The Journal of Open Source Software

gibbonNetR: an R Package for the Use of Convolutional Neural Networks and Transfer Learning on Passive Acoustic Monitoring (PAM) Data

Dena Jane Clink[1] **and Abdul Hamid Ahmad**[2]

1 K. Lisa Yang Center for Conservation Bioacoustics,Cornell Lab of Ornithology, Cornell University, Ithaca, New York, United States **2** Institute for Tropical Biology and Conservation,Universiti Malaysia Sabah (UMS), Kota Kinabalu, Sabah, Malaysia

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals
Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Automated detection of acoustic signals is crucial for effective monitoring of vocal animals and their habitats across large spatial and temporal scales. Recent advances in deep learning have made high performance automated detection approaches more accessible to more practitioners. However, there are few deep learning approaches that can be implemented natively in R. The 'torch for R' ecosystem has made the use of transfer learning with convolutional neural networks accessible for R users. Here we provide an R package and workflow to use transfer learning for the automated detection of acoustics signals from passive acoustic monitoring (PAM) data collected in Sabah, Malaysia. The package provides functions to create spectogram images from PAM data, compare the performance of different pre-trained CNN architectures trained on the ImageNet dataset, deploy trained models over directories of sound files, and extract embeddings from trained models. The R programming language remains one of the most commonly used languages among ecologists, and we hope that this package makes deep learning approaches more accessible to this audience. In addition, these models can serve as important benchmarks for more state-of-the-art approaches.

Statement of need

Passive acoustic monitoring

We are in a biodiversity crisis, and there is a great need for the ability to rapidly assess biodiversity in order to understand and mitigate anthropogenic impacts. One approach that can be especially effective for monitoring of vocal yet cryptic animals is the use of passive acoustic monitoring (Gibb et al., 2018), a technique that relies on autonomous acoustic recording units. PAM allows researchers to monitor vocal animals and their habitats, at temporal and spatial scales that are impossible to achieve using only human observers. Interest in use of PAM in terrestrial environments has increased substantially in recent years (Sugai et al., 2019), due to reduced price of the recording units and improved battery life and data storage capabilities. However, the use of PAM often leads to the collection of terabytes of data that is time- and cost-prohibitive to analyze manually.

Automated detection

Some of the early non-deep learning approaches for the automated detection of acoustic signals in terrestrial PAM data include binary point matching (Katz et al., 2016), spectrogram cross-correlation (Balantic & Donovan, 2020), or the use of a band- limited energy detector and subsequent classifier, such as support vector machine (Clink et al., 2023; Kalan et al., 2015). Recent advances in deep learning have revolutionized image and speech recognition (LeCun et al., 2015), with important cross-over for the analysis of PAM data. Traditional approaches to machine learning relied heavily on feature engineering, as early machine learning algorithms required a reduced set of representative features, such as features estimated from the spectrogram. Deep learning does not require feature engineering (Stevens et al., 2020) . Convolutional neural networks (CNNs) — one of the most widely used deep learning algorithms—are useful for processing data that have a 'grid-like topology', such as image data that can be considered a 2-dimensional grid of pixels (Goodfellow et al., 2016). The 'convolutional' layer learns the feature representations of the inputs; these convolutional layers consist of a set of filters which are basically two-dimensional matrices of numbers and the primary parameter is the number of filters (Gu et al., 2018). Therefore, with CNN's there is no feature engineering required. However, if training data are scarce, overfitting may occur as representations of images tend to be large with many variables (LeCun et al., 1995).

Transfer learning?

Training deep learning models generally requires a large amount of training data and computing resources, which can be hard to obtain with PAM data. Transfer learning is an approach wherein the architecture of a pretrained CNN (which is generally trained on a very large dataset) is applied to a new classification problem. For example, CNNs trained on the ImageNet dataset of > 1 million images (Deng et al., 2009) such as ResNet have been applied to automated detection/classification of primate and bird species from PAM data (Dufourq et al., 2022; Ruan et al., 2022). At the most basic level, transfer learning in computer vision applications retains the feature extraction or embedding layers, and modifies the last few classification layers to be trained for a new classification task (Dufourq et al., 2022). Recent advances have used embeddings from audio classification models trained on bird songs for new classification problems, and in most cases these embeddings led to better performance than general audio datasets (Ghani et al., 2023).

State of the field

The two most popular open-source programming languages are R and Python (Scavetta & Angelov, 2021). Python has surpassed R in terms of overall popularity, but R remains an important language for the life sciences (Lawlor et al., 2022). 'Keras' (Chollet & others, 2015), 'PyTorch' (Paszke et al., 2019) and 'Tensorflow' (Martín Abadi et al., 2015) are some of the more popular neural network libraries; these libraries were all initially developed for the Python programming language. Until recently, deep learning implementations in R relied on the 'reticulate' package which served as an interface to Python (Ushey et al., 2022). The transfer learning approaches included in this package have already been implemented in Python (Dufourq et al., 2022). Previous implementations of automated detection using deep learning in R relied on the 'reticulate' package Silva et al. (2022). However, the recent release of the 'torch for R' ecosystem provides a framework based on 'PyTorch' that runs natively in R and has no dependency on Python (Falbel, 2023). Running natively in R means more straightforward installation, and higher accessibility for users of the R programming environment. Keydana (2023) provides tutorials for transfer learning in the 'torch for R' ecosystem, and the functions in 'gibbonNetR' rely heavily on these tutorials.

Overview

This package provides functions to create spectrogram images, use transfer learning for six pretrained CNN architectures (AlexNet (Krizhevsky et al., 2017) , VGG16, VGG19 (Simonyan & Zisserman, 2014), ResNet18, ResNet50, and ResNet152 (He et al., 2016)) trained on the ImageNet dataset (Deng et al., 2009). The package also has functions to evaluate model performance, deploy the highest performing model over a directory of sound files, and extract embeddings from trained models to visualize acoustic data. We provide an example dataset that consists of labelled vocalizations of the loud calls of four vertebrates from Danum Valley Conservation Area, Sabah, Malaysia.

Usage

Data summary

We include spectrogram images of five classes: great argus pheasant (*Argusianus argus*) long calls (Clink et al., 2021), helmeted hornbills (*Rhinoplax vigil*), and rhinoceros hornbills (*Buceros rhinoceros*) (Kennedy et al., 2023), female gibbons (*Hylobates funereus*) and a catch-all "noise" category. The data come from two separate PAM arrays in Danum Valley Conservation Area, Sabah, Malaysia. The training and validation data come from a wide array of Swift autonomous recording units placed on ~750 m spacing (Clink et al., 2023), and the test data come from a different, smaller array (~250 m spacing) within the same area. We used a band-limited energy detector to identify signals that were 3-sec or longer duration within the 400-1600 Hz range, and then a single observer (DJC) manually sorted the detections into their respective categories (Clink et al., 2023).

Spectrogram images

The package currently uses spectrogram images (Figure 1), and 'gibbonNetR' includes a function that can be used to create spectrogram images from .wav files. The 'splits' option will assign the specified proportion of clips to either training, validation, or test folders. We highly recommend that your test data come from a different recording time and/or location to better understand the generalizability of the models (Stowell, 2022).

```
# Generate spectrogram images for training (70%) and validation (30%).
# Test data should come from a different source.
spectrogram_images(
  trainingBasePath = trainingBasePath,
  outputBasePath = outputBasePath,
  minfreq.khz = 0.4,
  maxfreq.khz = 1.6,
  splits = c(0.7, 0.3, 0),
  new.sampleratehz = 'NA'
)
```

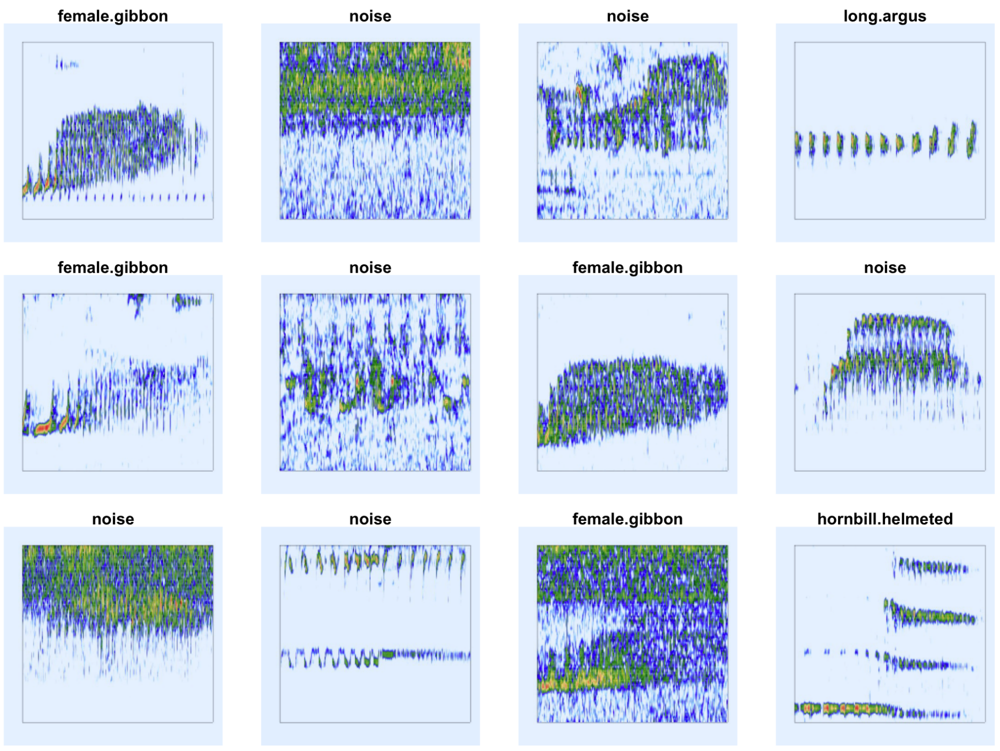

Figure 1: Figure 1. Spectrograms of training clips for CNNs.

Model training

The package currently allows for the training of six different CNN architectures ('alexnet', 'vgg16', 'vgg19', 'resnet18', 'resnet50', or 'resnet152'), and the user can specify whether or not to freeze the feature extraction layers. There is the option to train a binary or multi-class classifer.

```
# Location of spectrogram images for training
input.data.path <-  'data/examples/'

# Location of spectrogram images for testing
test.data.path <- 'data/examples/test/'

# User specified training data label for metadata
trainingfolder.short <- 'danummulticlassexample'

# We can specify the number of epochs to train here
epoch.iterations <- c(20)

# Function to train a multi-class CNN
gibbonNetR::train_CNN_multi(input.data.path=input.data.path,
                            architecture ='resnet18',
                            learning_rate = 0.001,
                            class_weights = rep( (1/5), 5),
                            test.data=test.data.path,
                            unfreeze.param = TRUE,
                            epoch.iterations=epoch.iterations,
                            save.model= TRUE,
                            early.stop = "yes",
                            output.base.path = "model_output/",
                            trainingfolder=trainingfolder.short,
                            noise.category = "noise")
```

Evaluate model performance

We can compare the performance of different CNN architectures (Figure 2). Using the 'get_best_performance' function we can evaluate the performance of different model architectures on the test dataset for the specified class.

```
PerformanceOutput <- get_best_performance(performancetables.dir=performancetables.dir,
                                          class='female.gibbon',
                                          model.type = "multi",
                                          Thresh.val=0)
PerformanceOutput$f1_plot
```

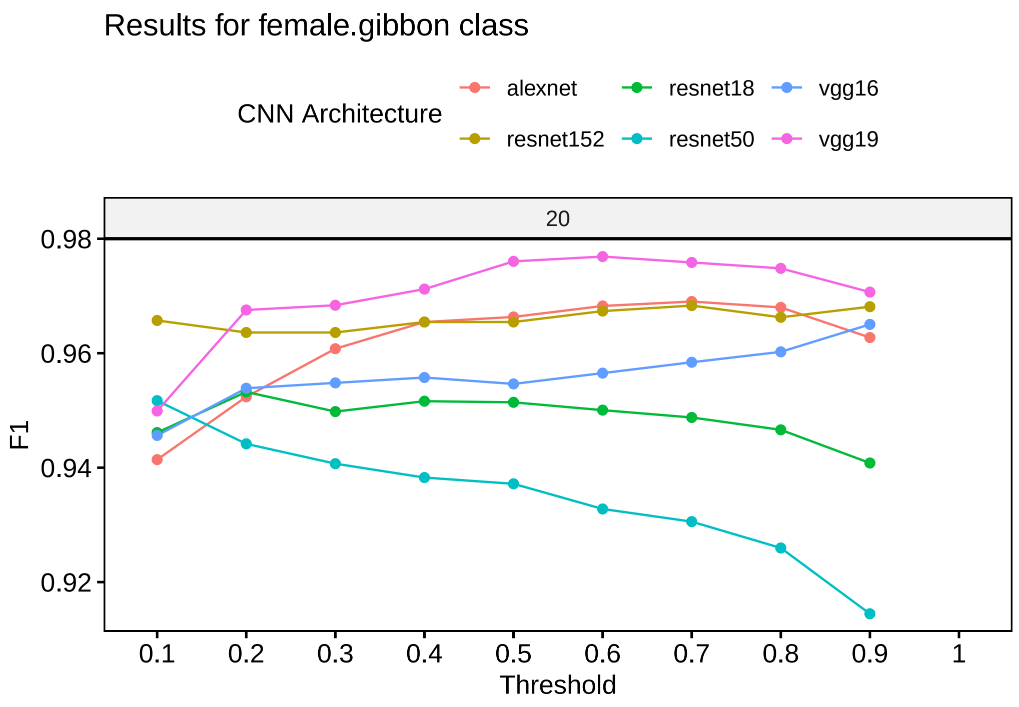

Figure 2: Figure 2. Evaluating performance of pretrained CNNs.

Extract embeddings

Embeddings from deep learning models can be used as features in unsupervised approaches, with promising results for call repertoires (Best, 2023) and individual identity (Lakdari et al., 2024). This package contains the function 'extract_embeddings' to use pretrained CNNs to extract embeddings, where the trained model path, along with test data location and target class are specified.

```
result <- extract_embeddings(test_input="data/examples/test/",
                                      model_path=ModelPath,
                                     target_class = "female.gibbon")
```

We can plot the unsupervised clustering results

In Figure 3 the top plot is a Uniform Manifold Approximation and Projection (UMAP) where each point represents one call, and the colors indicate the original class label. The bottom plot is the same UMAP plot, but with points colored based on cluster assignment by the 'hdbscan' algorithm (Hahsler et al., 2019).

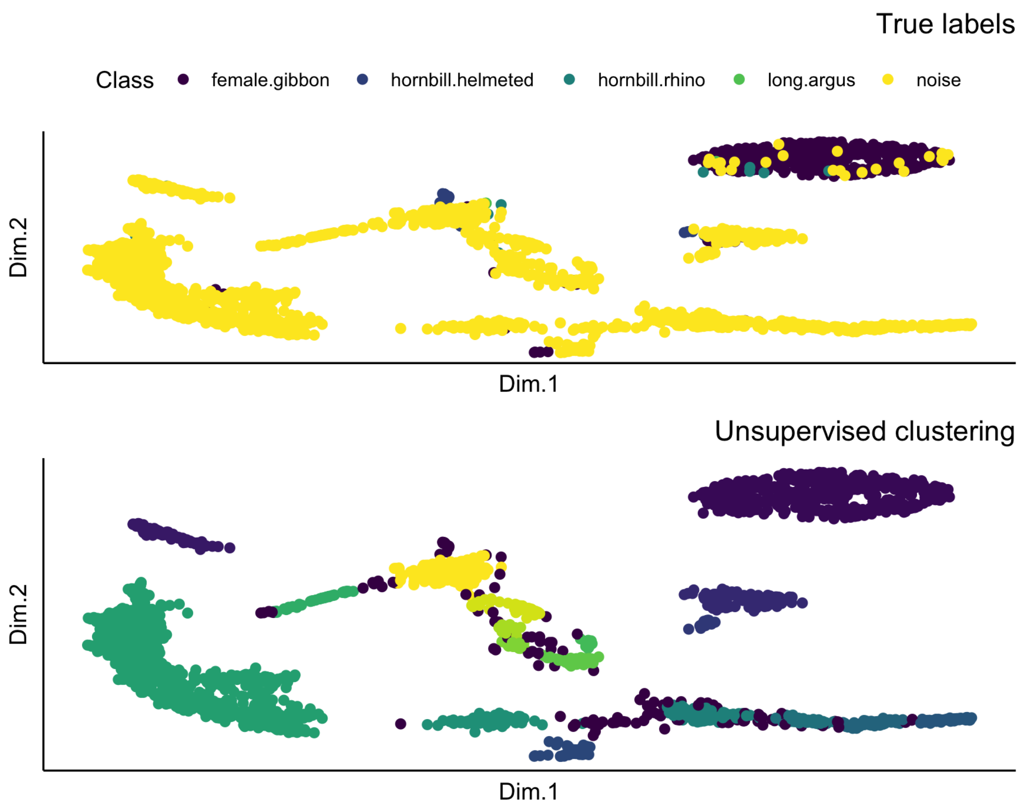

Figure 3: Figure 3. UMAP plot of embeddings from test data set colored by actual label (top) and unsupervised cluster assignment (bottom).

We can explore the unsupervised clustering results

Here we can see the Normalize Mutual Information score, which provides a value between 0 and 1, indicating the match between cluster labels and actual labels.

```
result$NMI
```

The confusion matrix is made using the 'caret' package(Kuhn, 2008) returns the results when we use 'hdbscan' (Hahsler et al., 2019) to match the target class to the cluster with the largest number of observations of that particular class.

```
result$ConfusionMatrix
```

Future directions

There have been huge advances in the fields of deep learning and automated detection approaches for PAM data in recent years. The approach presented in this package (as of 2024) is no longer state of the art, although it was just recently developed. More recent approaches use transfer learning from models that are explicitly trained on bioacoustics data, such as BirdNET (Ghani et al., 2023). However, there is a huge need in the field of bioacoustics to do benchmarking, wherein different model architectures and performance are compared across diverse datasets. Therefore, the methods presented here can provide important benchmarks for future work, and for understanding how and if advances improve performance over more traditional methods. The R package is available on Github, where issues can be opened.

Acknowledgments

We would like to thank the Sabah Biodiversity Centre and Danum Valley Conservation Area for granting us permission to conduct research.

References

Balantic, C., & Donovan, T. (2020). AMMonitor: Remote monitoring of biodiversity in an adaptive framework with r. *Methods in Ecology and Evolution*, *11*(7), 869877.

Best, S. A. G., Paul AND Paris. (2023). Deep audio embeddings for vocalisation clustering. *PLOS ONE*, *18*(7), 1–18. https://doi.org/10.1371/journal.pone.0283396

Chollet, F., & others. (2015). *Keras*. https://keras.io

Clink, D. J., Groves, T., Ahmad, A. H., & Klinck, H. (2021). Not by the light of the moon: Investigating circadian rhythms and environmental predictors of calling in bornean great argus. *Plos One*, *16*(2), e0246564.

Clink, D. J., Kier, I., Ahmad, A. H., & Klinck, H. (2023). A workflow for the automated detection and classification of female gibbon calls from long-term acoustic recordings. *Frontiers in Ecology and Evolution*, *11*. https://www.frontiersin.org/articles/10.3389/fevo.2023.1071640

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). *Imagenet: A large-scale hierarchical image database*. 248255.

Dufourq, E., Batist, C., Foquet, R., & Durbach, I. (2022). Passive acoustic monitoring of animal populations with transfer learning. *Ecological Informatics*, *70*, 101688. https://doi.org/https://doi.org/10.1016/j.ecoinf.2022.101688

Falbel, D. (2023). *Luz: Higher level 'API' for 'torch'*. https://CRAN.R-project.org/package=luz

Ghani, B., Denton, T., Kahl, S., & Klinck, H. (2023). Global birdsong embeddings enable superior transfer learning for bioacoustic classification. *Scientific Reports*, *13*(1), 22876.

Gibb, R., Browning, E., Glover-Kapfer, P., & Jones, K. E. (2018). Emerging opportunities and challenges for passive acoustics in ecological assessment and monitoring. *Methods in Ecology and Evolution*. https://doi.org/10.1111/2041-210X.13101

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & others. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354377.

Hahsler, M., Piekenbrock, M., & Doran, D. (2019). dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, *91*(1), 1–30. https://doi.org/10.18637/jss.v091.i01

He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. 770778.

Kalan, A. K., Mundry, R., Wagner, O. J. J., Heinicke, S., Boesch, C., & Kühl, H. S. (2015). Towards the automated detection and occupancy estimation of primates using passive acoustic monitoring. *Ecological Indicators*, *54*(July 2015), 217226. https://doi.org/10.1016/j.ecolind.2015.02.023

Katz, J., Hafner, S. D., & Donovan, T. (2016). Assessment of error rates in acoustic monitoring with the r package monitoR. *Bioacoustics*, *25*(2), 177196.

Kennedy, A. G., Ahmad, A. H., Klinck, H., Johnson, L. M., & Clink, D. J. (2023). Evidence for acoustic niche partitioning depends on the temporal scale in two sympatric bornean hornbill species. *Biotropica*, *55*(2), 517–528.

Keydana, S. (2023). *Deep learning and scientific computing with r torch*. CRC Press.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 8490.

Kuhn, M. (2008). Caret package. *Journal of Statistical Software*, *28*(5), 126.

Lakdari, M. W., Ahmad, A. H., Sethi, S., Bohn, G. A., & Clink, D. J. (2024). Mel-frequency cepstral coefficients outperform embeddings from pre-trained convolutional neural networks under noisy conditions for discrimination tasks of individual gibbons. *Ecological Informatics*, *80*, 102457.

Lawlor, J., Banville, F., Forero-Muñoz, N.-R., Hébert, K., Martínez-Lanfranco, J. A., Rogy, P., & MacDonald, A. A. M. (2022). Ten simple rules for teaching yourself R. *PLOS Computational Biology*, *18*(9), e1010372. https://doi.org/10.1371/journal.pcbi.1010372

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

LeCun, Y., Bengio, Y., & others. (1995). Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks*, *3361*(10), 1995.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, … Xiaoqiang Zheng. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). *PyTorch: An imperative style, high-performance deep learning library* (p. 80248035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Ruan, W., Wu, K., Chen, Q., & Zhang, C. (2022). ResNet-based bio-acoustics presence detection technology of hainan gibbon calls. *Applied Acoustics*, *198*, 108939. https://doi.org/https://doi.org/10.1016/j.apacoust.2022.108939

Ruff, Z. J., Lesmeister, D. B., Appel, C. L., & Sullivan, C. M. (2021). Workflow and convolutional neural network for automated identification of animal sounds. *Ecological Indicators*, *124*, 107419. https://doi.org/10.1016/j.ecolind.2021.107419

Scavetta, R. J., & Angelov, B. (2021). *Python and r for the modern data scientist*. O'Reilly Media, Inc.

Silva, B., Mestre, F., Barreiro, S., Alves, P. J., & Herrera, J. M. (2022). soundClass: An automatic sound classification tool for biodiversity monitoring using machine learning. *Methods in Ecology and Evolution*.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv Preprint arXiv:1409.1556*.

Stevens, E., Antiga, L., & Viehmann, T. (2020). *Deep Learning with PyTorch*. Simon; Schuster.

Stowell, D. (2022). Computational bioacoustics with deep learning: a review and roadmap. *PeerJ*, *10*, e13152. https://doi.org/10.7717/peerj.13152

Sugai, L. S. M., Silva, T. S. F., Ribeiro, J. W., & Llusia, D. (2019). Terrestrial passive acoustic monitoring: Review and perspectives. *BioScience*, *69*(1), 1525. https://doi.org/10.1093/biosci/biy147

Ushey, K., Allaire, J. J., & Tang, Y. (2022). *Reticulate: Interface to 'python'*.