Automatic Capture and Classification of Frog Calls

Eliza G. Foran[1], Tenecious A. Underwood[2], Winona Snapp-Childs[3], Sheri A. Sanders[4]

[1]Department of Biology, Indiana University, [2]Department of Computer Science, Kentucky State University, [3]Pervasive Technology Institute, IU, [4]National Center for Genome Analysis Support, IU

Introduction

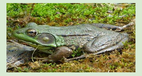
Global frog populations are threatened by an increasing number of environmental threats, such as habitat loss, disease, and pollution. Traditionally, in-person acoustic surveys of frogs have measured population loss and conservation outcomes among these visually cryptic species. However, these methods rely heavily on trained individuals and time-consuming field work. We propose an end-to-end workflow for the automatic recording, presence-absence identification, and web page visualization of frog calls by their species. The workflow encompasses recording of frog calls via custom Raspberry Pi's, data pushing to Jetstream's cloud computer, and species classification by three different machine learning models: Random Forest, Convolutional Neural Network, and Recursive Neural Network.

Goals of the Project

1. Create an end-to-end analysis system for frog surveys
2. Handle multiple frogs calling in the same recording
3. Identify variety of species, including closely related species
4. Handle unknown calls and sounds in recording (i.e. birds, talking)
5. Provide user transparent information on probability of call identification

Chorus Frog

Spring Peeper



Green Frog

American Toad

Methods - Data Collection

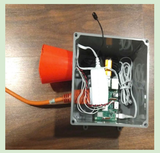


Figure 1. Open view of custom Raspberry Pi Recorder

Raspberry Pi - based collectors were assembled to provide a cheap, replicable means of recording frog calls.

- USB omnidirectional microphone for capturing sound
- USB drive for temporary data storage
- Water resistant casing
- Pre-scheduled recording times
- Optional atmospheric (temperature,and humidity) recording

Methods - Data Preprocessing

Five minute recordings were preprocessed using the warbleR package in R to automatically identify calls, using tuned parameters for call length, intensity, etc.

WarbleR extracts both isolated calls as WAV files and 26 different features call metadata measures. A total of 24,214 calls were used to train the Random Forest (9 species), and 20,264 calls were used to train the Neural Networks (8 species).

Input for Random Forests

- Call parameters were evaluated using sklearn, and only 14 of the features were integrated into the Random Forest model (feature reduction).
- Call recording month was added to capture important seasonal calling information.

Input for Neural Networks

- Individual calls were used for both Convolutional and Recursive Neural Networks.
- Training data was manually curated to remove non-frog calls from sound clips, using spectrograms, for accurate training data.
- Only sound clips over 0.5s were kept.

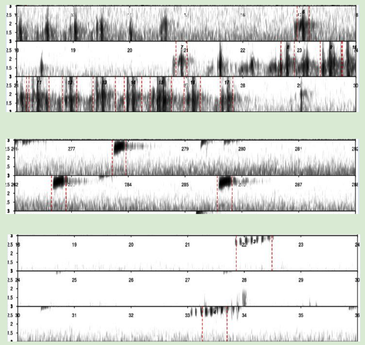


Figure 2. Automatically detected signals from *Lithobates sylvatica* (top spectrogram) and two false positives (bottom two).

Methods - Random Forest

Sklearn's RandomForestClassifier was used to develop the model.

- Models included 10-fold cross validation with a 10/90 split
- Models were tuned on training data to optimize accuracy.
- Cross validation output was visualized to select optimal hyperparameters (Table 1, Figure 3).
- Maximum tree depth was limited to 10 to prevent overfitting.

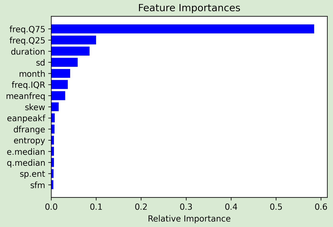


Figure 4. Feature importances from sklearn, with month added. Relative importance is calculated from gini importance.

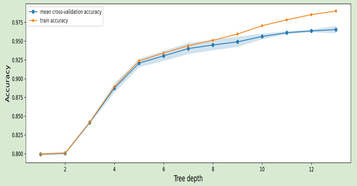


Figure 3. (left) Tuning Curve for maximum tree depth hyperparameter in Random Forest Classifier. 10-fold cross-validation was used.

Table 1. Hyperparameters of Random Forest Classifier

Hyperparameter	Value
criterion	gini
max_features	n_features
class_weight	balanced
n_estimators	100
min_samples_leaf	1
max_depth	10

Methods - Neural Networks Model

Model was developed using Keras, based on an instrument audio classifier.

The CNN and RNN required more pre-processing of call files to create sound "fingerprints" for each species (Figure 5). The filtering was done to increase the signal/ratio of the individual call files and remove white space.

The same cross validation structure was used (10-fold, 10/90 split), and class balance was adjusted in the training steps.

Convolutional Neural Network Architecture

- Commonly used for classification of Images
- Layers: 3 ReLU-activated Conv2D.
- Layers: Batch Normalization, Dropout, Flatten, 3 Dense.

Recurrent Neural Network Architecture

- Good for sound analysis due to capturing sequence
- Layers: 2 LSTM.
- Layers: 4 time-distributed Dense with ReLU activation
- Layers: Flatten,softmax-activated Dense.

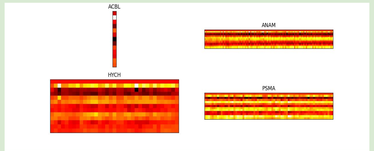


Figure 5 (left): Mel Frequency Coefficents can be seen as a fingerprint for NN's to analyze WAV files.

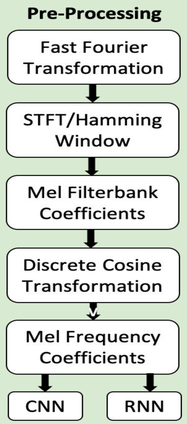


Figure 6. (above) Series of pre-processing steps applied before neural network analysis

Results

Goal 1: Create an end-to-end analysis system for frog surveys

- Developed an end-to-end automated system for analysis of frog calls using Raspberry Pi's capture the initial calls, that are then passed to the Jetstream server, processed through the WarbleR signal detection, and implemented into one of our three different Machine Learning methods within a pre-designed virtual machine (VM).
- Made models with VM's publically available on Jetstream Cloud:
  - Bioacoustic Data Collection - https://use.jetstream-cloud.org/application/images/997
  - Frog Call Analysis - https://use.jetstream-cloud.org/application/images/996

Goal 2: Handle multiple frogs calling in the same recording

- Previous Neural Network models did not accept warbleR preprocessing inputs, thus were unable to detect frogs calling on top of one another.
- All three models may now detect overlapping calls with warbleR preprocessing.

Goal 3: Identify variety of species, including closely related species

- The Random Forest model was able to accept all nine species from the training data.
- The Neural Networks accepted all except the Blanchard's cricket frog.
- Both were able to identify and separate closely related species such as common gray treefrog and Cope's gray treefrog.

Goal 4: Handle unknown calls and sounds in recording (i.e. birds, talking)

- Signals that were identified with lower than 80% positive probability were labeled as "unidentified"

Goal 5: Provide user transparent information on probability of call identification

- Output contains sound file name, positive probability of each species, and final prediction based on probabilities.

Table 2: Comparison of Classification Models

	Random Forest	CNN	RNN
Number of Species Predicted	9	8	8
CV Accuracy	97.80%	97.85%	97.38%
Build Time	12m 26s	50m 15s	51m 19s
Predict Time	0.0004s/call	0.36s/call	0.37s/call

Table 3: Multi-call Trials

Random Forest	CNN	RNN	Manual ID
HYCH, PSCR, NO ID	*HYCH, HYVE*	*HYCH, HYVE, NO ID*	*LIPI*
LISY, NO ID	*HYCH*	*HYCH, PSCR*	*ANAM, LIPI*
***ANAM**, HYCH, HYVE, LICL, PSCR, NO ID*	*HYCH*	*HYCH*	***ANAM**, LIPI*
NO ID	*HYCH, HYVE, LISY, NO ID*	***PSCR**, HYCH, NO ID*	***PSCR**, PSTR**
LISY, PSMA, NO ID	*HYCH, HYVE, LIPI, NO ID*	***PSCR**, PSMA, LIPI, NO ID*	***PSCR**, PSTR**
***PSCR**, HYVE, NO ID*	*LICL*	*LICL*	***PSCR**, LISY, PSTR**

Conclusions

- For final implementation, we recommend the Random Forest model because it detects all species, has the fastest prediction time, and is easy to optimize and rebuild
- The RNN is superior to the CNN in its ability to predict multiple calls within a single WAV file because of the LSTM layers within the RNN which allow the Recurrent Neural Network to learn at a faster rate and better distinguish multiple calls from different species.
- Future Directions:
  - Increase the threshold from 80% to reduce false positives
  - Hook up model and/or preprocessing VM's to GPU for faster analysis
  - Increase the number of species predicted
  - Improve the CNN by fixing the threshold and/or sampling rate to handle mutli-call WAV files
  - Improve the Random Forest by adding temperature of environment during recording as a call feature

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1445604 (Jetstream) and 1759906(NCGAS) Any opinions, findings, conclusions, or recommendations expressed in this material 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, Carrie Ganote and Laura Huber's 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.