HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 1 (2023)

Audio Classification of Bird Species Using Convolutional Neural Networks

Jocelyn Wang[1] and Guillermo Goldsztein[#]

[1]Jericho High School
[#]Advisor

ABSTRACT

As the total number of birds has declined in the billions over the last 50 years, an accurate method for classifying bird species is necessary for conservation efforts and population monitoring. One promising method is using machine learning models to classify birds by their sounds, which has emerged due to benefits such as being less affected by environmental factors (e.g., habitat, time of day), and lower disturbances to bird species during the data collection process, contrary to other processes such as image classification. As audio processing may eventually become the main method of classifying birds and may be used as an important conservation tool, it is imperative to understand the challenges that must be overcome before it can be successfully applied. In this work, the programming language Python and the machine learning model Convolutional Neural Networks were used to process and classify audio recordings from over 150 different bird species. This study demonstrates that although audio classification is a promising method of classification, many challenges are still present in the field, such as the amount of variety in the different calls of a single bird, the presence of background noises in many audio recordings, and the difficulty in efficiently representing an audio signal with images, highlighting the importance of overcoming these challenges for conservation efforts.

Introduction and Related Works

There are an estimated 17,000 to 18,000 bird species in the world (Barrowclough et al., 2016). Birds play crucial roles in environmental ecosystems, from preying on crop damaging insects and providing pest control, to pollination and seed dispersion (Sekercioglu et al., 2016). Additionally, birds may be used to measure environmental quality and ecosystem sustainability (Kalisińska, 2019), further highlighting their importance in ecosystems and conservation. Unfortunately, in the past fifty years, the population of breeding birds has suffered a net loss of 2.9 billion in numbers (Rosenberg et al., 2019), making it crucial to find a method that will accurately classify birds in order to monitor their population trends and gain a better understanding of their conservation statuses.

Multiple attempts have been made at bird population monitoring. Various types of monitoring were often used to collect data on birds, including traditional field surveys and citizen science surveys (Lepczyk, 2005), video monitoring (Verstraeten et al., 2010), and acoustic monitoring (Pérez-Granados & Traba, 2021). Initially, the most often used method of classification on collected bird data was visual classification, and many methods emerged for classifying images of birds. Classification of bird species based on images was often based on machine learning models such as Random Forests (Roslan et al., 2017), K Nearest Neighbors (KNN) (Budiman et al., 2022), and most commonly, Convolutional Neural Networks (CNN) (Kahl et al., 2017). However, recently, classification based on bird calls has come to light as a different way to classify birds, as audio classification has certain benefits over image classification that make it valuable. Firstly, audio recordings are not impacted by any visual factors such as light obstruction or time of the day, and audio monitoring methods do not cause disturbances to birds from human activity (Pérez-Granados et al., 2019). In addition, audio classification is similar to image classification in that audio representations must be converted to visual representations before being classified, so similar methods may be used between

image and audio classification with better effects in audio classification. However, audio classification requires more significant preprocessing steps. Therefore, numerous preprocessing methods have been applied to audio data (Wang et al., 2022), (Ramashini et al., 2022), before applying the aforementioned machine learning methods for classification (Kahl et al., 2017), (Ghani & Hallerberg, 2021), (Yang et al., 2021).

Technical Introduction

In this section, technical background, and common practice in machine learning with audio data are provided. The aim of this introduction is to familiarize readers with key technical concepts since concepts introduced in this section will be partly utilized in this study.

Spectrograms and Short Time Fourier Transform (STFT)

All audio signals can be expressed as the sum of many distinct frequencies. In order to convert signals from the time domain to the frequency domain, a fast Fourier transform (FFT) is typically applied. However, in a regular Fourier transform, information from the time domain is lost. Since information about time is also important in real world audio classification, the short-time Fourier transform (STFT) is often used instead. An STFT uses a sliding window time signal (often a Hamming or Hanning window) to compute FFTs for each windowed segment, then combines the results of each FFT into one graph in which frequency is rotated to be the y axis. In turn, a 3D graph is produced, including information on time, frequency, and amplitude. The amplitude at each point on this new graph is represented by a color (color is the 3rd dimension), with darker colors representing lower amplitudes and brighter colors representing higher amplitudes. This type of colored representation is known as a spectrogram, which is in the form of an image and can be trained upon with a classifier. The above process is displayed in Figure 1 (Gao & Yan, 2021), where a fast Fourier transform is performed on each window, and the results are combined to form one graph.

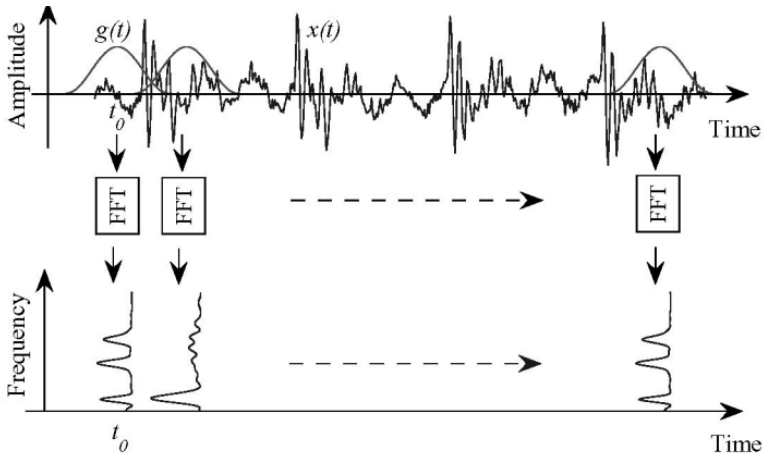

Figure 1. Illustration of short time Fourier transform (Gao & Yan, 2021)

However, initially, the combined graph usually displays very little information and likely will consist of very dark colors throughout. This is because most sounds humans hear fall into a narrow range of amplitudes. In fact, humans perceive amplitude logarithmically rather than linearly. Therefore, to account for the human perception of amplitude, the amplitudes of the spectrogram are converted to the decibel (dB) scale (a logarithmic scale of amplitudes where an increase of 10 dB means a tenfold increase in amplitude), and the colors of the spectrogram are represented by the decibel scale.

Mel Spectrogram

In addition to perceiving amplitudes logarithmically, humans also perceive frequencies logarithmically. On a linear scale, human ears do not correctly perceive distances between frequencies. Humans are more sensitive to lower frequencies than higher frequencies: for example, humans are much better at identifying the distance between 100 Hz and 200 Hz than the distance between 10,000 Hz and 11,000 Hz. Therefore, a logarithmic scale called the mel scale is used to represent frequencies. Frequencies of equal distance on the mel scale are also perceived to be equal in distance by human ears. Frequencies are converted from Hz to their corresponding mels based on the curve shown in Figure 2, whose logarithmic equation is commonly expressed in equation (1) (O'Shaughnessy, 1987) and is plotted in Figure 2. As shown in Figure 2, frequencies at lower Hz have greater distances between them in mels, and frequencies at higher Hz have smaller distances between them in mels. A spectrogram that uses the mel scale for frequencies is known as a mel spectrogram.

$$m = 2595 \cdot log_{10}(1 + \frac{f}{100})$$

$$f = 700 \cdot (10^{\frac{m}{2595}} - 1) \quad (1)$$

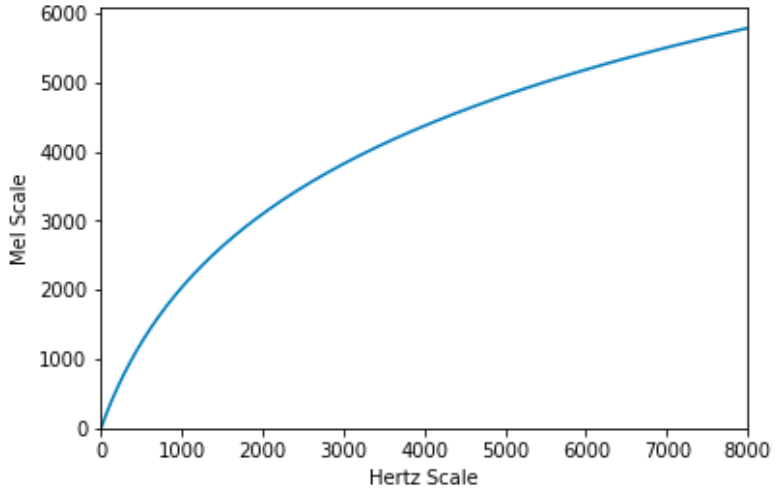

Figure 2. Relationship between mel frequency scale and Hz frequency scale

CNN

Convolutional Neural Networks (CNNs) have often been used in classification problems as they can learn features by themselves and do not require feature extraction. CNNs are known to successfully classify on images, representing each image as a matrix. Since many audio classification tasks can be converted to image classification tasks, CNNs are useful for audio classification as well. A typical CNN is composed of an input layer to begin, fully connected layers to end, and multiple hidden layers between, as shown in Figure 3. The hidden layers are composed of layers named convolution layers, batch normalization layers, and pooling layers.

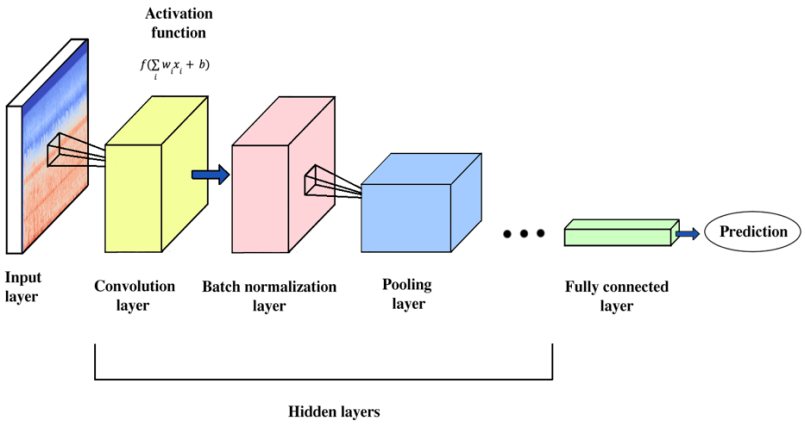

Figure 3. CNN structure

The convolution layer carries the main portion of the network's computational load. Convolution layers perform a dot product between a kernel matrix and a portion of the layer input matrix, called the receptive field. The convolution process is shown in Figure 4. Notice that the dot product is taken for each 2x2 matrix, by moving the receptive field (the gray area) by 1 unit (the stride length). Next is the pooling layer. Pooling layers come after convolution layers. Pooling layers summarize the previous convolution layer and help to reduce input sizes, while maintaining important features. Additionally, note that extra layers called batch normalization layers can be added to apply a transformation that makes the mean output close to 0 and the output standard deviation close to 1, which helps improve the training speed of the model. Finally, the last layers of the CNN are called fully connected layers. The fully connected layers take the inputs from the previous layers and outputs a single prediction (Alzubaidi et al., 2021).

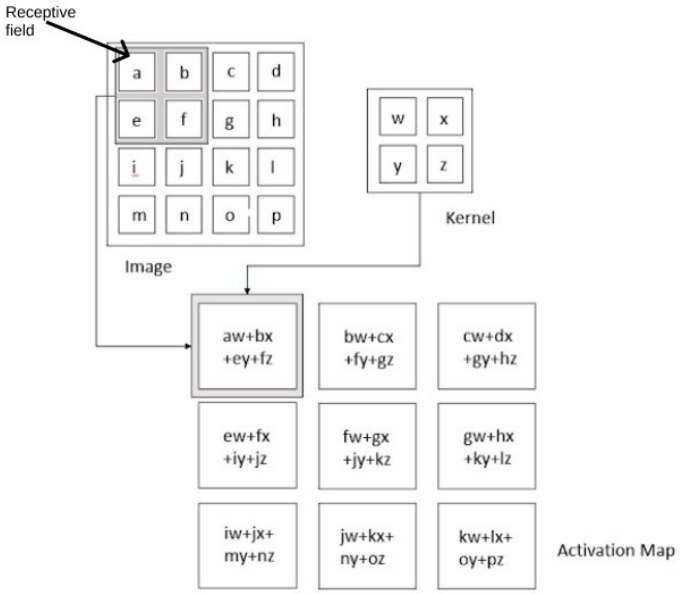

Figure 4. Convolution being performed with a 2x2 kernel when stride length is 1 (Goodfellow et al., 2016)

Methods

The paper is organized as follows. Preprocessing, image generation, and CNN creation steps are discussed in the methods section. Results evaluated based on four metrics are discussed in the results section, and future directions and a conclusion are presented in the conclusion section. The flowchart in Figure 5 demonstrates all research steps. All images and models used for classification were generated using Python.

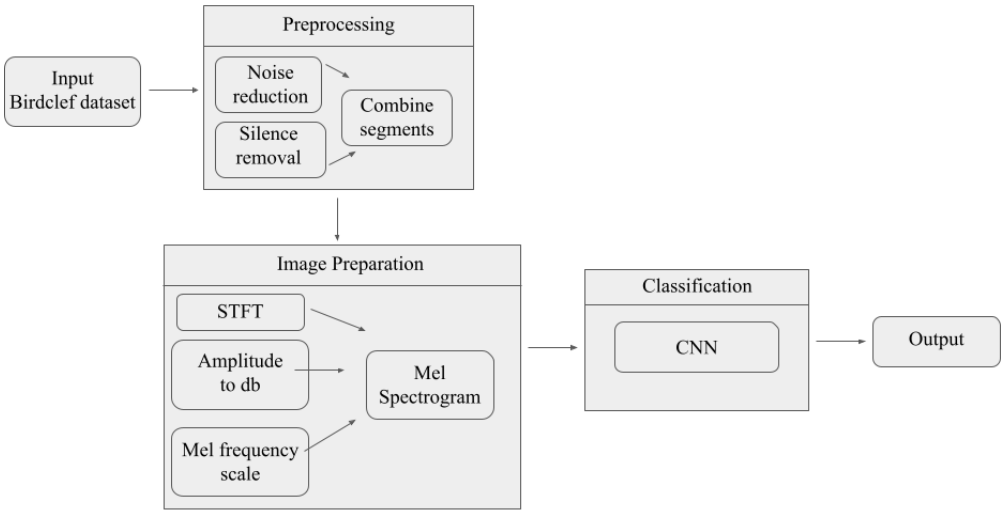

Figure 5. Flowchart of research steps

Dataset

In this study, a dataset of birds provided by the Kaggle BirdCLEF competition 2022 was used. The dataset is composed of crowdsourced labeled bird recordings from the xeno-canto database, and contains 151 labeled classes, each representing a different species of bird. Each class of bird contains multiple recorded ogg sound files, organized in folders for each class. In total, there are 16,753 audio recordings, ranging from a few seconds to up to a few minutes in length. All audio data were sampled at 44.1kHz.

Preprocessing

Upon generating mel spectrograms, it was discovered that many audio samples in the dataset contained significant background noise or silent segments, negatively impacting classification accuracy, as shown in Figure 6. Therefore, it became crucial to reduce the noise and remove the silence.

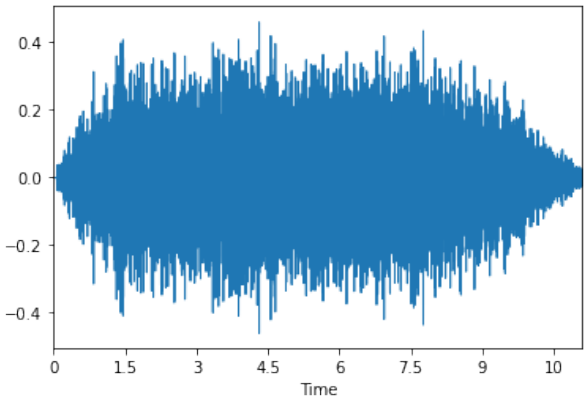

Figure 6. Noisy signal of African Silverbill bird audio recording

Noise reduction and silence removal libraries

In the first step of preprocessing, the function reduce_noise() from the noisereduce Python library (Sainburg, 2019) is used, implementing a form of noise gate called a spectral gate, which estimates a noise threshold for each frequency band of an inputted signal. The threshold is calculated through the usage of statistics: any frequency which is 1.5 standard deviations from the mean is classified as an outlier, which means noise. A noise mask is then computed using this threshold, and the mask gates all noise outside of the aforementioned frequency range, allowing for the separation and removal of background noise. The signal after noise reduction is displayed in Figure 7.

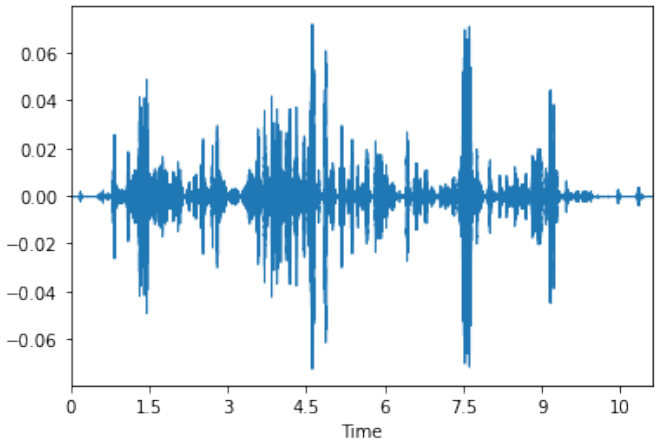

Figure 7. African Silverbill bird audio signal after noise reduction

Next, the durations of silence between bird calls were removed. In order to do this, the function silence_removal() from the Python library pyAudioAnalysis (Giannakopoulos, 2015), which takes audio signals and outputs the endpoints of audio events that are not silent, was used. The function achieves this by using support vector machines (SVM). An SVM is trained to distinguish between both low and high energy frames. 10% of the lowest and highest energy frames are used to train the SVM, giving a general range of the audio. Finally, for each frame sequence, the SVM outputs the probability that this frame sequence is not silent. All segments that are not silent are returned. Using Python splicing, these segments are then extracted, leaving out the silent segments, as shown in Figure 8.

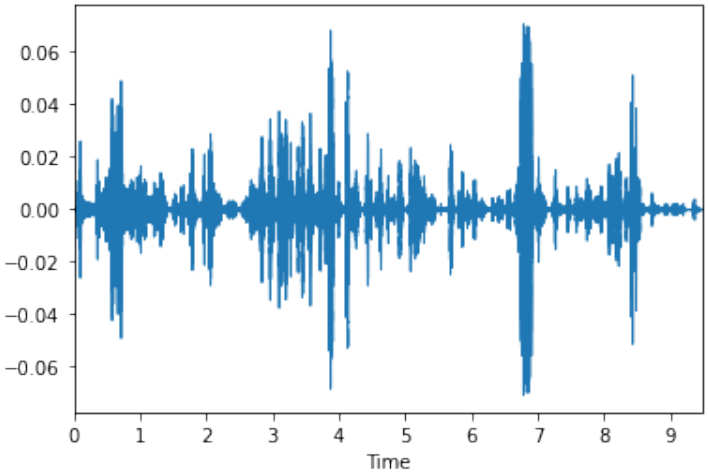

Figure 8. African Silverbill bird audio signal after noise reduction and silence removal

Finally, after both noise reduction and silence removal have been performed, all remaining segments are combined to form a single large file. This combined file is then split into 10 second time intervals, each of which will later be used to generate a spectrogram with the same label as the original large sample. If there are any time intervals less than 10 seconds, zero padding will be applied so that the interval is 10 seconds.

Image Generation

Image generation is carried out in several stages. In the BirdCLEF dataset, audio recordings were in the format ogg. and were represented as an array of numbers depicting the amplitude of the audio recording at each timestep. This array of numbers can then be expressed as a signal in the time domain. An example of such a signal representation is shown on the left in Figure 9, with time on the x axis in seconds and amplitude on the y axis. Upon applying an FFT to the original signal using the function np.fft.fft() from the NumPy Python library (Harris et al., 2020) the frequency spectrum was generated, as shown on the right in Figure 9.

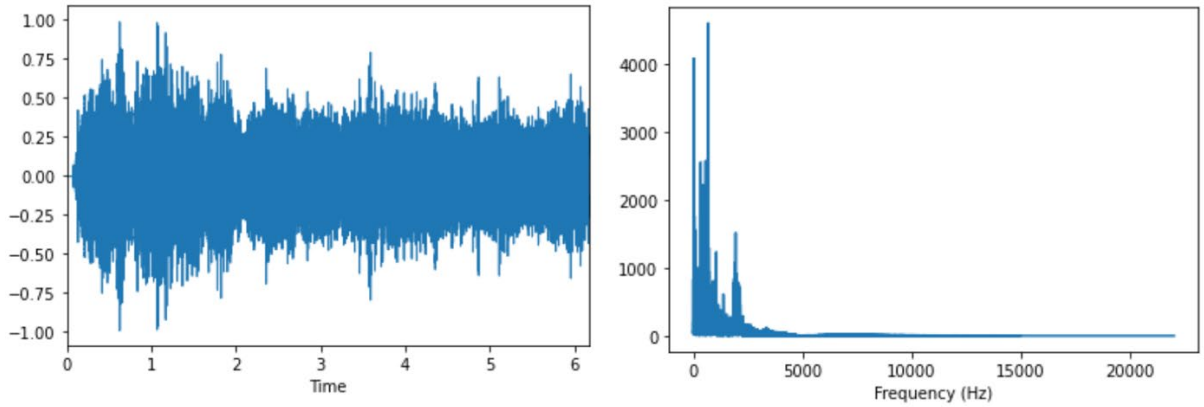

Figure 9. House Finch bird audio signal in time domain (left) and frequency domain (right)

In order to get a spectrogram of the original audio signal, an STFT was performed using the function librosa.stft() from the librosa python library (McFee et al., 2022). The resulting spectrogram is shown below in Figure 10.

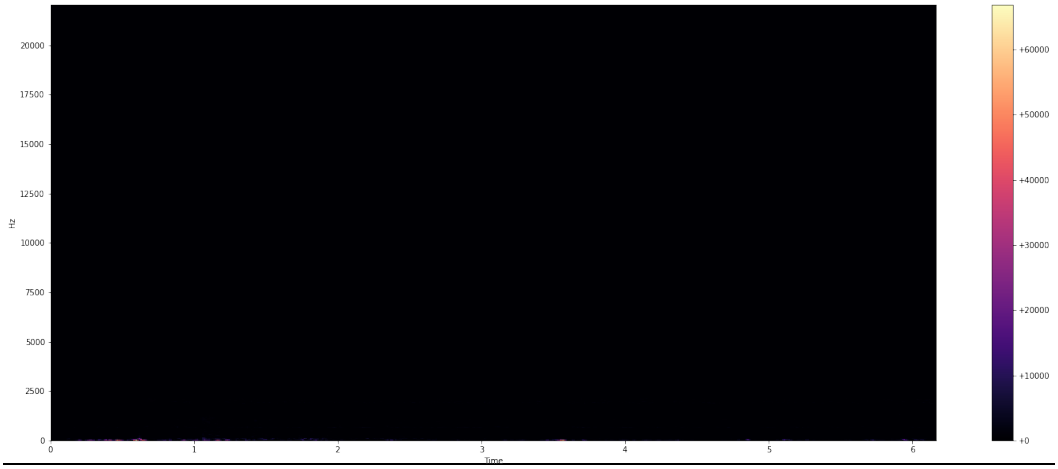

Figure 10. Spectrogram of House Finch bird audio recording

Notice that the above spectrogram (Figure 10) is fairly dark with little color concentration. In order to generate a more representative spectrogram, a logarithmic amplitude transformation using the function librosa.amplitude_to_db() from the librosa Python library was applied to generate the following spectrogram shown below in Figure 11.

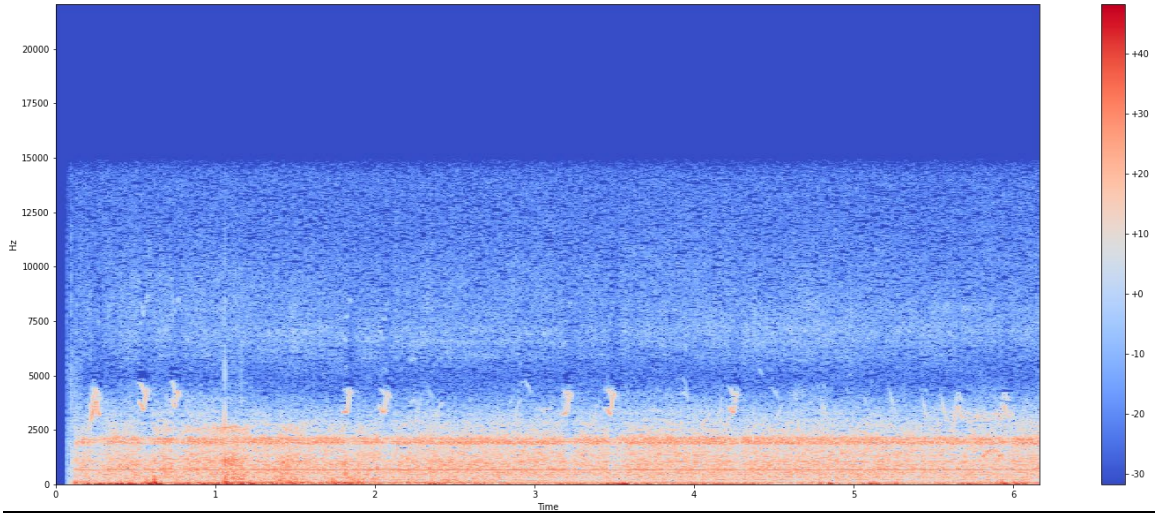

Figure 11. Spectrogram of House Finch bird audio recording with amplitude in dB

To convert the spectrogram to a mel spectrogram, the frequency scale was converted to a mel scale with 80 mel bands using the function librosa.feature.melspectrogram() from the librosa library. The final mel spectrogram is shown below in Figure 12.

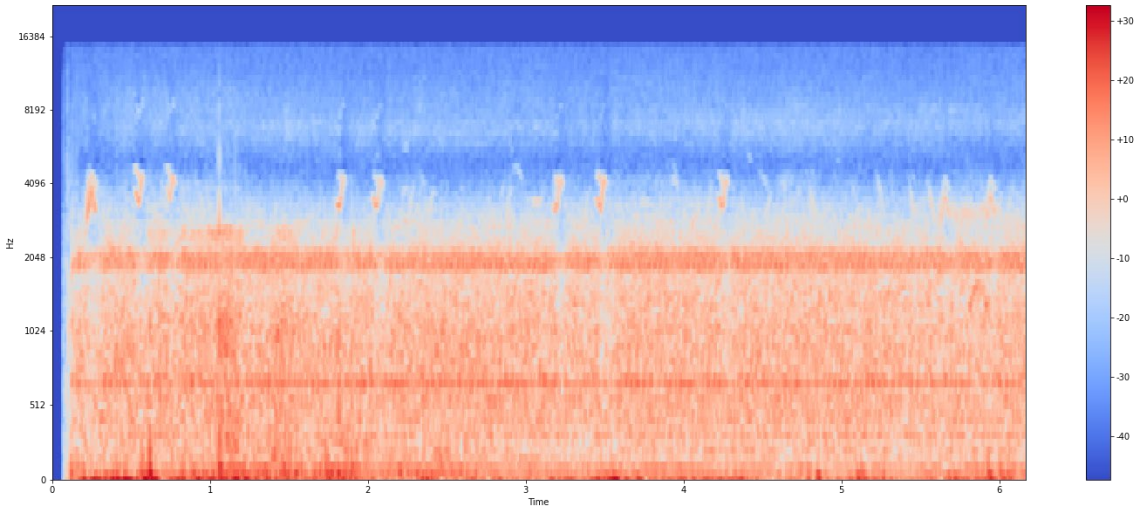

Figure 12. Finished Mel Spectrogram of House Finch Bird

This process of converting to mel spectrogram was repeated for the audio recordings of every bird. The respective mel spectrograms for each audio file were saved as images with their original labels.

CNN Architecture

In order to classify the data, a CNN was constructed using the Python library TensorFlow Keras (Abadi et al., 2016). In total, the CNN had 18 layers, including an input layer, 3 convolution layers, 3 pooling layers, a final dense layer, and other intermediary layers. Relu activation and same padding were used for each convolution layer, and batch normalization layers were added after convolutional and pooling layers. Finally, a flatten layer was included to make the multidimensional input one dimensional, and a dropout layer was included to prevent overfitting. All layers are displayed in Table 1.

Table 1. CNN architecture with all layers

Layer
Sequential layer and Input Layer
Conv 1: Input size: 442 x 858 x 3 • Number of filters = 32 • Kernel size = 3x3 • Stride = 2 • Number of parameters = 896 • Output size: 221 x 429 x 32
Batch normalization layer 1 • Number of parameters = 128
Max Pooling layer 1: pool size 2x2 • Output size: 110 x 214 x 32
Batch normalization layer 2 • Number of parameters = 128

Conv 2: Input size: 110 x 214 x 64 • Number of filters = 64 • Kernel size = 3x3 • Stride = 1 • Number of parameters = 18,496 • Output size: 110 x 214 x 64
Batch normalization layer 3 • Number of parameters = 256
Max pooling layer 2: pool size 2x2 • Output size = 55 x 107 x 64
Batch normalization layer 4 • Number of parameters = 256
Conv 3: Input size: 110 x 214 x 6 • Number of filters = 128 • Kernel size = 3x3 • Stride = 1 • Number of parameters = 73,856 • Output size: 55 x 107 x 128
Batch normalization layer 5 • Number of parameters = 512
Max Pooling layer 3: pool size 2x2 • Output size = 27 x 53 x 128
Batch normalization layer 6 • Number of parameters = 512
Flatten layer • Output size = 183168
Dense layer: 256 units, relu activation • Output size = 256 • Number of parameters = 46,891,264
Batch normalization layer 7 • Number of parameters = 1024
Dropout layer: frequency rate 0.5 • Output size = 256
Dense layer: softmax activation • Output size = 152 • Number of parameters = 39,064

For the optimizer function, the CNN used RMSProp, a gradient based optimizer that has an adaptive learning rate. Categorical cross entropy was used as the loss function.

Results and Discussion

The model used 100 epochs and took a total of 6 hours to run. 4 metrics were used to evaluate the model, as shown in Table 2. Refer to the confusion matrix in Figure 13 for better understanding.

Table 2. Metrics for evaluating model

Metric	Definition
Accuracy	(TP + TN)/(TP + TN + FP + FN)
Precision	TP/(TP + FP)
Recall	TP/(TP + FN)
Loss	The penalty for a prediction: 0 loss means a perfect prediction.

		True values	
		Positive	Negative
Predicted values	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Figure 13. Confusion matrix

The results of all 4 metrics for both the train and validation data over all 100 epochs are shown in figures 14, 15, 16, and 17 respectively.

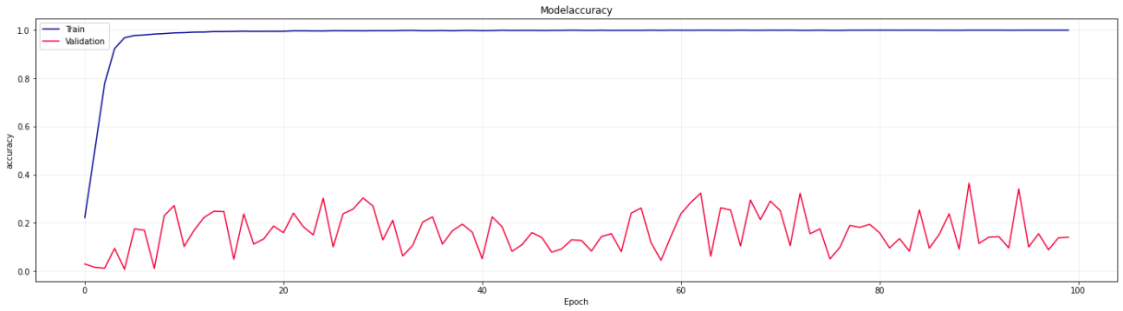

Figure 14. Model accuracy for both train and validation data

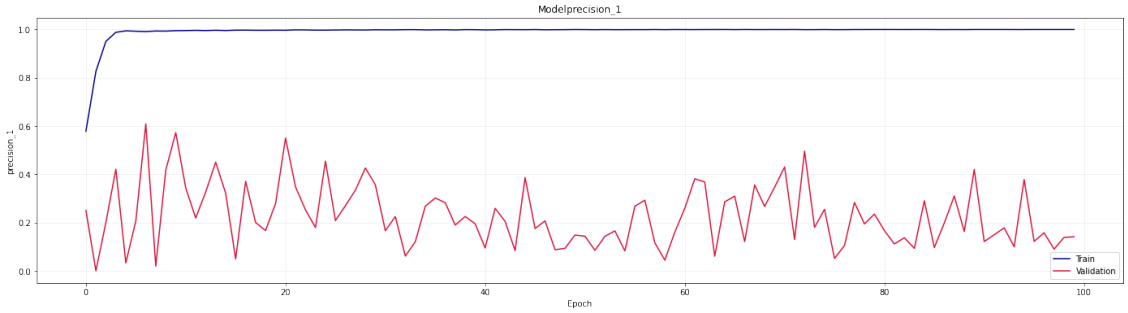

Figure 15. Model precision for both train and validation data

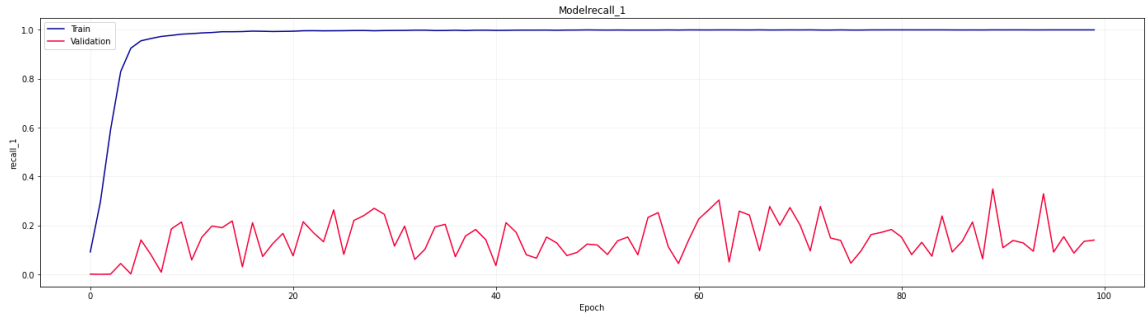

Figure 16. Model recall for both train and validation data

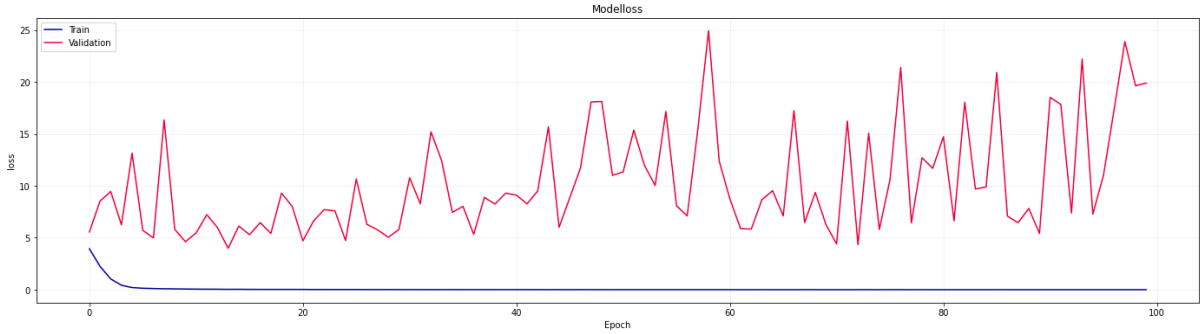

Figure 17. Model loss for both train and validation data

A summary of the results for each metric is displayed below in Table 3.

Table 3. Summary of 4 different result metrics

Category	Training	Validation
Accuracy	1	Between 0 and 0.4
Precision	1	Between 0 and 0.6
Recall	1	Between 0 and 0.4
Loss	0	Between 4 and 25

Clearly, the model achieved high results in training, reaching up to nearly perfect results, but achieved poor results in validation, indicating that there were some problems. One such problem is overfitting, which is likely present in this model, as much better training metrics compared to validation metrics is a clear sign of overfitting.

In order to reduce overfitting in this study, two methods were already utilized during training. Batch normalization and dropout were used on the convolutional and fully connected layers at the end of the CNN network. However, little improvement was reached from these efforts, showing that a deeper look into the image data is required. One likely reason for the low validation results may be due to certain aspects of the original data that were not sufficiently addressed during preprocessing, including variations in calls of the same bird species, multiple birds present in a recording, and no birds present in a recording. Firstly, birds typically have a wide variety of different vocalizations (eg. fighting, mating) for different tasks. An example of this is shown in Figure 18, which displays two very different spectrograms despite being the same bird.

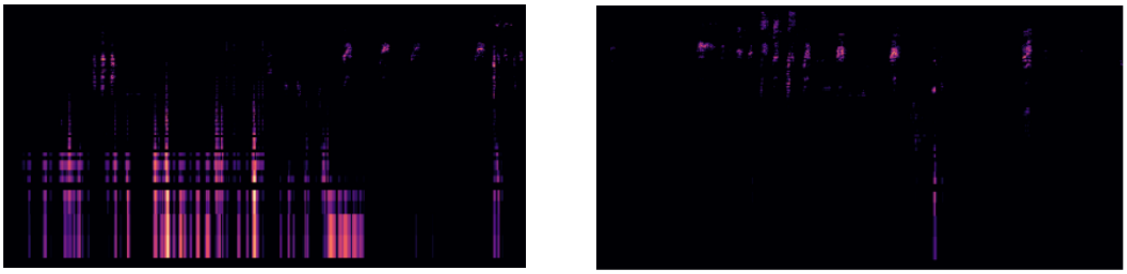

Figure 18. Two spectrograms generated from audio recordings of the bird Black-crowned Night-Heron

These alternate vocalizations of the same bird may be mistaken for vocalizations of another bird during classification. Next, due to the nature of data collection in terms of bird calls, audio recordings may have no bird calls or multiple bird calls in them, as recordings are inevitably subjected to the environment. These conditions must be accounted for during preprocessing.

Finally, the nature of spectrograms themselves may also contribute to the lower validation results, as spectrograms are different from regular images. For example, regular images that are shifted or rotated still carry the same information, leading to CNNs being skilled at recognizing images under transformations as the same image. However, in spectrograms, transformations often change the information of the spectrogram entirely. In a spectrogram, transformations such as rotations do not make sense, and other transformations, such as shifts along the frequency (y) axis, may cause two spectrograms to look similar for the CNN. However, these shifted spectrograms represent completely different audio signals with different pitches. This may cause the CNN to incorrectly identify two different birds as the same bird if they have similar frequency patterns but different pitches.

Conclusion and Future Works

In this work, a straightforward method of classifying on mel spectrograms with CNNs was applied, generating an audio classification model that did not produce high validation results. This study demonstrates that although audio classification of birds using machine learning is promising, it still faces multiple challenges before it can accurately be used for conservation purposes. Therefore, multiple future improvements could be made to different aspects of this project. In terms of preprocessing, a possible improvement would be to first predict when there is no call or more than one call present in any audio recording, then to create separate classes for these two instances. Classes with no bird calls should be removed from the training data, and classes with more than one call should be handled separately from those with one call. Additionally, further preprocessing work could be to use a K nearest neighbors model to divide each bird class into subgroups based on the appearance of spectrograms before actually classifying with a CNN. Finally, one last future research direction would be to further examine the qualities of mel spectrograms to find a way to reduce the impacts of certain transformations, such as vertical axis shifts.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*. https://doi.org/10.48550/arXiv.1603.04467

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data*, *8*(1), 1-74. https://doi.org/10.1186/s40537-021-00444-8

Barrowclough, G. F., Cracraft, J., Klicka, J., & Zink, R. M. (2016). How many kinds of birds are there and why does it matter? PLoS ONE 11(11): e0166307. https://doi.org/10.1371/journal.pone.0166307

BirdCLEF 2022 [data]. (2022). kaggle. Retrieved November 30, 2022, from https://www.kaggle.com/competitions/birdclef-2022/data

Budiman, I., Ramdania, D. R., Gerhana, Y. A., Putra, A. R. P., Faizah, N. N., & Harika, M. (2022, September). Classification of Bird Species using K-Nearest Neighbor Algorithm. In *2022 10th International Conference on Cyber and IT Service Management (CITSM)* (pp. 1-5). IEEE. https://doi.org/10.1109/CITSM56380.2022.9936012

Gao, R. X., & Yan, R. (2006). Non-stationary signal processing for bearing health monitoring. *International journal of manufacturing research*, *1*(1), 18-40. https://doi.org/10.1504/IJMR.2006.010701

Ghani, B., & Hallerberg, S. (2021). A randomized bag-of-birds approach to study robustness of automated audio based bird species classification. *Applied Sciences*, *11*(19), 9226. https://doi.org/10.3390/app11199226

Giannakopoulos, T. (2015). pyaudioanalysis: An open-source python library for audio signal analysis. *PloS one*, *10*(12), e0144610. https://doi.org/10.1371/journal.pone.0144610

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. Retrieved from: http://www.deeplearningbook.org

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357-362. https://doi.org/10.1038/s41586-020-2649-2

Kahl, S., Wilhelm-Stein, T., Hussein, H., Klinck, H., Kowerko, D., Ritter, M., & Eibl, M. (2017). Large-Scale Bird Sound Classification using Convolutional Neural Networks. In *CLEF (working notes)* (Vol. 1866).

Kalisińska, E. (Ed.). (2019). *Mammals and birds as bioindicators of trace element contaminations in terrestrial environments: an ecotoxicological assessment of the Northern Hemisphere*. Springer. https://doi.org/10.1007/978-3-030-00121-6

Lepczyk, C. A. (2005). Integrating published data and citizen science to describe bird diversity across a landscape. *Journal of Applied Ecology*, *42*(4), 672-677. https://doi.org/10.1111/j.1365-2664.2005.01059.x

McFee, B., Metsai A., McVicar M., Balke S., Thomé C., Raffel C., Zalkow F., Malek A., Dana, Lee K., Nieto O., Ellis Dan., Mason J., Battenberg E., Seyfarth S.,Yamamoto R., Morozov V., Morozov R., Choi K., Moore J., … Kim T. (2022). librosa/librosa: 0.9.2 (0.9.2). Zenodo. https://doi.org/10.5281/zenodo.6759664

O'Shaughnessy, D. (1987). *Speech communications: Human and machine (IEEE)*. Universities press.

Pérez-Granados, C., Bota, G., Giralt, D., Barrero, A., Gómez-Catasús, J., Bustillo-De La Rosa, D., & Traba, J. (2019). Vocal activity rate index: a useful method to infer terrestrial bird abundance with acoustic monitoring. *Ibis*, *161*(4), 901-907. https://doi.org/10.1111/ibi.12728

Pérez-Granados, C., & Traba, J. (2021). Estimating bird density using passive acoustic monitoring: a review of methods and suggestions for further research. *Ibis*, *163*(3), 765-783. https://doi.org/10.1111/ibi.12944

Ramashini, M., Abas, P. E., Mohanchandra, K., & De Silva, L. C. (2022). Robust cepstral feature for bird sound classification. *International Journal of Electrical and Computer Engineering*, *12*(2), 1477-1487. https://doi.org/10.11591/ijece.v12i2.pp1477-1487

Rosenberg, K. V., Dokter, A. M., Blancher, P. J., Sauer, J. R., Smith, A. C., Smith, P. A., ... & Marra, P. P. (2019). Decline of the North American avifauna. *Science*, *366*(6461), 120-124. https://doi.org/10.1126/science.aaw1313

Roslan, R., Nazery, N. A., Jamil, N., & Hamzah, R. (2017, October). Color-based bird image classification using Support Vector Machine. In *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)* (pp. 1-5). IEEE. https://doi.org/10.1109/GCCE.2017.8229492

Sekercioglu, Ç. H., Wenny, D. G., & Whelan, C. J. (Eds.). (2016). *Why birds matter: avian ecological function and ecosystem services*. University of Chicago Press. https://doi.org/10.1111/jofo.12214

Tim Sainburg. (2019). timsainb/noisereduce: v1.0 (db94fe2). Zenodo. https://doi.org/10.5281/zenodo.3243139

Verstraeten, W. W., Vermeulen, B., Stuckens, J., Lhermitte, S., Van der Zande, D., Van Ranst, M., & Coppin, P. (2010). Webcams for bird detection and monitoring: A demonstration study. *Sensors*, *10*(4), 3480-3503. https://doi.org/10.3390/s100403480

Wang, H., Xu, Y., Yu, Y., Lin, Y., & Ran, J. (2022). An Efficient Model for a Vast Number of Bird Species Identification Based on Acoustic Features. *Animals*, *12*(18), 2434. https://doi.org/10.3390/ani12182434

Yang, S., Frier, R., & Shi, Q. (2021, February). Acoustic classification of bird species using wavelets and learning algorithms. In *2021 13th International Conference on Machine Learning and Computing* (pp. 67-71). https://doi.org/10.1145/3457682.3457692