inconsistent.

Regarding the selected audio samples, one of the first challenges was the total number of available classes, which included **397 bird species**. Under our limited conditions, using all of them was not feasible. The classes within `train_short_audio` were also **highly imbalanced**, since some had as many as 500 samples, while others had only 8 samples.

To address this, we used a *PowerShell* script on *Windows* to list and sort the classes by the number of samples they contained. We then selected the **six classes** with the most samples. This approach helped us mitigate both computational limitations and dataset imbalance.

To standardize the input for our pipeline, we implemented a function to **trim** each audio sample to the same number of mel-spectrogram frames. The process is simple: it first calculates the average number of mel-spectrogram frames per file across all folders, then computes a **global average** across classes. The trimming is done from the **beginning and end** of the spectrograms, since based on our observation the bird vocalizations tend to occur in the middle of the recordings. We initially intended to perform a deeper statistical analysis to determine at what point in time birds were more likely to sing, in order to improve the trimming strategy. However, time constraints prevented us from exploring this further.

Eventually, we slightly reduced the target global average frame count to allow more samples to be included in the trimming process and in the final dataset. The original threshold resulted in too few usable samples, so this adjustment helped us balance quantity and quality.

After working on the audio length, we noticed an issue with the **labels**. In the *CSV* file provided with the audio samples, there were two columns related to labeling: one indicated the most probable bird species (`primary_label`), while the other listed less probable candidates (`secondary_labels`). To avoid introducing confusion during training, we decided to exclude any audio samples where one of the target birds appeared in the secondary labels column. This helped ensure that our model only trained on clean, unambiguous examples. Finally, we decided to convert the labels to numerical values to help us choose an appropriate loss function.

3.2 Models and optimization

The first architecture used in this project is based on a ***MobileNetV2***, specifically designed for classification tasks with limited computational resources. *MobileNetV2* is a lightweight architecture optimized for mobile devices, introducing two key concepts:

- **Inverted residual blocks**, which are a type of residual connection where the input is first expanded to a **higher-dimensional space**, then processed, and finally projected back to a **lower-dimensional space**. This is the opposite of traditional residual blocks, which typically reduce the number of channels and then increase them again. The inverted structure allows efficient feature transformation while keeping the number of parameters low [18].
- **Depthwise separable convolutions**, which break down a standard convolution into two simpler operations: a **depthwise convolution**, which applies a single filter per input channel (processing each channel separately), followed by a **pointwise convolution** (1×1) that mixes the channels. This significantly reduces computation and parameter count compared to regular convolutions [5].

The model begins with a **standard convolutional layer**, followed by several stages composed of inverted residual blocks. Each block first expands the number of channels using a 1×1 convolution, applies a depthwise convolution to process spatial information, and finally projects the channels back to a lower-dimensional space. This design allows the model to remain highly efficient in terms of both parameters and operations without sacrificing too much performance. **Residual connections** are used whenever the input and output tensor shapes match, helping to preserve gradient flow during training.

Regarding the intermediate layers, we used a configuration inspired by the default *PyTorch* implementation of *MobileNetV2*, but significantly reduced the number and depth of layers to fit our lightweight design goals. The original *PyTorch MobileNetV2* includes 17 inverted residual blocks, whereas our model only includes 5, with the following structure:

```
settings = [
    [1, 16, 1, 1],
    [6, 24, 2, 2],
    [6, 32, 3, 2],
    [6, 64, 2, 2],
    [6, 64, 1, 1]
]
```

Each element in the list follows the format `[t, c, n, s]`, where:

- `t` is the **expansion factor**, indicating how much we multiply the number of channels before the convolution.

- `c` is the **number of output channels**, which determines the target number of channels after the depthwise convolution and pointwise projection. This controls the capacity and depth of the stage.
- `n` is the **number of times the block is repeated**.
- `s` is the **stride** of the first block in the stage.

This structure keeps the core design principles of *MobileNetV2* while reducing the model size and computational cost.

The final classifier consists of a **global average pooling layer** (`AdaptiveAvgPool2d`), a **flattening layer** (`Flatten`), and a **fully connected layer** (`Linear`) that produces predictions for each class.

In our case, the model was adapted to handle **10-channel inputs** (mel-spectrograms) and outputs predictions across **6 classes**, corresponding to selected bird species from the *BirdCLEF* dataset. The total number of model parameters is approximately 200,000.

The selected loss function was `CrossEntropyLoss`, which is the standard for multiclass classification tasks. This function computes the cross-entropy between the model's output distribution (with an implicit softmax) and the true class encoded as an integer, based on the labels we have selected [14].

To determine the best optimization method for these models, we performed a test using ***Adam***, ***Adam with weight decay***, **RMSprop with momentum**, ***Adagrad***, and ***SGD with momentum*** to see which performed the best, with different hyperparameters. Our prediction is that either Adam or Adam with weight decay would yield the best results because of their robustness and efficiency in training deep neural networks.

Regarding the *ResNet18* architecture, it is a CNN comprising 18 layers. Its architecture is organized into an initial **convolutional stem** followed by four stages of **residual blocks**. Each residual block contains two 3×3 **convolutional layers** with **Batch Normalization** and ***ReLU*** activations, alongside an identity or projection shortcut that enables residual learning. The four stages progressively increase the number of filters (64, 128, 256, 512) and perform **downsampling** via **strided convolutions** at the beginning of each stage except for the first one.

The network begins with a 7×7 convolution (stride 2), followed by a 3×3 max pooling layer. After the residual stages, a global average pooling layer reduces spatial dimensions, and a fully connected linear layer outputs class probabilities.

In our implementation, we replaced the original 1000-class output with a 6-unit linear layer. This architecture enables efficient deep feature extraction while reducing vanishing gradients through the use of skip connections. The modified first convolutional layer and final classifier layer allow the network to effectively adapt to our 10-channel spectrogram input and 6-class output, respectively, while retaining the benefits of pretrained weights and residual learning.

We implemented our *ResNet18* it using *torchvision* [11], and pretrained it using the *ImageNet* dataset [1]. Since the original architecture is designed for 3-channel RGB images, we adapted the first convolutional layer to accept a 10-channel input, corresponding to our 10-spectrogram tensor. To initialize the new weights, we computed the mean of the original 3 input-channel filters and replicated this average across all 10 channels. This approach preserved the overall weight distribution and allowed for a smoother transition from pre training. We also experimented with directly copying the original 3 filters into the first 3 channels and random initialization for the remaining 7, but the averaged initialization resulted in more stable training and better convergence. This modification ensured compatibility with our custom input while maintaining the benefits of transfer learning. We modified the final linear layer to our 6 final classes.

3.3 Experiments

The primary goal of our experiment was to optimize a compact CNN based on the *MobileNetV2* architecture, in order to compare its performance with a well-known deep neural network, *ResNet18*. We aim to evaluate whether our audio encoding method preserves sufficient relevant information for effective neural network training, or if critical details are lost due to the structure of the proposed data representation.

For our *MobileNetV2*, we hypothesized that by carefully **tuning training strategies**, especially the choice of optimizer and learning rate, we could push the performance of our model. Our focus was on optimizer testing, as the optimizer plays a central role in CNNs.

To test this hypothesis, we conducted a series of controlled experiments using the exact same architecture and training data, varying only the optimizer and learning rate. The evaluated optimizers were:

- ***Adam***, known for its fast convergence and adaptive learning rate [15].
- ***Adam with weight decay***, a variant of Adam with decoupled weight decay, which improves generalization [13].
- ***SGD (with momentum)***, a strong baseline that is widely used in vision tasks but can require more careful tuning [17].
- ***RMSprop***, which is designed for non-stationary objectives and often used in recurrent tasks, tested here for completeness [16].
- ***Adagrad***, which is suitable for sparse data, included to compare its performance with more modern choices [12].

Each optimizer was evaluated with multiple learning rates (0.01, 0.0001), and performance was tracked via validation accuracy and loss curves. We deemed this metric to be adequate since we manually managed the amount of samples for each category, thus the dataset was balanced.

For all experiments we used the same number of epochs (20) and loss function (`CrossEntropyLoss`).

For the *ResNet18* model, we evaluated the impact of pretraining by comparing two versions of the same architecture, with and without pretrained weights. Since the model architecture was previously described, the focus here is on analyzing how initialization affects performance on our dataset.

Both models were trained under identical conditions (same learning rate, optimizer (*Adam*), batch size, and number of epochs (20)) to ensure a fair comparison. Evaluation was conducted using validation accuracy and validation loss. The experiment tests the hypothesis that transfer learning with pretrained weights will improve performance on our data. We also aimed to demonstrate how a well-established image recognition model like *ResNet18* performs on our dataset in order to better understand the performance observed with *MobileNetV2*. This comparison helps determine whether the possible issues with *MobileNetV2* were due to poor design decisions or limitations in the dataset itself. By using *ResNet18*, known for its robust performance in various vision tasks, we can isolate the effect of model architecture from other factors and gain clearer understanding of the strengths and weaknesses of our overall experimental setup.

For hyper parameter tuning we used a small learning rate 0.001, as the training was unstable. We used a batch size of 16, as we didn't have much data. We decided on cross entropy as loss functions, as it is a good loss function for classification of our distribution. We also used *Adam with weight decay* optimizer as we had a problem of overfitting and we wanted regularization. The number of epochs was 20.

4 Results

For our *MobileNetV2* model, after conducting the experiments for all the previously mentioned optimizers we obtained the results in Table 1.

Optimizer (learning rate)	**Validation Accuracy**	**Validation Loss**
Adam (0.01)	0.4375	1.5383
Adam (0.0001)	0.3295	1.6970
AdamW (0.01)	0.4830	1.5028
AdamW (0.0001)	0.3239	1.6340
RMSprop (0.01)	0.1818	1.8121
RMSprop (0.0001)	0.5227	1.4128
Adagrad (0.01)	0.4943	1.3037
Adagrad (0.0001)	0.2159	1.7905
SGD (0.01)	0.4773	1.5765
SGD (0.0001)	0.2216	1.7771

Table 1: Validation accuracy and loss for different optimizers and learning rates.

A better visual representation of the evolution of validation accuracy and loss for each optimizer is provided in Figure 1.

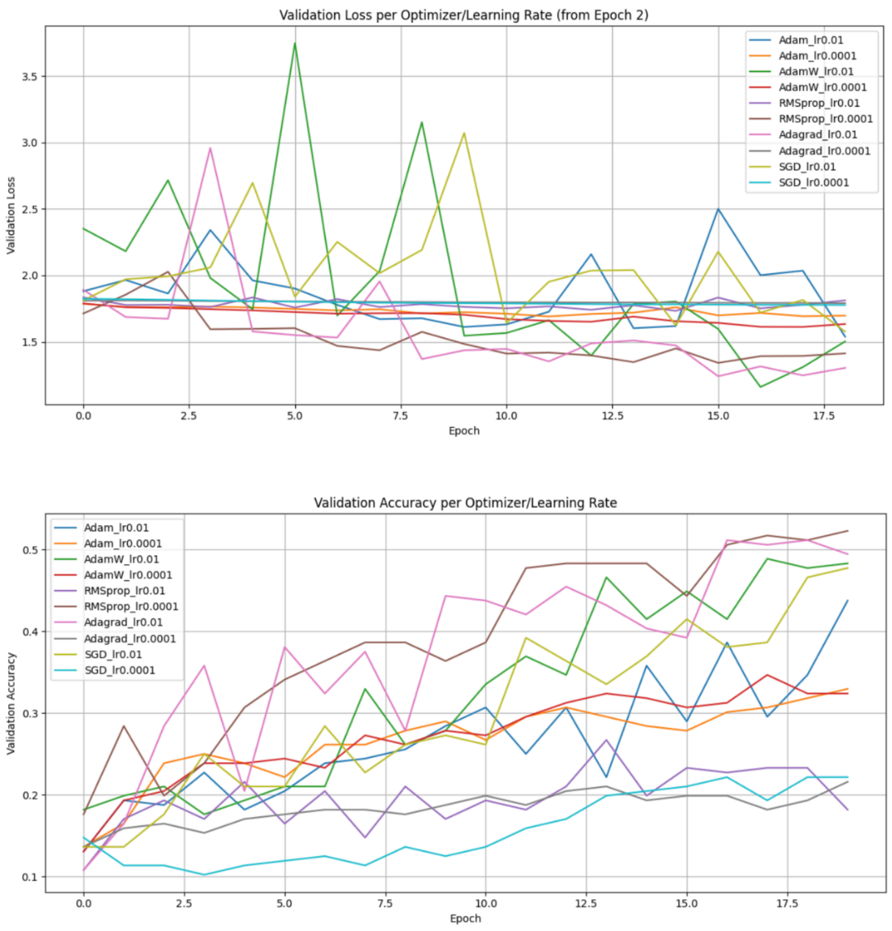

Figure 1: Evolution of validation accuracy and loss for each optimizer.

Both ***Adam*** with weight decay (learning rate 0.01 and weight decay 0.0001) and ***RMSprop*** (learning rate 0.0001) achieved approximately 0.5 validation accuracy. *RMSprop* performed particularly well, thanks to its adaptive learning rate mechanism, which helps stabilize training in the presence of noisy or sparse gradients [21]. The comparison between these two better-performing optimizers can be visualized in Figure 2.

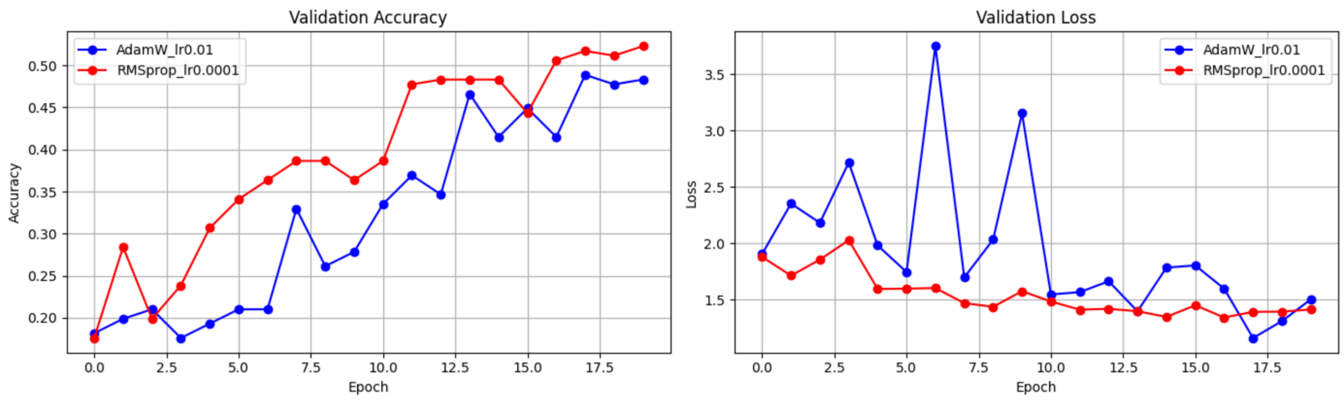

Figure 2: Comparison of validation metrics between *Adam with weights* and *RMSprop*.

For the *ResNet18* experiment, the final results can be seen in Table 2.

Model	**Validation Accuracy**	**Validation Loss**
Untrained ResNet18	0.51	2.5
Pretrained ResNet18	0.79	0.9

Table 2: Final validation metrics for pretrained vs. untrained ResNet18.

In Figure 3 we can see the evolution of the validation metrics across *ResNet18* models.

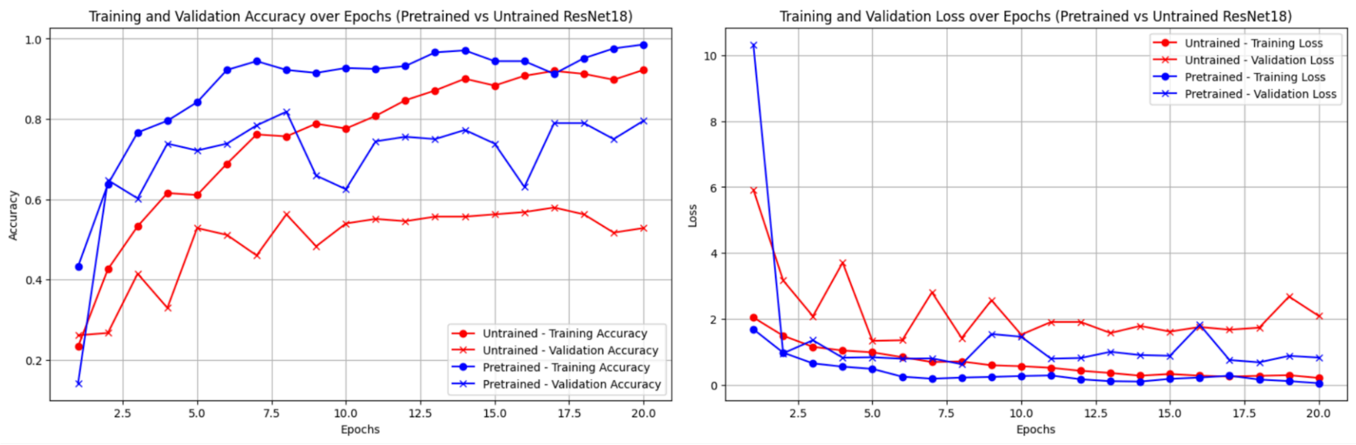

Figure 3: Final validation evolution for both the pretrained and untrained versions of *ResNet18*.

The final comparison between the results of both models gave us the values in Table 3.

Model	**Validation Accuracy**	**Validation Loss**
ResNet18 Untrained	52.84%	1.9803
ResNet18 Pretrained	77.84%	0.7783
MobileNetV2	51.70%	1.5313

Table 3: Final validation accuracy and loss for different CNN architectures.

Figure 4 is a good way to visualize the comparison between the evolution of the validation of both our *MobileNet* and the pretrained and untrained *ResNet18*.

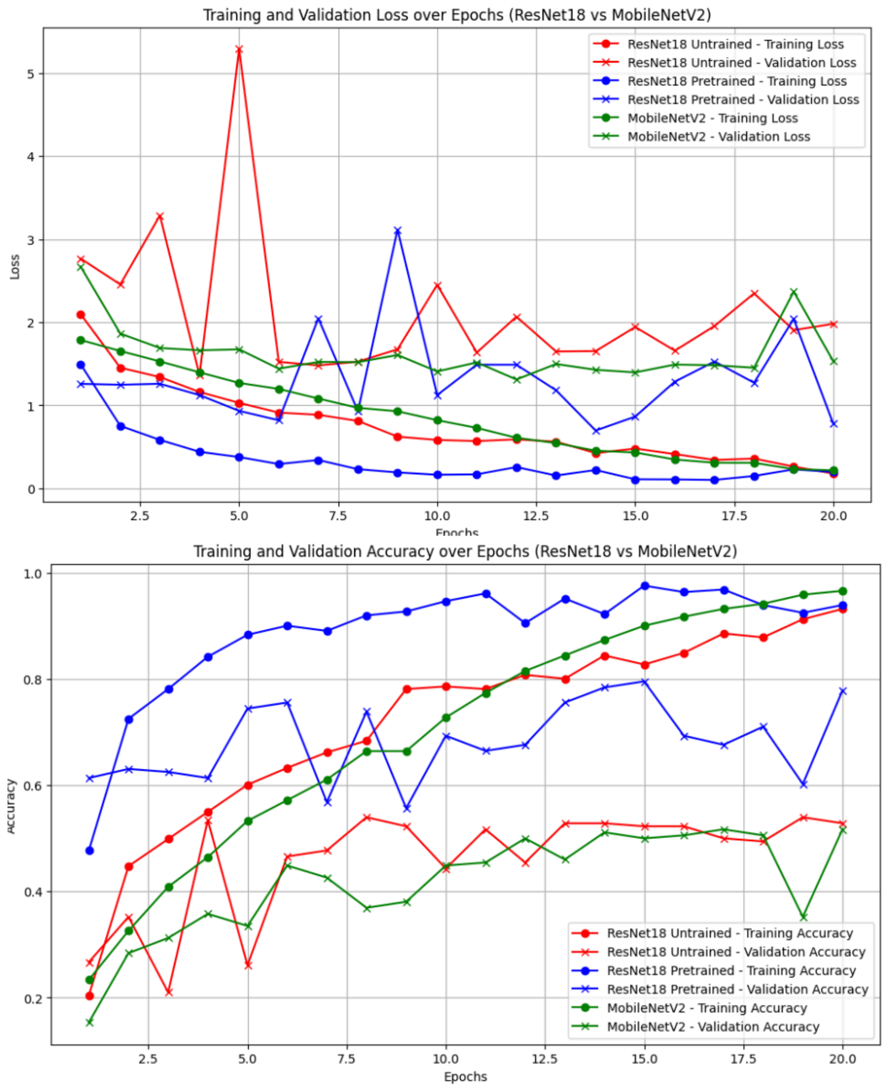

Figure 4: Final validation evolution for the *MobileNetV2* and both the pretrained and untrained versions of *ResNet18*.

5 Discussion and future work

Our experimental results provided valuable insights into the effectiveness of different model architectures and training strategies for bird audio classification using full-length spectrogram inputs.

MobileNetV2 did not perform as well as expected. While it achieved acceptable results, its performance was limited. When compared to *ResNet18* trained from scratch, both models yielded similar outcomes, suggesting that limited model capacity and insufficient data were detrimental to effective learning. Specifically, the **untrained** *ResNet18* struggled to converge meaningfully, likely due to the high variability in bird vocalizations and the relatively small dataset. In contrast, the **pretrained** *ResNet18* exhibited substantially better performance and training stability, reinforcing the value of transfer learning, even when the source domain (natural images in ImageNet) is significantly different from the target domain (audio spectrograms).

These results support our hypothesis that using entire spectrograms provides richer contextual information than segment-based approaches. However, they also highlight a deeper issue: the simplified version of the problem we attempted using a small subset of *BirdCLEF* data with single-label classification may not fully exploit the strengths of modern architectures. In fact, our results suggest that the real limitation may not be the models themselves, but the constrained problem formulation. Given that pretrained models can effectively transfer knowledge to spectrograms, a more promising direction may be to tackle the full *BirdCLEF* task directly, treating it as a multi-label classification problem with a deeper architecture (for example, *ResNet50*, *EfficientNet*, or transformer-based models). We also identify opportunities for improvement in our current pipeline, including the application of data augmentation strategies and pretraining on audio-specific datasets such as *AudioSet* [2].

Ultimately, we believe that moving toward solving the complete *BirdCLEF* challenge rather than simplified versions will provide a more realistic evaluation of model performance and unlock the full potential of deep learning for large-scale, real-world bird audio recognition.

We have to consider though, that facing the complete *BirCLEF* challenge would make our dataset unbalanced, rendering accuracy almost meaningless. Considering our main evaluation method consisted in looking at the accuracy, this change would greatly affect our project. We would probably need to use another metric such as the F1 score, which is the harmonic mean of precision ($\frac{\text{TP}}{\text{TP+FP}}$) and recall ($\frac{\text{TP}}{\text{TP+FN}}$), providing a balanced measure of a model's accuracy, especially useful for imbalanced datasets.

6 Conclusion

Our results support the idea that using full-length spectrograms as input can capture richer context for bird audio classification than traditional segment-based methods. This approach shows promise but requires larger datasets and more expressive models to reach its full potential. As a next step, we plan to scale up to the full BirdCLEF task and explore architectures better suited for complex, multi-label audio data.

Here are the main takeaways from our project:

- Treating full-length spectrograms as single inputs provides richer temporal context compared to traditional segment-based approaches.
- *MobileNetV2*, while efficient, struggled to capture sufficient discriminative features in this task.
- *ResNet18* pretrained on *ImageNet* significantly outperformed models trained from scratch, confirming the value of transfer learning even for spectrogram-based audio data.
- Model performance was likely constrained by the limited size and simplified nature of the dataset (6 classes, single-label classification).
- Results suggest that model architecture was not the main bottleneck—data scale and problem formulation were more limiting.
- Future work should target the full *BirdCLEF* multi-label task, use larger architectures (*ResNet50*, *EfficientNet*), and explore audio-specific pretraining (*AudioSet*).

References

[1] Jia Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[2] Jort F. Gemmeke et al. *AudioSet: An ontology and human-labeled dataset for audio events*. Google Research. Available at: https://research.google.com/audioset/. 2017. URL: https://research.google.com/audioset/.

[3] Max van der Hart. *BirdCLEF 2021 Mel Spectograms*. https://www.kaggle.com/datasets/defined/birdclef-2021-mel-spectograms. 2021.

[4] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.

[5] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV]. URL: https://arxiv.org/abs/1704.04861.

[6] Gao Huang et al. "Densely Connected Convolutional Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA: IEEE Computer Society, July 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.

[7] Alexis Joly et al. "BirdCLEF+ 2025: Species Identification in Soundscape Recordings of the Middle Magdalena Valley". In: *CLEF 2025 Working Notes*. 2025. URL: https://www.kaggle.com/competitions/birdclef-2025.

[8] Stefan Kahl et al. "Overview of BirdCLEF 2021: Bird call identification in soundscape recordings". In: *CLEF 2021 Working Notes – Conference and Labs of the Evaluation Forum*. Bucharest, Romania: CEUR Workshop Proceedings, Sept. 2021.

[9] Kirill Chemrov. *Discussion on BirdCLEF 2024: techniques and leaderboard analyses*. https://www.kaggle.com/competitions/birdclef-2024/discussion/512197. 2024.

[10] Daniel S. Park et al. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Proceedings of Interspeech 2019*. ISCA, 2019, pp. 2613–2617. DOI: 10.21437/Interspeech.2019-2680.

[11] PyTorch Core Team. *TorchVision 0.17 Documentation*. https://docs.pytorch.org/vision/0.17/. Accessed June 2025. 2025.

[12] PyTorch developers. *torch.optim.Adagrad*. https://docs.pytorch.org/docs/stable/generated/torch.optim.Adagrad.html. Accessed June 2025. 2025.

[13] PyTorch developers. *torch.optim.AdamW*. https://docs.pytorch.org/docs/main/generated/torch.optim.AdamW.html. Accessed June 2025. 2025.

[14] PyTorch Team. *torch.nn.CrossEntropyLoss — PyTorch 2.0 documentation*. 2024. URL: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html.

[15] PyTorch Team. *torch.optim.Adam — PyTorch 2.0 documentation*. 2024. URL: https://pytorch.org/docs/stable/generated/torch.optim.Adam.html.

[16] PyTorch Team. *torch.optim.RMSprop — PyTorch 2.0 documentation*. 2024. URL: https://pytorch.org/docs/stable/generated/torch.optim.RMSprop.html.

[17] PyTorch Team. *torch.optim.SGD — PyTorch 2.0 documentation*. 2024. URL: https://pytorch.org/docs/stable/generated/torch.optim.SGD.html.

[18] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *arXiv preprint arXiv:1801.04381v4* (2019). DOI: 10.48550/arXiv.1801.04381.

[19] S.S. Stevens, J. Volkmann, and E.B. Newman. "A scale for the measurement of the psychological magnitude pitch". In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190. DOI: 10.1121/1.1915893.

[20] Mingxing Tan and Quoc Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 6105–6114. URL: https://proceedings.mlr.press/v97/tan19a.html.

[21] Tijmen Tieleman and Geoffrey Hinton. *Lecture 6.5—RMSProp: Divide the Gradient by a Running Average of Its Recent Magnitude*. Coursera: Neural Networks for Machine Learning. University of Toronto. 2012. URL: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[22] Xeno-canto Foundation. *Xeno-canto: Sharing Bird Sounds from Around the World*. 2024. URL: https://www.xeno-canto.org.