Robot Emotion Recognition

Anna Buchele and Ariana Olson

The Goal

The goal of our project was to program the robot to detect emotion in a person's voice, and respond positively to positive emotion, and negatively to negative emotion.

The Learning Algorithm

Inspiration

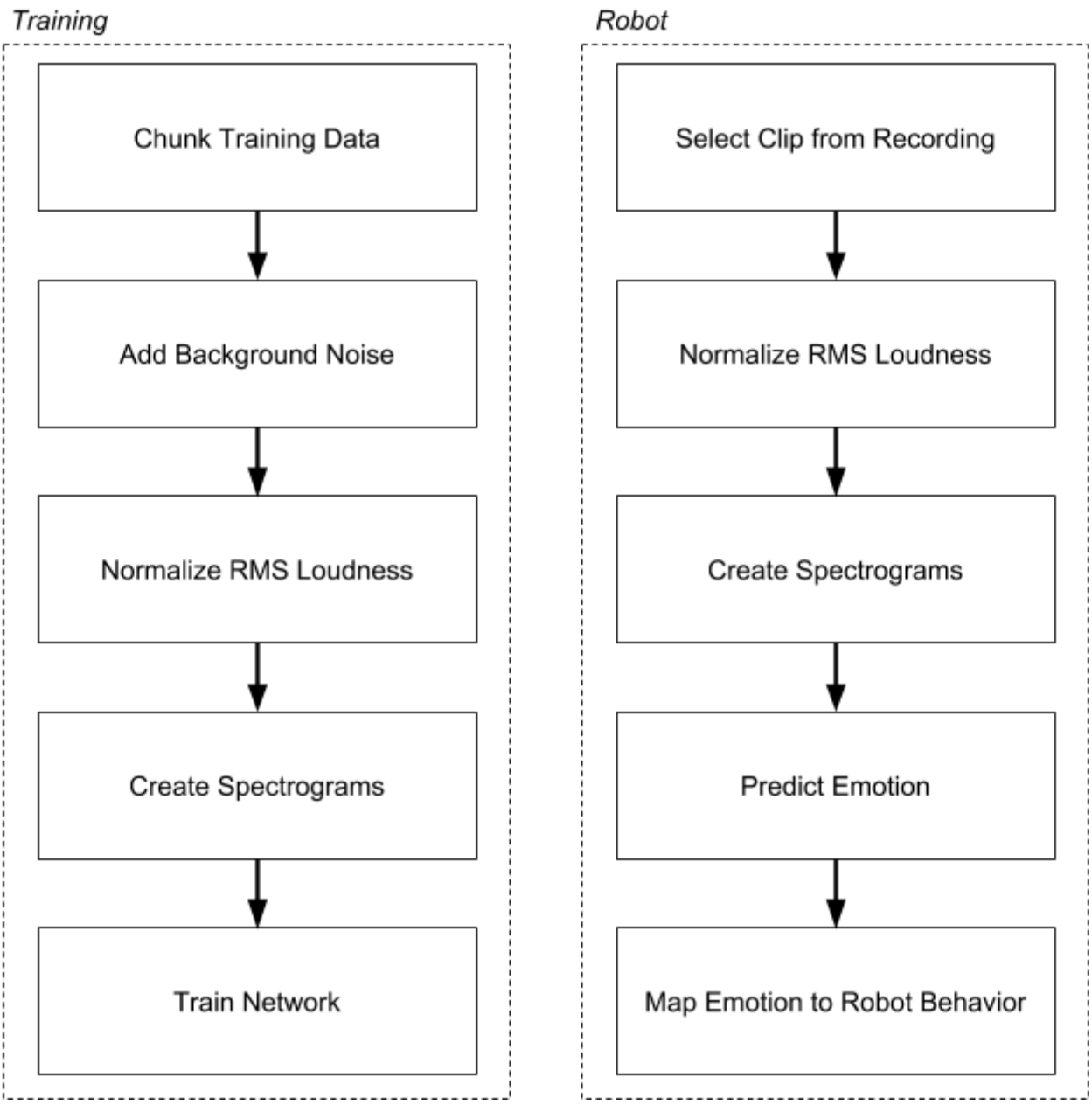
We started by taking inspiration from the paper "Speech Emotion Recognition Using Convolutional Neural Networks" (linked here) that had promising results with emotion recognition with convolutional neural networks. Convolutional neural networks seemed like a promising solution to this problem because audio spectrograms can be thought of as images, and CNNs are widely used for image recognition. In the paper, several datasets of actors speaking sentences with different emotions were used to train the network.

Datasets

We chose to use the Surrey Audio-Visual Expressed Emotion (SAVEE) dataset because the actors were English speakers, and it is free to use and easy to download. The dataset consists of recordings of 4 male voice actors speaking 15 sentences for each of 7 emotions: anger, disgust, fear, happiness, sadness, surprise, and neutral. For our purposes, we grouped the emotions into two categories. "Positive" emotions were happiness, neutral, and surprise. The "negative" emotions were anger, disgust, fear, and sadness. We also used the background noise recordings from the TensorFlow Speech Commands dataset. The dataset contained both background noise recordings and computer generated background noise, but we only used recorded noise for our purposes. This dataset was chosen because it contained several files of recorded background noise and was easy to access and use.

Block Diagram

The block diagram of our solution can be seen below.



Chunk Audio Data

The first step was to divide the dataset audio into smaller chunks to make spectrograms out of. This was done in order to have more data to use for training, to standardize the size of our spectrograms, and because we assumed that the network would not require the full audio recording in order to classify emotions. This step was done both on the dataset audio and on audio recorded from the robot.

Add Background Noise

Second, we added background noise to the chunked audio. Random sections of background noise recordings from the Speech commands dataset were added to the samples in order to make the network more robust against background noise, which we anticipated having a fair amount of when using recordings from the robot. Adding background noise was only done for the training data and not on the recordings from the robot.

Standardize Data

The final step before creating spectrograms was to normalize the loudness of the audio. This was done in order to reduce bias the network may have to classify emotions on the volume of the sample alone. We calculated the overall RMS loudness of each chunk of audio (with noise added) and normalized the level down to a target of -24dBFS. This target was chosen because it was quieter than most of the samples, which minimized the need to boost signals which comes with the risk of clipping. Normalization was done on both the training data and recordings from the robot.

Spectrogram

Finally, we converted the processed audio into spectrograms. The spectrograms had a window size of 0.128s, and an overlap of ¼ the window size. These spectrograms were all generated at once and saved in order to speed up the training process when fitting the learning model.

Learning Model

We implemented our own deep learning network to solve this problem. The network consisted of convolutional layers paired with max pooling layers and dropout layers as well as a final dense layer. We tried several architectural variants using this pattern, but we did not see significant success with any of these, so we cannot recommend a specific architecture. The output layer had two neurons: one that gave a “positive” score, and one that gave a “negative” score. Each score was between 0 and 1.

Integration with the Neato

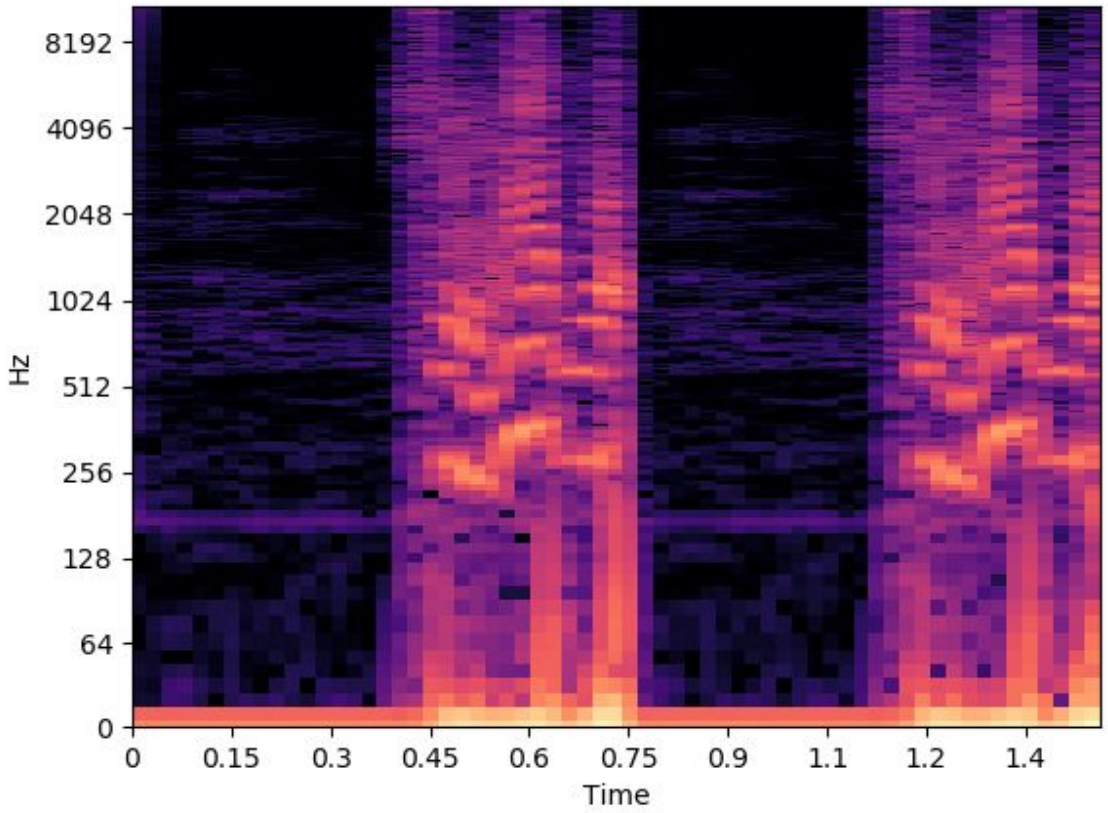
The trained model was then used on the robot. The robot would record input from the microphones, and this input would be processed according to the steps outlined above. If the positive score was higher, the robot would move forward. If the negative score was higher, the robot would spin in a circle. In the event that the positive and negative scores were equal, the robot would not move.

Comparison with a “Classical” Approach

For comparison, we trained a SVM classifier for emotion recognition. Using input from a single speaker, we were able to reach accuracy of 88% in identifying positive or negative emotion. However, when we trained it on all four speakers, the accuracy dropped to 61%, which is barely better than chance. We think this is why we had such trouble getting the neural network to work - there was simply too much variation in how the speakers expressed emotion, and too wide of a range of emotion, for our network to successfully discern positive and negative emotion.

Design Decisions

One of the design decisions we made was for what kind of machine learning we would use, and how we would input the data. We ended up using spectrograms because we felt that a spectrogram was a good way to characterize the “feel” of a sound, and might easier approximate it to an emotion than other ways of characterizing audio data. We also found in our research that many other people had used spectrograms to characterize emotion in audio, so it seemed like a good place to start.



Above, an example of a spectrogram input into the network. The speaker was speaking the sentence "Kindergarten children decorate their classrooms for all holidays." with anger. The spectrogram only represents a segment of the utterance.

Challenges

We had a lot of trouble getting our neural network to do better than chance accuracy. Our limited amount of data meant that there was a lot of variability in the performance of our model based on how the data was partitioned, and we kept running into problems with overfitting. Also, creating the spectrograms took a bit of time, so the fact that we had to remake them whenever we made changes to our processing made things more difficult. We tried varying our data input, by shortening or lengthening the audio files, or by increasing or decreasing the amount of noise. When that didn't help, we made a lot of changes to the neural network, but that also didn't increase the accuracy.

Improvements for the Future

If we had more time, we would tweak the neural network more to try and make it work. Or perhaps we would examine our input data and try with a different dataset.

Lessons Learned

One thing that could have helped this project succeed is to have smaller, more easily represented goals. It might have been better to start out the neural network with a test of distinguishing two very visually (in spectrogram) different emotions, such as anger and happiness. Then, after meeting that goal, we could move to more nuanced emotions. Trying to distinguish between “good” and “bad” when some of the emotions in each category are visually very similar made things harder on the network. Similarly, having such a large visual range of what belongs in one category is also likely part of the problem.