Automatic recognition of frog vocalizations using Jetstream

Eliza G. Foran*
egforan@iu.edu
Indiana University
Bloomington, Indiana

Evan D. Suggs
mdk988@mocs.utc.edu
University Of Tennessee At Chattanooga
Chattanooga, Tennessee

Tenecious Underwood
TUnderwood65@students.livingstone.edu
Livingstone College
Salisbury, North Carolina

Winona G. Snapps-Childs
wsnappch@iu.edu
Indiana University
Bloomington, Indiana

Sheri A. Sanders
ss93@indiana.edu
Indiana University
Bloomington, Indiana

ABSTRACT

Recording animal calls and vocalizations is a time-honored data collection method in various fields of biological and environmental science. In the past, the only method available for analyzing such recordings involved extensive training of human experts. Now, however, machine learning techniques make automatic recognition of such vocalizations possible. Automatic recognition of animal calls and vocalizations is desirable on two fronts: it reduces the burden of (at least initial) data analysis, and supports non-intrusive environmental monitoring. Here, we outline a proof-of-concept workflow that will make the quest, from gathering to interpreting data, more attainable for researchers. We simulate this data collection process by collecting animal (frog) calls using recording devices and Raspberry Pi's, then feed this data into a database and virtual machine hosted on XSEDE resources (i.e. Jetstream and Wrangler). We then show how database pulling, machine learning, and visualization works on Jetstream.

CCS CONCEPTS

• **Bioacoustic Identification** → **Machine Learning**; • **Frog Call** → *Jetstream cloud computing*; • **Keras** → Neural Networks; • **Field station support** → Automation;

KEYWORDS

machine learning, neural networks, bioacoustics, cloud computing

ACM Reference format:
Eliza G. Foran, Evan D. Suggs, Tenecious Underwood, Winona G. Snapps-Childs, and Sheri A. Sanders. 2019. Automatic recognition of frog vocalizations using Jetstream. In *Proceedings of PEARC '19: Practice and Experience in Advanced Research Computing, Chicago, IL, July 28– August 01, 2019 (PEARC '19),* 4 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Foran, Suggs, and Underwood contributed equally to this research.

PEARC '19, July 28– August 01, 2019, Chicago, IL

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Using field station cyberinfrastructure to collect and handle field data is an emerging resource for biology researchers. The Jetstream national research cloud [29] assists scientists in the "long tail of science" with high-performance computing. We developed a Jetstream work flow for automatic recognition of frogs with greater accuracy than the average citizen science frog surveyor (at least an 80% chance of correct identification). Frog surveys emerged from scientific interest in amphibians as environmental health markers. Frogs are especially vulnerable to disease, pollution, and habitat loss, making them valuable indicators of environmental disruption. This is especially relevant to global trends in amphibian biodiversity, where around 81% of species are decreasing rapidly and 18% are increasing [25]. The National Center for Genome Analysis Support (NCGAS) created automatic recording devices and simple bioacoustic analysis, however that system required extensive proofing. We expanded the NCGAS/Jetstream workflow with deep machine learning to eliminate proofs and enhance authority in identification.

Neural networks are machine learning systems that can extract information from patterns of raw data to mimic human learning [19]. They use operations on nodes similar to neurons that can be built on top of one another in layers, which create "deep" learning within networks [19]. We tested three different deep neural networks to determine greatest application for frog call identification: one convolutional neural network (CNN) based on simple frequency over time spectrograms (image-based), one CNN using mel frequency cepstrum coefficients (audio-based), and an audio-based recurrent neural network (RNN). All of the models we chose have separate advantages, from computer storage to run-time. The Jetstream cloud computer allowed us to create and run these models efficiently and store large amounts of testing and training samples for all of the models.

2 METHODS

2.1 Data Collection and Processing

All audio samples were collected from Cornell's Macaulay Library archives of wildlife sounds [32]. Unfortunately, many of the sound files contained false positives. To solve this problem, we used the R package WarbleR [12] to create 9-second spectrograms of each of the lossless sound samples. We then parsed through over 5000 of these images to eliminate false positives. We then subset the raw audio to match our proofing on the spectrogram images. Overall,

we used an average of 707 frog calls 4 for each of the following four species:

- Spring Peeper
- Chorus Frog
- American Toad
- Green Frog

In both the CNNs and RNNs, we created visual representations of sound. For the audio-based CNN, we created a simple frequency over time spectrogram in grey scale. This simple spectrogram gives the frequency over time of each call. We chose to make each segment 9 seconds long to capture several frog calls in each segment, while giving the model a moderately sized number of individual samples. Previous work on using CNNs for acoustic data includes Haomin Zhang, Ian McLoughlin, and Yan Song [33]. Other previous work centered specifically on frog call recognition focused on audio spectrum analysis instead of machine learning techniques [13].

Audio processing often applies much more complex visual representations, such as Fourier transformations. Done correctly, these transformations eliminate empty information that would otherwise hinder a neural network. For inputs with time stamps, popular spectrograms are short-time Fourier transform and mel frequency spectrogram. According to a study of music preprocessing for neural networks, the mel frequency spectrogram has a slight advantage in accuracy (around 0.2 to 0.7 of 100%) for all data sets in Area Under Curve (AUC) tests. Above 64% data set usage, the differences are negligible as the AUCs converge [14]. Fourier transforms, in particular short-time Fourier transforms, are commonly used for preprocessing audio [14]. Where a standard Fourier transform is not a time frequency, the short-time Fourier transform has a window for values, and all values outside this window are zero-valued. The series is thus calculated as shorter windowed Fourier transforms instead of a single large Fourier. These sub-signals are then processed together [17].

For the audio-based RNN and CNN, we performed a series of transformations on the raw audio, shown in Figure 1. Instead of using a simple frequency-over-time spectrogram, we wanted to include both the power spectra, with frequency and time domains. To do this, we first used a fast Fourier transformation to create a change-in-time-over-frequency spectrum, which eliminated unimportant features of the audio. Additionally, we used short-time Fourier transformation (STFT) with Hamming Windows, Filterbank coefficient energies (logarithmic value of 26 filters), and processed through the Discrete Cosine Transform (DCT) to eliminate over-correlated coefficients of higher frequencies. Lower frequency coefficients created the end product in Mel-Frequency Cepstrum Coefficients. MFCCs include compressed data on time, frequency, and power density spectra [3–6].

2.2 Jetstream and Associated Packages

Speed is a competitive function of any machine learning application. We used a Jetstream [30] instance with Conda [11] environments to develop three neural networks. Our current instance configuration has a Jetstream g1.v100x-8q flavor. This flavor includes 4 VCPUs, 22GB RAM, 20GB size, and Centos 7 Devel OS with Anaconda v.1.9.

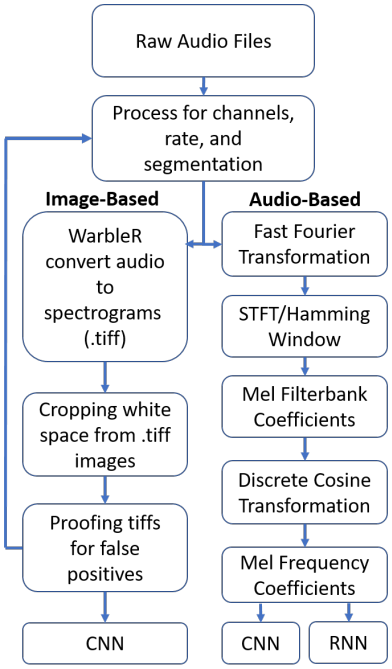

Figure 1: Data Processing Workflow

Similar to other research on neural networks, this project used extensive software libraries and applications to run them. The programming languages used were R and Python. The CNN was built with Keras [15] with a Tensorflow backend[1]. The Python package manager utilized for these was Anaconda [11], including other packages:

- Python 3.7 and 2.7.3 [26, 31]
- Numpy [23]
- Scipy [20]
- Pandas [22]

2.3 Image-Based Convolutional Neural Network

A common starting point for testing neural network applications is convolutional neural networks. Here, we have created a CNN that differentiates two and four frog species by inputting images of spectrograms. These spectrograms show frequency over time and can be differentiated by the naked eye, as shown in Figure 2.

Our CNN was set up with Keras for testing based on a CNN tutorial by Dakila Ledesma [21]. This model was originally designed for comparing RGB images of cats and dogs, though it readily accepted our frog call spectrograms. Input was a 64 * 64 resized image with 3 channels for RGB data. The first layer is a Conv2D using the ReLU activation functions to extract important portions of the data. The next layer, MaxPool2D, subtracts dimensions from extracted features found in the convolutional layer. The overall effect is padding to strengthen edge detection on the spectrograms versus pure white space. This is followed by another Conv2D layer and a dropout layer. The dropout layer prevents "over-fitting" the model to the training data by randomly selecting neurons to ignore. Over-fitting prevents the model from identifying inputs outside of

the training set data. The last three layers are two Dense layers sandwiching a Batch Normalization layer. The Dense layers keep the model "densely connected" and normalize inputs for probability distributions. The Batch Normalization layer reduces co-variance shift and makes the layers of the entire neural network behave more independently. The model ends with a Dense layer. The model was then compiled with mean-squared error (MSE) loss and Adam optimizer, then trained over 100 epochs along with a batch size of 256 [19].

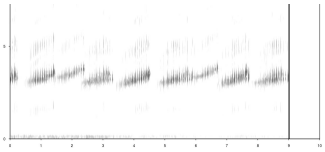

Figure 2: Frequency (kHz) over time spectrogram of Chorus Frog from Macauley Library, 2019

Figure 3: Four local Indiana frog species sampled for calls [16, 24, 27, 28]

2.4 Audio-Based Neural Networks

We adapted a previously developed Keras RNN/CNN and audio processing workflow (Figure 1) from Seth Adams on Github and YouTube. His models also uses the Tensorflow backend and Keras modeling functions. Both models were originally created to identify between ten different instrument sounds, such as trumpets and fiddles. We could simply replace the audio inputs with frogs and reduce the number of categories (instruments to frog species) to fit our needs [2, 7–10].

Seth Adams's model script includes both a CNN model and an RNN model, however the preprocessing for both is identical.

The RNN model uses nine different layers, where the input layer is a long short-term memory (LSTM). These layers retain information from multiple inputs and pass it to subsequent layers, which is useful for processing sequences or time series of data such as audio [19]. There are two explicit ReLU-activated LSTM layers at the top of this model. Deeper layers include four time-distributed Dense layers. These each perform simple rectified linear unit (ReLU) functions on inputs. The last two layers are a Flatten and Dense layer. The Flatten layer removes all but one dimension from inputs so that the final Dense layer does not need to be time-distributed, and the activation function of the final layer becomes softmax. Softmax layers normalize all the inputs. The outputs of this model are compiled with categorical cross-entropy loss and Adam optimizers. The cross-entropy loss or log loss outputs model accuracy as a probability between 0 and 1. Adam is a popular deep learning optimizer that combines advantages of AdaGrad and RMSProp [2, 7–10, 19].

The audio CNN also uses nine different layers. The first three layers are Conv2D fitted with ReLU activation. The next three layers are MaxPool2D, Dropout, and Flatten. The model is completed with three more Dense layers. All the layers described here are explained in the Image-Based Neural Network section. Similar to the RNN, this model has categorical cross-entropy outputs, Adam optimizers, and accuracy metrics. The final output of the model determines the accuracy of network's ability to identify two or four different frog species. [2, 7–10]

3 RESULTS

In order to calculate the accuracy of each model, we ran predictions ten times with model rebuilding between each run (Figure 6). We found that the image-based CNN performed with the greatest accuracy. In a Krustal-Wallis rank sun test, we found that the audio-based RNN had the weakest performance.

	CNN-image	CNN-audio	RNN-audio
2 Species	Training: 1244 (86.15%) Testing: 200 (13.85%)	Training: 1015 (83.54%) Testing: 200 (16.46%)	Training: 1015 (83.54%) Testing: 200 (16.46%)
4 Species	Training: 2428 (85.86%) Testing: 400 (14.14%)	Training: 1911 (82.69%) Testing: 400 (17.31%)	Training: 1911 (82.69%) Testing: 400 (17.31%)

Figure 4: Proportion of audio samples selected for model training versus prediction

	Training Time with 2 species, 4 species (min)		Predicting Time with 2 species, 4 species (min)	
CNN on spectrograms	9:45	21:39	0:33	1:14
RNN on audio	10:01	24:36	1:31	2:53
CNN on audio	13:16	28:28	1:28	2:38

Figure 5: Training times for the three neural networks

	CNN-image (avg ± std dev. %)	CNN-audio (avg ± std dev. %)	RNN-audio (avg ± std dev. %)
2 species	97.72 ± 0.80	99.50 ± 0.00	99.45 ± 0.16
4 species	97.35 ± 0.38	88.88 ± 0.89	89.83 ± 0.69

Figure 6: Prediction accuracy of the three models (avg + std dev %) after 10 runs

4 DISCUSSION

The accuracy of all three neural networks theoretically exceeds the accuracy for traditional citizen science frog surveys. All three of our models produced over 88% accuracy (Figure 6).

When fully integrated, our image-based CNN model can translate a frog calling in a remote location to automatic identification on a web page. Previous Jetstream undergraduate students created a custom Raspberry Pi recording device that could push audio directly to a web page for basic bioacoustic visualization (as a spectrogram and principle component analysis plot) [18]. Our research would complete automated identification. The data collection and visualization instance is already free for use on Jetstream and can be automated to communicate with our machine learning instance. This could potentially be used to generate high-authority contributions to national frog surveys and field station cyberinfrastructure.

Even though accuracy lagged for the audio-based neural networks, there are benefits to each of them. The preprocessing transformations to create mel-spectrogram cepstrum coefficients take up less storage than raw audio files, and these outputs also will not contain many interferences and noise. In our research, the raw

audio and necessary files for transformations makes only 2.03 GB, while the spectrogram images required 4.69 GB of space total. Mel spectrograms isolate individual features over time and frequency domain, eliminating competing noise. When we ran predictions, the overall number of audio actually used was cut down to match the CNN, thus the accuracy suffered from reduced training and testing samples. With long-term use and continuous training, the audio-based neural networks will improve more than the image-based CNN over time.

Our neural networks are one of the final necessary applications for a fully integrated frog identification that can be used both in the field or as a supplement for national frog surveys. This work could even apply to other bioacoustic analyses, such as crickets or birds. Observing the natural world through automation creates greater authority and accessibility for researchers and citizen scientists abroad.

ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grant No. 1445604 (Jetstream) and 1759906 (NCGAS). Opinions, findings, conclusions, and recommendations expressed herein are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Special thanks to Seth Adams and Dakila Ledesma's neural network resources; Jefferson Davis, Harmony Jankowski for revising and editing this paper, Carrie Ganote and Laura Huber for help setting up and debugging our neural networks, the Jetstream system administration team (Steve Bird, Mike Lowe, and George Turner); and Bhavya Papudeshi for guidance with Jetstream and Conda.

REFERENCES

[1] 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, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

[2] Seth Adams. 2018. Convolutional Neural Network - Deep Learning for Audio Classification p.5. (2018). https://www.youtube.com/watch?v=_nOu_CHogWw&list=PLhA3b2k8R3t2Ng1WW_7MiXeh1pfQJQi_P&index=5 [Online; accessed <7/17/2019>].

[3] Seth Adams. 2018. DSP Background - Deep Learning for Audio Classification p.1. (2018). https://www.youtube.com/watch?v=Z7YM-HAz-IY&list=PLhA3b2k8R3t2Ng1WW_7MiXeh1pfQJQi_P&index=2&t=0s [Online; accessed <7/17/2019>].

[4] Seth Adams. 2018. Loading Data - Deep Learning for Audio Classification p.2. (2018). https://www.youtube.com/watch?v=-GddLd2_0ok&list=PLhA3b2k8R3t2Ng1WW_7MiXeh1pfQJQi_P&index=2 [Online; accessed <7/17/2019>].

[5] Seth Adams. 2018. Model Preparation - Deep Learning for Audio Classification p.4. (2018). https://www.youtube.com/watch?v=rFt-3J0brE8&list=PLhA3b2k8R3t2Ng1WW_7MiXeh1pfQJQi_P&index=4 [Online; accessed <7/17/2019>].

[6] Seth Adams. 2018. Plotting & Cleaning - Deep Learning for Audio Classification p.3. (2018). https://www.youtube.com/watch?v=mUXkj1BKYk0&list=PLhA3b2k8R3t2Ng1WW_7MiXeh1pfQJQi_P&index=3 [Online; accessed <7/17/2019>].

[7] Seth Adams. 2018. Predictions - Deep Learning for Audio Classification p.8. (2018). https://www.youtube.com/watch?v=gfhx4dr6gJQ&list=PLhA3b2k8R3t2Ng1WW_7MiXeh1pfQJQi_P&index=8 [Online; accessed <7/17/2019>].

[8] Seth Adams. 2018. Recurrent Neural Network - Deep Learning for Audio Classification p.6. (2018). https://www.youtube.com/watch?v=Lq1rnT-MOos&list=PLhA3b2k8R3t2Ng1WW_7MiXeh1pfQJQi_P&index=6 [Online; accessed <7/17/2019>].

[9] Seth Adams. 2018. Saving Data and Models - Deep Learning for Audio Classification p.7. (2018). https://www.youtube.com/watch?v=H-X9vpARvhI&list=PLhA3b2k8R3t2Ng1WW_7MiXeh1pfQJQi_P&index=7 [Online; accessed <7/17/2019>].

[10] Seth Adams. 2019. Audio-Classification. (2019). https://github.com/seth814/Audio-Classification [Online; accessed <7/16/2019>].

[11] Anaconda. 2016. Anaconda Software Distribution. (2016).

[12] M. Araya-Salas and G. Smith-Vidaurre. 2017. warbleR: an r package to streamline analysis of animal acoustic signals. (2017). http://dx.doi.org/10.1111/2041-210X.12624

[13] Wen-Ping Chen, Song-Shyong Chen, Chun-Cheng Lin, Ya-Zhung Chen, and Wen-Chih Lin. 2012. Automatic recognition of frog calls using a multi-stage average spectrum. *Computers & Mathematics with Applications* 64, 5 (2012), 1270–1281.

[14] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. 2018. A comparison of audio signal preprocessing methods for deep neural networks on music tagging. In *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 1870–1874.

[15] François Chollet et al. 2015. Keras. (2015). https://keras.io

[16] Contrabaroness. [n. d.]. Male Green Frog - Hunterdone County, NJ. ([n. d.]). https://commons.wikimedia.org/wiki/File:Male_Green_Frog_-_Hunterdon_County,_NJ.jpg Online; Accessed July 24th, 2019.

[17] Ahmet Elbir, Hamza Osman İlhan, Gorkem Serbes, and Nizamettin Aydın. 2018. Short Time Fourier Transform based music genre classification. In *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*. IEEE, 1–4.

[18] Eliza Foran, Jazzly Anderson, Thomas Slayton, Emmanuel Guido, Thomas Doak, and Sanders. 2019. Developing a workflow for bioacoustic recording devices and frog call analysis within Jetstream. CEWiT Poster Competition.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[20] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–. SciPy: Open source scientific tools for Python. (2001–). http://www.scipy.org/ [Online; accessed <today>].

[21] Dakila Ledesma. 2019. cnn-tutorial.md. (2019). https://github.com/bgq527/intro-ml/blob/master/cnn-tutorial.md [Online; accessed <7/2/2019>].

[22] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 51 – 56.

[23] Travis Oliphant. 2006–. NumPy: A guide to NumPy. USA: Trelgol Publishing. (2006–). http://www.numpy.org/ [Online; accessed <today>].

[24] Peter Paplanus. 2018. Illinois Chorus Frog (Pseudacris illinoensis). (2018). https://www.flickr.com/photos/2ndpeter/44546804495 Online; Accessed July 19th, 2019.

[25] R Alexander Pyron. 2018. Global amphibian declines have winners and losers. *Proceedings of the National Academy of Sciences* 115, 15 (2018), 3739–3741.

[26] R Core Team. 2013. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/

[27] National Park Service. [n. d.]. American Toad (Anaxyrus americanus) NPS Photo. ([n. d.]). https://www.nps.gov/chat/learn/nature/american-toad.htm Online; Accessed July 19th, 2019.

[28] U.S. Fish & Wildlife Service. [n. d.]. Spring peeper. USFWS photo. ([n. d.]). https://www.fws.gov/chesapeakebay/Newsletter/Spring07/spring%20wildlife%2007/spring.htm Online; Accessed July 19th, 2019.

[29] Craig A Stewart, Timothy M Cockerill, Ian Foster, David Hancock, Nirav Merchant, Edwin Skidmore, Daniel Stanzione, James Taylor, Steven Tuecke, George Turner, et al. 2015. Jetstream: a self-provisioned, scalable science and engineering cloud environment. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*. ACM, 29.

[30] Craig A Stewart, David Y Hancock, Matthew Vaughn, Jeremy Fischer, Tim Cockerill, Lee Liming, Nirav Merchant, Therese Miller, John Michael Lowe, Daniel C Stanzione, et al. 2016. Jetstream: performance, early experiences, and early results. In *Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale*. ACM, 22.

[31] Guido Van Rossum and Fred L Drake Jr. 1995. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

[32] Brad Walker Wil Hershberger, H C Gerhardt. 2019. Spring Peeper, Chorus Frog, Green Frog, American Toad. (2019). https://www.macaulaylibrary.org/ [Online; accessed <7/14/2019>].

[33] Haomin Zhang, Ian McLoughlin, and Yan Song. 2015. Robust sound event recognition using convolutional neural networks. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 559–563.