Classification of Bird Calls with 3 Types of Neural Networks

Abstract

This project uses three types of neural networks (feed forward, convolutional, and recurrent) to classify bird calls from the surrounding Seattle area. Neural networks will address these three tasks: (1) perform binary classification between two bird call classes, (2) perform multiclass classification between twelve different bird call classes, and (3) classification of three .mp3 sound files. The models performed slightly better than random guessing, but later discussion will show possible improvements to the models, namely decreasing the learning rate.

Introduction

Neural networks are an especially helpful tool for performing classification on objects not so clearly delineated along one feature or another. For example, when distinguishing between species of animals, there might not be a clear relationship between one feature, wing span for example, and the type of species. Instead, one might use some large combination of wing span and other features. To further motivate the worry, this linear combination of variables might not even be able produce anything like a clear linear boundary in the data. To work around this linearity, neural networks introduce non-linearity in the form of activation layers. This project will classify bird calls using neural networks.

The website, xeno-canto, is dedicated to extracting sounds from wildlife, many of which are stored as .mp3 files. The neural networks will be trained on bird calls from the Seattle area. What makes the data particularly difficult to work with is its unstructured nature. Namely, the sound clips of the birds are in an audio format, as opposed to a tabular structure that a .csv file would have.

The data comprise 12 species.

Variable Name	Bird Species
amecro	American Crow
amerob	American Robin
bewwre	Bewick’s Wren
bkcchi	Black-capped Chickadee
daejun	Dark-eyed Junco

houfun	House Finch
houspa	House Sparrow
norfli	Northern Flicker
rewbla	Red-winged Blackbird
sonspa	Song Sparrow
spotow	Spotted Towhee
whcspa	White-Crowed Sparrow

The number of samples vary widely, from 37 to 630.

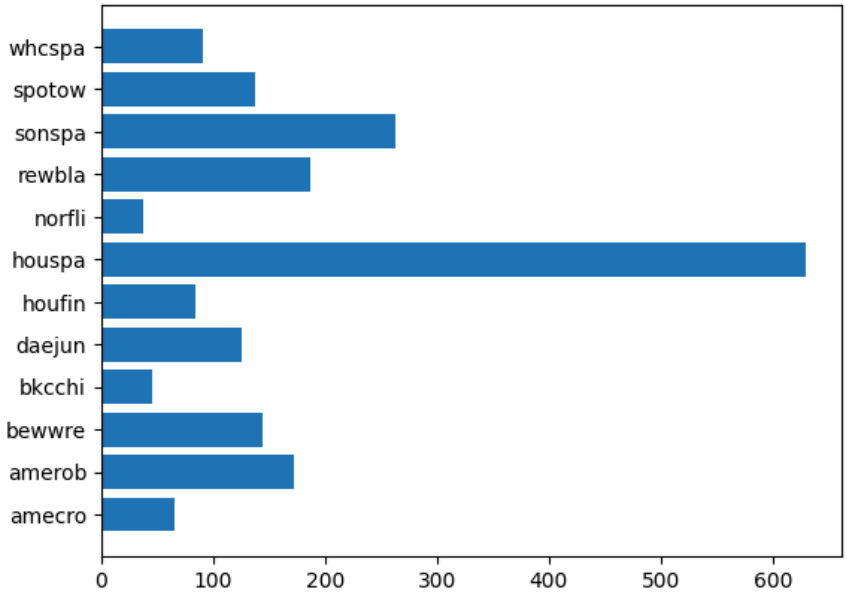

Figure 1. Histogram of counts of each species

House sparrows are rather common whereas the northern flicker is rather uncommon. This imbalance in the data will be troublesome for classification efforts and will be discussed later in the methodology section.

While xeno-canto records the sounds in the form of an .mp3 file, the project will be making use of a spectrogram, namely a mel-spectrogram. A spectrogram is a visual representation of the frequency of the sound over time. A spectrogram object is analogous to a picture or a two dimensional graph, with time on the x-axis and frequency on the y-axis. The librosa package allows the following visualization.

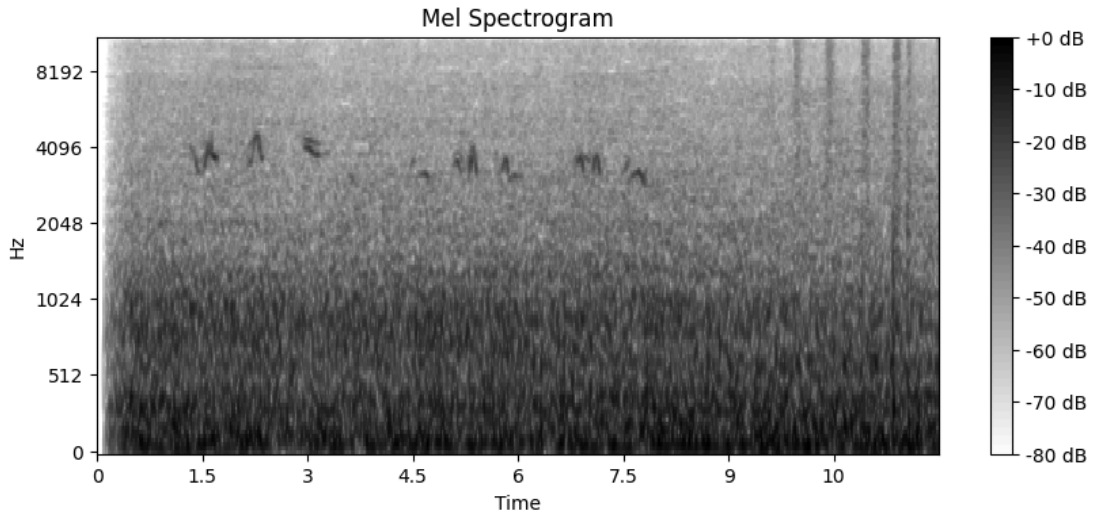

Figure 2. Spectrogram of a bird call

Hz is plotted along with the time. Then, given a time and frequency, the plot provides the decibels, or intensity of the sound. The data has already been scaled by the librosa package, so the time is just a 3 second snippet. The process of converting the sound file into a spectrogram involves the following:

1. The first 3 seconds are extracted from the clip.
2. The clip is then filtered to take frequencies from 0 Hz to 22050 Hz.
3. A spectrogram is made for the 3 second clip

Because the neural network is trained on 3 second increment spectrograms, the models will only take in 3 second increments. This will affect how the model is being run. Namely, making a prediction of a clip longer than 3 seconds will require the clip to be broken up into 3 second chunks.

Theoretical Background

Neural networks are a type of model that uses a combination of activation functions to model the data. As James et al. describe, a neural network takes several inputs, applies non-linear transformations to them, and then aggregates the inputs by taking a linear combination of them for its output. Find their illustration below in their 2020 text (405).

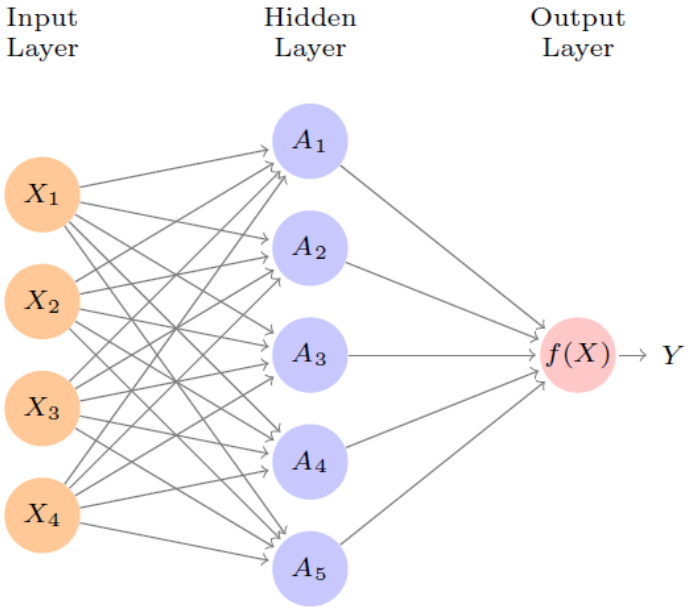

Figure 3. A simple neural network with 1 layer

Consider the above neural network as an example. The model takes in four inputs, each of the X terms. These could represent different samples being fed to the model. These inputs are then sent to hidden layers, containing activation functions. An activation function is just a function that adds some type of non-linearity to the data. This non-linearity is what allows a neural network to potentially perform better than a linear model such as linear regression. Examples of such activation functions are the ReLu function and the sigmoid function. The ReLu function takes in a number and outputs either that number, if it is greater than 0, or 0, if that number is less than or equal to zero. This is good at introducing non-linearity to a function by potentially breaking it to many piecewise functions.

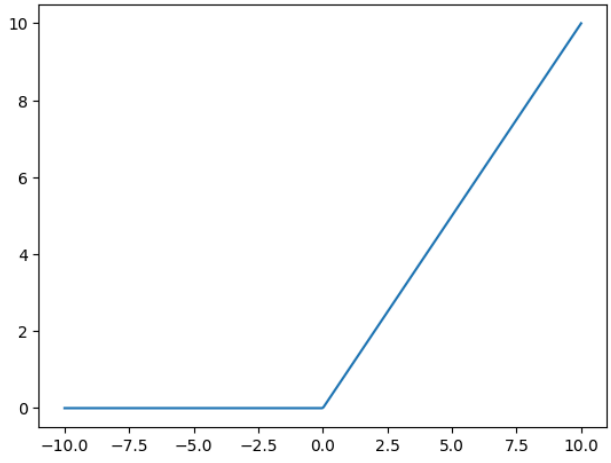

Figure 4. An illustration of the ReLu function

The sigmoid function, on the other hand, takes a number and sends it to a number that is extremely close to 1 or 0, essentially converting a numerical result into a categorical result.

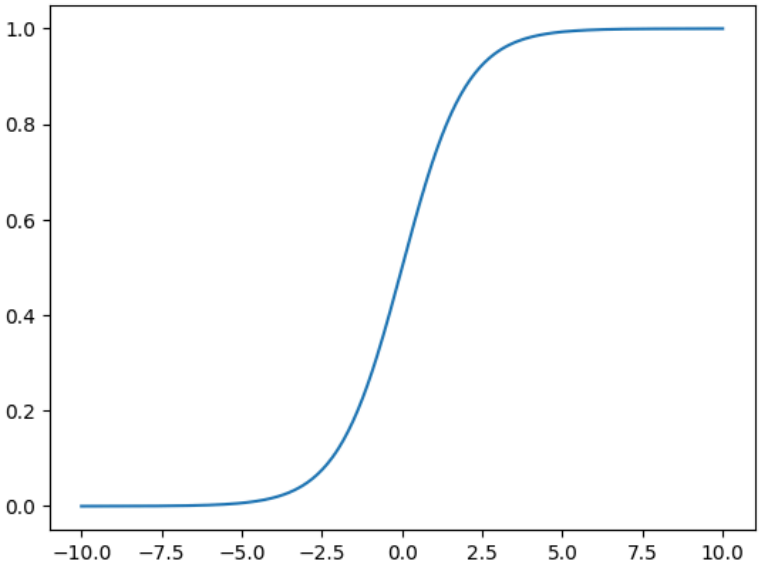

Figure 5. An illustration of the sigmoid function

A linear combination of these activation functions helps create a complex and flexible model.

Classification and Encoding

It seems that a linear combination, however, would intuitively be numeric. While such a result might not be problematic for binary classification, since the sigmoid function can help the network predict between the two classes with 1s and 0s, multiclass classification seems a little more difficult. The neural network will make use of one-hot encoding, which allows one to convert an object like a string into a vector of numbers. This converts the output data, namely bird titles, into numerical arrays to be decoded into labels later.

Figure 6. One hot coding example for a dictionary of 4 terms

Feed Forward Neural Networks:

A feed forward neural network has already been described as above. However, there are several tools and parameters worth describing.

Epoch:

An epoch can be described as a cycle of training. What a neural network does, essentially, is perform training cycles multiple times, attempting to decrease the loss function(accuracy in our case). After each epoch, the weights and biases of the model are adjusted to better minimize the loss function[1]. Intuitively, one might think of additional epochs as being strictly better. However, there are some drawbacks to additional epochs. First is the computational cost. It takes either time or computational resources to train for longer periods of time. Second, however, is the risk of overfitting. While a neural network does make use of a type of cross validation when training its model, if the data are sufficiently small enough, the model might end up fitting for that noise even if it has not seen it in the training set. However, lowering the learning rate can help prevent overfitting.

Learning Rate:

As mentioned with epochs, the model is constantly updating its weights and biases in each layer. However, one potential issue is that the model might overfit, altering its weights and biases too much and not decreasing the loss. This can be addressed by setting a smaller learning rate. The learning rate could be understood as the step size that the model takes when it updates its parameters (the weights and biases). By lowering the step size, the model will avoid making changes that are too drastic, therefore preventing the accuracy from oscillating.

[1] The weights and biases mentioned here could be understood as the slope and y-intercept, respectively, of our linear combination when we take the activation function. The activation function is non-linear, but the linear transformation we apply to our inputs between each layer involves addition and multiplication by scalars and vectors of numbers.

Layer number:

The image given above describes a neural network with only one hidden layer. However, given that the advantage of a neural network is that they have access to lots of complexity and non-linearity, it would be reasonable to try to increase the number of layers. The addition of more non-linear layers would also add more parameters (in the form of additional weights and biases) for the model to use. However, as with epochs, there is an issue with limited computational resources as more parameters would use up more memory in training.

Dropout Layers:

One layer that may be useful for the above models will be a dropout layer. Recall that neural networks, in comparison to linear models, are flexible. With this flexibility comes the risk of overfitting, where the data fits to the noise but not the more general structure of the data. On data that is sufficiently rich and large enough, overfitting is less of an issue. However, there are still risks to overfitting. One can use an additional layer, namely a dropout layer. A dropout layer, as the name implies, "drops out" some data on each iteration, or epoch. The model does not see it during training and does not fit to the noise of that data. When a dropout layer is used in the investigation, it is set to drop only 10% of the data.

Convolutional Neural Networks (CNNS):

Convolutional neural networks are a type of a neural network that involves changing, or *convolving*, the input objects. A convolution is like a filter, or layer, applied to the model that slightly alters or shifts the image. What this is meant to do is to, as James et al. describe, to highlight local features of different objects (414). This is because objects sent through a convolutional layer will still keep their local features. For example, a picture of a tiger sent through many features will still retain many important features, such as its teeth, nose, etc. Afterwards, the neural network condenses the object by making use of a pooling layer, which shrinks the input object. This will likely eliminate noise that does not contribute a relevant feature to the image. Convolutional neural networks average out noise by emphasizing local features.

Convolutional neural networks are relevant for this investigation. The example given above was of an image of a tiger. It seems that convolutional neural networks can be useful when classifying images. However, spectrograms are not fully analogous to pictures. However, a graph could be understood as a picture and can serve as an image for the purposes of the classification task because bird calls would likely have some level of structure. Therefore, a convolutional neural network would be able to plausibly capture some of these local features.

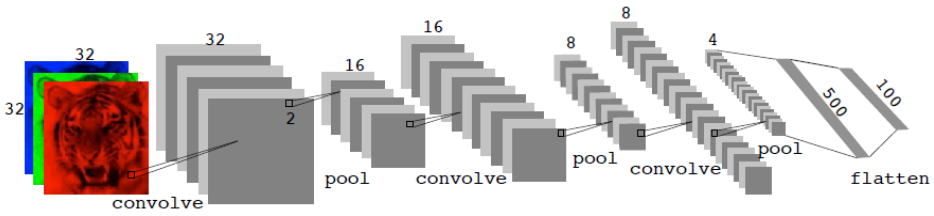

Figure 7. James et al.'s diagram of the layers in a convolutional neural network (416)

Recursive Neural Networks (RNNS):

A recursive neural network is another specialization of a neural network that is effective with data arranged chronologically. This neural network makes use of previous features and accommodates for the temporal nature of the data. By arranging the features timewise, the neural network can be "aware", in a sense, that the objects are sequential, in a similar way to how the words in a sentence are sequenced.

One particular benefit, though potentially a drawback, for a recurrent neural network, is weight sharing. Weight sharing is when the same object, but across different time slices, will have the same weights and biases as an object from earlier time slices. This is beneficial insofar as it lets the model train while keeping in mind that time should not drastically change the classification of an object. For example, a bird call at the start of a sound clip and a bird call at the end of a sound clip should plausibly be categorized as the same bird call[2]. The bird calls are in a sequence (time), so a recurrent neural network seems like a promising candidate for the model.

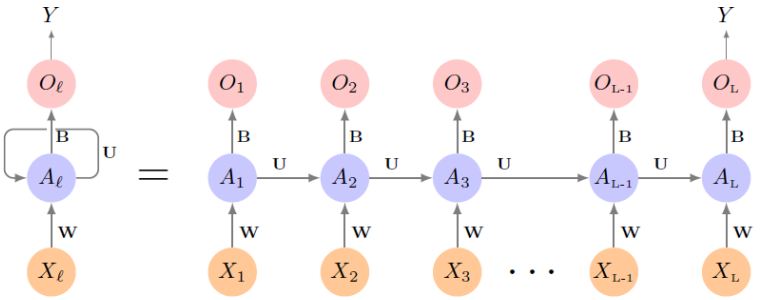

Figure. 8 James et al.'s diagram of a recurrent neural network (422)

[2] Shared weights might be an issue for objects that are significantly affected by time, namely stock market prices over decades. It is plausible that our weights would in fact be changed over time.

Methodology

Data Imbalancing

Recall earlier that the data was particularly unbalanced, because of the large number of house sparrows. If all of the data were used for training, a neural network would likely have a potentially 50% accuracy rate classifying house sparrows, just because about half of the data was house sparrows.

A straightforward, though rather coarse, solution to imbalanced data is to sample only as much data as the minimum amount of data points across all classes. In other words, since the Northern Flicker has the least samples with 37, the model will take 37 samples per class to prevent imbalancing the data. This just has the small issue that the data will be of a slightly lower quality.

Data Re-shaping

For a feed forward neural network, one should *flatten* the dimensional spectrogram matrix into a 1 dimensional vector, losing some structure. For convolutional and recurrent neural networks, the input matrix is converted to a 3 dimensional matrix, without losing many features.

Learning Rate Fine Tuning

A high learning rate can lead to a great amount of oscillation in the model's accuracy. A prime example is the following graph of binary classification accuracy of one of the models. The left hand plot has a high learning rate while the right hand plot has a low learning rate. While both plots have validation accuracies that drastically change, the model with a lower learning rate has experienced much smaller oscillations in accuracy compared to the model with higher learning rate. One can fine tune the learning rate such that the variance in the accuracy decreases. It is true that a lower learning rate could lead to a potentially higher validation accuracy, it seems that the model would be very inconsistent and its parameters would not converge. We make use of Keras' RMS prop's learning rate variable to lower the learning rate to 0.0001 from 0.001. Combined with more epochs, the model could potentially converge.

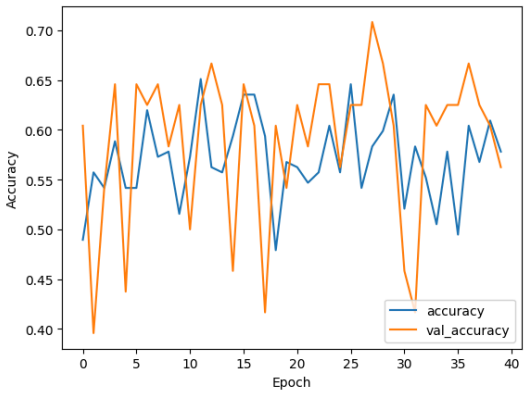

Figure 9. Accuracy without fine tuning

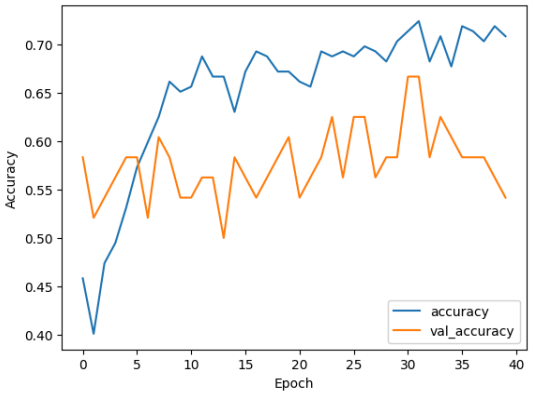

Figure 10. Accuracy with learning rate fine tuning

Epoch Fine Tuning

Epochs were altered to allow convergence of a model. The number of epochs would vary between 10 and 40. However, as can be noticed in figures 9 and 10 above, increasing the number of epochs does not necessarily guarantee convergence. Additionally, a low level of epochs might converge quickly. It appears that there is no strict relationship between increased epochs and improved convergence.

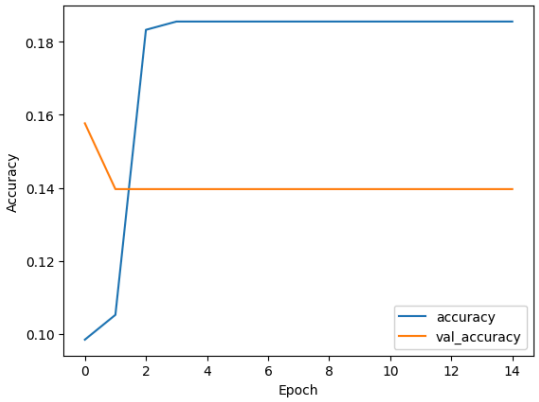

Figure 9. Low epoch amount that converges quickly

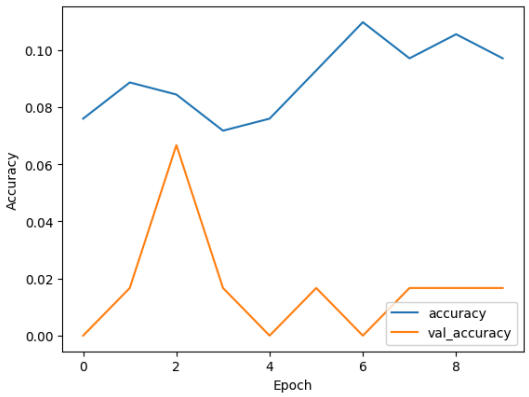

Figure 10. Low epoch amount that oscillates

Training Parameters:

While there are three main types of models, these models can differ greatly in their performance given the relevant hyper parameters. For the current investigation, learning rate and epochs will be altered for fine tuning. However, several other parameters were kept relatively fixed throughout:

1. Batch size: This parameter was kept around 16, since any higher would get fairly close to the size of the sample (37 in the case of multiclass classification) size of a species.

2. Loss function: Binary cross entropy, or log loss, is used for binary classification. Categorical cross entropy is used for multiclass classification.

Specific Model Architectures:

The feed forward neural network, CNN, and RNN used in the binary multiclass classification task all had a simple architecture that is noted for replicability.

1. Feed forward: This neural network uses a dense activation layer with a ReLu function, followed by a dropout layer of either 0.1 or 0.4, followed by another dense layer.
2. CNN: This neural network uses a convolution layer, followed by a pooling layer, followed by another convolution layer, followed by another pooling layer, followed by a dropout function of either 0.1 or 0.4, followed by two dense layers for flattening.
3. RNN: This network uses a simple RNN layer followed by a dense activation layer with a ReLu function, followed by a dropout layer, followed by another dense layer.

Results

Binary Classification:

For the binary classification task, the model classified the American Robin and Red-winged Blackbird. This was because these datasets were of similar size, having about 130 samples each.

Neural Network	Test Accuracy
Feed forward	46.22%
Feed forward (lowered dropout rate)	53.78%
Convolutional	53.78%
Convolutional (increased epochs, lowered layers, removed dropout layer)	59.66%
Recurrent	45.38%
Recurrent (decreased learning rate)	58.98%

These results are not at all impressive. However, a strength of these models is that they did not severely overfit, as evidenced by the following history. Increased epochs at least did not lead to the test accuracy decreasing.

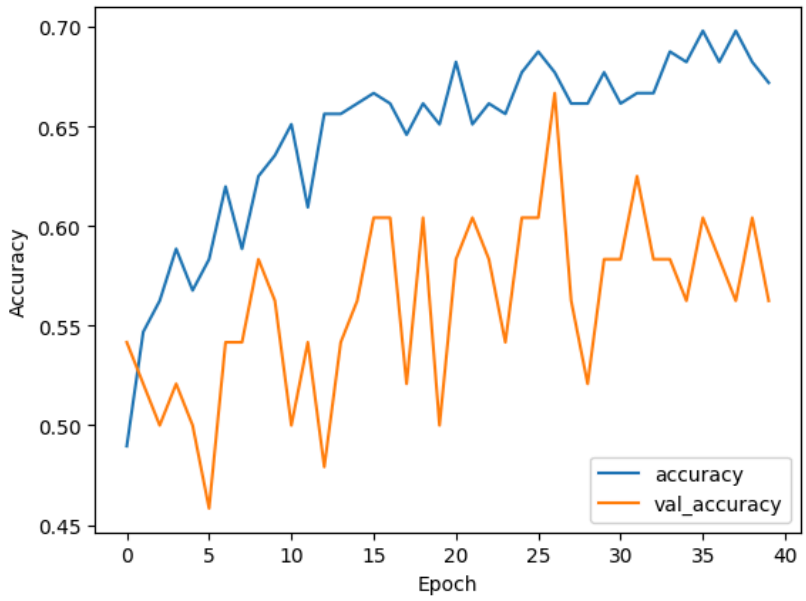

Figure 13. RNN accuracy with epoch increase

It seems that the specialized models, with hyperparameter finetuning, have done rather poorly for the binary classification task. It appears that the decreased dropout rate has improved the accuracy simply due to the low amount of data.

The following are confusion matrices for the binary classification.

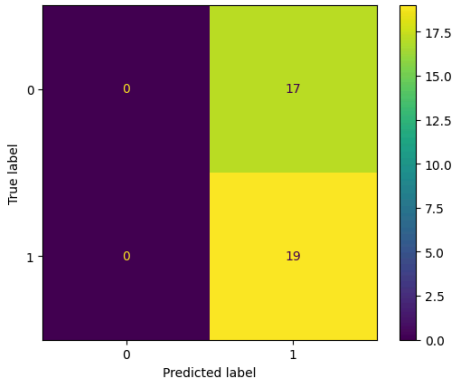

Figure 14: Feed forward neural network binary classification confusion matrix

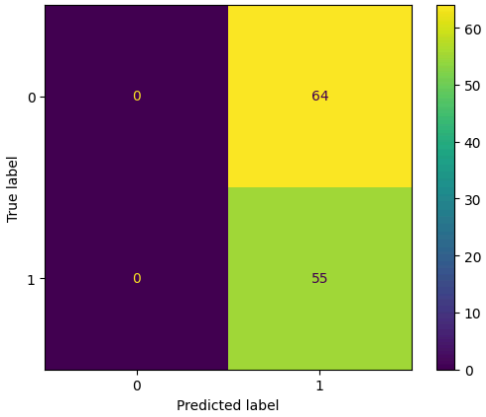

Figure 15. CNN binary classification confusion matrix

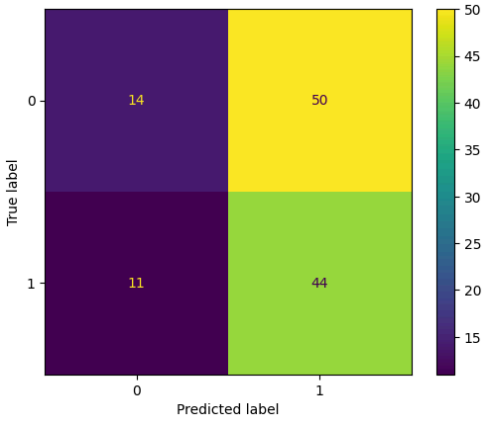

Figure 16. RNN binary classification confusion matrix

The feed forward and CNN models classified all the samples as a single species. Only the RNN classified the test set into two sets. However, even the RNN classification had a low accuracy. The feed forward neural network and CNN likely had such misclassification due to data imbalance leading to bias. The RNN had poor classification results likely due to a lack of complexity in the model.

Multiclass Classification:

Neural Network	Test Accuracy
Feed forward	7.48%
Convolutional	9.52%
Recurrent	12.93%

It appears that the recurrent neural network performed the best, with a non-trivial margin above the other methods. However, this accuracy is still low, because the model is only doing slightly better than random guessing, and the convolutional neural network and feed forward neural network have performed almost as badly as random guessing. However, this is likely due to data imbalance.

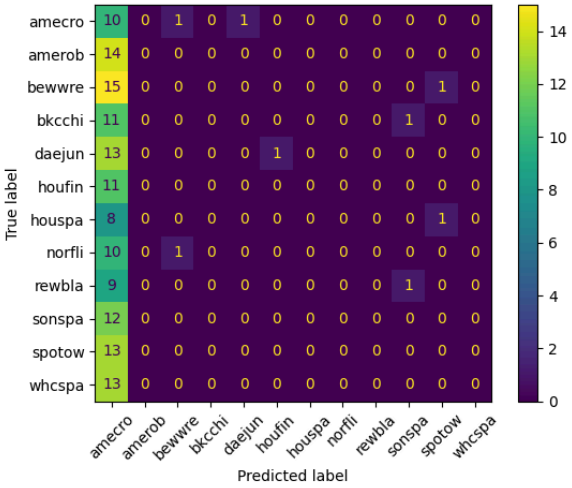

Figure 17. Feed forward multiclass classification confusion matrix

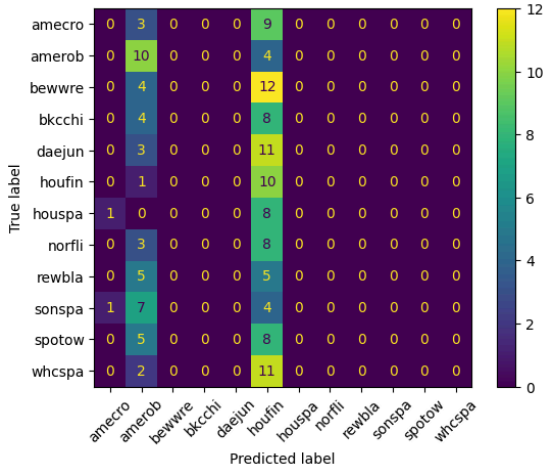

Figure 18. CNN multiclass classification confusion matrix

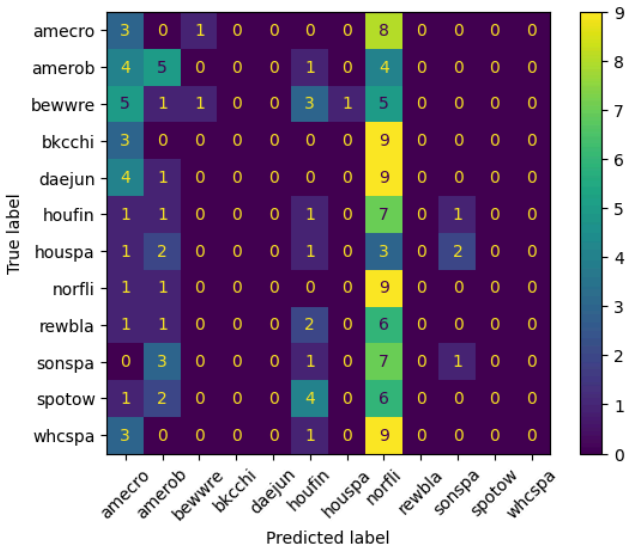

Figure 19. RNN multiclass classification confusion matrix

The above 3 confusion matrices confirm that there is a large amount of bias in the data due to imbalance. If there were no issues with imbalance, the data would be more evenly spread out.

Classification Task for Random Test Files:

The best performing model, namely the RNN, will be used to classify three sample clips. One additional complication is that the clips are not simply 3 seconds long and may feature several birds. Therefore, the clip will be broken down into 3 second increments and the majority will determine the bird type.

The following table describes the results for the first test clip.

Time Interval (seconds)	Classification 1st Choice	2nd Choice	3rd Choice
0 to 3	Northern Flicker	American Crow	House Sparrow
3 to 6	Northern Flicker	American Crow	House Sparrow
6 to 9	Northern Flicker	American Crow	House Sparrow
9 to 12	Northern Flicker	American Crow	House Sparrow
12 to 15	Northern Flicker	American Crow	House Sparrow
15 to 18	Northern Flicker	American Crow	House Sparrow
18 to 21	Northern Flicker	American Crow	House Sparrow

The RNN categorizes all increments as the Northern Flicker, followed by the American Crow and House Sparrow. This is unsurprising because the model, from the above confusion matrices, is biased towards the Northern Flicker.

The following table describes the results for the second test clip. Because the clip was only 5 seconds, the first and last 3 second intervals were used.

Time Interval (seconds)	Classification 1st Choice	2nd Choice	3rd Choice
0 to 3	Northern Flicker	American Crow	House Sparrow
2 to 5	Northern Flicker	American Crow	House Sparrow

The RNN still categorizes all increments as the Northern Flicker, the same as the first test clip. Once again, this is likely due to the bias of the model as mentioned earlier.

The following table describes the results for the third test clip.

Time Interval (seconds)	Classification 1st Choice	2nd Choice	3rd Choice
0 to 3	Northern Flicker	American Crow	House Sparrow
3 to 6	American Crow	Black-Capped Chickadee	House Sparrow
6 to 9	American Crow	Black-Capped Chickadee	House Sparrow
9 to 12	Northern Flicker	American Crow	House Sparrow
12 to 15	Northern Flicker	American Crow	House Sparrow

This is the first instance where the model provided a classification that was not the Northern Flicker as the first choice. However, the Northern Flicker still remained the majority choice, so the model predicted Northern Flicker for all three test cases.

Discussion

Comparison of Different Architectures:

While all models performed rather poorly, the CNN and RNN performed slightly better than the standard feed forward neural network. This is likely because flattening the training

matrices took away useful structure from the input data. The CNN and RNN models were likely able to make use of this additional structure.

It is also likely that some positive features of CNNs and RNNs enabled them to perform well. For example, both types of models have a type of weight sharing. It is potentially this weight sharing that allowed the models to perform better than the feed forward neural network. What likely allowed the RNN to perform even better than the CNN was that the RNN was time focused, potentially allowing it to capture the temporal structure of the bird calls.

In multiclass classification, the RNN model also performs better in terms of avoiding overly biased results. For example, predictions were spread across 4 species as opposed to only 2, in the case of the CNN, and 1, in the case of the feed forward neural network. The ability of the RNN to make use of prior states likely enables it to have a richer representation of the data.

Limitations:

It appears that, with the high level of oscillation in our accuracy, that the models could have served to be improved. However, there were a few limitations to our methods. The most important restriction here is computational power. On several instances, the models, in particular the convolutional neural network, did not have enough memory to continue running. To compensate for that loss of memory, the batch size was decreased and the number of layers to lower the amount of parameters. That computational gain likely leads to a loss in accuracy.

Consider the following trainable parameters for two models:

Model	Parameters
Binary CNN	23,626
Multiclass CNN	16,975,756

Noticeably, the multiclass convolutional neural network was much larger than the binary convolutional neural network. When considering time as a resource, due to the small number of samples in the dataset, no training run took more than 10 minutes, given that there were at most 170 samples in the dataset. As above, this likely led to some losses in the model.

Improvements for Future Work:

There are several ways that the models could have performed better, given that many results were almost as bad as, if not worse than, random guessing. There are several ways that future models could improve and offer

1. Lack of layers: The lack of layers in the model prevented the neural networks from being complex enough. By adding more layers, the model would have more parameters, as mentioned above.
2. Undersampling: In the multi classification model, the data was severely undersampled, with some being as low as 37. This hurt the model by not having enough rich data.
3. Lack of scaling: The model was not scaled in the initial stages, which may have interfered with the weights being used in each layer of the neural networks. Future attempts should scale the data in the preprocessing stages.
4. Use of sklearn's train/test split: To attempt cross validation, a train/test split is used to separate training and test data. However, this exacerbates the issue of undersampling. Not only is the data further unsampled, but due to the lower number of samples per species, the data could be heavily unbalanced as well. One should consider avoiding a train/test split in future attempts, because the neural network already performs a validation test during training.

Predictions Regarding Different Species:

It appeared that models tended to be very biased in all cases. This can be seen especially in the case of the feed forward neural network, which almost only classified birds as American Crows. The CNN almost only classified birds as House Finches or American Robins. The RNN performed the best in terms of avoiding bias by splitting its predictions between 4 different species, the American Crow, American Robin, Northern Flicker, and House Finch. What this might imply is that the model grouped distinct birds together. Therefore, House Finches might be distinct from American Robins, leading them to be categorized separately and the model would then group birds as either House Finches or American Robins, depending on which was similar to the other.

To determine which birds were primarily misclassified as each other, it might also be worth considering the results of the tests performed. Using the RNN, it seems that the Northern Flicker, American Crow, and House Sparrow are all categorized as each other. There are two explanations. The first is that the bias of the model led them to have high probabilities. The second is that the species are quite similar.

While many calls were misclassified, some might have been excluded entirely due to their pitch. Recall that preprocessing began with taking frequencies between 0 and 8192 Hz. However, this seems very restrictive. After all, it is quite plausible that many birds could make sounds above 8000 Hz. In particular, a quick search reveals that the Bewick's Wren makes sound at a high frequency, according to the Cornell Lab of Ornithology. One should then consider allowing higher pitches to fully capture the range of some of these birds.

Considerations of Other Models:

While the current discussion primarily involves neural networks, it is worth considering other models as well, such as linear decision trees, linear regression by least squares, or support vector machines. The primary issue with applying these models will just be how the data are formatted. The models mentioned above take in data in nice tabular form. The only way to simplify the data might be to flatten each of the matrices, but this would still yield an unwieldy amount of data. The reason neural networks are used is because the data are too complicated.

An initial way one might consider using other models is to aggregate several slices of the data so that one can take the average decibels per time chunk and per frequency chunk. However, this would greatly diminish the quality of the data. It is also unclear how such transformations would preserve any structure in the data. Classification would be done on the average frequency of bird calls.

Conclusion

There is much work to complete when it comes to classifying birds. While the models only performed slightly better than random guessing, there remain many hyperparameters to fine-tune. In particular, tuning the learning rate afforded much less oscillation in model accuracy. Additionally, it is worth considering increasing the training size set to strengthen the models as well. With this data, one could potentially train models that, like many birdwatching apps, can be used to identify birds by their call, as opposed to some subjective features.

References

[1]Cornell Lab of Ornithology. (n.d.). Bewick's Wren sounds. All About Birds. Retrieved May 12, 2025, from https://www.allaboutbirds.org/guide/Bewicks_Wren/sounds

[2] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (1st ed.) [PDF]. Springer.

[3] skadaver (https://stats.stackexchange.com/users/254303/skadaver), Cross Entropy vs. Sparse Cross Entropy: When to use one over the other, URL (version: 2020-02-03): https://stats.stackexchange.com/q/420730

[4] Stack Abuse. (2023, July 19). Don't Use Flatten: Global Pooling for CNNs with TensorFlow and Keras. https://stackabuse.com/dont-use-flatten-global-pooling-for-cnns-with-tensorflow-and-keras/