Exploring Convolutional Neural Networks and Transformers for Bird Call Detection in the BirdCLEF 2024 Challenge

Philipp Unger
University of Passau

Spencer Apeadjei-Duodu
University of Passau

SS 2024

Abstract

This project presents the methodology and findings from participation in the BirdCLEF 2024 Kaggle challenge [1], aimed at detecting bird calls within noisy audio soundscapes. Significant background noise and overlapping frequencies with bird calls in the dataset presented substantial challenges. The primary objectives were efficient data processing and achieving high detection accuracy.

A range of preprocessing techniques, including filtering and spectrogram conversion, were utilized. Various models were evaluated, including Convolutional Neural Networks (CNNs), ResNet18, Inception v3, and the pretrained Audio Spectrogram Transformer (AST) fine-tuned on AudioSet [2, 3]. Among these, the ResNeXt model [4], optimized using Optuna [5], demonstrated superior performance.

Challenges encountered included inefficient data conversions and the underperformance of simpler CNN models. The findings emphasize the critical role of model selection and preprocessing in audio classification tasks. This project offers valuable insights into managing noisy datasets and optimizing models for bioacoustic applications.

1 Introduction

The BirdCLEF 2024 Challenge seeks to enhance the state-of-the-art in automatic identification of under-studied Indian bird species through audio recording analysis. Birds, with their high mobility and diverse habitat needs, serve as indicators of biodiversity restoration success or failure based on changes in species assemblages and population numbers [1].

Audio classification has become a crucial aspect of numerous applications, from environmental monitoring to voice recognition systems. This field poses significant challenges due to the complex and variable nature of audio signals, which often contain substantial background noise and overlapping frequencies. Addressing these challenges requires sophisticated data processing techniques and robust classification models to achieve high accuracy.

Both authors contributed equally to this research.

The text in this paper has been improved using ChatGPT.

In this paper, the approach to addressing the BirdCLEF 2024 challenge, which focuses on detecting bird calls in noisy audio environments, is detailed. The methodology involves converting audio signals into spectrograms to leverage image classification models. Key steps in audio processing included noise reduction and high-pass filtering, with a multithreaded approach used for efficient data handling.

Spectrograms were generated from 5-second audio snippets for model training. Initial trials with simple CNNs yielded suboptimal accuracy. Improved results were achieved with pre-trained models like ResNet18, Inception v3, and ResNeXt. The Audio Spectrogram Transformer (AST) was also tested, though not successfully implemented due to technical issues.

The ResNeXt model, optimized with Optuna, showed significant accuracy, with performance measured using the ROC curve, a critical metric for the challenge. This study underscores the importance of preprocessing in audio classification and demonstrates the effectiveness of treating audio problems as image recognition tasks, contributing valuable insights to the field of bioacoustics.

2 Background

2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep learning models particularly well-suited for processing data with grid-like topologies, such as images and audio spectrograms. They automatically learn spatial hierarchies of features through the use of convolutional layers that apply filters to detect local patterns, followed by pooling layers that reduce dimensionality while retaining essential information. A typical CNN architecture includes convolutional layers, activation functions like ReLU, pooling layers, and fully connected layers. These networks excel at learning feature representations at various levels of abstraction, from simple edges in early layers to complex shapes in deeper layers [6]. In audio classification, CNNs process spectrograms - visual representations of the audio's frequency spectrum over time - utilizing their powerful pattern recognition capabilities to identify and classify audio events [7]. CNNs have proven effective in diverse audio tasks, such as speech recognition, music genre classification, and - as used in this paper - environmental sound detection.

2.2 PyTorch

PyTorch, an open-source deep learning framework developed by Facebook's AI Research lab (FAIR), is highly popular in research and industry for its dynamic computational graph, ease of use, and flexibility. It allows developers to build and train neural networks using a Pythonic interface, suitable for both beginners and experts. A key feature of PyTorch is the ability to dynamically define and modify network architectures, aiding research and experimentation. It offers robust GPU acceleration, enabling efficient training of large-scale models. With a comprehensive library of pre-built neural network layers, loss functions, and optimization algorithms, PyTorch supports rapid prototyping and deployment of machine learning models. Additionally, PyTorch integrates seamlessly with Python libraries like NumPy and SciPy, further enhancing its versatility and making it a powerful tool for deep learning development and experimentation.[8].

2.3 Evaluation Metrics

Evaluation metrics are crucial for assessing machine learning models' performance, guiding model selection, and understanding their effectiveness. Accuracy measures the proportion of correct predictions among all predictions. For binary classification, the Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) are particularly valuable. The ROC curve plots the true positive rate against the false positive rate across different thresholds, providing a detailed performance overview. The AUC summarizes this performance into a single value, with values closer to 1 indicating better discrimination between positive and negative classes [9].

2.4 Audio Filters

2.4.1 Noisereduce

NoiseReduce is a Python library designed to reduce noise in audio recordings, leveraging advanced algorithms to remove unwanted sounds while preserving the primary signal's quality. This is particularly useful in applications like bioacoustic research, where background noise can hinder the accuracy of bird call detection models. By improving the signal-to-noise ratio, NoiseReduce enhances audio clarity, facilitating more accurate analysis and classification. The libr0ary supports various noise reduction methods, such as spectral gating and Wiener filtering, and offers an easy-to-use API, making it a valuable tool for researchers and developers working with noisy audio datasets [10].

2.4.2 Butterworth

Highpass filtering with a Butterworth filter is favored in signal processing due to its maximally flat frequency response, which avoids ripples that can distort the signal. This makes it effective for removing low-frequency noise while preserving high-frequency components, such as bird calls in bioacoustic studies [11]. The Butterworth filter's relatively linear phase response maintains the waveform's integrity, crucial for applications like real-time audio processing and temporal pattern analysis [12]. The implementation of Butterworth filters is conveniently available in the SciPy package, providing an accessible and efficient tool for signal processing tasks [13].

3 Bird Call Classification

3.1 Dataset

The dataset for this study is sourced from the BirdCLEF 2024 Challenge on its Kaggle webpage, with audio recordings generously uploaded by users of Xenocanto.org, a community-driven platform for sharing wildlife sounds. The recordings used in the challenge were primarily made in the Western Ghats biodiverisity hotspot.

The dataset is organized into several directories and files:

- **train audio:** Contains short audio recordings of individual bird calls in ogg format, downsampled to 32 kHz.

- **test soundcapes:** Populated with approximately 1,100 4-minute length audio recordings used for scoring when the notebook is submitted.
- **train metadata.csv:** Provides metadata for the training data, including species labels, geographic coordinates, and file names.
- **sample submission.csv**: A sample submission file with fields for row IDs and bird ID columns, where participants provide the probability of each bird species' presence.

The competition however uses a hidden test set which is made available after the submitted notebook is scored.

3.2 Data Preprocessing

Preprocessing is the most crucial step in preparing our bird call audio data for the model training. It involves various techniques to enhance the quality of the data and convert it into a format suitable for the CNNs to train on.

3.2.1 Initial Preprocessing Attempts

Initially, a variety of preprocessing techniques were explored to enhance the audio data quality. These techniques included:

1. **High-Pass Filtering:** Applied to remove low-frequency noise below a cutoff frequency, typically set at 1000 Hz, using a Butterworth filter [13].
2. **Silence Removal:** Utilized to detect and remove silent sections from the audio using the 'pydub' library, aiming to reduce irrelevant data.
3. **Normalization:** Adjusted the audio signals to a uniform range to minimize the impact of varying recording conditions [14].
4. **Amplitude-Based Filtering:** Filtered out low-amplitude segments to focus on more significant parts of the audio.
5. **Noise Reduction:** Employed spectral gating and adaptive filtering techniques to remove background noise [15].
6. **Compression:** Applied dynamic range compression to manage variations in audio amplitude.

Despite the comprehensive nature of these preprocessing steps, it was found that many were either redundant or had minimal impact on the performance of the machine learning models. Consequently, the preprocessing pipeline was streamlined to improve efficiency and effectiveness.

3.2.2 Optimized Preprocessing Pipeline

The final preprocessing pipeline has been optimized to include only the most impactful techniques. The simplified steps are:

1. **noisereduce:** Utilizing the 'noisereduce' library to effectively reduce background noise, improving the signal-to-noise ratio[10].

2. **Segmentation:** Dividing long audio recordings into fixed-length segments of five seconds each. This ensures that each segment can be processed independently, facilitating parallel processing and reducing computational load [16].

3. **Spectrogram Generation:** Creating Mel-scaled spectrograms [17] from the audio segments, focusing on frequencies between 2000 and 8000 Hz. For this task, the library 'librosa' is utilized [18]. These spectrograms are then converted to grayscale images of a fixed square size (224x224 pixels), enhancing the model's ability to process the data uniformly.

Furthermore, the entire preprocessing pipeline has been parallelized to make use of multiple CPU cores, significantly speeding up the processing time. This parallelization is achieved using Python's 'multiprocessing' library [19], allowing simultaneous processing of multiple audio files.

Figure 1 shows the various stages of preprocessing.

3.3 Classification with CNNs

3.3.1 Baseline CNN Implementation

For the study, a baseline Convolutional Neural Network (CNN) was developed utilizing TensorFlow to classify bird spectrogram images. The purpose of the baseline model was to establish a reference performance metric, allowing the efficacy of more complex architectures in subsequent experiments to be gauged. The preprocessing pipeline involved resizing the spectrogram images to 224x224 pixels and converting them into TensorFlow tensors. The dataset was partitioned into an 80-20 split for training and validation purposes.

The baseline model architecture consisted of three convolutional layers with filter sizes of 32, 64, and 128, respectively, and a kernel size of 3. Each convolutional layer was followed by a Batch Normalization layer and a max-pooling layer with a kernel size of 2 and a stride of 2. The flattened output from the convolutional layers was fed into a dense layer with 128 neurons, followed by a dropout layer with a probability of 0.5 to prevent overfitting. The final layer was a dense layer that produced output corresponding to the number of bird species classes. The architecture of the baseline CNN is illustrated in Figure 2.

The model was originally intended to be trained for 30 epochs with a batch size of 32 using the Adam optimizer and the sparse categorical cross-entropy loss function. To augment the training data, a data augmentation pipeline was implemented, including random horizontal and vertical flips and rotations. Model checkpoints were saved every 500 steps to facilitate recovery and continuation of training. However, the training was stopped after 5 epochs because it took too much time. It was later learned that the data augmentation flipping and rotating did not improve the model and could potentially harm the model, as the bird

calls in the spectrograms are supposed to have their original orientation and unique look. However, it must be said that this is purely speculative, as this was not tested explicitly.

This basic model served as a proof of concept for training a CNN with the spectrogram data provided. The results from this model served as a benchmark for evaluating the performance improvements achieved by more sophisticated models developed in later stages of the project.

3.3.2 Resnet-18

In the study, the Resnet-18 architecture, a well-established deep learning model introduced by He et al. [20], was the first pre-trained model used for the task.

Architecture and Design: The Resnet-18 model consists of 18 layers, incorporating convolutional layers, batch normalization, ReLU activations, and fully connected layers. The model's architecture is built upon residual blocks, each comprising two convolutional layers with shortcut connections that bypass one or more layers, enabling the network to learn residual functions relative to the layer inputs.

Training Process: The model was trained for 25 epochs. During each epoch, checkpoints were saved, and metrics such as accuracy, loss, and ROC were calculated and exported. The code was designed to allow for training to be paused and resumed from any checkpoint. However, a random seed was not set.

Upon pausing at epoch 22, an extension of the training to 50 epochs was planned. After resuming, a significant jump in accuracy was observed, as shown in Figure 3. This was traced back to the random reshuffling of the test and validation splits due to the missing seed. The consistency of the training process was disrupted by this reshuffling.

Impact of Missing Seed: Restarting the training without setting a random seed led to the following issues:

- **Different Data Splits**: Different training and validation sets were used in the resumed run, causing variations in performance metrics.
- **Inconsistent Results**: Significant variations in the model's performance were observed compared to the first run, complicating the determination of whether observed differences were due to actual model changes or random splits.
- **Evaluation Variability**: Non-comparable evaluation results were produced due to different validation sets in each run.

3.3.3 Inception v3

Inception v3 is a deep convolutional neural network architecture that has been widely used for image classification tasks. It was introduced by Szegedy et al. [21]. The architecture is known for its efficiency and effectiveness in achieving high accuracy on image classification benchmarks.

Architecture and Design: Inception v3 is built upon the success of previous Inception architectures (e.g., Inception v1 and v2), incorporating several improvements to enhance both the performance and computational efficiency. The key components of Inception v3 include:

- **Factorization into Smaller Convolutions**: Instead of large convolutional filters, Inception v3 uses smaller convolutions (e.g., 1x1, 3x3) to reduce computational cost while maintaining the representational power.
- **Asymmetric Convolutions:** Inception v3 employs asymmetric convolutions, such as 3x3 followed by 1x1, to further reduce the number of parameters.
- **Auxiliary Classifiers:** Intermediate auxiliary classifiers are used during training to improve gradient flow and prevent the vanishing gradient problem. In our implementation, we do not use it to reduce the computational complexity.
- **Grid Size Reduction:** The architecture incorporates grid size reduction techniques to downsample feature maps efficiently without losing important spatial information.

The architecture of our implementation is illustrated in Figure 4

3.3.4 ResNeXt-50

ResNeXt-50 is a deep convolutional neural network architecture that has been widely used for image classification tasks. It was introduced by Xie et al. in their paper "Aggregated Residual Transformations for Deep Neural Networks" (2017). The architecture is known for its efficiency and effectiveness in achieving high accuracy on image classification benchmarks.

Architecture and Design: ResNeXt-50 builds upon the success of the ResNet architecture by incorporating the concept of "cardinality" (the size of the set of transformations) to improve model performance without significantly increasing computational complexity. The key components of ResNeXt-50 include:

- **Residual Blocks:** Similar to ResNet, ResNeXt-50 uses residual connections to facilitate gradient flow and training of deep networks.
- **Aggregated Transformations:** Instead of using a single transformation, ResNeXt-50 aggregates multiple transformations in parallel, which increases the model's representation capacity.
- **Cardinality:** The number of parallel transformations, which is a new dimension to scale up the model alongside depth and width.

Training Process The training process for the ResNeXt-50 model involved leveraging Optuna for hyperparameter tuning to optimize performance on the roc-auc metric. Initially, spectrogram images generated from the audio data were preprocessed using standard transformations such as resizing, normalization, and conversion to tensors. The dataset was then split into training and validation sets, with 80 percent of the data allocated for training and 20 percent for validation. The final fully connected layer was modified to match the number of classes in the dataset.

Key hyperparameters, specifically batch size and learning rate, were optimized using Optuna. During each trial, the model was trained for 5 epochs using the Adam optimizer, and its performance was evaluated based on the validation set. Metrics such as training and validation losses, validation accuracy, and ROC-AUC scores were recorded for each epoch.

Optuna's pruning feature was used to halt underperforming trials early, thus conserving computational resources. After the optimization process, the best-performing model was identified, and its training and validation metrics were plotted for further analysis. This approach ensured that the most effective hyperparameter configuration was achieved, significantly enhancing the model's performance.

A seed was also set for reproducibility.

3.4 Failed approach with MIT-AST

In exploring various approaches for the challenge, an attempt was made to utilize the Audio Spectrogram Transformer (AST) to directly process the audio data, bypassing the need for spectrogram images that the CNNs rely on. This approach required different preprocessing steps, as the transformer could potentially handle raw audio inputs.

Initially, there were problems with the audio file format, as the transformer could not work on ogg files and 32kHz sampling rate. Consequently, the audio data had to be converted and resampled to 16kHz. For this conversion, SoX [22] and PyDub [23] were utilized.

However, significant implementation difficulties were encountered with the AST model. An initial issue arose with a cryptic error message stating:

```
AssertionError: choose a window size 400 that is [2, 1]
```

After extensive trial and error and consulting with the model's author on GitHub, it was determined that the problem was due to the use of an incorrect feature extractor:

```
feature_extractor = ASTFeatureExtractor.from_pretrained(model_name)
```

Switching to an alternative feature extractor:

```
feature_extractor = Wav2Vec2FeatureExtractor.from_pretrained(model_name)
```

resolved the feature extraction step. Despite this progress, subsequent cryptic errors during model training proved to be significant obstacles within the project's time constraints. Specifically, the new errors lacked sufficient documentation or community support, making troubleshooting exceptionally challenging. The combination of these technical hurdles and the limited time available for the competition led to the decision to discontinue further attempts with the AST model.

The differences in preprocessing between the CNN and the AST approaches are notable. For CNNs, the preprocessing involved converting audio signals into spectrogram images, which capture the frequency spectrum over time and provide a visual representation suitable for image-based models. In contrast, the AST approach aimed to process raw audio data directly, which would have required extracting relevant audio features without the intermediate step of spectrogram conversion.

Given the circumstances, focus was redirected to the ResNeXt model, which demonstrated more promising and reliable performance in the classification tasks. The shift in focus not only ensured the efficient use of available resources but also underscored the importance of adaptability and problem-solving in machine learning research.

4 Results

4.1 Performance Metrics

The performance of the models was evaluated using accuracy and ROC curves on the validation dataset, providing a comprehensive comparison of their classification capabilities. Accuracy, defined as the proportion of correctly classified bird species among all samples, offers a direct measure of model performance. Additionally, ROC (Receiver Operating Characteristic) curves, which plot the true positive rate (sensitivity) against the false positive rate (1-specificity) across various thresholds, allow for a detailed comparison of the models' ability to distinguish between positive and negative classes. Table 1 presents the accuracy scores for each model, while the Figures in the appendix visually compares their ROC curves, highlighting differences in their classification performance. This combined evaluation helps in understanding not only the overall effectiveness but also the robustness of each model.

Model	**Train Loss**	**Validation Loss**	**Validation Accuracy**	**Validation AUC**
Baseline-CNN	1.0518	4.3762	0.4466	0.8830
Resnet-18	0.5508	1.2262	0.7211	0.9447
InceptionV3	0.2055	1.5339	0.7467	0.9839
ResNext-50	0.2433	1.3121	0.7511	0.9880

Table 1: Metrics across all our trained models

4.2 Discussion

The results indicate a clear progression in model performance with increasing complexity and sophistication of the architectures. The ResNeXt-50 model, in particular, demonstrated notable improvements over the baseline CNN and other models tested. This suggests that more advanced architectures can significantly enhance the classification accuracy for bird spectrogram images.

5 Future Work

Building on the findings and challenges of this project, several areas for improvement have been identified for future work.

Early Optimization One critical insight from the study is the importance of using optimization tools, such as Optuna, early in the process. The power of these tools can be utilized to save a considerable amount of time by systematically tuning hyperparameters and improving model performance from the start.

Advanced Validation Techniques With more time, different validation methods such as k-fold cross-validation could be explored. This approach would provide a more robust evaluation of model performance by ensuring that the model is tested on multiple subsets of the data, reducing the risk of overfitting and improving generalizability.

Transformer Models A significant future direction is the implementation of transformer-based models. Unlike CNNs that rely on spectrogram images, transformers can work directly on raw audio files. The performance of transformers in terms of training time and model accuracy could be investigated to provide valuable insights and potentially enhance the detection capabilities. This comparison could reveal whether transformers offer substantial advantages over CNNs for audio classification tasks.

6 Conclusion

This study aimed to explore advanced approaches to bird call detection in the BirdCLEF 2024 Challenge, leveraging Convolutional Neural Networks (CNNs) and Transformers. Through extensive experimentation with various models and preprocessing techniques, the ResNeXt model, optimized using Optuna, was identified as the most effective in handling the noisy audio dataset.

Despite the promising results, several key challenges and lessons emerged from this project:

Kaggle Submission Struggles A significant challenge was encountered in the timely submission of results on Kaggle. Unfortunately, this critical step was postponed until the last week. Due to unexpected complications, the submission process could not be completed. In future projects, this aspect will be prioritized early on to receive timely feedback and adjust approaches accordingly.

Consistency in Strategy The approach was initially broad and exploratory, leading to a wide range of attempted methods and models. While valuable insights were provided, this also resulted in time constraints and a lack of focus towards the end of the competition. Establishing a solid strategy early on and adhering to it could have streamlined efforts and improved overall results.

Community Engagement Another missed opportunity was the engagement with the Kaggle community. Interaction with other participants could have provided different perspectives and innovative approaches that might have been adapted early in the challenge. Future participation will include active community engagement to leverage collective knowledge and enhance solutions.

7 Individual Work

We supported each other throughout the entire project and maintained constant communication. The exchange of ideas and teamwork were excellent, making it a positive experience. There were many discussions in which we consistently found effective solutions for how to proceed. Unfortunately, early in the project, we lost our third team member as he decided to withdraw. However, we adapted to this change and redistributed the workload between the two remaining members.

Philipp Unger My main focus was testing various preprocessing methods. I experimented with multiple approaches and different combinations of filters.

The model part, Spencer and I developed in parallel. My focus was initially on the first Baseline-CNN approach, which was in TensorFlow and was quickly discarded, and later on ResNet-18 and its optimization. Later on, I worked on the Audio Spectrogram Transformer (AST), which unfortunately did not succeed. In the end, I dedicated many of my working hours to the Kaggle submission.

Spencer Apeadjei-Duodu Using the generated spectrogram images from Philipp's preprocessing implementation, I initially experimented with a Baseline CNN model in PyTorch. Following this, I planned to explore the Inception v3 and ResNeXt-50 models. For the ResNeXt-50 model, I set up hyperparameter tuning trials using Optuna to optimize its performance. However, despite my efforts to help with submission, I encountered challenges with the inference stage and was unable to successfully submit to Kaggle.

References

[1] Kaggle. Birdclef 2024 competition, 2024. Accessed: 2024-07-12.

[2] MIT. ast-finetuned-audioset-10-10-0.4593, 2024. Accessed: 2024-07-12.

[3] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer, 2021.

[4] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017.

[5] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[7] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification, 2017.

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[9] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. ROC Analysis in Pattern Recognition.

[10] Tim Sainburg. Noisereduce: A python package for noise reduction, 2024.

[11] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-Time Signal Processing*. Prentice-hall Englewood Cliffs, second edition, 1999.

[12] Steven W. Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Publishing, USA, 1997.

[13] SciPy. scipy.signal.butter, 2024.

[14] Matthias Mauch and Sebastian Ewert. The audio degradation toolbox and its application to robustness evaluation. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, pages 83–88, 2013.

[15] Steven F Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(2):113–120, 1979.

[16] Daniel PW Ellis. Classifying music audio with timbral and chroma features. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pages 339–340, 2003.

[17] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, 2000.

[18] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. *Librosa: Audio and Music Signal Processing in Python*, 2024. Accessed: 2024-07-26.

[19] Python Software Foundation. *multiprocessing — Process-based parallelism*, 2024. Accessed: 2024-07-26.

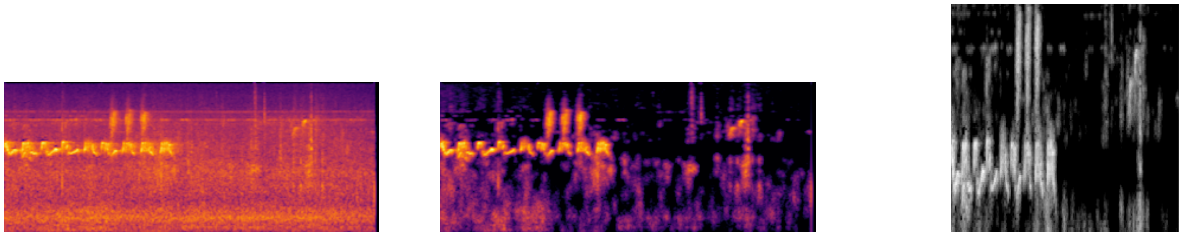
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2015.

[21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2818–2826, 2015.

[22] SoX Developers. Sox - sound exchange. `http://sox.sourceforge.net/`, 2021.

[23] PyDub Developers. Pydub. `https://github.com/jiaaro/pydub`, 2021.

A Appendix



(a) Without filters (b) Applied noicereduce (c) Squared and greyscale

Figure 1: The progressive stages of preprocessing the spectrogram images.

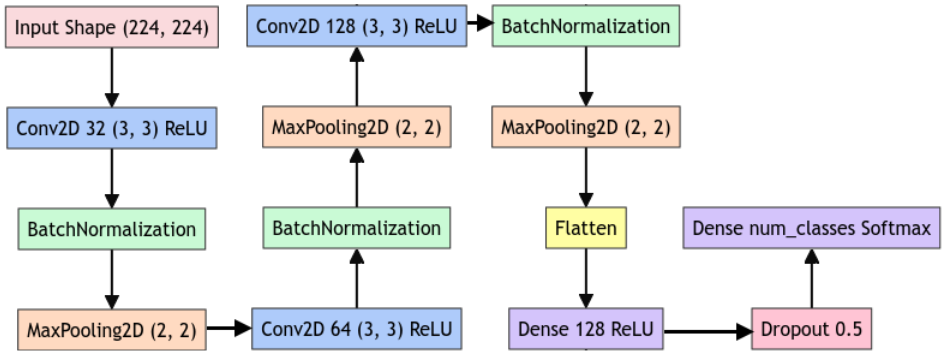


Figure 2: Baseline CNN Architecture for Bird Spectrogram Classification

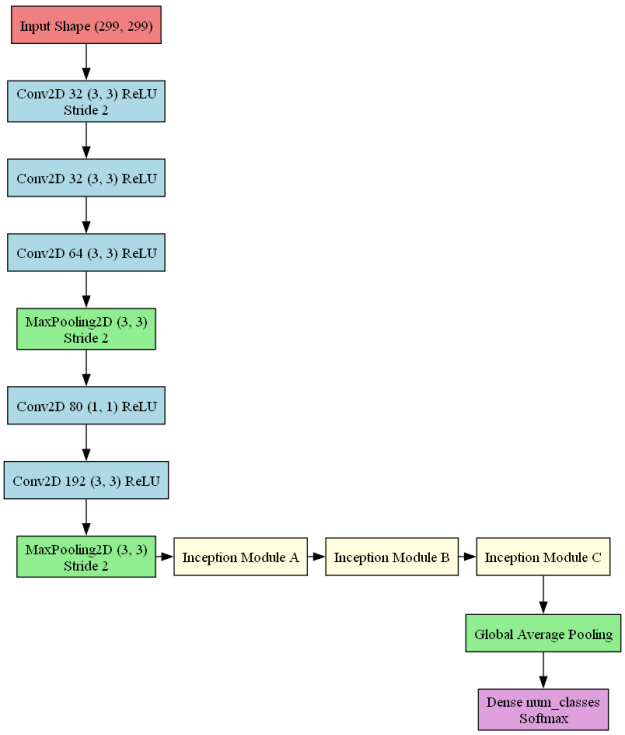


Figure 4: Inception-v3 Architecture

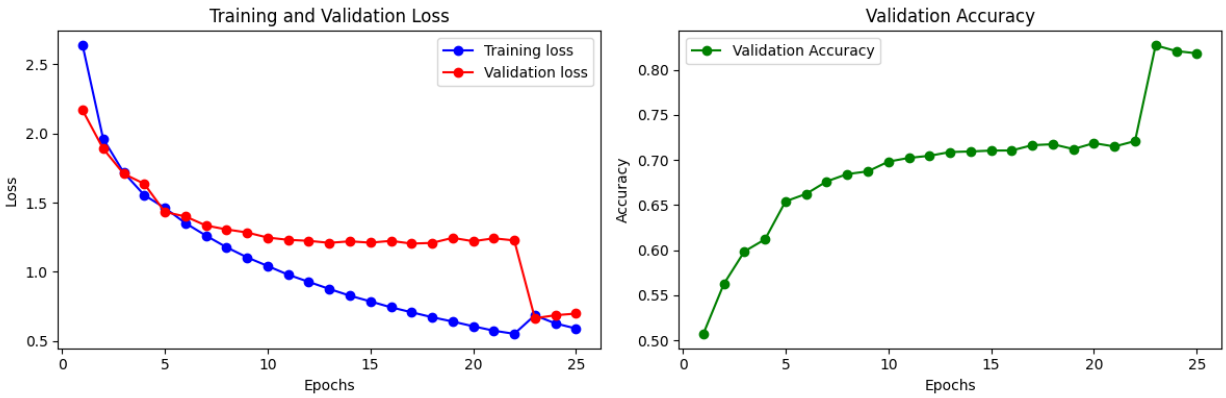


Figure 3: Accuracy and Loss Change After Restart

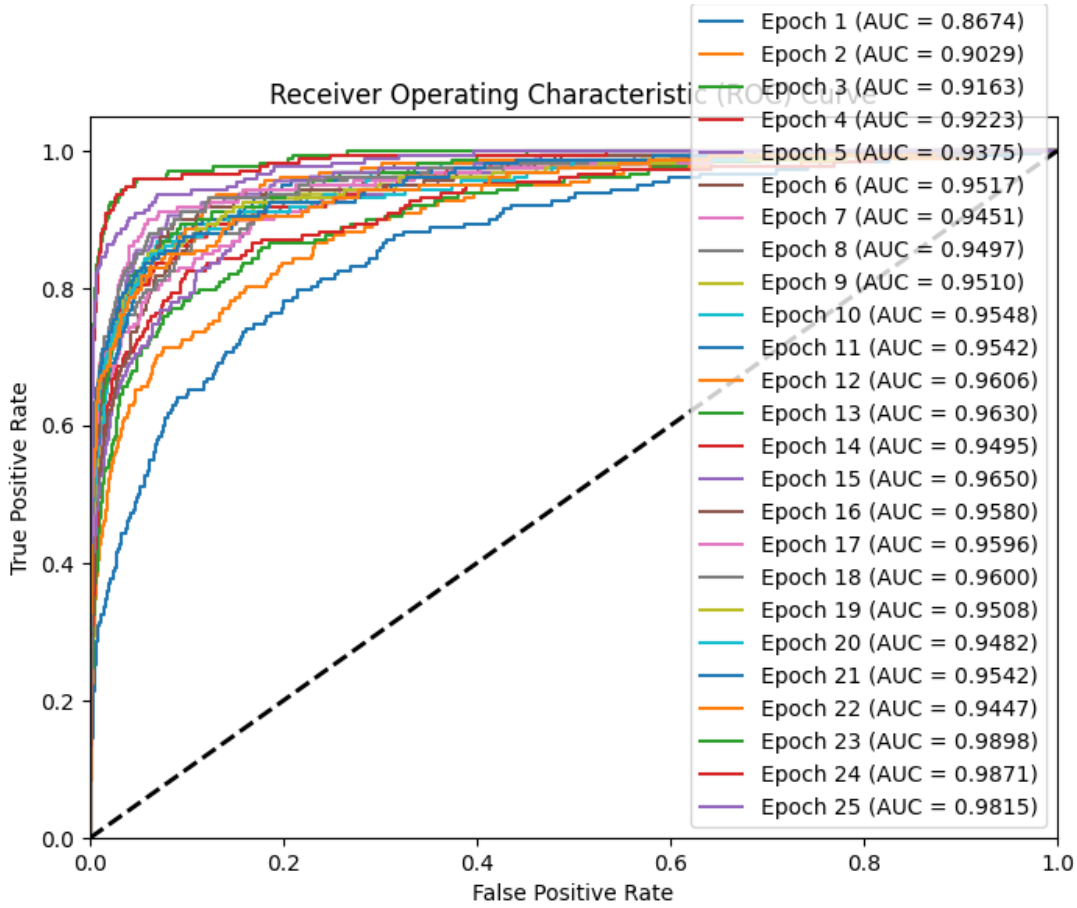
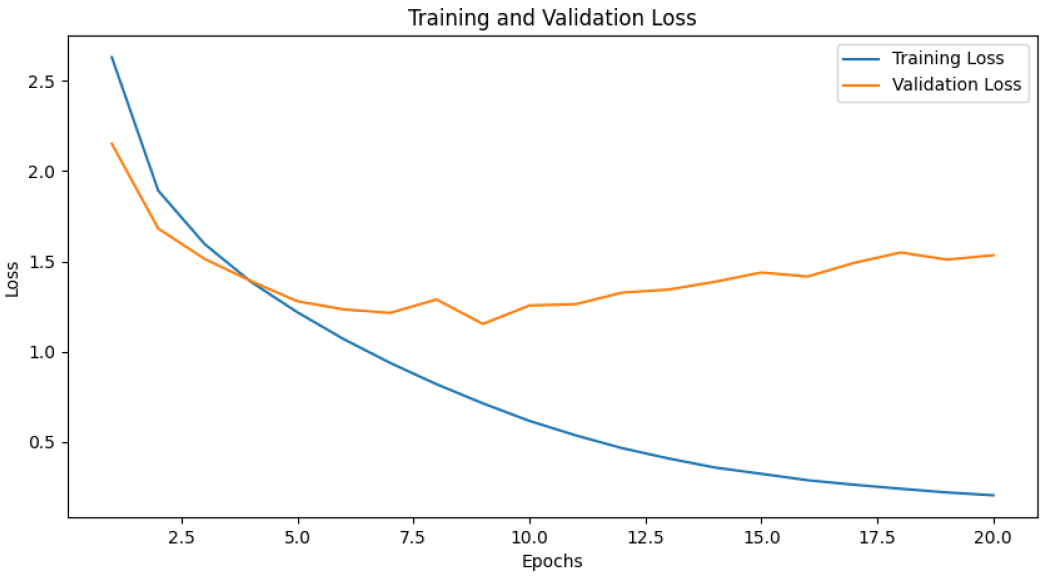
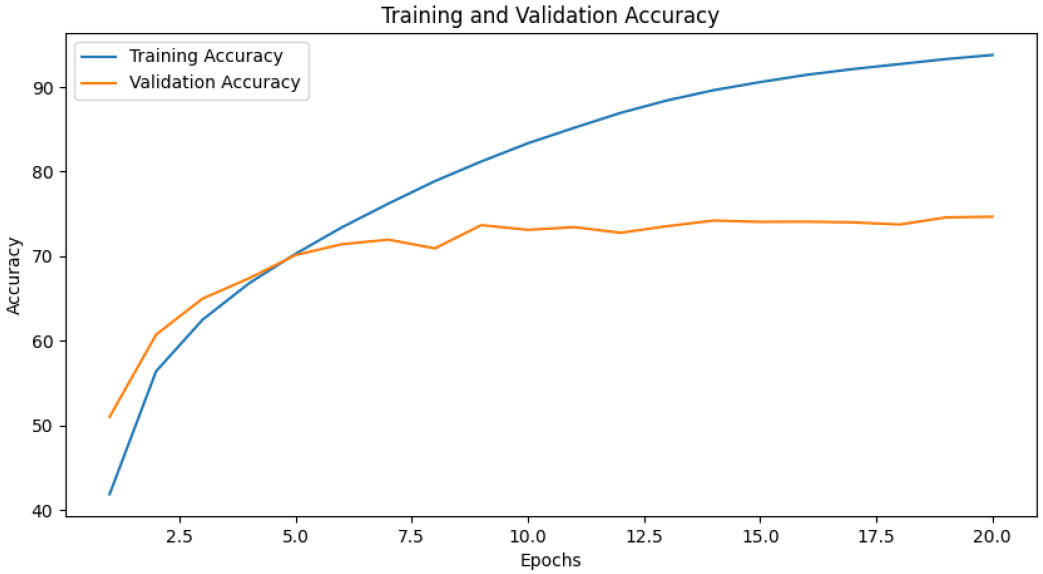


Figure 5: ROC Curve for ResNet-18 with 25 epochs of training



(a) Inception v3 Loss Plot



(b) Inception v3 Accuracy Plot

Figure 6: Inception Performance Metrics with 20 epochs

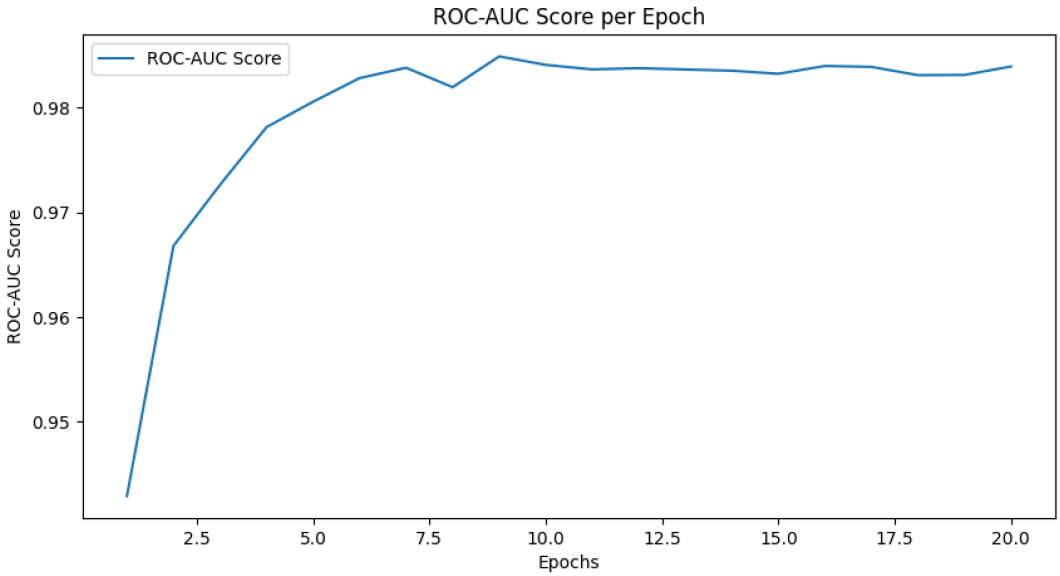
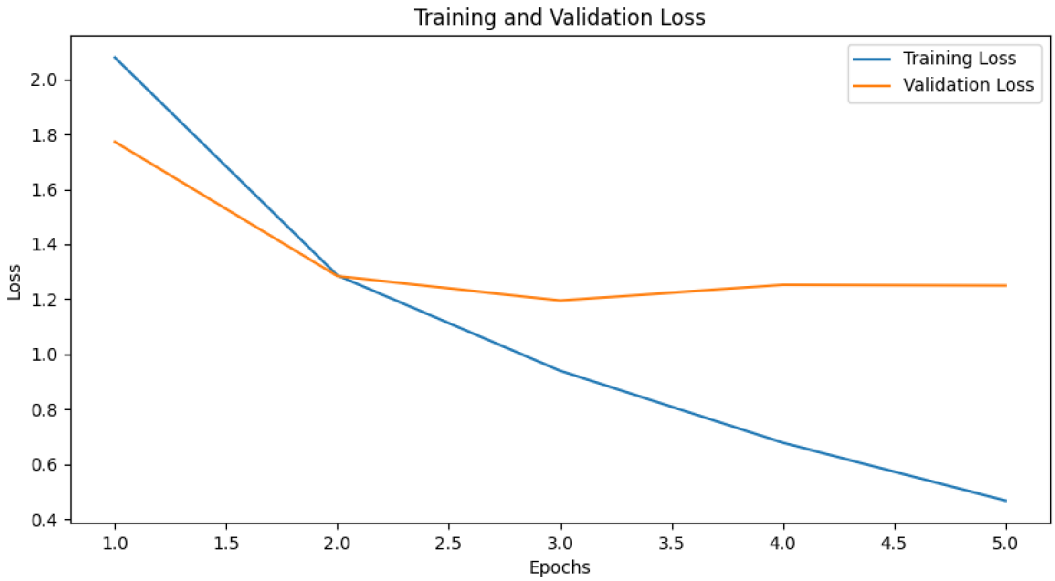
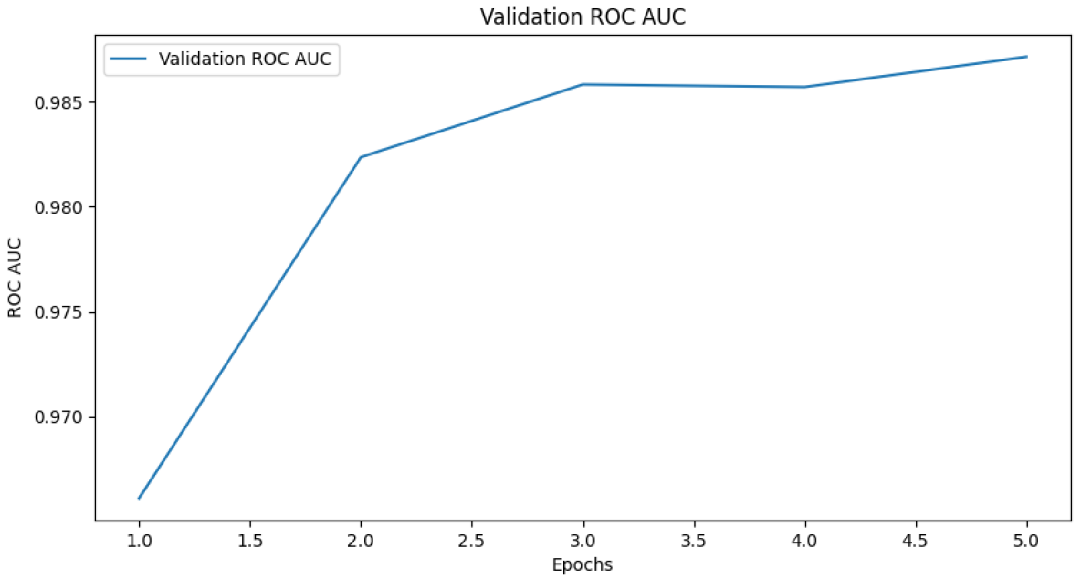


Figure 7: Inception v3 ROC-AUC Curve



(a) Resnext-50 Loss Plot



(b) Resnext-50 ROC-AUC Curve

Figure 8: Resnext-50 hyperparameter tuning of 5 epochs