Vocalization Detection
Final Report for
CS4624: Multimedia, Hypertext, and Information Access

Authors

Dat Bui

Instructor: Dr. Edward Fox

May 11, 2020
Virginia Tech, Blacksburg VA 24061

Vocalization Detection
Final Report for
CS4624: Multimedia, Hypertext, and Information Access

(ABSTRACT)

DeepSqueak is a deep-learning based system for detection and analysis of ultrasonic vocalizations. The original DeepSqueak model was created by Kevin R. Coffey, Russel G. Marx, and John F. Neumaier. Rodents engage in social communication through ultrasonic vocalizations, and Dr. Bowers is utilizing DeepSqueak's technology to study rats in his lab.
AviSoft is another software package that has been used by Dr. Bowers, to record and manually analyze sound files gathered from the rats. Dr. Bowers would like to use all available data to train DeepSqueak's classification model, to further improve its accuracy, and to reduce manual analysis and labeling work. The purpose of the Vocalization Detection project is to assist with that effort, leveraging the available data, the two software packages, and our processing.
Initial efforts involved studying DeepSqueak, AviSoft, and the available data files. Further exploration considered automating use of the tools, and helping with the training of DeepSqueak models. Then the work pivoted, to develop matching methods to take data processed with AviSoft, to transform that into labeled data to improve the training of DeepSqueak models.

Vocalization Detection
Final Report for
CS4624: Multimedia, Hypertext, and Information Access

(GENERAL AUDIENCE ABSTRACT)

This project aims towards creating a program that brings together the analytical power of DeepSqueak's classification model and the results of data labeling with AviSoft. A classification model is a way for a machine to take information and categorize it. In this instance, audio files of rat calls will be classified according to an approved list of types of vocalizations.

Acknowledgments

I would like to thank Dr. Michael Bowers (bmike1@vt.edu) for offering the opportunity to work on this project, as well as providing the data needed for the classification models.
I would also like to thank Amr Aboelnaga, a Master's student working with Dr. Bowers, for mentoring me and guiding me throughout this project.
Finally, I would like to thank Dr. Edward A. Fox for facilitating meetings and providing the opportunity to learn about this project.

Contents

List of Tables

List of Abbreviations

Faster R-CNN: Faster Region-based Convolutional Neural Network

MATLAB: Matrix Laboratory, a multi-paradigm numerical computing environment

RPN: Region Proposal Network

USV: Ultrasonic Vocalization

Chapter 1

Introduction

The Vocalization Detection project aims to allow data labeled with AviSoft to help improve the classification power of DeepSqueak. In particular, we hope to construct a training data set by transforming the data processed with AviSoft so it can be used to help improve upon the existing classification model of DeepSqueak.

DeepSqueak is used with Dr. Bowers's research on rodent vocalizations, which is a part of his research on autism spectrum disorder (ASD). Dr. Bowers has identified a certain gene variant in the FOXP family that, when expressed, leads to attributes of autism in humans, specifically regarding vocalization. He is validating its effect through CRISPR/CAS9 work with rats, studying how that variant affects rat vocalizations. The purpose of this project is to facilitate the automatic classification of rat calls for Dr. Bowers and his team.

1.1 Objective

Our objective is to improve on the classifier that is currently being used in DeepSqueak. Earlier, Dr. Bowers and his team were classifying the rodent calls manually with the aid of AviSoft, which takes many more person-hours than if they had an appropriate automatic classifier. The DeepSqueak classifier has limitations, so efforts are being made to improve upon that classifier by training it with some of the manually labeled data.
Another objective is to create software for Dr. Bowers and his team to use, so additional manually labeled data can be included in the training set.

1.2 Client

Our client is Dr. Michael Bowers, a professor from the Virginia Tech Department of Neuroscience. Dr. Bowers and his team at the Corporate Research Center gathered audio data from the rats, and manually classified them using AviSoft [1]. They are also using DeepSqueak to automatically classify the rat calls, with subsequent checking and editing to address DeepSqueak limitations.

1.3 Roles

Group Member	Role
Dat Bui	Take notes during meetings, learn DeepSqueak, and match data to add to the training set

1.4 Timeline

In this timeline we give details about milestones, both for the course, and for the overall project. The most notable roadblock has been finding out that AviSoft does not fit our needs, and that automating its operation was not feasible. Accordingly, the last part of the project involves matching data labeled with AviSoft with related DeepSqueak data, so whatever additional training data can be extracted and utilized.

Table 1.1: Vocalization Detection Project Timeline

2/3	Granted access to Kaggle data.
2/17	Process data from Kaggle.
2/20	Research alternatives to AviSoft.
3/2	Start research on how to pipeline data into DeepSqueak.
4/6	Interim Report.
4/7	Refinement of Classification Model.
4/21	Conclusion of data matching.
4/25	Final product complete.

Chapter 2

Requirements

The first goal for this project was to have an application with a graphical user interface suitable for someone who is non-technical to use. By the end of the semester, we hoped to have a working product for Dr. Michael Bowers with the capability to take in an audio file, preprocess the file, and then feed it into DeepSqueak's classification algorithm.

Originally, everything was being done manually. Dr. Bowers and his team had to get an audio file, run it through AviSoft, and use that tool to speed up their manual labeling of vocalizations. Later, they found they could run the data through the DeepSqueak MATLAB library, have it automatically label the vocalizations, and then only check and edit the DeepSqueak output.

After a thorough study, we decided against further work on automation with AviSoft. The problem with AviSoft is that it is a licensed product, and there is no practical way to automate the processing of AviSoft.

Another requirement of the project which will continue beyond this semester is to improve upon the classification algorithms for DeepSqueak. At the moment, the algorithm is not the most accurate, and so it could be improved upon. The person who is working on improving the classification is a Master's student named Amr Aboelnaga. Amr has been experimenting with other classification algorithms.

To improve the classification, a large amount of training data is needed. Since there are many files produced with AviSoft and manual labeling, the final plan is to use a matching approach to transform those files into additional training datasets to help improve the DeepSqueak processing.

We currently have a lot of data, and we need to make sure everything lines up. An issue with machine learning and classification is that sometimes one object can be described in multiple different ways. In this project, what can easily be perceived by a human as a single vocalization from a rodent may be characterized as multiple vocalizations by a computer.

In this project, data matching is important for training the classifiers. We want the DeepSqueak output to be similar to the AviSoft data. It is not important that the calls are being classified at the same exact points in time, but for this project, it is important that when they are being classified and detected that they match up [2]. For instance, if the AviSoft data shows that a call at time 2:03 is labeled as 'X', the purpose of data matching is to make sure that

the call detected by DeepSqueak between roughly 2:02 and 2:04 is labeled as ‘X’

Chapter 3

Design

3.1 Automation Study

Our first effort was to study automating operations with AviSoft. Though this effort has been terminated, this section describes that work.

From a design standpoint, we are free to go about it however we wish, but we seem to have run into a big roadblock when it comes to the design of an appropriate GUI and application for Dr. Michael Bowers. As previously stated, we are unable to use AviSoft if we want to integrate it with DeepSqueak. AviSoft is a proprietary software package that runs only on Windows, and source code is not available. The only way to automate AviSoft processes is to use automation software that relies on graphical cues, sort of like programming a computer to click at certain points on the screen at certain points in time. This is not a practical way to implement automation, and is counterproductive to what we want Dr. Michael Bowers's end product to do.

We have had multiple design ideas and implementations that we have attempted. Our first idea was to have AviSoft to write to a database, and then to feed that data into DeepSqueak from the database. The program would be a simple pipeline from the raw data of the vocalizations, into the classification network of DeepSqueak. We decided that this was not an appropriate way to proceed, because having to set up a database, feed it the output data, and then feed it into DeepSqueak would not actually save much time, if any. Our next approach was to try and automate the AviSoft process of getting a file, outputting data, and then feeding it into DeepSqueak. The automation process was attempted using PyWinAuto, but the issue was that PyWinAuto relies on graphical UI interaction [4]. This was not practical, nor was it really the automation that we were hoping to get. Our last attempt with automating AviSoft was to reverse engineer the software. The team attempted to use NSA's open source reverse engineering software known as "Ghidra." Ghidra works by disassembling the source code into raw assembly, and attempts to piece it back together into C code [5]. However, that proved to be cumbersome, ultimately impossible since we had no experience in reverse engineering disassembled code. Since AviSoft is licensed software, with no open source code available, we are looking into other software that can give us the same functionality as AviSoft.

3.2 Matching Study

See Section 4.3 and Section 5.4 for more about the work on matching.

Chapter 4

Work Progression

Throughout the semester, there have been multiple shifts in effort on the project. Initially, there was a plan to pipeline AviSoft's output with DeepSqueak's input. This process would have involved a database that takes in the output data of AviSoft, from some audio file, and then would have fed the data into DeepSqueak for classification. Then, focus shifted to attempting to automate AviSoft. These proved to be unsuccessful, so the team moved away from using AviSoft. At that point, my focus shifted to matching data.

4.1 Integration of AviSoft and DeepSqueak

At the beginning of this capstone project, my main task was to find some way to integrate the two technologies of AviSoft and DeepSqueak. Classification labels and data from AviSoft has to be put manually into DeepSqueak. Initially, we wanted to have a database that stores the AviSoft output and manual classifications, so as to cross-validate with the new classifier being developed. This procedure would save a lot of time in the improvement of the classifier network, because it takes out the busywork of having to actually grab the data. The database approach was thought to be a good way to go, but then we decided we would rather automate the process of pipelining AviSoft data into DeepSqueak. Next we proceeded to try and automate the processing of AviSoft.

4.2 Automation of AviSoft

Automating AviSoft began with attempts to use a Python library known as PyWinAuto to try and get the AviSoft data without any user input. This approach was clunky and challenging. PyWinAuto works by scanning the windows and dialog options to perform actions [4] . This is not what we had in mind for automating processes, as it is not practical; it can not be run in the background, and it takes up too many resources. From PyWinAuto, we decided to move on and try to reverse engineer AviSoft. Reverse engineering was attempted by using a tool created by the NSA called Ghidra. Ghidra disassembles the code, and reconstructs it in C format [5]. Unfortunately, this also proved to be extremely difficult, as the whole process of reverse engineering AviSoft would take a lot of time, and expertise

in reverse engineering. For now, we have decided to move away from AviSoft, and the new technology is to be determined. Moving forward, the team will be focused on data matching.

4.3 Data matching

At the moment, the team's task is to assist Amr Aboelnaga with his research by performing data matching on the data that we currently have. The data we currently have consists of manual detections from Dr. Michael Bowers's team, and the DeepSqueak detections.

Chapter 5

Implementation

5.1 Data

Dr. Bowers and his team periodically supply us with data to use from the rat vocalizations. The files are usually output data from AviSoft, as well as the input WAV files. We then run the WAV files through DeepSqueak to get a classification. The classification shows us whether or not our neural network is on the right track.

5.2 Detection

Detection of the rodent calls is done using a Fast Region-based Convolutional Neural Network, or Fast R-CNN for short. Fast R-CNN is a state of the art neural network used in object detection in photos, so why are we using it to detect sounds? In essence, a WAV file can be interpreted using an image, on which we are able to train a Neural Network to detect different types of sounds, i.e., vocalizations. The WAV files usually contain a lot of blank space, with occasional wave changes to signify an ultrasonic vocalization.

5.3 Classification

Classification of the rodent calls is currently being done with an unsupervised k-means clustering model in the original DeepSqueak [3]. K-means clusters together datapoints and classifies them based on what cluster they belong to. Other methods of classification are being experimented with, as k-means clustering does not suit our purpose as well as other methods. K-means clustering is good for when all the categories can be placed into distinct clusters. For example, considering letters and numbers, the letter 'A' is very distinct from 'B', and the number '0' is distinct from '5'. K-means clustering would not work with USV classifications, because there are more minor differences between each category that are not pronounced enough to place them in distinct clusters. Amr Aboelnaga is currently working on a Wright Classifier Network, which is better suited for DeepSqueak's USV classifications.

5.4 Data Matching

In the context of machine learning, data matching is the process of matching pieces of information in large sets of data. The purpose is to find duplicates, or find subjects that are related to each other in the database. There are two types of approaches to data matching, deterministic, and probabilistic. Deterministic data matching aims to say definitively whether or not two objects match up with each other. For instance, with something like driver's licenses, deterministic data matching would be like having two people named "John Smith", and determining whether or not they are the same person by checking their driver identification number. Probabilistic data matching is using multiple data fields to calculate the probability that two objects match up with each other. For instance, given one homework assignment given to two different students, there isn't a data field that definitively says whether or not they cheated, but someone may look at different details, such as how similar their answers are, to determine the probability that the students cheated [2]. For DeepSqueak, the team will be focused on probabilistic data matching, since there do not seem to be any fields that outright determine whether or not two USVs are the same or not.

Chapter 6

Future Work

In the future, we hope to improve on the classification model further. This is an ongoing problem that will extend for many semesters after this. We also hope that there will be a more robust, open-source technology that can give us similar raw data to what AviSoft gives us.

The data matching will continue into the future, as these are massive datasets that we are working with, and it is important to be able to properly match together the many different pieces of information we are given. As work goes forward, Dr. Bowers will continue his research on the rats, and this project may expand into more complex territories beyond Ultrasonic Vocalization classifications.

Chapter 7

User's Manual

This manual will describe how to use the DeepSqueak MATLAB package.

7.1 Prerequisites

The user should have MATLAB 2019a installed with the following toolboxes installed:
Computer Vision System Toolbox™
Curve Fitting Toolbox™
Image Processing Toolbox™
Deep Learning Toolbox™ (formerly Neural Network Toolbox™)
Parallel Computing Toolbox™
Signal Processing Toolbox™
The user should also have a dedicated GPU installed on their PC.

7.2 Setting up MATLAB and DeepSqueak

The user should clone the DeepSqueak repository using the command
git clone https://github.com/DrCoffey/DeepSqueak.git
In MATLAB, simply input the command DeepSqueak to import and run the program.

Chapter 8

Developer's Manual

The users that this system needs to support are non-technical people. They do not need to know a lot about programming to use this system, but they do need a basic understanding of how to navigate a program. The goal of these users is to be able to upload a sound file, and get it classified with a high degree of accuracy.

8.1 Main Goal

The main goal of this project is to provide a program that is user-friendly. The tasks include refining the classification model, pipelining raw data into the classification model, and to build an interface that asks as little from the user as possible.

8.2 Refining the classification model

For this task, the data needs to be analyzed, and data for training and validation need to be created.

8.3 Analyzing the data

Given the data, it needs to be fed into the classification model, and then using the results of those classifications, the model needs to trained on the training dataset, and have its performance measured by the validation dataset.

8.4 Creating training and validation data

To create the training and validation datasets, the data, in this case WAV files, need to be classified by hand.

8.5 Pipeline raw data into the training classification model

For this task, an alternative to AviSoft must be found, that program must be automated using its source code, and it should be integrated with DeepSqueak.

8.6 Finding an alternative to AviSoft

There must be an alternative found for AviSoft that accomplishes the same features desired by Dr. Bowers' team. The alternative must also have source code that is available to the public.

8.7 Automation and integration of the program

Automation of the program can be done by directly using the source code. Integration can be done by modifying the source code to work with DeepSqueak's source code. Building a user-friendly interface For this task, the optimal parameters must be found, as well as research on UX and UI. Since this program should be suited for non-technical users, the required input should not be more complex than it needs to be.

Finding the optimal parameters The user should be able to use the interface with very little input. When the user provides an audio file, it is expected that they receive the best possible classification.

UX/UI Research Consultations with Dr. Bowers' team should be made to clarify the exact specifications of the interface.

8.8 Prerequisites

The user should have Python 3.7 installed, as well as NumPy, pandas, and SciPy.

8.9 Installation

The user should first install Anaconda by going to https://https://www.anaconda.com/ and following the installation instructions. After installing Anaconda, the user can install NumPy, pandas, and SciPy by typing the following commands in their terminal:

conda install pandas
conda install -c anaconda scipy
Upon the installation of these libraries, the user will be able to run the matching script.

8.10 Running the matching script

The user should run the matching script by first browsing to the folder in which the Python script is installed via the terminal. They should then type in the following: "python path_to_labeled_excel_file path_to_deepSqueak_classified_file" with the appropriate paths for each of the variables.

8.11 Editing the matching script

The matching script can be viewed and edited in any text editor. Simply open the .py file in any editor of your choice.

8.12 Structure of matching code

The matching script takes in two files, an .xlsx file which contains data from Avisoft, and a .mat file, which contains the classification data from DeepSqueak. The .xlsx files contain thousands of rows of data, as well as many columns of categories, but the categories that should be focused on are the 0th unlabeled column, and the column labeled "duration". The 0th unlabeled column is where the manual classification is located, and the "duration" column can give us an idea of the running time. The running time is kept by going through the rows and summing up the numbers in the duration column; the AviSoft file does not keep track of the time, only the durations and intervals. Maintenance should be done whenever the format of the input files change.

Chapter 9

Summary

In summary, this project has been an interesting foray into the insights of neuroscience, and how it can tie in with concepts in Computer Science. DeepSqueak was interesting to work with, and Dr. Bowers's research on the FOXP gene is groundbreaking. It has been a privilege to work on this project. This is a project with much more work still to be done, and by the end of it all, there are exciting discoveries to be made.

Bibliography

AviSoft Bioacoustics (2020). Avisoft. http://www.avisoft.com/.

Christen, P. (2012). Data matching. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*.

Coffey, K., Marx, R., and Neumaier, J. (2019). Deepsqueak: a deep learning-based system for detection and analysis of ultrasonic vocalizations. *Neuropsychopharmacology*, 44.

McMahon, M. (2006–2018). Pywinauto. https://pywinauto.readthedocs.io/en/latest/.

National Security Agency (2020). Ghidra. https://ghidra-sre.org/.