# *Embedded System with Machine Learning for Black-tailed Prairie Dog Warning Call Detection and Data Collection in the Field with Voice Recognition Control*

Frederick Pelon
Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, CO, USA
fred.pelon@colostate.edu

***Abstract*—Machine learning is being used to identify many natural phenomena today such as birds from their calls [1]. These inferences identify species based on audio samples collected via smart phone. This project proposes a further step: attaching meaning or context to the animals' verbalizations. This project also adds a research dimension to the project by collecting audio samples deemed interesting by the user via model training data for further model training and classification. This project is specifically trained for the black-tailed prairie dog. Field data was collected and subsequently used to add a class to the trained network.**

***Keywords—machine learning, prairie dog, spectrogram, voice recognition***

## I. Introduction

Field researchers studying prairie dog behaviors have identified vocalizations that can be described as language [2]. The warning calls indicate predator activity including descriptions of the possible predator. Field researchers have typically observed prairie dog behavior from a blind (to characterize non-warning behaviors) or by provoking them with different types of volunteers of differing descriptions (short, tall, color of clothing, etc.) These observations have all typically been recorded and annotated in the field and subsequent correlating signal processing performed post-data collection in the laboratory. In lieu of prairie dog vocalization data labeled with warning bark source, I have chosen to classify prairie dog warnings based on physical location of the source colony given the labeling of available data. My solution provides real-time feedback of prairie dog vocalizations with a facility to store interesting vocalization samples to re-train the machine learning model with an eye to adding classifications to the detection and inference model. I have trained my model with recorded prairie dog warning barks from three distinct colonies of prairie dog [3]. I have also trained a second copy of the same audio machine learning model with a small vocabulary of command words used to control the Convolutional Otological Network (CON) enabling headless operation to extend battery life for extended field work.

In this paper I will show:

- An accurately trained convolutional neural network based on available data.
- A similarly trained copy of this neural network trained to recognize an attention phrase and command words used to control CON.
- Data collected in the field using CON to retrain the model and add a class.
- Data collected in the field with the newly trained model with inference accuracy results.

## II. Related Work

Scientists studying prairie dogs' behaviors have recorded prairie dogs' warning barks in the field and have even identified characteristics of the subject of the warning [4,5,6]. Experiments were conducted with test subjects as predators inducing warning barks by traversing the area of the prairie dog's colony. The subjects were given varying colors of articles of clothing to wear as they traversed the colony. The color of the articles of clothing each subject wore were deduced from the recorded barks by analyzing them in the lab. Further information could also be deduced from this data such as the subject's physical features like tall or short, blond, or brunette [7]. These observations and analyses formed the basis for further experiments and illustrated the rich content available in a prairie dog's warning barks. There also exist papers with datasets from data taken in a blind in the field that characterized the full gamut of prairie dog behaviors such as grooming, socializing, and fighting [2]. Again, long hours of observation and careful notetaking were necessary to correlate each behavior with a recording. In the realm of machine learning and animal vocalizations, Cornell has created an app (Merlin) that uses a smartphone to record and identify bird species from their calls or songs [8]. Since the Macaulay Library - the bird library associated with the Cornell Lab of

Ornithology - and its machine learning algorithms and code are open source[9], classifying bird behaviors is easily conceivable. My solution uses an embedded system designed to collect data for scientific research negating the need for many of the features available on a smartphone such as an internet connection.

## III. Equipment

I started with a Raspberry Pi 4 Model B with 8GB of RAM. The processor is a Broadcom BCM2711 which is quad-core ARM v8 Cortex-A72. I added a Coral.AI USB Accelerator which is a co-processor known as a Tensor Processing Unit (TPU) meant to accelerate machine learning inference. This device requires a TensorFlow Lite model that is completely quantized to 8-bits including the input and output [9]. I had implemented this quantization scheme, and used the suggested tool maintained by google to compile the model specifically to be run on the accelerator but had very poor performance in inference. My initial design point was to run the processor on the Raspberry Pi to perform inference on the voice recognition model and then run the prairie dog detection model on the TPU simultaneously. In the end, I used the Coral tools and just switched models between inference needs, control or collect. The Coral tools claim that if the accelerator is presented with an uncompliant model, the model would be loaded to the main processor and inference run there. Using un-quantized TensorFlow Lite models for inference gave acceptable performance. Next, I added a Yeti Blue USB microphone. This device has a frequency response of 15 Hz - 22 kHz or normal human hearing range. I had explored higher frequency response devices, but these are specialty devices and the several prairie dog papers I read showed them using normal microphones. A possible area of future research is whether or not prairie dogs' communications extended beyond normal human hearing range and whether having that information would assist in classification. Future idea. The machine is finished off with a set of speakers and a USB-C 30 Ah battery pack. Headphones may be a better choice for the future to not interfere with prairie dog behavior as much as possible.

## IV. Firmware

The controlling software for CON consists of a simple state machine written in python running in the Raspberry Pi's Linux distribution from an onboard micro-SD card which also serves as the hard storage for this device. Raspberry Pi's OS is standard Linux based on Debian. The state machine starts from bootup using the cron table and the first state is the voice recognition portion of CON asking for command input. Several commands are implemented in CON, but for the sake of field trials, one command triggers inferring and recording prairie dogs, and another quits. The other commands are ignored, and the command input state is re-entered until an implemented command is encountered. Once the command to infer and record is given, the state machine cedes control to the recording portion of the software. This sub-state machine waits for audio to be detected and then performs inference on the detected audio. The original recording is of 30-second duration and the inference is performed on 3-second chunks of that 30 seconds. When prairie dog warning calls are detected in the chunks, the audio samples are saved to hard memory in an endless loop. Otherwise, control returns to the voice recognition and control state. Silence or audio without prairie dogs detected are return-to-control cases. See Figure 1.

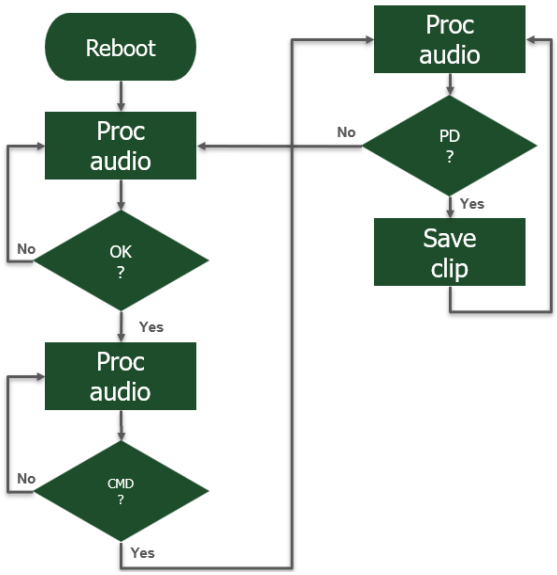



Fig. 1. Flowchart for CON's firmware.

.

## V. Data

I was given data from [3] which was a series of recordings of black-tailed prairie dog warning calls split and labeled into three different colonies. The files were all .WAV files. I processed these audio files into 3-second chunks where each sample had been normalized such that all samples would be of equal maximum amplitude. The quiet sections of each recording were culled. I sorted these samples into a directory structure such that tensorflow would recognize the directories as classes [11]. I added another class to the data called Other. These were .WAV audio samples of various random sounds from the web (horns honking, chickens, etc.) and added my voice in as well. The Other class serves to differentiate between prairie dog vocalizations I collect in the field and want to save versus data to be discarded. The Other data is pre-processed the same as the prairie dog data. This data is then transformed into spectrograms for presentation as 2D data to the machine learning model. My initial resolution for the spectrograms was 129 x 129 pixels which worked very well to train my model, but inference proved to be too unreliable on my system. One issue was signal processing. Since the tensorflow short-time Fourier transform (STFT) was not available on my inference device, I used the equivalent STFT function in the python scipy library. But these two functions were not equivalent in how they calculated the transform. Moving to a much higher resolution of 517 x 513 for the spectrograms solved this problem. The data for the voice control portion of CON was generated by

instances of my recording the commands (and start phrase) myself, the data was pre-processed the same as the prairie dog data and the model trained the same way. Figures 2 and 3 show examples of time domain and spectrograms for an example of the Other class and one of the prairie dog classes.

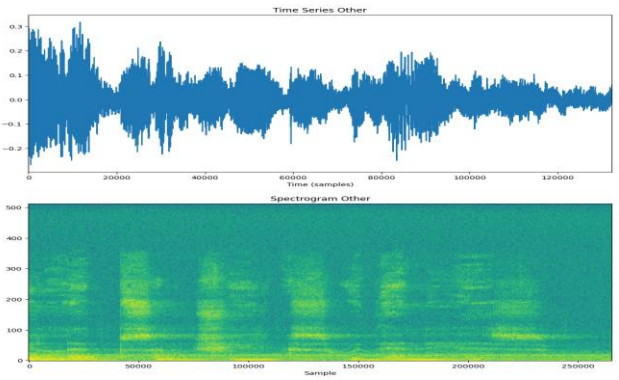



Fig. 2. Other time series and spectrogram.

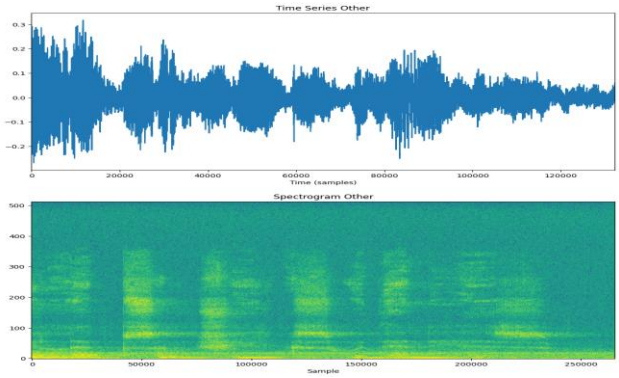
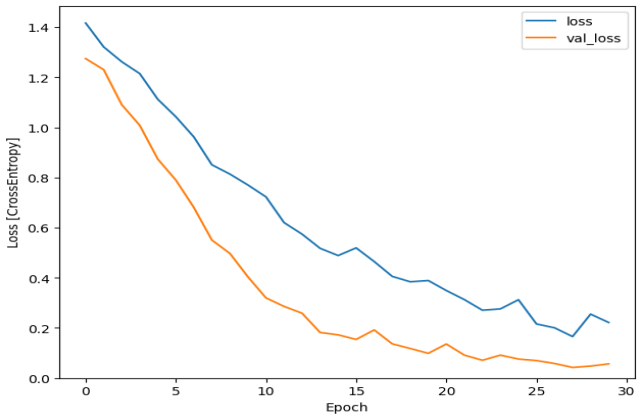
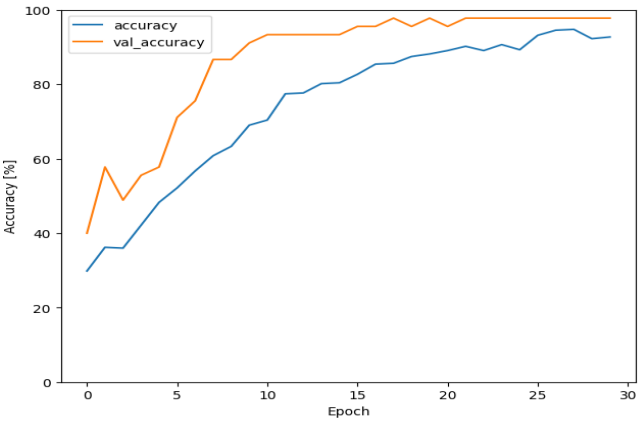



Fig. 3. Other time series and spectrogram.

## VI. Methods

I used a Jupyter Notebook to pre-process the data and train the model. I started by following the tensorflow example given in TensorFlow's online resource [11] and modified each step along the way to accommodate my unique data. I also desired a higher accuracy than the example achieved, so I modified the simple model given to be more complicated to achieve that higher accuracy. My model started overfitting with this more complicated model, so I added more regularization in the form of max pooling layers and dropout layers. I also adjusted the amount of dropout used to contribute to the necessary regularization. I also added a softmax layer at the output to better gauge inference effectiveness given the normalized class outputs. Without this softmax layer, when I performed inference on the device, the score given to each class by the inference was not distributed between 0 and 1 which would have necessitated gaining experience with the data and values to understand relative score of each classification. See Figure 4 for a plot of the final model. I used 150 samples of each class for training data. The training data was all recorded at a 44.1 kHz sampling rate mono in a 16-bit WAV format. The recordings made with CON all followed the same format. I followed the labeling nomenclature of the original recordings for site designations yielding the classes as CRE, PRE and TCE mono as the recording standard. This precludes generalization to any speaker, but for the sake of this project, success of voiced word inference was demonstrated. The models were then converted to Tensorflow Lite models for running inference on CON. The incoming data is processed. The data was split into training and validation at an 80%/20% split. The validation data was further split into a test set to determine overall model accuracy. Using TensorFlow's fit() function with a learning rate of 0.001 using the Adam optimizer, the model was successfully trained to discriminate between prairie dog classes with a > 96% accuracy compared to the 85% given by the example network. See the loss and accuracy training plots in Figure 5. Figure 6 gives a confusion matrix further illustrating the high accuracy of this trained model. For the second model used for voice recognition, I used an exact copy of the prairie dog model and trained that model with a small database using phrases that I recorded in my lab with my own voice. I recorded and trained



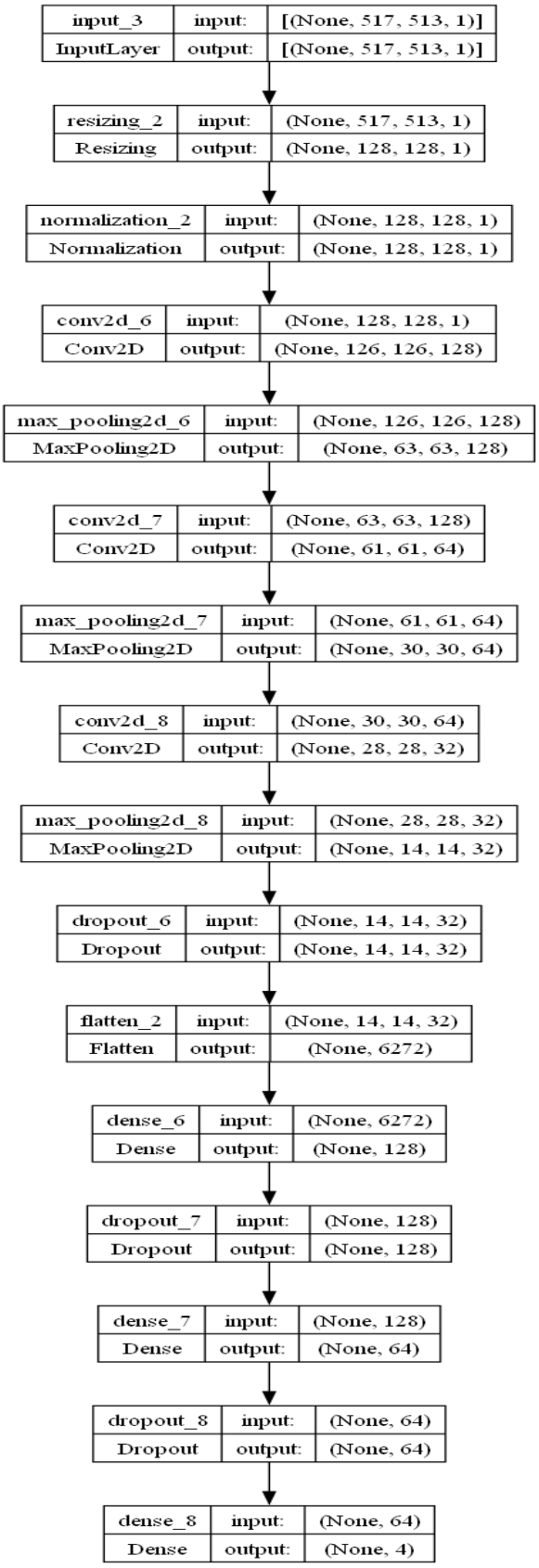



Fig. 4. 2D Convolutional Neural Network model.

for "OK CON" (attention command), "Play", "Save" (state machine exit command), "Infer", and "Record" (state machine continue command). See the confusion matrix for CON's voice recognition in Figure 7. Again, I used a 44.1 kHz sampling rate, in a similar fashion to the original training data. First, the data is tested for a minimum amplitude value. If the entire field sample is low amplitude the simple state machine flags silence and CON returns control to the voice control algorithm. If the data has some meaningful audio content, the entire 30-second sample is amplitude-normalized by the maximum sample value and amplified by a fixed value. In this way, quiet sections of the sample can be tested for audio energy and discarded if below some threshold. The remaining parts of the sample are then run through the classifier. Since this device was created as a tool for field research, careful attention was given to energy consumption considerations. Implementing a headless system obviated the need for a power-hungry display [12].

## VII. Experiments

I performed experiments with CON. The first experiment also served as CON's debut field trial. In this experiment, CON was used in the field at a fifth site – Jasper Farms (JF) [13] and data was collected for various locations in and around the pastureland inhabited by this colony of prairie dogs. The hardware and software worked as expected in the field. An important first test in this field trial was whether or not CON would boot up and work as expected since this machine was not only headless, but untethered as well – there was no internet connection in the field. CON inferred, detected, and collected prairie dog bark data for several hours and this data was brought back to the lab for download and inspection. The resulting raw data was curated and examined for missed detections and while most of the data was indeed prairie dog barks there was not a 96% inference accuracy. This was determined empirically. After culling the unusable audio samples, this data was used to re-train the prairie dog model including this new fifth class. The model performed in training almost as well as with the original data and achieved 92% accuracy. Note that the confusion matrix shown in Figure 8 denotes no mis-detects for this new class. For the second experiment, I used this newly trained model with the JF class added to again detect the same set of prairie dogs' warning barks. The firmware had performed as expected in the first experiment and was used for the second experiment at this same revision level. For this field trial, the recordings were field-curated such that inference was only run while prairie dogs were actively emitting warning barks so as to hasten the data dissemination back in the lab. CON recognized the JF class in 71 of 113 samples yielding a 63% accuracy for inference. Looking at Table 1, we can see that the TCE site class was the second most identified colony at 26%.

## VIII. Conclusions

The first conclusion is that the hardware and software worked as expected via the first experiment showed and that the device performed admirably in the field as hoped and recorded prairie dogs' warning barks in the field The second conclusion

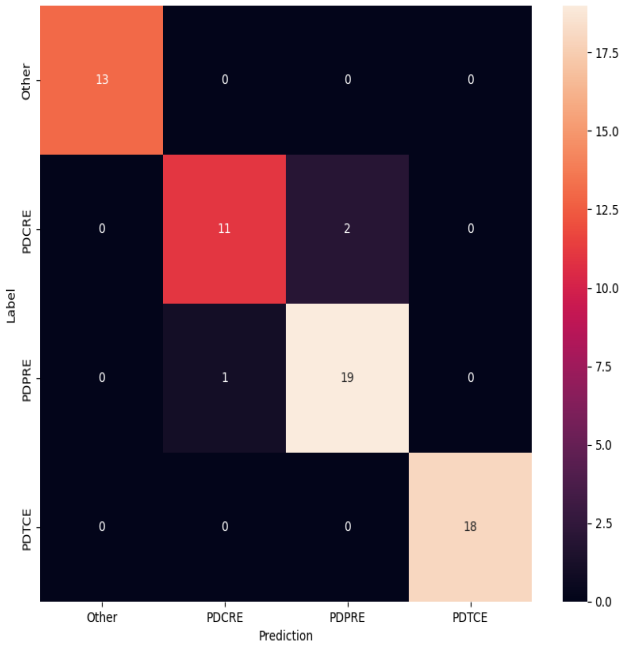



Fig. 6. 4-class prairie dog confusion matrix.

shows that the data collected during the first experiment from a new class could successfully be used to introduce a new class to the machine learning model. The third conclusion is that this newly-introduced class performed well in field trials but did not achieve the accuracy of the training data with the Tensorflow model. The balance of conclusions I drew from this machine and its field trials take the form of notes and observations from this project. There were several field notes that highlighted shortcomings in the current implementation. CON's inferences on the spoken commands were empirically less reliable intermittently. There were occasions that the voice recognition

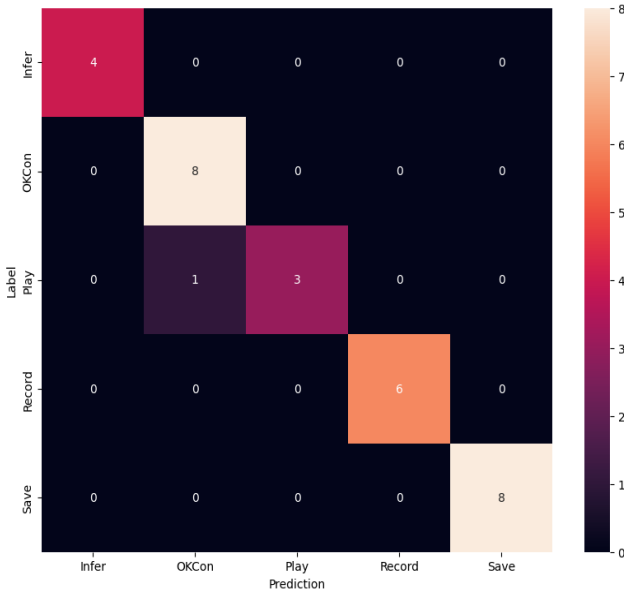



Fig. 7. Voice control confusion matrix.

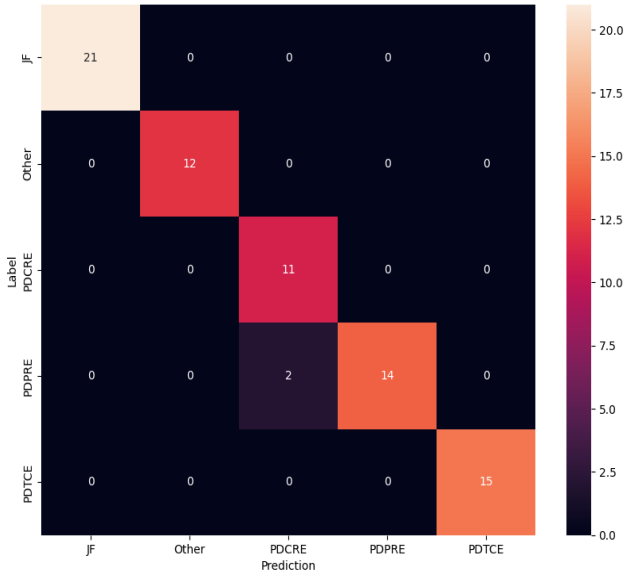



Fig. 8. Confusion matrix with additional prairie dog class from collected data.

worked perfectly. At other times, there seemed to be a condition that the machine entered a state where no inference was possible. Future versions of this machine should include reliability testing and evaluation for more robust firmware versions. I suspect the amount and homogeneity of the training data contributed to this. The training data was all recorded in a studio-like setting for example. Often recordings were made that should have corresponded to prairie dog warning barks but obviously were not. I suspect this was due to the dearth of the Other class of sounds. I learned that I should have included gate opening and footsteps for example as these are common sounds on the farm and that I activated during the first experiment for the express purpose of testing the Other class. There were occasions too where the software logic did not follow the code CON could be made more robust. Perhaps a Linux kernel could be generated to specifically work for this system abandoning extraneous support functions and streamlining operation for speed, space, and energy use.

TABLE I. INFERENCE ACCURACY FOR FIVE CLASS PRAIRIE DOG CLASSIFIER

| *Colony detected* | *Number detected (of 113)* | | *Percentage detected* |
|---|---|---|---|
| *JF* | *71* | | *63%* |
| *TCE* | *29* | | *26%* |
| *CRE* | *10* | | *9%* |
| *PRE* | *3* | | *3%* |

## IX. Acknowledgements

I need to thank Dr. Graeme Shannon at Bangor University in Wales who provided the initial database of prairie dog warning calls that formed the basis of my trained model. I chose to classify the data based on location of colony for ease of implementation, but the data contains a wealth of labels that could prove to form the basis for more research with machine learning and prairie dogs.

I would also be sorely remiss if I did not thank my wife Amber Pelon who pointed out this class to me and offered invaluable feedback for this paper and its associated presentation.

## References

[1] Cornell Lab, Merlin, merlin.allaboutbirds.org

[2] J. Connell, L. Porensky, A. Chalfoun, and J. Scasta, "Black-tailed prairie dog, Cynomys ludovicianus (Sciuridae), metapopulation response to novel sourced conspecific signals", Animal Behaviour

[3] Shannon, Graeme et al. (2019). Data from: Vocal characteristics of prairie dog alarm calls across an urban noise gradient [Dataset]. Dryad. https://doi.org/10.5061/dryad.vmcvdncp9

[4] Slobodchikoff, C. N. and R. Coast. 1980. Dialects in the alarm calls of prairie dogs. Behavioral Ecology and Sociobiology 7: 49-53.

[5] Slobodchikoff, C. N., C. Fischer, and J. Shapiro. 1986. Predator-specific words in prairie dog alarm calls. American Zoologist 26: 557 (Abstract)

[6] Slobodchikoff, C. N., Judith Kiriazis, C. Fischer, and E. Creef. 1991. Semantic information distinguishing individual predators in the alarm calls of Gunnison's prairie dogs. Animal Behaviour 42: 713-719.

[7] Radiolab program on NPR, https://www.npr.org/transcripts/132650631

[8] The Cornell Lab of Ornithology https://www.macaulaylibrary.org/machine-learning/

[9] BirdNET-Analyzer user's guide, https://github.com/kahst/BirdNET-Analyzer/blob/main/README.adoc#technical-detail

[10] https://www.tensorflow.org/lite/performance/post_training_quantization

[11] Tensorflow examples, https://www.tensorflow.org/tutorials/audio/simple_audio

[12] https://www.usenix.org/legacy/event/atc10/tech/full_papers/Carroll.pdf, An Analysis of Power Consumption in a Smartphone, A. Carroll, G. Heiser

[13] Jasper Farms, this paper's author's home, a small beef cattle and vegetable farm in Berthoud, CO