Christopher Yang
DATA 5322
May 11, 2025

Sound of Seattle Birds: Convolutional Neural Networks to Classify Bird Species Using Audio Spectrograms

Abstract

This report presents a deep learning based approach for classifying bird species based on audio recording spectrograms of their calls using convolutional neural networks (CNNs). Two custom built CNNs were designed, interpreted, and evaluated for two classification tasks. One for binary classification between just 2 of 12 bird species common in the Seattle area, and another for multiclass classification across all 12 species. The spectrograms used in this analysis are representations of audio clips that were processed into 128 x 517 Mel spectrogram images, which served as our CNN inputs. To optimize the performance of each model, hyperparameters were tuned and complexities (layers of models) were experimented with. Our analysis also included 3 external test audio mp3 files which were converted to spectrograms then predicted on. Despite class imbalance and the computational restraint of a 2019 macbook, the best performing model for binary classification achieved 84% test accuracy with 0.9 overall AUC, and the best performing model for multiclass classification achieved ~65% test accuracy across the 12 species.

Introduction

The goal of this project was to develop and evaluate custom CNN models capable of identifying bird species from spectrogram representations of audio recordings of their calls provided in a preprocessed HDF5 file. The dataset used for training the models are a subset of a Kaggle Bird Calling Competition dataset originating from Xeno-Canto [1], which is a crowd-sourced bird sounds archive containing bird calls of over 264 species. As mentioned before, both binary and multiclass classification models were developed and evaluated using just 2 species of birds for binary (House Sparrow and Song Sparrow) and all 12 species (American Crow, American Robin, Bewick's Wren, Black-Capped Chickadee, Dark-Eyed Junco, House Finch, House Sparrow, Northern Flicker, Red-Winged Blackbird, Song Sparrow, Spotted Towhee, and White-Crowned Sparrow) for multiclass. For both binary and multi-class classification, a more deep and complex network of 5 layers with some hyperparameter tuning yielded the best results and captured more insight within our data.

Theoretical Background

Convolutional Neural Networks (CNNs) are a type of deep learning model that has become a standard approach for image classification tasks due to their ability to learn hierarchical features from input data. In this project, we use CNNs to classify audio spectrograms (image representation of sounds like bird calls), which convert time-frequency patterns into images that CNNs can analyze.

A CNN is typically made up of multiple layers, each providing a specific function in the learning process:

- **Convolutional Layers**: These layers apply a set of learnable filters (kernels) to the input data to extract local features. Each filter is trained to detect specific patterns, such as frequency modulations or temporal chirps in the spectrograms. One of the key hyperparameters here is the number of filters at each layer. Too few filters may not capture enough features, while too many filters can risk overfitting and lead to long runtimes. Another key hyperparameter here is the filter size, which determines the receptive field of the feature detector.
- **Activation Functions**: Non-linear activation functions like ReLU (rectified linear unit) are applied after convolution to introduce non-linearity into the model. This allows the CNN to learn complex, non-linear relationships between the features. Without activation functions, a CNN would behave like a linear model regardless of depth.
- **Pooling Layers**: These layers reduce the spatial dimensions of the feature maps, which help generalize the learned features and reduce overfitting. Max pooling is most commonly used with pool sizes like 2x2 or 3x3. Pooling not only reduces computational costs, but also adds some translation invariance into the model (output remains unchanged regardless of how its input is shifted or translated).
- **Dropout Layers**: Dropout is a regularization technique that randomly "drops" or deactivates a subset of neurons in the model during training. This helps prevent the model from relying on a specific node and leads the model to learn more robust features. The dropout rate is a crucial hyperparameter since dropping too many neurons prevents the model from learning, while dropping not enough might lead the model to overfit.
- **Fully Connected (Dense) Layers:** These layers are used at the end of the network to combine extracted features and make the final classification. They are prone to overfitting due to the large number of parameters and might require regularization.
- **Global Average or Max Pooling:** Instead of flattening the feature maps and feeding them into a large dense layer (which creates too many parameters), global pooling aggregates the spatial information into a single vector per feature map. This reduces the parameter count and improves the training efficiency and generalization of the model.
- **Batch Normalization:** This layer normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. This stabilizes and speeds up the training process. This also helps mitigate internal covariate shift, which is a phenomenon where the distribution of each layer's inputs change during training.

When CNNs are too deep or trained on smaller datasets, they are prone to overfitting. The model tries to memorize instead of learn and generalize. Overfitting often happens when the models are too complex (too many filters, large dense layers, etc.) or trained too long without proper regularization. It is essential to tune the hyperparameters to build and train more robust models that can generalize well. In addition to architectural components, hyperparameters must be tuned for CNNs to perform effectively:

- **Filter Size and Number:** Determines the size and the number of pattern detectors at each layer. Larger filter sizes (5x5, 7x7) are more general and might lose detail, while the smaller filters (3x3) are more fine-grained with a risk of overfitting.
- **Stride and Padding:** The stride controls how much the filter moves per step. A stride of 1 retains more detail, while larger strides downsample the output. Padding ('same' or 'valid') determines whether the spatial dimensions are preserved during convolution.
- **Dropout Rate:** Dropout rates between 0.2-0.5 are commonly used and should be adjusted based on the layer's position and depending on whether the model overfits.
- **Learning Rate:** Another important hyperparameter. This determines how much the model weights are updated during training. For the Adam optimizer, the default is 0.001, but values like 1e-4 or 1e-5 are sometimes better when the training is unstable. Learning rate schedules, like ReduceLROnPlateau, can dynamically adjust the weights based on the validation performance of the model.
- **Batch Size:** This defines the number of samples used per gradient update. Smaller batch sizes (16-32) introduce more noise, which can help generalization, while larger sizes (64-128) train faster but might generalize worse.
- **Epochs:** The number of complete passes through the training dataset. Setting this too high can lead to overfitting. Early stopping can be used to prevent the model from training further once validation loss stops improving.
- **Optimizer:** The Adam optimizer is widely used because of its adaptive learning rate mechanism, but Stochastic Gradient Descent with momentum or RMSprop can also be used. Using different optimizers might lead to different convergence speeds and sometimes model performance as well.

For binary classification, sigmoid activation functions are used with binary cross entropy because we are classifying between 2 classes. For multiclass classification, softmax activation functions are used with categorical cross entropy because we are classifying across many different classes (more than 2 classes).
Evaluation metrics include:

- **Accuracy**: This checks the proportion of correct classifications/predictions
- **Confusion matrix**: This helps us visualize the correct/incorrect classifications of our model especially for multiclass
- **Precision/Recall/F1-Score**: Accuracy can be misleading especially when the data contains class imbalance, so looking at precision, recall, and f1-score provides a more meaningful understanding of false positives, true positives, and the balance between the two

- **AUC**: Used mainly for binary classification, but can technically be used for multiclass as well even though it is much more complicated with multiclass. For binary at least, it measures the ability to separate between the two classes.

Methodology

Data Preparation

The data was provided in preprocessed HDF5 format [2]. Each sample represents a 2 second spectrogram of a bird call with the dimensions of (128 x 517). For binary classification, the data was further subsetted to include only the two species of interest (Song Sparrow and House Sparrow) since these two species contained the most samples, and models typically perform better when they have a lot of data to train on. For multiclass classification, each sample was labeled an integer corresponding to one of the 12 bird species (0 = American Robin, 1 = American Crow, etc.).
There was also a noticeable class imbalance in our data. For example, House Sparrow had 630 samples while Northern Flicker had 37 samples. So to try and address the class imbalance, class weights were applied once to give more weight to species with less samples. Although techniques like Synthetic Minority Over-sampling Technique (SMOTE) exist for generating additional samples for minority classes, SMOTE was not used in this project. This is because SMOTE is not ideal when there is potential for class overlap, which is the case with our bird calls that may sound similar. Also, SMOTE assumes linear interpolation between features is meaningful, which is not suitable for image data, like our spectrograms.
For both classification tasks, each spectrogram was normalized to a [0, 1] range. The training and test data were split using stratified sampling with a ratio of 80/20.

Binary Classification Flow

(1) Gather input spectrograms for the two species (Song Sparrow and House Sparrow)
(2) Preprocess by normalizing and reshaping to correct 4D tensors
(3) Model Architecture:

	Model 1	Model 2	Model 3	Model 4
Layers	2	3	5	5
Flatten vs Global Pooling	Flatten	Global Average Pooling	Global Average Pooling	Global Average Pooling
Activation Function	ReLu	ReLu	ReLu	ReLu
Regularization	No	0.01	0.0001	0.0001
Label smoothing	None	None	0.05	0.05

Batch size	16	32	16	32
Dropout	None	0.3	Scaled 0.2-0.4	Scaled 0.2-0.4
Learning Rate	0.001	0.001	0.0001	0.0001
Batch Normalization	No	No	Yes	Yes

(4) Optimization: Adam optimizer with adjusted learning rates, early stopping, and class weights to try and handle some class imbalance
(5) Evaluation: test accuracy, AUC, precision, recall, f1-score from classification report

Multiclass Classification Flow

(1) Organize spectrograms for all 12 species
(2) Preprocess by label encoding, one hot transformation, and class weights calculation
(3) Model Architecture:

	Model 1	Model 2	Model 3
Layers	3	4	5
Flatten vs Global Pooling	Global Average Pooling	Global Average Pooling	Global Average Pooling
Activation Function	ReLu	ReLu	ReLu
Regularization	None	None	None
Batch Size	64	32	32
Dropout	0.3 (only in dense layer)	0.5 (only in dense)	0.5 (only in dense)
Learning Rate	0.001	0.0001 (with reduce on plateau)	0.001 (with reduce on plateau)
Batch Normalization	No	Yes	Yes
Class Weights	No	Yes	No

(4) Optimization: Adam optimizer early stopping, reduce learning rate on plateau, and class weights
(5) Evaluation: test accuracy, precision, recall, f1-score

3 External Audio Predictions

(1) Raw MP3 clips were converted to spectrograms [3]

(2) Preprocessed spectrograms to grayscale, resized, and normalized
(3) Made final predictions using Model 3 of multiclass classification with the top 3 species reported

Results

Link to the code

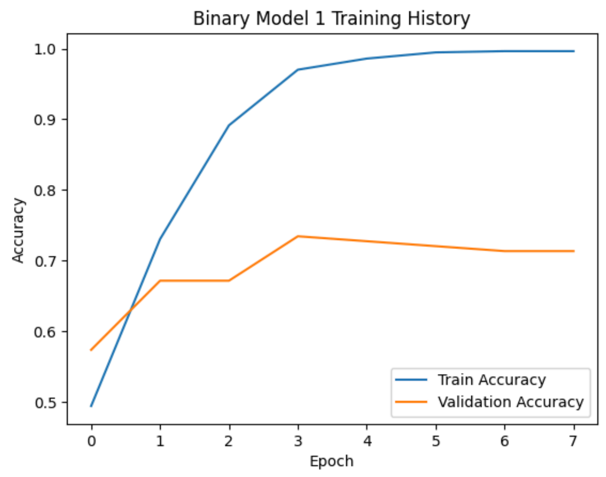
Binary Classification Results

Below is a summary table comparing the performance of the initial baseline model and the best performing model. The first model, a simple boilerplate architecture, performed reasonably well given its simplicity. It achieved an accuracy of 72% and an AUC of 0.65, which is somewhat better than random guessing and suggests that the model was able to capture some meaningful patterns in the data. Notably, it classified House Sparrows pretty accurately with a precision of 0.79 and recall of 0.81. However, it struggled to identify Song Sparrows indicated by the lower precision of 0.52 and recall of 0.49. This difference is likely because of class imbalance seen with the 53 samples of Song Sparrows and 126 samples of House Sparrows, which skewed the model's performance in favor of the majority class.
In contrast, the deeper model (Model 3) had more tuning and performed a bit better. It achieved an accuracy of 86% and an AUC of 0.90 indicating it was able to properly distinguish between the two species. Despite the class imbalance, model 3 was able to better balance performance across the two classes seen with the macro f1-score of 0.82. The precision and recall for both classes was much higher and more balanced than in model 1. This performance did come at the cost of runtime as model 1 only took ~2 minutes to run while model 3 took ~23 minutes.

Metric	Model 1	Model 3
Accuracy	72%	86%
AUC	0.65	0.90
Precision (Song Sparrow)	0.52	0.82
Precision (House Sparrow)	0.79	0.87
Recall (Song Sparrow)	0.49	0.68
Recall (Song Sparrow)	0.81	0.94
Samples (Song Sparrow)	53	53
Samples (House Sparrow)	126	126
Macro F1-Score	0.65	0.82

Time	~2m	~23m

Below are also the plots of training and validation accuracy over the epochs trained. We can see that the simple model (model 1) had some overfitting seen with the large, increasing gap between the training and validation accuracy as the training progressed, while the deeper model (model 3) had little to no overfitting seen with the similar training and validation accuracy, which is what we want to see. Another thing to note is that compared to the deeper model, the simpler model stopped very early on due to early stopping, which is why the runtime was also much shorter.



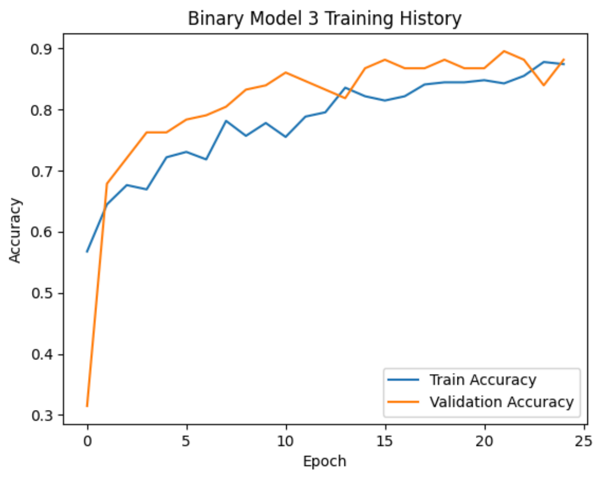


Figure 1 (above left) shows the training and validation accuracy across epochs for Model 1. **Figure 2 (above right)** shows the training and validation accuracy across epochs for Model 3.

Multiclass Classification Results

Below is a summary comparing the results of the initial and best performing multi-class models. The first model, a simple boilerplate architecture, performed decently in terms of accuracy. Although it was able to achieve 32% accuracy (better than random guessing ~8.33%), the macro precision of 0.0264 and macro recall of 0.0833 indicates that the model was performing poorly. The low macro precision and macro recall is likely due to the class imbalance mentioned earlier. The model was not able to learn or generalize the data well. The recall of 0.0833 is equal to 1/12 indicating the model only predicted on one class (majority class House Sparrow).
In contrast, the deeper model had more tuning and performed much better. It achieved an accuracy of 66%. The macro precision of 0.5913 and macro recall of 0.5035 indicate that the model was able to generalize well across all species (roughly half), which is much better than the simple model. However, the more complex model had significant overfitting seen with the high training accuracy and much lower validation accuracy. The more complex model took 45 minutes to run (5 times longer than the simple model), however, the trade-off was worth it since the more complex model performed much better and was able to learn somewhat effectively compared to the simple model.

Metric	Model 1	Model 3
Accuracy	32%	66%
Train vs Validation Accuracy	31.82% vs 31.74%	95.39% vs 65.99%
Macro Precision (across all species)	0.0264	0.5913
Macro Recall (across all species)	0.0833	0.5035
Macro F1-Score (across all species)	0.0402	0.5233
Time	~ 9m	~ 45m

Below is the confusion matrix for the simple model. Like the macro recall indicated, our simple model only classified and predicted on one species, which was our majority class species (House Sparrow).

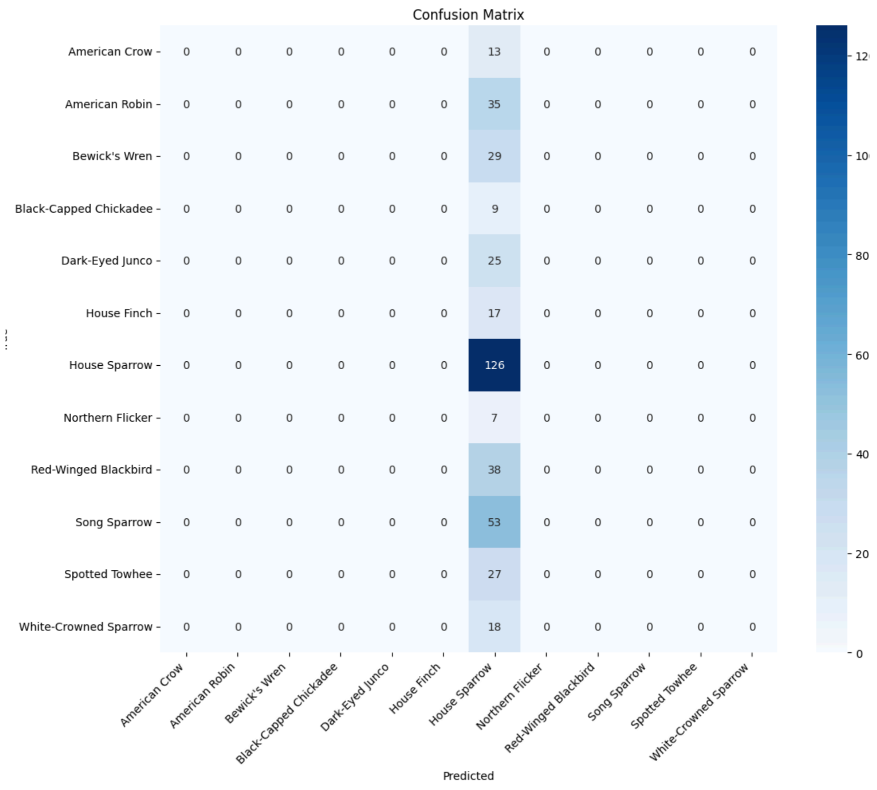


Figure 3 (above) shows the confusion matrix of multiclass model 1

Below is the confusion matrix for the more complex model. We can see the nice diagonal line that indicates our model was able to generalize well across all species. Again, due to the class imbalance, our House Sparrow had the most ideal results. In contrast, the species with less samples like Spotted Towhee and Bewick's Wren were somewhat mixed frequently since our model was not able to generalize too well. This model did do much better than the simple model, though.

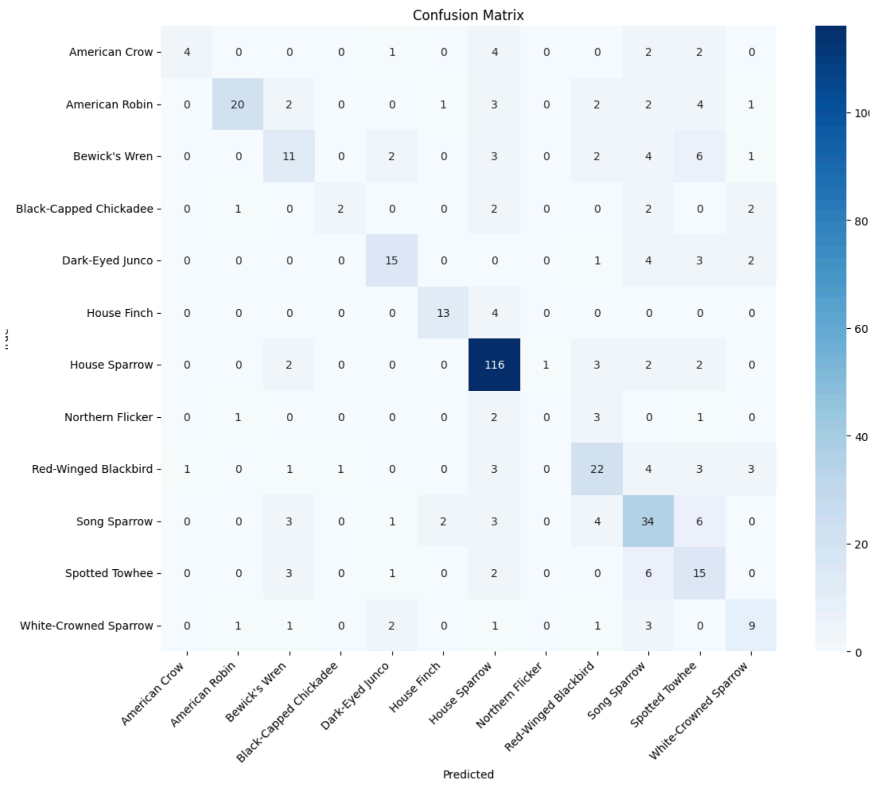


Figure 4 (above) shows the confusion matrix of multiclass model 3

One important thing to mention is that I did try to use class weights in the 4 layer model. However, because the results indicated the model did worse than random guessing, I decided to leave class weights aside to focus on tuning the hyperparameters to get a solid baseline model, which could be modified with class weights in the future.

External Clip Predictions

Below is the table containing the predictions of the complex and tuned multi-class model. Interestingly enough, the model did not classify any predictions to be House Sparrow despite the obvious class imbalance we had when building the model. The model was able to generalize fairly well since we have some variety in the predictions. Specifically, we can see that the model had a bias towards Red-Winged Blackbird and possibly Song Sparrow. American Robin and Black-Capped Chickadee were also solid contenders. It does seem very likely that the third audio sample has multiple birds. It also seems somewhat possible that the first audio sample has multiple birds as well.

Unknown Spectrogram	1st Prediction	2nd Prediction	3rd Prediction	Multiple Birds?
1	Red-Winged Blackbird - 52.28%	Song Sparrow - 20.45%	American Robin - 8.88%	Possibly
2	Red-Winged Blackbird - 39.43%	American Robin - 14.12%	Black-Capped Chickadee - 13.47%	No
3	Red-Winged Blackbird - 45.33%	Song Sparrow - 43.59%	Black-Capped Chickadee - 6.51%	Yes

Discussion

In both the binary and the multi-class classification models, the more complex and tuned models fortunately proved to yield better results. The binary models did have relatively better performance than the multi-class models, showing the challenges of trying to classify multiple species.

Binary Classification (Song Sparrow and House Sparrow)

The binary classification models showed the benefits of a deeper architecture and meaningful regularization techniques. The simplest, boilerplate model was able to achieve 72% accuracy and 0.65 AUC suggesting that even basic CNNs can extract meaningful insights from the spectrograms. However, the simple model did show clear bias towards the majority class (House Sparrow) with the high precision and recall. There was also some overfitting seen with the increasing gap between the training and validation accuracy, which confirmed the need for more complex tuning of the model's parameters.

Once we tuned the model to be deeper and more complex by scaling dropouts, padding, label smoothing, and adjusting the learning rate, it led to a more robust model. The deeper model was able to achieve 86% accuracy with an AUC of 0.9 plus much better class balance between the two species. The improvement in F1-score and the little to no overfitting in the training curves indicate that the model generalized well even with some imbalance in the data. Although the deeper and more complex model did take longer to train, the tradeoff was worth it since we ended up much better results.

Multi-class Classification (All 12 species)

The multi-class classification models struggled relative to the binary models. The overall sample size of the dataset (a couple thousand samples) proved to be inefficient in providing strong models, especially across a large number of classes with large class imbalances. House Sparrow had 17 times more samples than Northern Flicker. Despite the complications of the class imbalance and the small sample size, the deeper model did relatively well. The deeper model was able to predict and classify across all species and not just the majority ones. Both models also did better than random guessing, which is great. It shows the models were able to at least learn something rather than nothing.

External Audio Clip Predictions

Predictions on the 3 external audio clips using the deeper multi-class model gave varying results. Although our model did lean heavily towards the Red-Winged Blackbird for all 3 predictions, the 2nd and 3rd ranked predictions showed some variety, which is promising. Again, while the model did learn something and was able to generalize between the species, it was unable to generalize well enough, especially with the minority class species, to make more interesting (possibly accurate) predictions. With more tuning and a larger, more balanced dataset, the models might be able to distinguish between the classes more effectively.

Limitations

A limitation that I ran into was with hardware. Because I was using visual studio code on a 2019 macbook, it took a long time to train the models and some variations would not end so I had to use less aggressive parameters. The results in the above sections took 10-45 minutes at most, but the ones that I did not use or include ran for over 3 hours without stopping even with early stopping.

A very clear limitation was also the size of the dataset and the imbalance as well. The dataset had a total of 1981 samples across 12 species, and the samples for each species ranged from 37-630. So, not only did we have only a few samples to begin with, but we also had a large class imbalance among a relatively large number of species.

Species-Specific Challenges

For binary classification, it was not too challenging to predict the minority class Song Sparrow. There was at least some decent balance in precision and recall for both classes.

For multi-class classification, it was much harder to predict than binary classification since we had a lot more classes to predict. Although the deeper model was able to do decent on the test samples, it did seem to mix up the minority classes somewhat. For example, the Spotted Towhee and the Bewick's Wren were mixed up on occasion.

The predictions did have some variety, which indicates the model did learn some patterns, but it was not enough to make more unique predictions. It is very unlikely the 3 audio samples were all Red-Winged Blackbird given they all sound different just from listening to them.

Regarding whether there were multiple species, the model indicated the third test sample had multiple species. Given the model predicted two species with similarly high probability indicates that it possibly identified both. Whereas the predictions on the first audio sample indicates the model identified one with some uncertainty on which species it identified. It could also be the case that if multiple species were in the audio samples, the more distinct, loud, or higher frequency call would overshadow the other.

Alternatives

There are some alternative methods we could have used to approach this project. One alternative we could have tried was using pre-trained audio models like YAMNet which is trained on large audio datasets and applied transfer learning to fit our needs. Another possible approach could have been to use Recurrent Neural Networks (RNNs) or even LSTMs. Although RNNs and LSTMs could be a little slower to train and also require a lot of data to train on, they might have been able to capture trends and patterns in the temporal aspects of the image spectrograms we had. Another possibility could have been to use SVMs or Random Forests using spectral contrast to compute numerical values that the SVM or Random Forest can then use as features. Regardless, CNNs seems to be the logical choice since our data is image spectrograms with species specific traits, which CNNs are great at detecting and learning especially when spatial locality matters like in our case. Also, CNNs are more computationally efficient than the alternative mentioned above, so it had a good balance between performance and computational runtime compared to the alternatives. It is possible one of the other methods could prove to be more effective than the CNNs, but who knows, theory can differ from practices at times.

Conclusion

This project showed the feasibility and limitations of using CNNs to classify bird species from audio spectrograms. The binary classification models showed that effectively regularized and deep models can perform well even with class imbalance and encourages the use of CNNs in ecological monitoring applications where data for target species is abundant.

At the same time, the multi-class classification models showed that the limitations in data quality and quantity can also limit the performance of the model itself.

These results suggest that for real-world applications using deep learning:

- Balanced and abundant data is essential
- Decent hardware is also beneficial

Despite the hardware and data limitations, this project offers a valuable baseline for future work. It reiterates the importance of data focused model development and shows how even small changes in architecture or preprocessing can lead to significant performance differences.

Works Cited

1. Rao, Rohan. *Xeno-Canto Bird Recordings Extended (A–M)*. Kaggle, 2024, https://www.kaggle.com/datasets/rohanrao/xeno-canto-bird-recordings-extended-a-m.

2. Mendible, Ariana. *Bird Spectrograms Dataset*. 2025. GitHub, https://github.com/mendible/5322/blob/main/Homework%203/bird_spectrograms.hdf5.

3. Mendible, Ariana. *Test Bird Audio Clips*. 2025. GitHub, https://github.com/mendible/5322/tree/51bd4705bf06d4f46e1848f9261da9b2db3333c0/Homework%203/test_birds.

Appendix

Species Code	Common Name
amecro	American Crow
amerob	American Robin
bewwre	Bewick’s Wren
bkcchi	Black-Capped Chickadee
daejun	Dark-Eyed Junco
housfin	House Finch
houspa	House Sparrow
norfli	Northern Flicker
rewbla	Red-Winged Blackbird
sonspa	Song Sparrow
spotow	Spotted Towhee
whcspa	White-Crowned Sparrow