Emma Oriol
DATA 5322
5/14/2023

Neural Network Bird Call Classification

Abstract:

This report builds convolutional neural network models to perform audio classification on bird calls for twelve of Seattle's most common birds. The models are trained on spectrograms of bird calls. The aim of this study is to develop accurate bird recognition to contribute to the appreciation, study, and conservation of Seattle's bird population.

Introduction:

Seattle has a large and diverse bird population, with over 200 species regularly sighted in King County. Seattle provides key habitat for many year-round resident birds as well as birds passing through on their migration routes (King County, 2016). Identifying birds using their calls can be a tool for monitoring, researching, and protecting bird populations in Seattle.

The goal of this report is to build classification models that identify bird species using spectrograms of bird calls from twelve of Seattle's most common birds. The first model built is a convolutional neural network model model that performs binary classification to identify crows and black-capped chickadees. The second model is a convolutional neural network model that performs multi-categorical classification to identify the American Crow, Barn Swallow, Black-capped Chickadees, Blue Jay, Dark-eyed Junco, House Finch, Mallard, Northern Flicker, Red-winged Blackbird, Steller's Jay, Western Meadowlark, and White-crowned Sparrow. The spectrograms are sourced for Xeno-Canto, a crowd-sourced bird sounds archive.
.

Theoretical Background:

Neural network models are considered to be among the most complex and powerful machine learning models. Neural networks process data in layers/nodes and are designed to mimic the way brains process information using neurons (Castillo, 2022). At each layer, an activation function is applied to the data. Weights are assigned to each layer and adjusted during training to optimize model accuracy. Neural networks can have any number of processing layers that can take a variety of forms (James et al., 2022). They are best suited for complex and nuanced classification problems, however they are very computationally expensive and require high machine power, memory, and long processing times.

Convolutional neural networks (CNNs) are a class of neural networks that are best suited to process data in a grid-like formation, like pixels in images. CNNs repeat convolution and pooling layers until it reaches its final flattening layer. At each convolution layer, a filter is applied to the image to detect key features. Then, pooling layers condense and summarize the data. The

programmer determines the number and size of convolution layers to be used, and the activation function and weight of each layer are learned by the model through training. Data must be in matrix format to be processed by CNNs (James et al., 2022).

There are several tuning parameters to be selected during model construction. First, the number, type, and size of the activation layer is selected. Second, dropout learning can be used at each layer to reduce the size of the training data used and reduce overfitting. Lastly, the batch size determines the size of data sampled for each epoch, and the epoch is the number of passes through the data set (James et al., 2022).

Methodology:

The data used to measure bird call sounds were in the form of spectrograms. Spectrograms are a visual representation of audio frequencies over time. Convolutional neural networks were chosen as an appropriate model because the spectrograms are a matrix of pixels.

Prior to starting analysis, the spectrograms were obtained from the Xeno-Canto website, from which ten mp3 sound clips of each of the twelve birds were chosen. The first model built was a CNN that classifies American Crows and Black-capped Chickadees. The ten spectrograms for the crow and chickadee calls were combined into a four-dimensional matrix. The corresponding names of each bird species were saved in the form of a binary matrix in sparse-matrix format. The data was then randomly split into a training set (80% of data) and test set (20% of data).

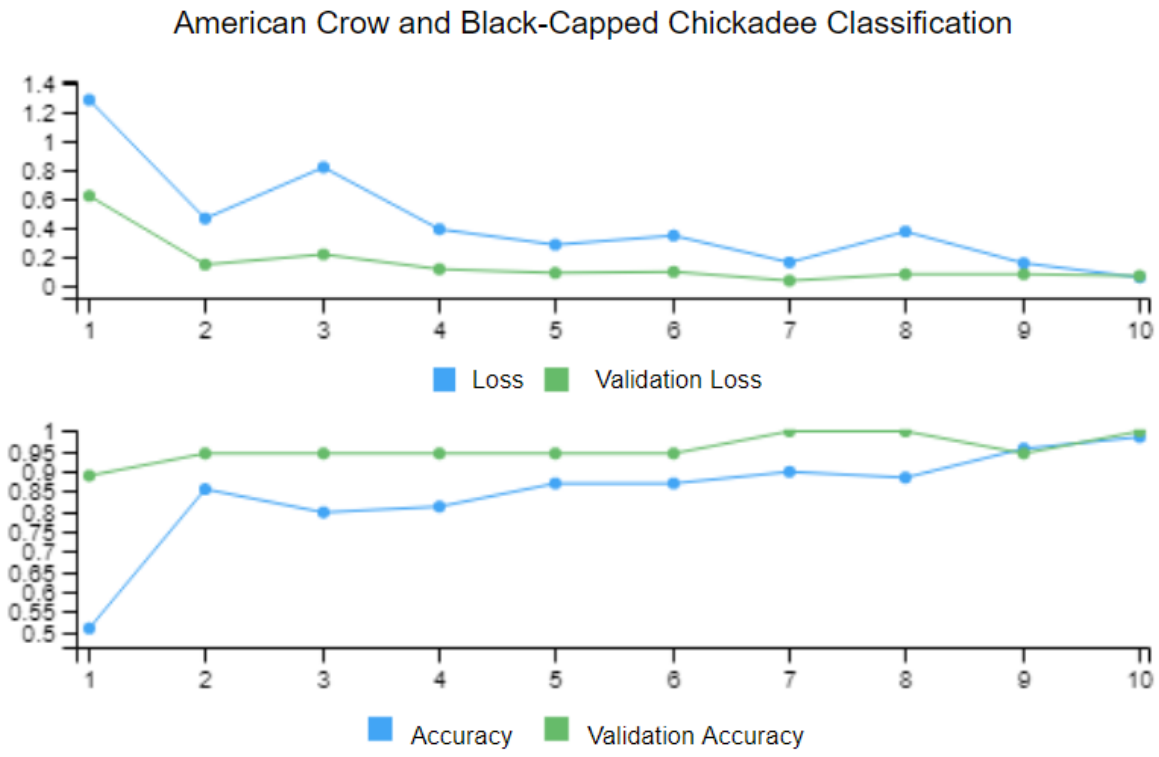
Next, the model was built using five convolution and pooling layers. A dropout rate of 0.5 was used to drop 50% of the training data at each layer. The epochs parameter was set to ten, meaning the model iterated over the entire data set 10 times at each layer. The batch size was set to five so the model processes five samples before updating its weights. Lastly, the validation split parameter was set to 0.2, so that 20% of the training data is used for validation during training. The output of the model was a binary classification of bird species (American Crow or Black-capped Chickadee).

The second model built was a multicategorical classification CNN that classified all twelve species of birds. The spectrograms for every bird species were combined into a matrix, and the corresponding name of each bird species was saved in a matrix in sparse-matrix format. The data was then randomly split into a training set (80% of data) and test set (20% of data).

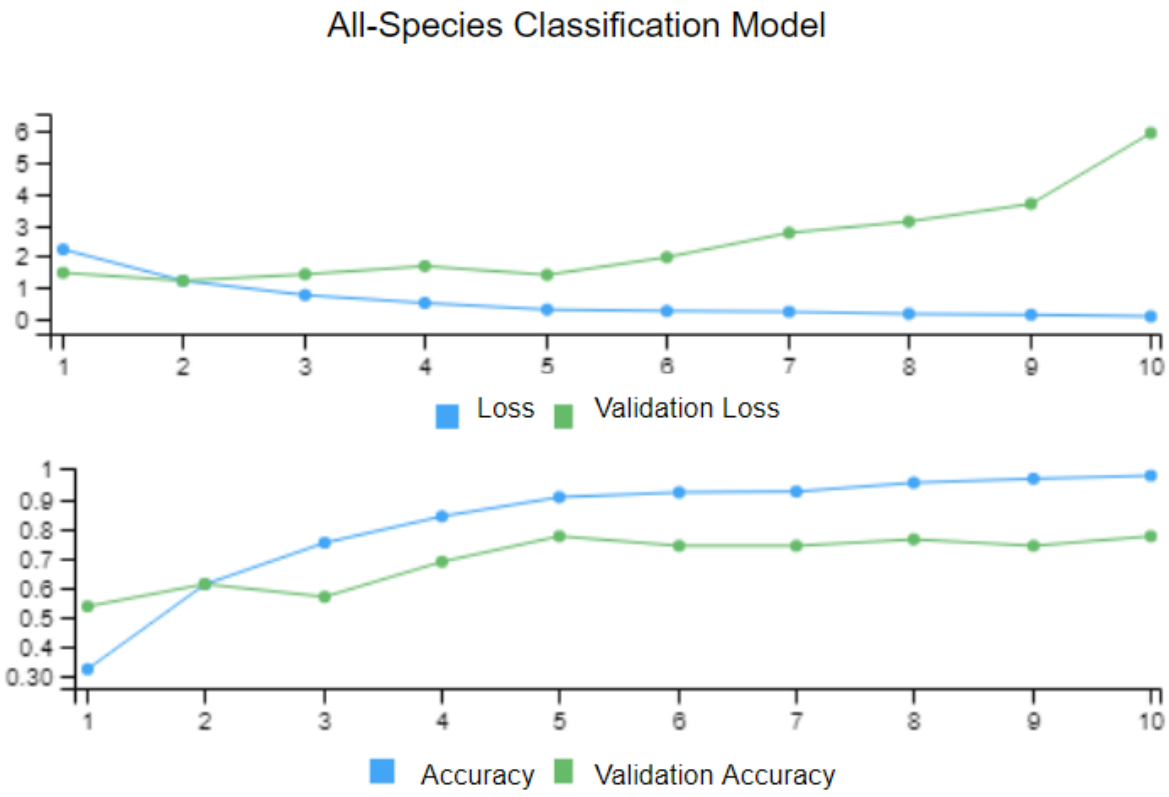
The model was built using five convolution and pooling layers. A dropout rate of 0.5 was used to drop 50% of the training data at each layer. The epochs parameter was set to ten, meaning the model iterated over the entire data set 10 times at each layer. The batch size was set to five so the model processes five samples before updating its weights. Lastly, the validation split parameter was set to 0.2, so that 20% of the training data is used for validation during training. The output of the model was a classification for one of the twelve species of birds used.

Computational Results:

The binary classification model identified American Crows and Black-capped Chickadees with an accuracy rate of 90.9%. The plot below demonstrates the loss, validation loss, accuracy, and validation accuracy for each epoch while the model trained. The plot demonstrates that the loss and validation loss, which are the data misclassified by the model, decreases and begins to plateau. Meanwhile, the accuracy and validation accuracy, which is the data correctly classified by the model, increase and plateau as well. This suggests that the model is tuned well and does not need more epochs, because the performance improves and then plateaus as epochs increase.



The second model that classified all twelve bird species performed similarly well, with a test accuracy rate of 95.9%. As with the first model, the loss and validation loss decreases and begins to plateau while the accuracy and validation accuracy increase and plateau. This suggests that the model is tuned well and does not need more epochs, because the performance improves and then plateaus.



Discussion:

The largest limitation in this study was the processing time and machine power it takes to run the models. Each model took about 10 minutes to train, which made it very slow to tweak parameters and re-run the model. The sample sizes were also small (only ten spectrograms per bird) in order to reduce processing time. To further improve the models, increased machine power and storage space would be advantageous. The models could be tuned better and trained on larger data sets. Larger data sets would improve accuracy and provide more diverse examples of bird calls, allowing for more nuanced classification.

One repeated mistake of the second model was that White-crowned Sparrows were misclassified as House Finches. Both birds have a similar-sounding high pitched warbling call. This resulted in similar spectrograms which were easily confused by the model.

Neural networks were a highly effective choice for this application due to their ability to handle very complex and nuanced data. Convolutional neural networks were particularly well-suited for processing spectrograms because they are designed for image processing and feature identification. Alternatively, random forest decision tree models could have been used for this task. Decision tree models are commonly used for classification tasks, and random forest models are an ensemble approach by combining multiple decision trees. However, decision

trees are not as well suited for the complexity and nuance of spectrograms, which neural networks are particularly well-equipped to handle.

Conclusion:

Overall, the convolutional neural network models performed remarkably well and can be a powerful and effective tool for bird species classification. Improving bird classification can further understanding, appreciation, and conservation of Seattle's bird population. Furthermore, neural networks could have a vast range of applications in audio classification. For example, they could identify voices, other animal species, and environmental sounds. Further research is needed to determine the efficacy of audio classification in other contexts.

In conclusion, convolutional neural networks are highly effective in bird species classification and have the potential to improve audio classification in various applications. By improving our ability to recognize and classify environmental sounds, convolutional networks can pave the way for improved conservation strategies and ecological research.

Bibliography:

Castillo, D. (2022, January 4). *Neural network models explained*. Seldon. https://www.seldon.io/neural-network-models-explained

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2022). *An introduction to statistical learning: With applications in R*. Springer.

King County. (2016, November 10). *Birds in King County*. Birds in King County - King County. https://kingcounty.gov/services/environment/animals-and-plants/biodiversity/defining-biodiversity/species-of-interest/birds.aspx

Appendix:

````
---
title: "Written Homework 3"
author: "Emma"
date: "2023-05-08"
output:
  pdf_document: default
---
```{r}
library(tidyverse)
library(keras)
library(e1071)
library(dplyr)
library(abind)
library(Matrix)
library(forecast)
```
#Load in data
```{r}
load("spectrograms/amecro.dat")
amecro = species
load("spectrograms/barswa.dat")
barswa = species
bkcchi <- load("spectrograms/bkcchi.dat")
bkcchi = species
blujay <- load("spectrograms/blujay.dat")
blujay = species
daejun <- load("spectrograms/daejun.dat")
daejun = species
houfin <- load("spectrograms/houfin.dat")
houfin = species
mallar3 <- load("spectrograms/mallar3.dat")
mallar3 = species
norfli <- load("spectrograms/norfli.dat")
norfli = species
rewbla <- load("spectrograms/rewbla.dat")
rewbla = species
stejay <- load("spectrograms/stejay.dat")
stejay = species
wesmea <- load("spectrograms/wesmea.dat")
wesmea = species
whcspa <- load("spectrograms/whcspa.dat")
whcspa = species
````
````

```

#Binary Model Prediciton <br>

Prep data <br>

Function to One-hot encode

```{r}

one_hot <- function(labels, num_classes) {
  label_indices <- match(labels, unique(labels))
  sparseMatrix(
    i = seq_along(label_indices),
    j = label_indices,
    x = 1,
    dims = c(length(labels), num_classes),
    dimnames = list(NULL, levels(labels))
  )
}
```

Prepare and sample training and testing data
```{r}
#Combine crow and barn swallow into one matrix
x = abind(amecro, bkcchi, along=1) #bird calls
x <- abind(x, along = 4) #add dimension
y = c(rep('amecro', dim(amecro)[1]), rep('bkcchi', dim(bkcchi)[1])) #saves names of bird species
y = one_hot(y, 2)#one hot encode my y variable

#Choose training and testing data
set.seed(1)

sample_size = floor(0.8*nrow(x)) #sample size for training data set
picked = sample(seq_len(nrow(x)),size = sample_size) #sample 80% of data

#x data
xtrain =x[picked,,,]
xtest = x[-picked,,, ]
#ydata
ytrain = y[picked,]
ytest = y[-picked,]

#reshape x data
```
```

```
xtrain <- array_reshape(xtrain, c(dim(xtrain)[1], dim(xtrain)[2], dim(xtrain)[3], 1))
xtest <- array_reshape(xtest, c(dim(xtest)[1], dim(xtest)[2], dim(xtest)[3], 1))
```
Build model
```{r}
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3),
    padding = "same", 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),
    padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3),
    padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 256, kernel_size = c(3, 3),
    padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = "relu") %>%
  layer_dense(units = 2, activation = "softmax")
summary(model)
```
Accuracy function
```{r}
accuracy <- function(pred, truth) {
  mean(drop(as.numeric(pred)) == drop(truth)) }
```
Run model
```{r}
model %>% compile(loss = "categorical_crossentropy",
   optimizer = optimizer_rmsprop(), metrics = c("accuracy"))
#history <- model %>% fit(x_train, y_train, epochs = 30,
history <- model %>% fit(xtrain, ytrain, epochs = 10,
   batch_size = 5, validation_split = 0.2)

model %>% predict(xtest) %>% round() %>% accuracy(ytest) #test data accuracy

```
Accuracy of 90.9%

Try again with adjusted parameters
```{r}
model %>% compile(loss = "categorical_crossentropy",
   optimizer = optimizer_rmsprop(), metrics = c("accuracy"))
#history <- model %>% fit(x_train, y_train, epochs = 30,
history <- model %>% fit(xtrain, ytrain, epochs = 12,
   batch_size = 3, validation_split = 0.1)

model %>% predict(xtest) %>% round() %>% accuracy(ytest)

```

Accuracy of 90.9%. No improvement of accuracy, but the plots of loss, validation loss, accuracy, and validation accuracy, show more fluctuations than the previous model. The previous model therefore performed better.

#Multicategorical Model Prediciton

Prep data

```{r}
#Combine all bird spectrograms into one matrix
x = abind(amecro, barswa, bkcchi, blujay, daejun, houfin, mallar3, norfli, rewbla, stejay, wesmea, whcspa, along=1) #bird calls
x <- abind(x, along = 4) #add dimension

#Species lables
y = c(rep('amecro', dim(amecro)[1]), rep('barswa', dim(barswa)[1]), rep('bkcchi', dim(bkcchi)[1]), rep('blujay', dim(blujay)[1]),
     rep('daejun', dim(daejun)[1]), rep('houfin', dim(houfin)[1]), rep('mallar3', dim(mallar3)[1]), rep('norfli', dim(norfli)[1]),
     rep('rewbla', dim(rewbla)[1]), rep('stejay', dim(stejay)[1]), rep('wesmea', dim(wesmea)[1]), rep('whcspa', dim(whcspa)[1])) #saves names of bird species
y = one_hot(y, 12)#one hot encode y variable
```
Training and testing data
```{r}
#Choose training and testing data
set.seed(2)

sample_size = floor(0.8*nrow(x)) #sample size for training data set
picked = sample(seq_len(nrow(x)),size = sample_size) #sample 80% of data

xtrain =x[picked,,,]
```

```
xtest = x[-picked,,, ]
ytrain = y[picked,]
ytest = y[-picked,]

#reshape x data
xtrain <- array_reshape(xtrain, c(dim(xtrain)[1], dim(xtrain)[2], dim(xtrain)[3], 1))
xtest <- array_reshape(xtest, c(dim(xtest)[1], dim(xtest)[2], dim(xtest)[3], 1))
```
Build model
```{r}
model2 <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3),
    padding = "same", 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),
    padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3),
    padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 256, kernel_size = c(3, 3),
    padding = "same", activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = "relu") %>%
  layer_dense(units = 12, activation = "softmax")
summary(model2)
```
Run and test model
```{r}
model2 %>% compile(loss = "categorical_crossentropy",
   optimizer = optimizer_rmsprop(), metrics = c("accuracy"))
#history <- model %>% fit(x_train, y_train, epochs = 30,
history <- model2 %>% fit(xtrain, ytrain, epochs = 10,
   batch_size = 5, validation_split = 0.2)

accuracy <- function(pred, truth) {
  mean(drop(as.numeric(pred)) == drop(truth)) }
model2 %>% predict(xtest) %>% round() %>% accuracy(ytest) #test prediction accuracy

```
Accuracy of 95.9%.

Which birds were misclassified?

Print out predictions

```{r}
predictions <- round(predict(model2, xtest))
labels <- c(rep('amecro', dim(amecro)[1]), rep('barswa', dim(barswa)[1]), rep('bkcchi',
dim(bkcchi)[1]), rep('blujay', dim(blujay)[1]),
    rep('daejun', dim(daejun)[1]), rep('houfin', dim(houfin)[1]), rep('mallar3', dim(mallar3)[1]),
rep('norfli', dim(norfli)[1]),
    rep('rewbla', dim(rewbla)[1]), rep('stejay', dim(stejay)[1]), rep('wesmea', dim(wesmea)[1]),
rep('whcspa', dim(whcspa)[1])) #saves names of bird
length(labels)
labels <- labels[-picked]
length(labels)
predictions <- cbind(predictions, labels)
predictions
```

White-crowned sparrow frequently misclassified as house finch.
```{r}
whcspa_matrix <- as.matrix(whcspa[,,1])  # Convert the desired slice to a matrix
image(whcspa_matrix, useRaster = TRUE)

```

```{r}
houfinch <- as.matrix(houfin[,,1])  # Convert the desired slice to a matrix
image(houfinch, useRaster = TRUE)
```