Aishwarya Saibewar
DATA 5322
May 11, 2023

Predicting bird species based on their sound using Neural Networks

Abstract

In today's world, many new species of birds are being discovered and many more species are on the verge of becoming extinct. The issue that arises is to identify and differentiate the species so that we can classify them appropriately and take necessary measures to preserve the species. This demonstrates the necessity of developing an algorithm that can accurately predict bird species from their sounds. This research uses data from Xento-Canto to perform analysis and examine the accuracy of Neural Networks in predicting bird species based on their sound. Two models were developed to make predictions. The first model using a Convolution Neural Network (CNN) to perform binary classification of two species has predicted the bird species with a test accuracy of 96.9%. The second model using CNN to perform multi-class classification of 12 species has predicted the bird species with a test accuracy of 68.3%.

Introduction

It is becoming increasingly significant to monitor bird species as they are considered critical components of biodiversity and ecology. To accurately recognize and classify a bird into the right species would require an image or audio file. In this paper, we will be using data from Xento-Canto for developing models to accurately predict the bird species based on their sound using neural networks. Xento-Canto is a repository for recordings of bird sounds, which allows researchers or bird enthusiasts to upload and share recordings of wildlife sounds from all over the world. This publicly accessible website has collected 575,000 sound recordings of over 10,000 different bird species worldwide. This analysis would help researchers identify and prioritize species on the verge of extinction when developing conservation strategies.

For this study, we are utilizing Xento Canto data containing recordings of 12 bird species. The recordings of each species are in mp3 format. They were converted into spectrograms to feed as input to the neural network. The data of all 12 species of birds were considered for multiclass classification, while the data of two species of birds, the amecro and the barswa, were considered for binary classification.

Theoretical Background

The Neural Network is a machine learning algorithm that was developed with the intention of attempting to imitate the human brain's function. The primary goal of this kind of machine learning algorithm is to identify patterns in the data and make predictions. The neural network uses neurons to transmit information. Building relationships between multiple layers, such as an input layer, one or more hidden layers, and an output layer, is how the fundamental logic of identifying patterns in the data is carried out.

A neural network with just one hidden layer between the input and output layers is known as a single-layer neural network. The linear combinations of observations are fed through the input layer and are then passed to the hidden layer. Every neuron applies a non-linear activation function to this weighted sum and passes it as an input to the output layer. The output produced is a linear combination of activation functions. A neural network with more than one hidden layer between the input and output layers is known as a multi-layer neural network. It is a popular kind of neural network that can be used for image and speech recognition.

The model's performance on the training dataset and the type of predictions it can make are both influenced by the activation function chosen in the hidden and output layers. Some of the commonly used activation functions to introduce non-linearity into the output of a neuron are ReLU, Sigmoid, and Softmax. ReLU (Rectified Linear Unit) is widely known for implementation in hidden layers and maps all the negative values to 0. Mathematical operations can be performed more quickly and easily because only a few neurons are activated. The sigmoid function is typically applied at the binary classification output layer (0 or 1). The Softmax function is commonly used in the output layer of multiclass classification to obtain the probabilities distributions for different classes. Additionally, it guarantees that the probability of all classes added together equals 1.

Convolution Neural Networks (CNN) are primarily used for image recognition and are designed to extract the desired features from the image data using convolution layers. In addition to the layers present in the multi-layer neural network, the CNN contains other layers such as the convolution layer, and pooling layer. The convolution layer extracts the most important features such as edges, and corners through the use of corresponding filters. The output of the convolution layer is a smaller image that features the areas where the input image matches the filter. By computing a filtered value that relates a group of pixels together, it is used to find meaningful patterns in the images. Therefore, the parameters that need to be specified in the convolution layer are the number of filters, kernel size, padding, activation function, and input shape. The Kernel size determines the spatial extent (number of pixels covered in each direction) of features and is usually a square matrix with odd dimensions such as 3x3, 5x5, or 7x7. The smaller kernel size is used to extract smaller features and the larger kernel is used to extract larger features. Padding is utilized for preserving the size of the input image by padding the border with zeros. The input shape parameter in the CNN represents the dimension of the input image.

A pooling layer can be added in order to reduce the size of the data while maintaining the important features. Average pooling selects the average values, whereas max pooling selects the maximum element from the region of the feature map. A flattening procedure would be required to transform this data into a 1D array for use in the subsequent layer. The output from the convolution layers is flattened to produce a single long feature vector that is connected to the final classification model via the dense layers. In order to prevent overfitting of the training data, dropout layers are added to regularize by defining the percentage of data that can be ignored in a given layer.

Once the layers are specified in the model, the model needs to be compiled. The parameters required for compiling the model are loss, optimizer, and metrics. The loss parameter is set to understand the model's ability to accurately predict output for a given input. The commonly used loss functions for regression problems are MAE (Mean Absolute Error) and MSE (Mean Square Error) and for classification problems are binary cross-entropy and categorical cross-entropy. An Optimizer is used to reduce overall loss and increase accuracy. Some of the commonly used optimizers are RMSProp, Stochastic Gradient Descent, Adagrad, and Adam. The metrics are specified to evaluate the performance of the model. The commonly used metrics for a regression problem are MSE, MAE, and R-Squared(R2), and for classification problems are accuracy.

In predicting the bird species based on their sound, the audio signals are converted into spectrograms. The spectrograms are the visual representation of audio signals at different frequencies over time. They are created by dividing the audio signals into frames (shorter time segments). The Fast Fourier Transformation (FFT) is then applied to the frames to get spectrograms. The most common application for these spectrograms is in speech recognition. These spectrograms are then used as input to the neural network model to predict and classify the sound type.

Methodology

Data Preparation

12 bird species from the Xento Canto data were considered for predicting the bird species based on sound. MP3 files were the format of this data. As it was easier to analyze the frequencies over time with spectrograms, these audio files were converted into them. The spectrogram data of 12 bird species as .dat files were then considered for predicting the bird species based on sound. A single array was created by combining the data from each species into a single stack of data. Labels were created for the spectrogram of each species because the model must be trained on labeled data in order to accurately predict and classify the species for the test data. To prepare the data for building a neural network, the labels (categorical data) were one-hot encoded. In addition, the data was reshaped to feed as an input to the Convolution Neural Network. The data already consisted of three dimensions the number of samples (n_samples), timesteps (343), and frequencies (256). To feed as an input to the Neural network the fourth dimension representing the number of channels was added. The number of channels for this data is 1 as the spectrogram is a greyscale image. Therefore, the dimension of the input data is n_samples x 343 x 256 x 1.

Models

First, the dataset was divided into training and testing, with 70% of the data used as a training dataset to train the models, and the remainder was reserved for testing purposes. Next, the labels were one-hot encoded and the input to the neural network had a dimension of n_samples x 343 x 256 x 1.

For binary classification, a convolution neural network was developed to predict the bird species based on sound. The two species considered for binary classification are “amecro” and “barswa”. The binary CNN model consists of two convolution layers with ReLU activation, two max-pooling layers, one drop-out layer, one flattening layer, and two dense layers. Because the model needs to predict between the two classes, the output layer only has one unit with a sigmoid activation function.

For multiclass classification, a convolution neural network was developed to predict the bird species based on sound. All 12 species were considered for performing the multiclass classification. The multiclass CNN model consists of two convolution layers with ReLU activation functions, two max-pooling layers, one drop-out layer, one flattening layer, and two dense layers. Because the model must predict between the 12 classes and provide a probability distribution for each class, the output layer has 12 units with a Softmax activation function.

To train the neural network in less time and in an efficient way, an Adam optimizer was used. The loss was binary cross-entropy for binary classification and categorical cross-entropy for multi-class classification and the performance metric was the accuracy of the model. After choosing a batch size and epoch for the model, the models were fitted to the training data. Additionally, validation data was assigned to 20% of the training data. Test data served as the basis for the predictions.

Computational Results

Binary Classification

A binary classification model was developed to classify the bird species as amecro or barswa based on their sound. A Convolution Neural Network was developed to perform this classification. As shown in Figure 1 the model's loss and accuracy are plotted against the increasing number of epochs for the training and validation dataset. The model has a batch size of 25 and converges at the seventh epoch and the accuracy of the training data increases with the number of epochs. And the loss of the training data tends to decrease as the number of epochs increases. The initial model without dropout layers has predicted with a test accuracy of 93.3%. To improve the accuracy of the model, a dropout layer that ignored 30% of the training data was added to the CNN model. As a result of this, the accuracy of the test data has slightly increased from 93.3% to 96.9%. Finally, the best model predicted the training data with an accuracy of 98%, validation data with an accuracy of 97%, and the test data with an accuracy of 96.9%. This model was able to classify the species more accurately because the sounds of the bird species amecro or barswa are quite different and the model was able to understand the sound patterns of the species and classify the new observations accordingly.

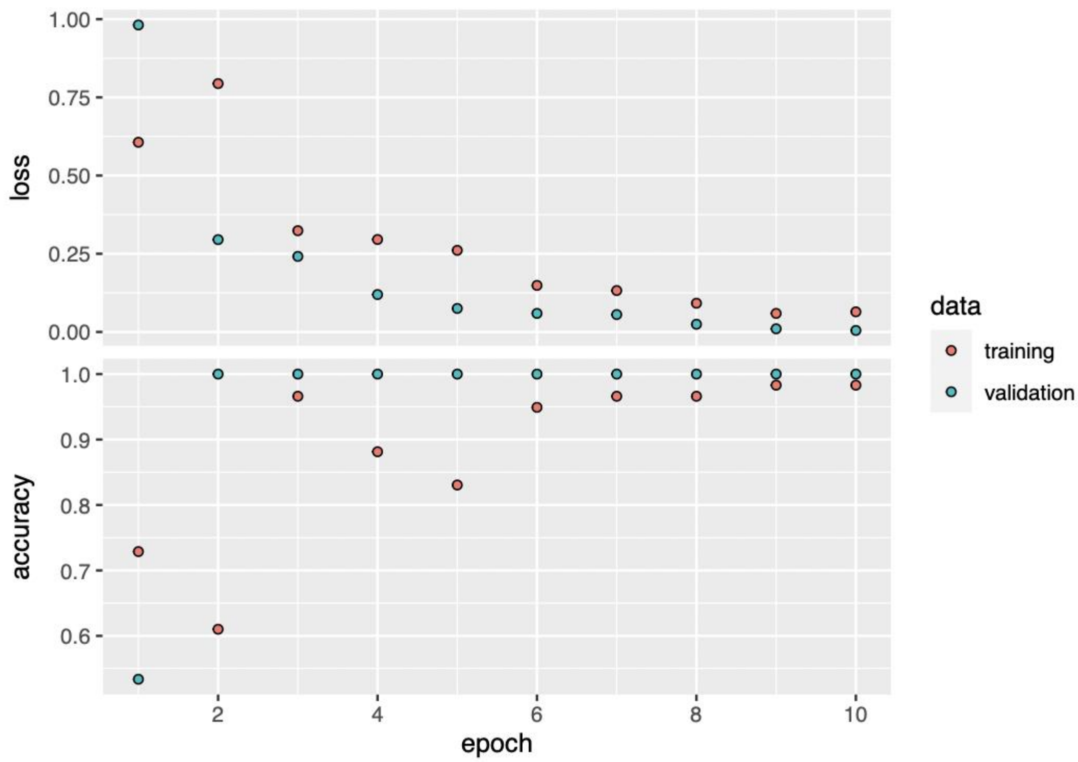


Figure 1 Performance Metrics for Binary Classification

Multiclass Classification

A multiclass classification model was developed to classify the 12 bird species based on their sound using a Convolution Neural Network. As shown in Figure 2 the model's loss and accuracy are plotted against the increasing number of epochs. As shown in Figure 2, the model's accuracy and loss are plotted against the training and validation datasets for the increasing number of epochs. It can be observed that the accuracy of the training data increases with the number of epochs and the model with a batch size of 25 converges at the 8th epoch. And the loss of the training data tends to decrease as the number of epochs increases. The initial model without dropout layers has predicted the training data, validation data, and test data with an accuracy of 90.8%, 70.1%, and 59.3% respectively. It can be seen that the model is overfitting the training data and is unable to generalize the patterns in test and validation datasets. To overcome this issue, an additional dropout layer that ignores 40% of the training data was added to the CNN model. This has resulted in decreasing the accuracy of training data from 90.8% to 82.8% and increasing the accuracy of test data from 59.3% to 68.3%.

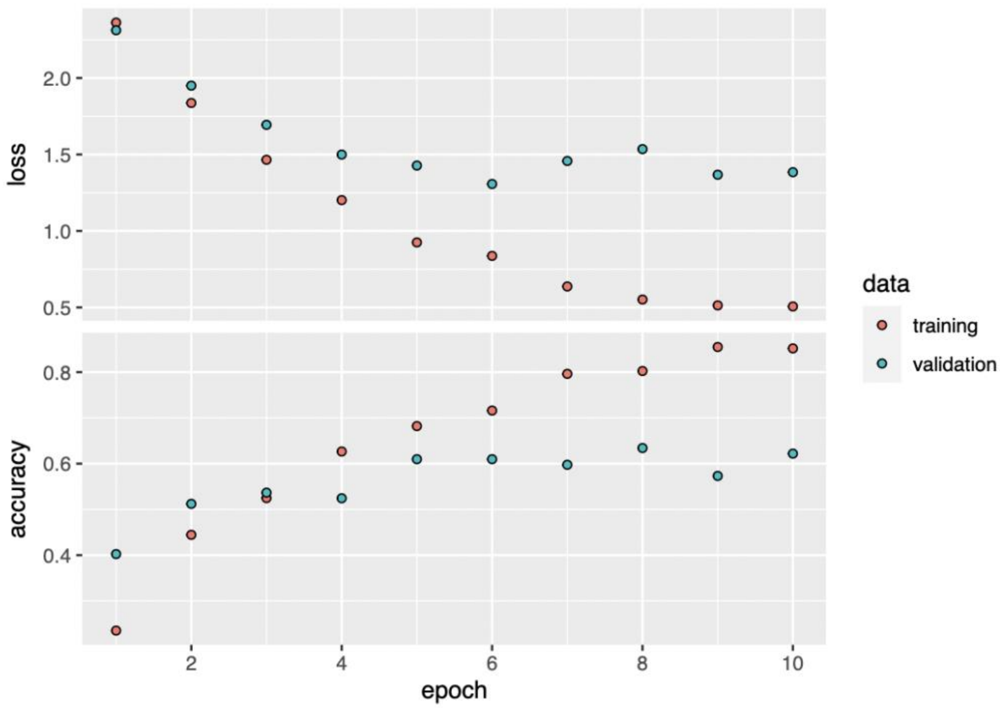


Figure 2 Performance Metrics for Multiclass Classification

The confusion matrix of the multiclass classification model is shown in Figure 3. The diagonal elements represent the correct classifications and the others represent the misclassifications. It very well may be seen from the confusion matrix that there are misclassifications of the observations. Some of the incorrect classifications include barswa being predicted as houfin and whcspa, bkcchi as barswa and daejun, and norfli as blujay and daejun. This could be due to the similarity of these bird species sounds. Additionally, the spectrograms used for the analysis included data on just the louder parts of the audio clip. Although important patterns of the species may be present in the clip's louder parts, some species that had their identity in the clip's softer parts may have been incorrectly classified.

	Actual											
Predicted	amecro	barswa	bkcchi	blujay	daejun	houfin	mallar3	norfli	rewbla	stejay	wesmea	whcspa
amecro	11	0	0	0	0	0	1	0	0	0	0	0
barswa	0	13	0	1	1	3	0	0	3	0	0	5
bkcchi	0	2	8	0	2	0	0	1	1	0	0	1
blujay	0	3	0	9	0	0	0	0	5	1	1	0
daejun	0	0	2	0	14	0	0	0	0	0	0	1
houfin	0	0	0	0	0	6	0	0	1	0	3	0
mallar3	1	0	0	0	0	1	12	0	0	0	0	0
norfli	0	0	1	4	3	1	0	16	0	0	2	0
rewbla	0	0	0	0	0	1	0	0	5	0	0	0
stejay	0	0	0	2	0	0	0	0	2	6	0	0
wesmea	0	0	0	0	0	0	0	0	0	0	5	0
whcspa	0	0	0	0	0	1	0	0	0	0	0	12

Figure 3 Confusion matrix of Multiclass Classification

Discussion

According to the findings of this research, Neural Networks can be used to predict bird species based on sounds. This research produced three models, the first of which was a binary classification model which classified the bird species as amecro or barswa based on their sound. A Convolution Neural Network was developed, and it predicted with a test data accuracy of 96.9% and the time taken to train the model was 167 sec. The accuracy of the model on the test data has slightly improved from 93.3% to 96.9% with the addition of a dropout layer. The second model was designed for multi-class classification. Using CNN, it could classify a given input into one of 12 bird species based on their sound. This model has performed with an accuracy of 68.3% and the time taken to train the model was 927.13 sec. The multiclass model was overfitting the training data. Therefore, an additional dropout layer was added to address this issue. This improved the accuracy of the model on the test data from 59.3% to 68.3%. The training time is more for the multiclass model as it was trained to predict 12 bird species.

The model for binary classification was able to classify the species into two classes more accurately as the sounds of those bird species (amecro and barswa) are distinct. Therefore, the model was able to effectively generalize the data and accurately make predictions. In the case of the multiclass classification model, twelve species were taken into consideration, and some of the species have been misclassified by the model. This may be due to the similarity in the sounds of the species or could be because only the louder segments of the audio clip were considered for the analysis. This may have totally overlooked the species which had their uniqueness in softer segments of the audio clip. As a result, developing a model based on the dataset that takes into account both the louder and softer parts of the clip may help lower the misclassification rate. Additionally, a few of the sound clips contain background noise, which may also have caused difficulty for the model in recognizing the species by sound. In addition, overfitting exists in the multiclass classification model despite the fact that few measures were taken to prevent it.

Therefore, a larger dataset would be considered for future research, which would aid neural networks in generalizing, recognizing patterns in the data, and making more accurate predictions on the test dataset. The incorporation of L1 and L2 regularizations, as well as the addition or modification of neural network layers, are a few additional strategies that will be utilized to prevent overfitting.

Conclusion

According to the findings of this research, Neural Networks can be used to predict bird species based on sounds as they can easily understand the underlying patterns and are more suitable when working with larger datasets. Neural Networks are more powerful compared to other machine learning algorithms when working with audio files as they could easily understand and classify the spectrogram data. However, to improve the accuracy of the model, future research would include considering the spectrogram data of both the louder and softer segments which could help in effectively generalizing the patterns. Additionally, a few other strategies such as incorporating L1 and L2 regularizations and adding or modifying the layers in the model architecture could be used to prevent overfitting of the training data. Experts and researchers can use these findings to prioritize their efforts to save endangered species and identify any human activity, such as deforestation, that may be contributing to the species extinction.

References

- Xeno-canto :: Sharing Wildlife Sounds From Around the World. xeno-canto.org.
- About :: Xeno-canto. xeno-canto.org/about/xeno-canto.
- Wikipedia contributors. "Xeno-canto." Wikipedia, Feb. 2023, Available at: https://en.wikipedia.org/wiki/Xeno-canto
- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani - An Introduction to Statistical Learning
- "What Is a Spectrogram? | Pacific Northwest Seismic Network." Pacific Northwest Seismic Network, Available at: https://pnsn.org/spectrograms/what-is-a-spectrogram
- Spectrogram - Edge Impulse Documentation. Available at: https://docs.edgeimpulse.com/docs/edge-impulse-studio/processing-blocks/spectrogram
- What Are Convolutional Neural Networks? | IBM. www.ibm.com/topics/convolutional-neural-networks.
- Deep Learning, DATA 5322, Class Notes
- IBM. Available at: www.ibm.com/topics/convolutional-neural-networks.

CodeImplementation3

Aishwarya Saibewar

5/5/2023

```
#Load the required libraries
library(keras)
```

```
## Warning: package 'keras' was built under R version 4.1.2
```

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.1.2
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(abind)
library(tools)
```

Predicting bird species based on Sound

#Binary Classfication

```
# Set the working directory
setwd("/Users/aishwaryasaibewar/Documents/SeattleUniversity-MSDS/Courses/SU Course Work/SPRING_2023/Sta


#The folder includes 2 .dat files of amecro and barswa bird species
folder_path <- "/Users/aishwaryasaibewar/Documents/SeattleUniversity-MSDS/Courses/SU Course Work/SPRING_

#.dat files list
```

```r
dat_files <- list.files(folder_path, pattern = ".dat")

# List of spectrogram data of all bird species
data_list <- list()

# List of names of all bird species
speciesname_list <- list()

#List of 1st dimension for assigning labels
species_label<-list()

# Iterating through the files and reading and proocessing them
for (i in 1:length(dat_files)) {
  data <- load(dat_files[i]) #Load the data from the dat files
  data <- species  #Initial data is a large array and the data is stored in species object
  data_list[[i]] <- data #Store the spectrogram data of each species in a list
  fname_without_exten <- file_path_sans_ext(dat_files[i]) #extract the names of each species
  speciesname_list[[i]] <- fname_without_exten #Then append it to the list and store it
  species_label[i]<-dim(species)[1] #Choose the 1st dimension for assigning labels to each species

}
```

```r
# Using abind function to combine data stored in list for all species along a particular axis.It will b
data_array <- abind(data_list, along = 1)

#Check Dimension of the data
dim(data_array)
```

```
## [1] 107 343 256
```

```r
#Assigning labels for the species, 0 to amecro and 1 to barswa
labels_assigned <- c(rep(0, species_label[1]), rep(1, species_label[2]))
```

```r
#One-hot encode the labels
one_hot_labels <- array(labels_assigned,dim=c(length(labels_assigned),1))
```

```r
# The dataset was divided into training and testing, with 70% of the data used as a training dataset to
set.seed(1)
train <- sample(nrow(data_array), 0.7 * nrow(data_array))

x_train <- data_array[train,,]
y_train <- one_hot_labels[train]

x_test <- data_array[-train,,]
y_test <- one_hot_labels[-train]

y_train <- matrix(y_train, ncol = 1)
y_test <- matrix(y_test, ncol = 1)
```

```r
#Dimensions of test and training
dim(x_train)
```

```
## [1]  74 343 256
```

```r
dim(x_test)
```

```
## [1]  33 343 256
```

```r
dim(y_train)
```

```
## [1] 74  1
```

```r
dim(y_test)
```

```
## [1] 33  1
```

```r
#Reshape the data to a 4D array to feed as an input to the neural network
x_train <- array_reshape(x_train, c(dim(x_train)[1], dim(x_train)[2], dim(x_train)[3], 1))
x_test <- array_reshape(x_test, c(dim(x_test)[1], dim(x_test)[2], dim(x_test)[3], 1))
```

```r
#Dimensions of test and training
dim(x_train)
```

```
## [1]  74 343 256   1
```

```r
dim(x_test)
```

```
## [1]  33 343 256   1
```

```r
# Build a Neural Network
binary_model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3),padding = "same", activation = "relu", input_shape =
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3),padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  #layer_dropout(rate = 0.2) %>% #Adding dropout layer
  layer_flatten() %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dropout(rate = 0.3) %>% #Adding dropout layer
  layer_dense(units = 1, activation = "sigmoid")
```

```
## Loaded Tensorflow version 2.4.1
```

```r
# Compile model by considering loss as binary_crossentropy and optimizer as adam and performance metric
binary_model %>% compile(loss = "binary_crossentropy",
  optimizer = "adam",
  metrics = list("accuracy")
)
```

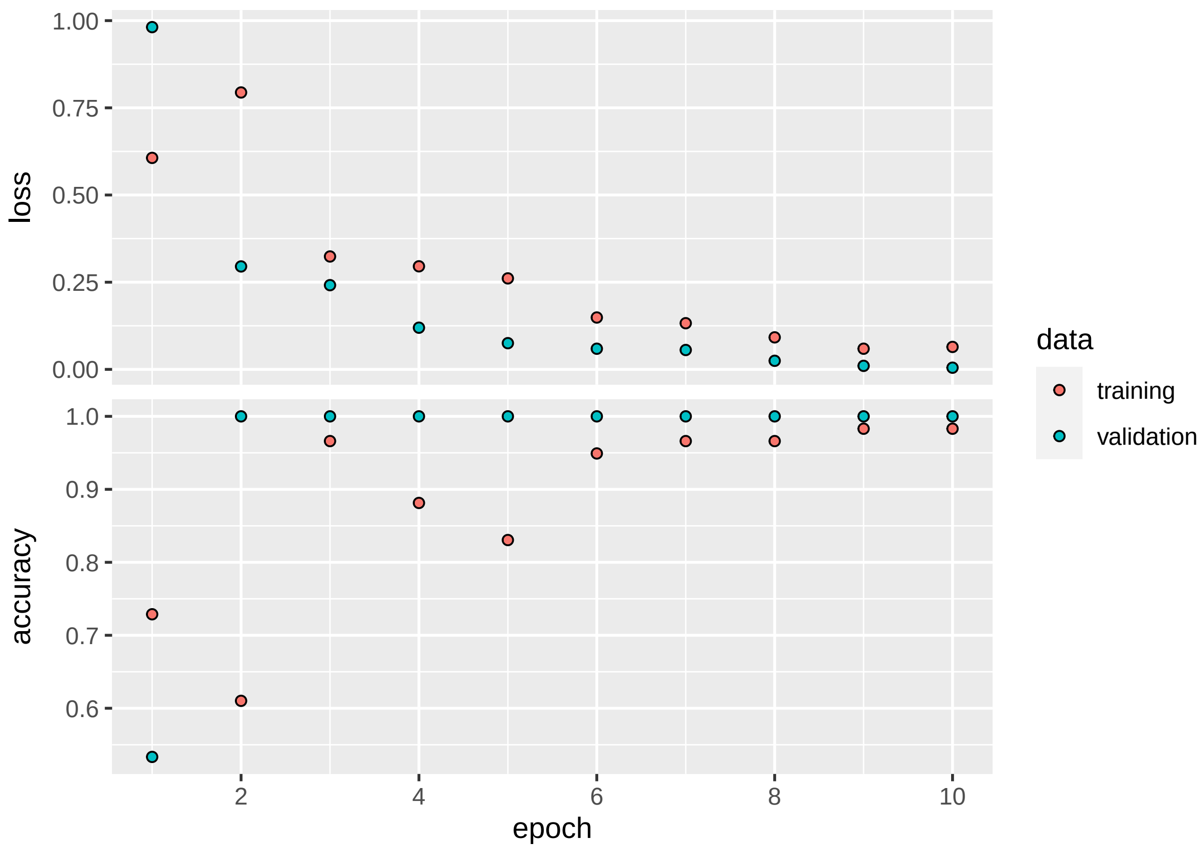
```r
#Summary of the binary classification model
summary(binary_model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                    Param #
## ================================================================================
## conv2d_1 (Conv2D)                   (None, 343, 256, 32)            320
## ________________________________________________________________________________
## max_pooling2d_1 (MaxPooling2D)      (None, 171, 128, 32)            0
## ________________________________________________________________________________
## conv2d (Conv2D)                     (None, 171, 128, 64)            18496
## ________________________________________________________________________________
## max_pooling2d (MaxPooling2D)        (None, 85, 64, 64)              0
## ________________________________________________________________________________
## flatten (Flatten)                   (None, 348160)                  0
## ________________________________________________________________________________
## dense_1 (Dense)                     (None, 32)                      11141152
## ________________________________________________________________________________
## dropout (Dropout)                   (None, 32)                      0
## ________________________________________________________________________________
## dense (Dense)                       (None, 1)                       33
## ================================================================================
## Total params: 11,160,001
## Trainable params: 11,160,001
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
# Fit the model and estimate the time taken to train the model.
system.time(
   history <- binary_model %>%
      fit(x_train, y_train, epochs = 10, batch_size = 25,
        validation_split = 0.2)
 )
```

```
##    user  system elapsed
## 162.689  40.566  38.643
```

```r
#Plot the history of the model.
plot(history, smooth = FALSE, main = "Performance metrics for Binary Classification")
```




```r
#Verify the dimensions
dim(x_test)
```

```
## [1]  33 343 256   1
```

```r
dim(y_test)
```

```
## [1] 33  1
```

```r
# Evaluate model on test data
test_binary_accuracy <- binary_model %>% evaluate(x_test, y_test)
cat("Test accuracy for binary classification model using CNN is ", test_binary_accuracy)
```

```
## Test accuracy for binary classification model using CNN is  0.04085369 0.969697
```

A binary classification model was developed to classify the bird species as amecro or barswa based on their sound. A Convolution Neural Network was developed to perform this classification. As shown in Figure 1 the model's loss and accuracy are plotted against the increasing number of epochs for the training and validation dataset. It can be observed that the accuracy of the training and validation data increases with the number of epochs, and the model converges at the 10th epoch with a batch size of 25. Similarly, the loss tends to decrease as the number of epochs increases. The initial model without dropout layers has predicted with a test accuracy of 93.3%. To improve the accuracy of the model, a dropout layer that ignored 30% of the training data was added to the CNN model. As a result of this, the accuracy of the test data has slightly increased from 93.3% to 93.9%. Finally, the best model predicted the training data with an accuracy of 98%, validation data with an accuracy of 97%, and the test data with an accuracy of 96.9%. This model was able to classify the species more accurately because the sounds of the bird species amecro or barswa are quite different and the model was able to understand the sound patterns of the species and classify the new observations accordingly.

==

MULTICLASS CLASSIFICATION

```r
# Set the working directory
setwd("/Users/aishwaryasaibewar/Documents/SeattleUniversity-MSDS/Courses/SU Course Work/SPRING_2023/Sta


#The folder includes 2 .dat files of amecro and barswa bird species
folder_path <- "/Users/aishwaryasaibewar/Documents/SeattleUniversity-MSDS/Courses/SU Course Work/SPRING_

#.dat files list
dat_files <- list.files(folder_path, pattern = ".dat")

# List of spectrogram data of all bird species
data_list <- list()

# List of names of all bird species
speciesname_list <- list()

#List of 1st dimension for assigning labels
species_label<-list()

# Iterating through the files and reading and proocessing them
for (i in 1:length(dat_files)) {
  data <- load(dat_files[i]) #Load the data from the dat files
  data <- species  #Initial data is a large array and the data is stored in species object
  data_list[[i]] <- data #Store the spectrogram data of each species in a list
  fname_without_exten <- file_path_sans_ext(dat_files[i]) #extract the names of each species
  speciesname_list[[i]] <- fname_without_exten #Then append it to the list and store it
  species_label[i]<-dim(species)[1] #Choose the 1st dimension for assigning labels to each species

}
```

```r
# Using abind function to combine data stored in list for all species along a particular axis.It will b
data_array <- abind(data_list, along = 1)

#Check Dimension of the data
dim(data_array)
```

```
## [1] 580 343 256
```

```r
#Assigning labels for the 12 species
labels_assigned <- c(rep(0, species_label[1]), rep(1, species_label[2]),rep(2, species_label[3]),rep(3,
```

```
# The dataset was divided into training and testing, with 70% of the data used as a training dataset to
set.seed(1)
train <- sample(nrow(data_array), 0.7 * nrow(data_array))

x_train <- data_array[train,,]
y_train <- to_categorical(labels_assigned[train])

x_test <- data_array[-train,,]
y_test <- to_categorical(labels_assigned[-train])
```

```
#Check dimensions of test and training data
dim(x_train)
```

```
## [1] 406 343 256
```

```
dim(x_test)
```

```
## [1] 174 343 256
```

```
dim(y_train)
```

```
## [1] 406  12
```

```
dim(y_test)
```

```
## [1] 174  12
```

```
#Reshape the data to a 4D array to feed as an input to the neural network
x_train <- array_reshape(x_train, c(dim(x_train)[1], dim(x_train)[2], dim(x_train)[3], 1))
x_test <- array_reshape(x_test, c(dim(x_test)[1], dim(x_test)[2], dim(x_test)[3], 1))
```

```
#Dimensions of test and training
dim(x_train)
```

```
## [1] 406 343 256   1
```

```
dim(x_test)
```

```
## [1] 174 343 256   1
```

```
# Build a Neural Network with 2 convolution layers,2 max pool layers, one flatten, 3 dense layers and a
multiclass_model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu", input_shape = c(343,256,1)) %
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dropout(rate = 0.4) %>% #Added for overfitting
  layer_dense(units = 12, activation = "softmax")
```

```r
# Compile model by considering loss as binary_crossentropy and optimizer as adam and performance metric
multiclass_model %>% compile(loss = "categorical_crossentropy",
  optimizer = "adam",
  metrics = list("accuracy")
)
```

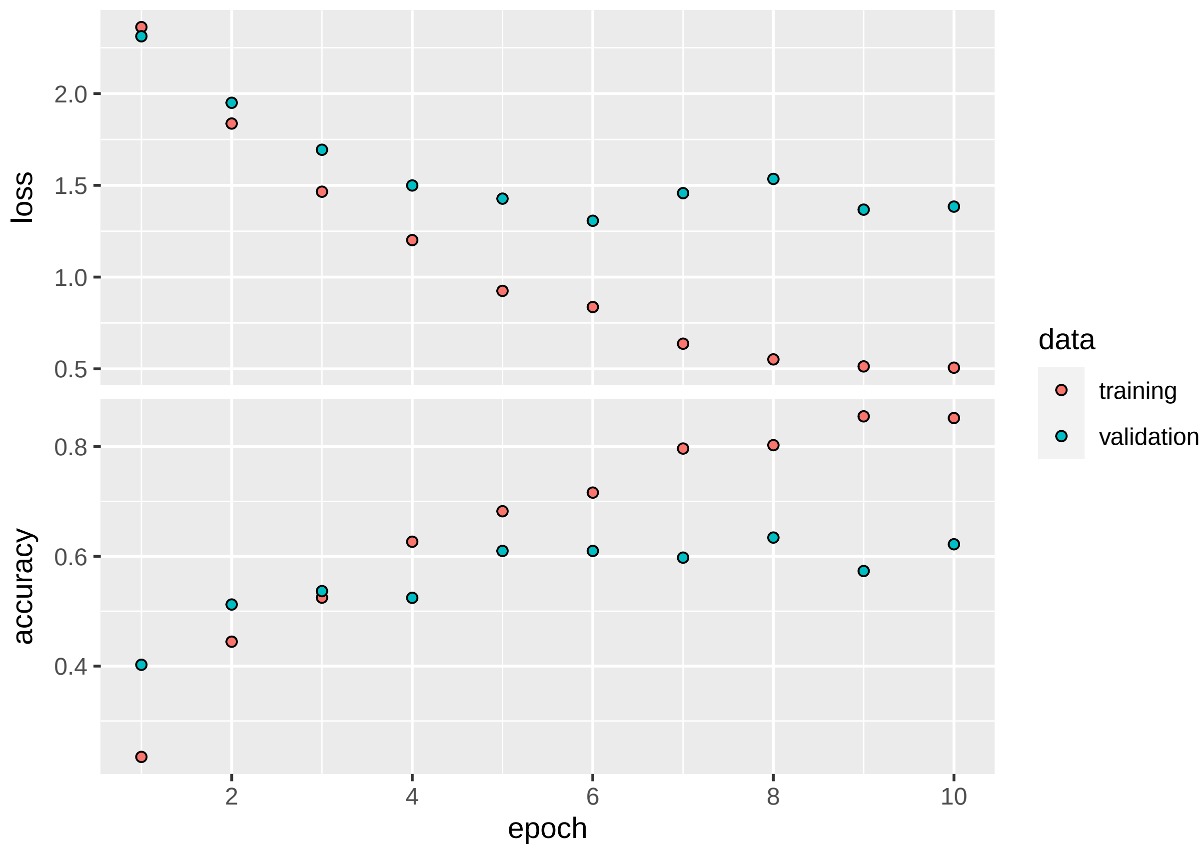
```r
#Summary of the multiclass classification model
summary(multiclass_model)
```

```
## Model: "sequential_1"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                     Param #
## ================================================================================
## conv2d_3 (Conv2D)                   (None, 341, 254, 32)             320
## ________________________________________________________________________________
## max_pooling2d_3 (MaxPooling2D)      (None, 170, 127, 32)             0
## ________________________________________________________________________________
## conv2d_2 (Conv2D)                   (None, 168, 125, 64)             18496
## ________________________________________________________________________________
## max_pooling2d_2 (MaxPooling2D)      (None, 84, 62, 64)               0
## ________________________________________________________________________________
## flatten_1 (Flatten)                 (None, 333312)                   0
## ________________________________________________________________________________
## dense_4 (Dense)                     (None, 64)                       21332032
## ________________________________________________________________________________
## dense_3 (Dense)                     (None, 32)                       2080
## ________________________________________________________________________________
## dropout_1 (Dropout)                 (None, 32)                       0
## ________________________________________________________________________________
## dense_2 (Dense)                     (None, 12)                       396
## ================================================================================
## Total params: 21,353,324
## Trainable params: 21,353,324
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
# Fit the model and estimate the time taken to train the model.
system.time(
   history <- multiclass_model %>%
      fit(x_train, y_train, epochs = 10, batch_size = 25,
        validation_split = 0.2)
 )
```

```
##     user   system  elapsed
## 1016.035  256.451  237.463
```

```r
#Plot the history
plot(history, smooth = FALSE)
```




```r
#Check the dimensions of test and training
dim(x_test)
```

```
## [1] 174 343 256   1
```

```r
dim(y_test)
```

```
## [1] 174  12
```

```r
# Evaluate model on test data
test_multi_accuracy <- multiclass_model %>% evaluate(x_test, y_test)
cat("Test accuracy for multiclass classification model using CNN is ", test_multi_accuracy)
```

```
## Test accuracy for multiclass classification model using CNN is  1.297022 0.683908
```

A multiclass classification model was developed to classify the 12 bird species based on their sound using a Convolution Neural Network. As shown in Figure 2 the model's loss and accuracy are plotted against the increasing number of epochs. As shown in Figure 2, the model's accuracy and loss are plotted against the training and validation datasets for the increasing number of epochs. It can be observed that the model converges at the 10th epoch with a batch size of 25, and the accuracy of the training and validation data increases with the number of epochs. And the loss tends to decrease as the number of epochs increases. The initial model without dropout layers has predicted the training data, validation data, and test data with an accuracy of 90.8%, 70.1%, and 56.3% respectively. It can be seen that the model is overfitting the training data and is unable to generalize the patterns in test and validation datasets. To overcome this issue, an additional dropout layer that ignores 40% of the training data was added to the CNN model. This has resulted in decreasing the accuracy of training data from 90.8% to 82.8% and increasing the accuracy of test data from 56.3% to 67.2%.

```r
#The names of the species
speciesname_list
```

```
## [[1]]
## [1] "amecro"
##
## [[2]]
## [1] "barswa"
##
## [[3]]
## [1] "bkcchi"
##
## [[4]]
## [1] "blujay"
##
## [[5]]
## [1] "daejun"
##
## [[6]]
## [1] "houfin"
##
## [[7]]
## [1] "mallar3"
##
## [[8]]
## [1] "norfli"
##
## [[9]]
## [1] "rewbla"
##
## [[10]]
## [1] "stejay"
##
## [[11]]
## [1] "wesmea"
##
## [[12]]
## [1] "whcspa"
```

```r
# Predict the model
predict <- multiclass_model %>% predict(x_test)
colnames(predict) <- speciesname_list
head(predict)
```

```
##         amecro       barswa       bkcchi       blujay       daejun       houfin
## [1,] 0.9997925 4.517176e-06 1.915617e-06 5.047864e-08 1.076086e-04 2.138709e-05
## [2,] 0.9999909 4.274308e-07 2.081657e-08 2.068129e-09 2.571410e-06 1.890889e-06
## [3,] 0.9997440 1.297887e-05 3.757193e-05 2.051244e-07 7.234476e-05 1.326729e-05
```

```
## [4,] 0.9937984 2.060163e-04 4.995524e-04 1.824290e-05 8.574291e-04 4.871043e-04
## [5,] 0.9995331 2.773515e-05 5.978975e-06 4.587525e-07 1.834058e-04 8.928256e-05
## [6,] 0.9106319 4.406515e-03 1.816162e-03 3.082101e-04 5.581217e-04 5.430363e-03
##           mallar3       norfli       rewbla       stejay       wesmea
## [1,] 3.261410e-06 2.205689e-07 3.824868e-05 2.434721e-05 3.947842e-06
## [2,] 2.399539e-06 4.018204e-08 1.064326e-06 5.290655e-07 2.209569e-08
## [3,] 2.196760e-05 2.941223e-06 7.865383e-05 1.009489e-05 3.685392e-06
## [4,] 7.109014e-04 1.111504e-04 1.126674e-03 1.506087e-03 3.947592e-04
## [5,] 2.992274e-05 1.352223e-06 2.537644e-05 9.550908e-05 2.807312e-06
## [6,] 3.549807e-02 5.711262e-04 2.143132e-03 2.960632e-02 1.247055e-03
##            whcspa
## [1,] 1.917262e-06
## [2,] 5.815135e-08
## [3,] 2.212154e-06
## [4,] 2.835986e-04
## [5,] 5.099837e-06
## [6,] 7.783043e-03
```

```
#Ensure that the sum of the probbabilities is equal to 1
sum(predict[1, ])
```

```
## [1] 0.9999999
```

```
sum(predict[2, ])
```

```
## [1] 1
```

```
# https://www.rdocumentation.org/packages/nnet/versions/7.3-12/topics/predict.nnet

#Fetch the actual and predicted values
Predicted = max.col(predict) - 1
Actual = max.col(y_test) - 1
```

```
#Confusion matrix for multiclass classification
multi_data_cf <- table(Predicted, Actual)
multi_data_cf
```

```
##          Actual
## Predicted  0  1  2  3  4  5  6  7  8  9 10 11
##        0  12  0  2  0  0  0  0  0  0  0  0  0
##        1   0 14  0  0  0  2  0  0  0  0  0  1
##        2   0  0  5  0  0  0  0  0  0  0  0  0
##        3   0  1  0  7  0  0  0  0  3  1  1  0
##        4   0  1  2  1 17  0  0  0  1  0  0  4
##        5   0  0  0  2  0  6  0  0  1  0  2  0
##        6   0  0  0  0  0  0  4  0  0  0  0  0
##        7   0  1  2  2  3  1  0 17  0  0  1  1
##        8   0  0  0  1  0  2  0  0 11  0  0  0
##        9   0  0  0  2  0  0  1  0  0  6  0  0
##        10  0  0  0  0  0  1  0  0  1  0  7  0
##        11  0  1  0  1  0  1  8  0  0  0  0 13
```

```r
#Rename the confusion matrix
colnames(multi_data_cf) <- c("amecro","barswa","bkcchi","blujay","daejun","houfin","mallar3","norfli","
rownames(multi_data_cf) <- c("amecro","barswa","bkcchi","blujay","daejun","houfin","mallar3","norfli","
```

```r
#Final Confusion matrix for multiclass classification
multi_data_cf
```

```
##          Actual
## Predicted amecro barswa bkcchi blujay daejun houfin mallar3 norfli rewbla
##   amecro      12      0      2      0      0      0       0      0      0
##   barswa       0     14      0      0      0      2       0      0      0
##   bkcchi       0      0      5      0      0      0       0      0      0
##   blujay       0      1      0      7      0      0       0      0      3
##   daejun       0      1      2      1     17      0       0      0      1
##   houfin       0      0      0      2      0      6       0      0      1
##   mallar3      0      0      0      0      0      0       4      0      0
##   norfli       0      1      2      2      3      1       0     17      0
##   rewbla       0      0      0      1      0      2       0      0     11
##   stejay       0      0      0      2      0      0       1      0      0
##   wesmea       0      0      0      0      0      1       0      0      1
##   whcspa       0      1      0      1      0      1       8      0      0
##          Actual
## Predicted stejay wesmea whcspa
##   amecro       0      0      0
##   barswa       0      0      1
##   bkcchi       0      0      0
##   blujay       1      1      0
##   daejun       0      0      4
##   houfin       0      2      0
##   mallar3      0      0      0
##   norfli       0      1      1
##   rewbla       0      0      0
##   stejay       6      0      0
##   wesmea       0      7      0
##   whcspa       0      0     13
```

The confusion matrix of the multiclass classification model is shown in Figure 3. The diagonal elements represent the correct classifications and the others represent the misclassifications. It very well may be seen from the confusion matrix that there are misclassifications of the observations. Some of the incorrect classifications include barswa being predicted as houfin and whcspa, bkcchi as barswa and daejun, and norfli as blujay and daejun. This could be due to the similarity of these bird species sounds. Additionally, the spectrograms used for the analysis included data on just the louder parts of the audio clip. Although important patterns of the species may be present in the clip's louder parts, some species that had their identity in the clip's softer parts may have been incorrectly classified.