Noise Recorder using the Pi Zero (`NRP0`)

Ada Pezzuti Dyer

Homeschool in Albuquerque, New Mexico

https://youtube.com/watch?v=basqPd7etqU&feature=share

Introduction

The oily sizzle of an egg in a hot frying pan, the melodic birdsong coming from the tree beside our house, the song I play during piano practice, the obnoxious rev of a pickup truck as it speeds down our street, the high-pitched hum of a plane en route to the airport, and the gasoline-powered roar of our neighbor's weedwhacker and leaf blower.

These are the sounds my family and I hear on a daily basis, and clearly, some are more pleasant than others. Aside from being annoying, those last three noises I mentioned have harsher repercussions than you might imagine.

Let's take, for example, my neighbor's leaf blower. From about 50 feet away, the sound it makes is about 85 decibels. This means that if I'm outside in my yard for two hours while he's blowing leaves, I could be damaging my hearing.[1] What about the traffic noise coming from our street? That's about 80 decibels, and can cause damage in 8 hours.

Hearing loss isn't the only way noise pollution can hurt you, especially if you're my age or younger. Studies show that a child between 8 and 10 years old that has been exposed to 3 times more traffic noise than their peers can show memory development that is 27% slower and attention ability development that is 8% slower[2]. Stress levels can also increase in both children and teens.[3]

Even something as extreme as hearing loss is not uncommon. The CDC reported that in the United States, it is twice as common as diabetes or cancer[4], and that 14.9% of children are subject to hearing loss[5].

Although an individual may not have the power to improve conditions for everyone, with the help of my device, they can be empowered to make a lasting impact on their own community. It would allow a concerned citizen to collect data and prove to authorities that there is a problem to solve, and even specify how to solve it. That's why my chosen theme is creating a safer world for people who live in cities.

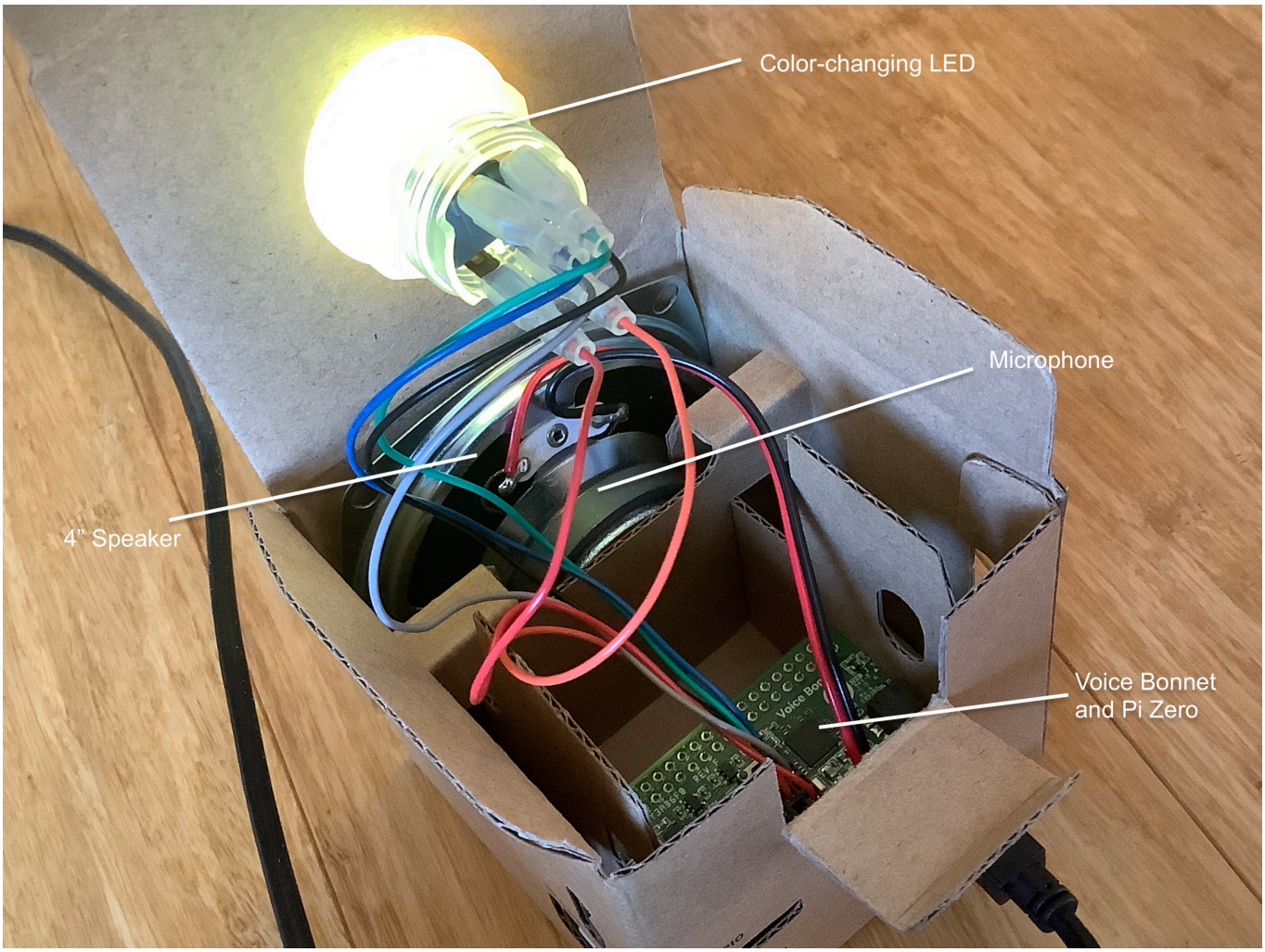
My current prototype is called the Noise Recorder using the Pi Zero, or the NRP0. It is also affectionately nicknamed "the noise police". It creates sound clips of noises that are above a threshold, which are then classified using a machine learning model.

This system is still not seamless enough to be used by someone who doesn't have technology-related knowledge, but its purpose is to make sound-related data collection easier and cheaper

for the average person. The device itself costs about $40, about the same price as a voice-activated sound recorder[6], and the software that I created would be free.

Technology Summary

To create the NRP0, I repurposed the AIY Voice Box, a do-it-yourself virtual assistant that uses AI created by Google. This device is mainly comprised of a Pi Zero, an AIY Voice Bonnet, a speaker, a 4" microphone, and a color-changing button.



A labeled picture of the NRP0, with the cardboard casing open to show its electronic components

When the NRP0 records sounds, the button turns yellow, and the device checks the noise level - measured in RMS - every 0.3 seconds. RMS stands for Root Mean Square, and it is a unit that describes a sound's average power over a length of time[7]. If the RMS exceeds 0.5, then the device makes a 5-second recording that hopefully captures the source of the noise.

After the desired amount of recordings are made, they are classified into categories (such as "Car", "Birds", and "Plane", among others) by a supervised machine learning model, which I

had previously trained and will be described in more detail later.

I. Setting up the Voice Box

The first thing I had to do was a bit of assembly. Following the instructions on the AIY Voice Kit website[8], I constructed the Voice Box.

Then, to be able to add programs to it, I made a `wpa_supplicant` file with my local network's name and credentials, uploaded it onto the Voice Box's chip, and plugged it in. This step allowed me to access all the code on my Voice Box through a local website it was hosting, named `http://orcspi-voice.local` with a port number of `5555`.

Now that I was able to code it, I was ready to turn my Voice Box into the NRP0.

II. Recording sound clips

When the NRP0 is turned on and the following Python program is run, it starts measuring the sound level of its environment. The variables `recdur` (the duration of the recording, in seconds) and `threshold` (the RMS threshold for recording sounds) are set. An "audio stream" is started - this is a period of time in which the NRP0 listens to its environment - and the button turns yellow to indicate this.

```
recdur = 5
threshold = 0.5

stream.start_stream()

buttonLight.update(buttonLight.rgb_on((255, 100, 0)))
```

When a loud noise is detected, the button is turned red. The audio stream, which was used to determine the sound level, is closed. An angry print statement is put in the console (for debugging purposes). A timestamped ".wav" file with a duration of `recdur` is recorded and saved.

```
                if (rms > threshold):
            # red light
            buttonLight.update(buttonLight.rgb_on((255, 0, 0)))

            # stop stream
            stream.stop_stream()
            stream.close()
```

```
            # angrily record loud people
            print("You're being too loud - I will record you!")

            date = time.strftime("%Y%m%d-%H%M%S")
            fn = "loud-" + date + ".wav"
            record_file(AudioFormat.CD, filename=fn, filetype='wav',
                wait=lambda: time.sleep(recdur))
```

After it's finished, the button turns yellow again and a new audio stream is opened.

```
                        buttonLight.update(buttonLight.rgb_on((255, 100,
0)))

            # start stream
            stream = p.open(format=p.get_format_from_width(WIDTH),
                input_device_index=DEVICE,
                channels=1,
                rate=RATE,
                input=True,
                output=False,
                stream_callback=callback)
```

This cycle is repeated every 0.3 seconds.

```
                # refresh every 0.3 seconds
        time.sleep(0.3)
```

The main purpose of turning the button red when it is recording was to make the visual identification of sound clips using the `NRP0` easy. When the button turned red, I could write down what I thought the source of the noise was and the time, so I could then use it to train the machine learning model.

*See the full code in the Appendix.

III. Identify the sounds Pt. 1

The next part of my project was to identify the sounds my device recorded. To do this, I made a machine learning model using CreateML using the labeled sound clips I collected to train it.

To create the training data, I recorded around 36 minutes of MP3 files during the morning, afternoon, and night in my backyard. While I was doing this, I wrote down the sources of all the

notable sounds I heard and the time that they occurred. This process probably took a total time of about 17 hours, along with waiting for sounds to occur so I could record them.

Then, I used a Python program I wrote to slice the audio recordings into 4-second chunks, and identified each chunk using what I wrote down while I was recording. Another reason that I wrote this code is so that if someone wanted to record sounds and classify them but didn't have an NRP0, they could still use my machine learning model.

I begin by importing the files I need from `pydub`. This Python library is really useful for manipulating sound files.

```
from pydub import AudioSegment
from pydub.utils import make_chunks
```

Then, I create a function called `sliceWav` which takes in the parameters `segLength` and `fileName`. The first thing this function does is retrieve an audio file with type `wav` and the name that we pass in and assign it to the variable `myaudio`.

```
myaudio = AudioSegment.from_file(fileName + ".wav", "wav")
```

After that, I convert the chunk length that we pass in to milliseconds.

```
chunk_length_ms = segLength * 1000
```

I then make an array called `chunks` (using the infinitely useful function from the `pydub` library) that is composed of the `wav` file that we retreived sliced into chunks of the desired length.

```
chunks = make_chunks(myaudio, chunk_length_ms)
```

Finally, I loop through the array `chunks` setting the variable `chunk_name` to include the original file name along with the time segment it includes, and describe the folder I want to put it in...

```
chunk_name = "slicedAudio/" + fileName + "_{0}-{1}.wav".format(i *
segLength, (i + 1) * segLength)
```

... print out the file name for clarity...

```
print("exporting", chunk_name)
```

... and save the file on my computer.

```
chunk.export(chunk_name, format="wav")
```

Now, I ask the user for the name of the file to be split, and run the function explained above on it.

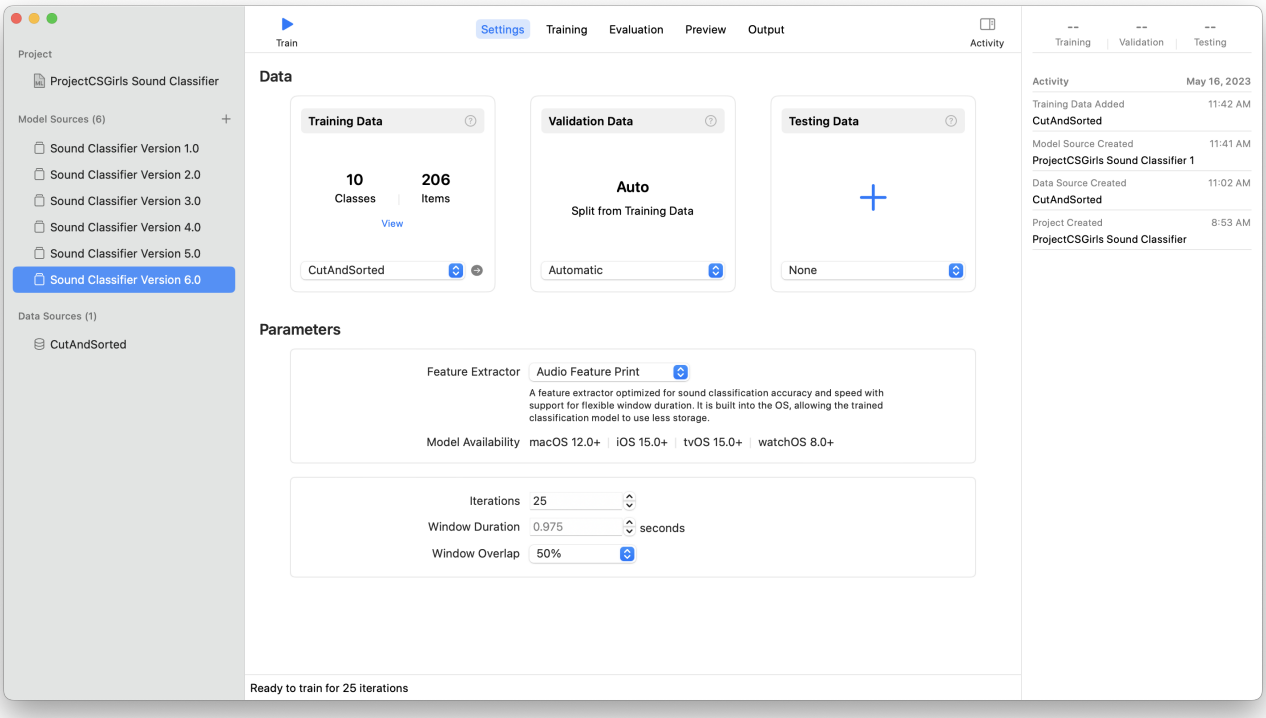
```
input = input("File name:       ")
sliceWav(4, input)
```

*See the full code in the Appendix.

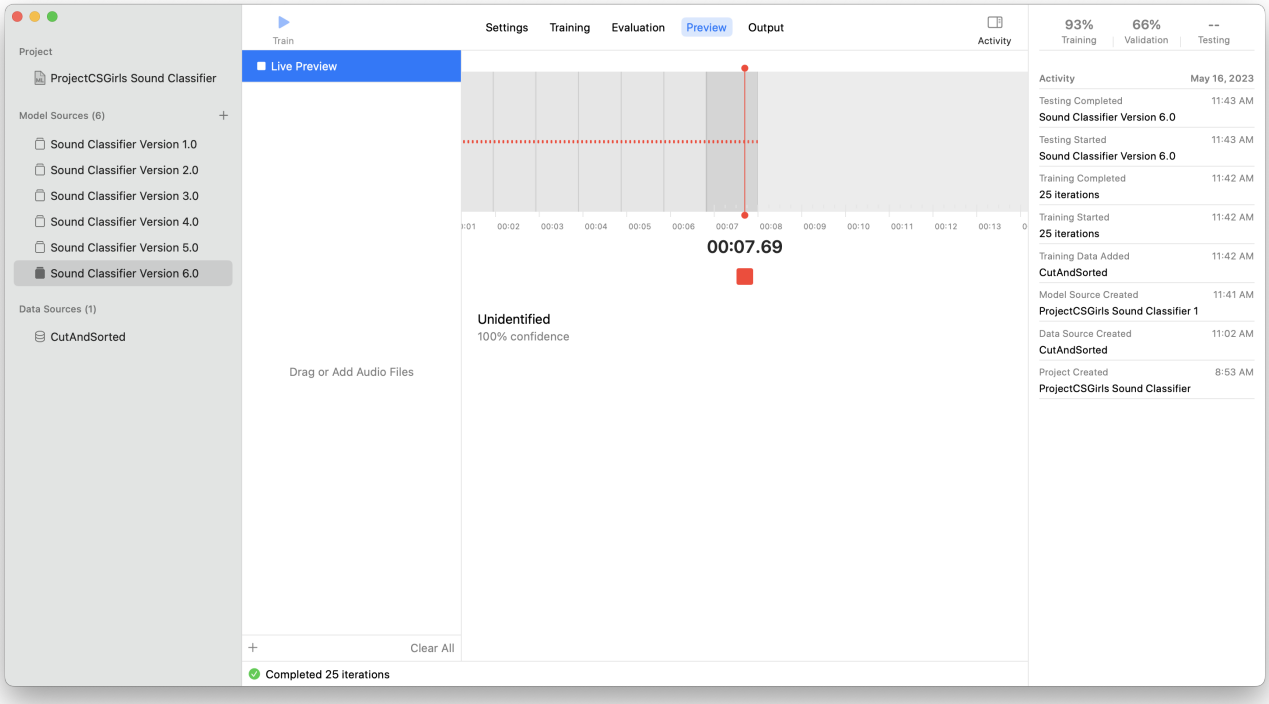
IV. Identify the sounds Pt. 2

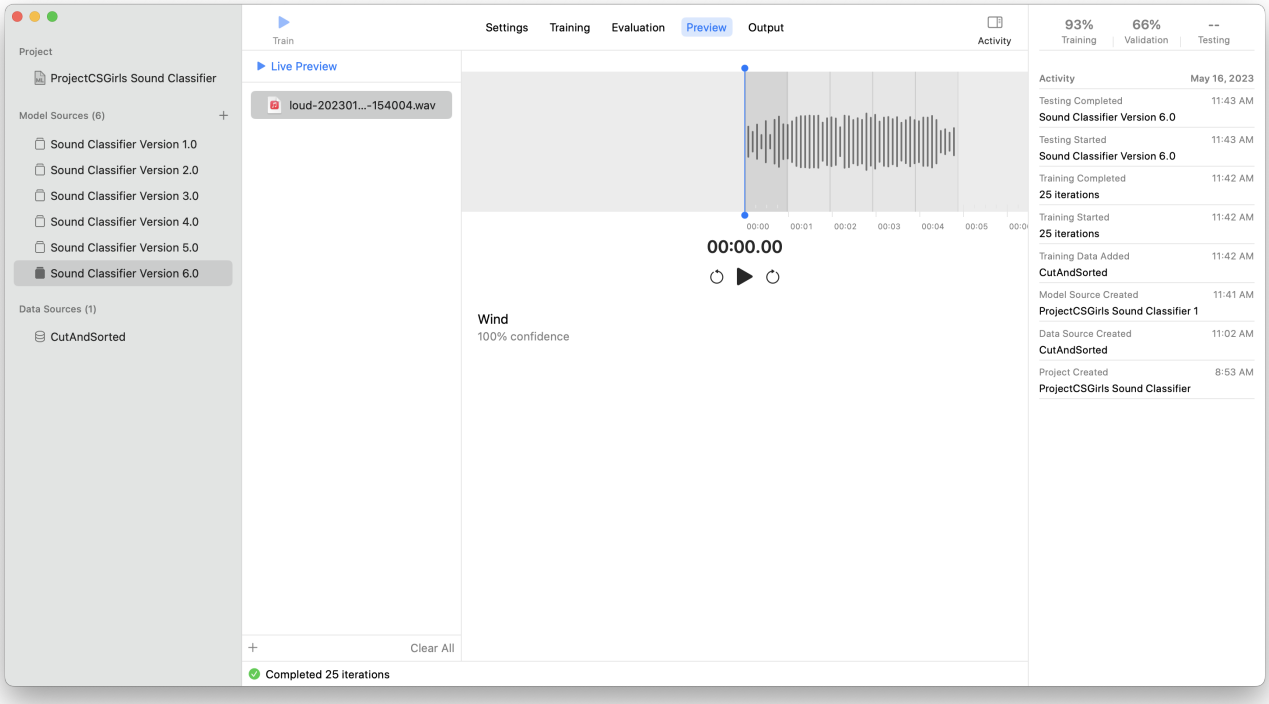
Now that I had training data, I made a machine learning model powered by supervised learning using the Mac application CreateML[9]. I trained it using the labeled sound clips I collected.

First, I create a new CreateML file with a model type of "Sound Classification" and import my training data. The data is in the format of 206 audio clips sorted into 10 folders.

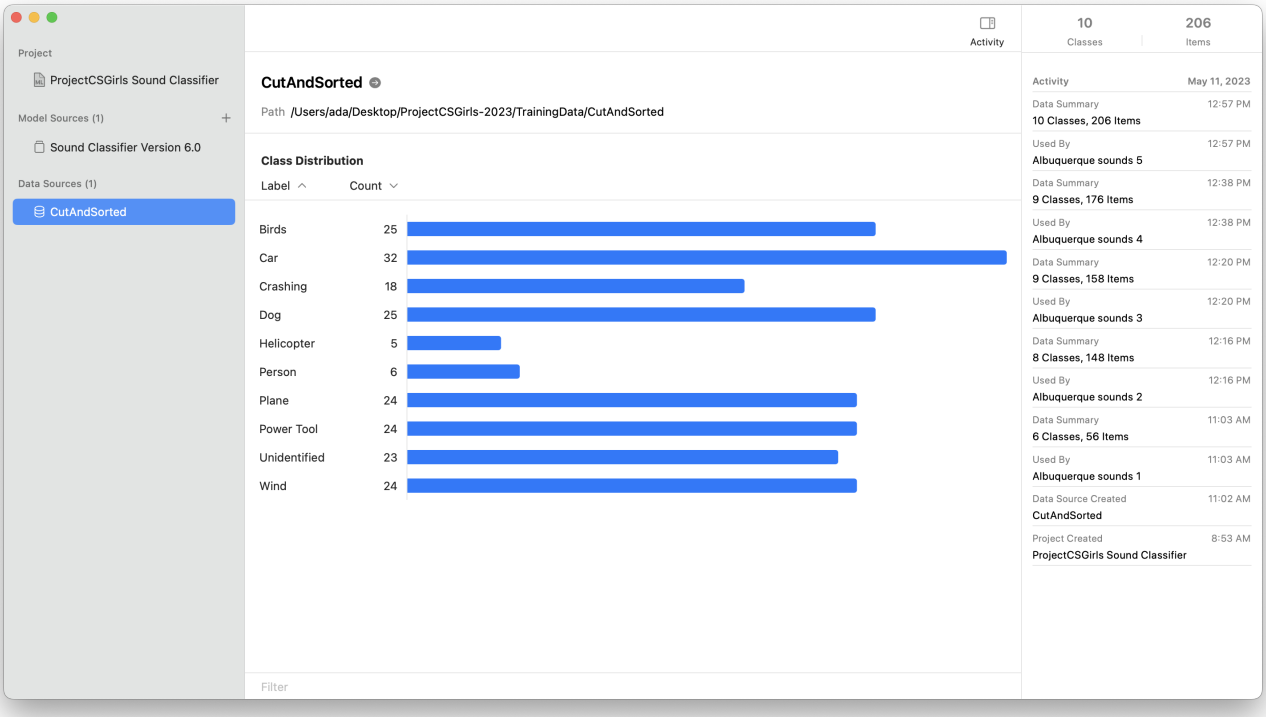


Once the model is trained on this data, you can test it out by starting a live sound stream or uploading a file for it to classify.





You can also get information about the data by finding it in the navigation bar.



This model is available for download on Github at https://github.com/apzzd/ProjectCSGirls-2023/blob/main/Sound%20Classifier%20Version%206.0.mlmodel.

V. Putting it all together

Now with my NRP0 and MLModel finished, I had a way to collect and analyze data.

Sadly, towards the end of my project, the NRP0 suddenly stopped connecting to Wifi, and so I was unable to access files on it. Because of this, I was unable to collect this data with the device, and had to record it using my laptop.

Results

Improvement

The NRP0 is a very problematic device. It caused me an abundance of issues while I was developing it, and a few weeks before my project was completed it stopped working entirely. Further work towards a deployable device is needed.

My machine learning model isn't perfect either. Due to its limited set of training data, it yielded an accuracy of only **89%**.

	Actual Birds	Actual Car	Actual Crashing	Actual Dog	Actual Helicopter	Actual Person	Actual Plane	Actual Power Tool	Actual Unidentified	Actual Wind
Predicted Birds	9	0	0	0	0	0	0	0	1	0
Predicted Car	1	9	0	0	0	0	0	0	0	0
Predicted Crashing	0	1	10	0	0	0	0	0	0	0
Predicted Dog	0	0	0	10	0	0	0	0	0	0
Predicted Helicopter	0	0	0	0	5	0	0	0	0	0
Predicted Person	0	0	0	0	0	6	0	0	0	0
Predicted Plane	0	0	0	0	0	0	10	0	0	0
Predicted Power Tool	0	0	0	0	0	0	0	10	0	0
Predicted Unidentified	0	0	0	0	0	0	0	0	9	0
Predicted Wind	0	0	0	0	0	0	0	0	0	10

The confusion matrix above (created using the Mac application Numbers[10]) shows me that to improve this model's accuracy, I should include relatively more "Bird", "Car", and "Unidentified" clips because it is weak in those areas.

Next Steps

For this project, I made a bulky and barely working prototype, using code with a runtime that could probably be way shorter. What would this look like if produced on a commercial level?

To start, the device would definitely not be as prone to technical issues, and its structure would be made more compact. Like Google's AIY Voice Kit, it would cost around $40. As for the software, I believe it would be very helpful to connect all the different platforms my device and model work on by way of an app to make it more user-friendly. The app would pair with an NRP0 through bluetooth, and use the machine learning model to give you a "daily report" of the loud noises that were recorded. Of course, the app would be free.

Appendix

End Notes

[1] https://www.cdc.gov/nceh/hearing_loss/what_noises_cause_hearing_loss.html

[2] https://www.theguardian.com/environment/2022/jun/02/traffic-noise-slows-childrens-memory-development-study-finds

[3] https://pubmed.ncbi.nlm.nih.gov/29313308/

[4] https://www.cdc.gov/nceh/hearing_loss/public_health_scientific_info.html

[5] https://www.cdc.gov/ncbddd/hearingloss/data.html

[6] https://www.amazon.com/Digital-Voice-Activated-Recorder-Lectures/dp/B07KBWN8L1/ref=asc_df_B07KBWN8L1?tag=bngsmtphsnus-20&linkCode=df0&hvadid=80264466333888&hvnetw=s&hvqmt=e&hvbmt=be&hvdev=c&hvlocint=&hvlocphy=&hvtargid=pla-458386399300239O&psc=1

[7] https://en.wikipedia.org/wiki/Audio_power#Continuous_power_and_%22RMS_power%22

[8] https://aiyprojects.withgoogle.com/voice/

[9] https://developer.apple.com/machine-learning/create-ml/

[10] https://www.apple.com/numbers/

Bibliography

RMS Noise

https://en.wikipedia.org/wiki/Audio_power#Continuous_power_and_%22RMS_power%22
https://resources.pcb.cadence.com/blog/2019-what-is-rms-noise-and-how-does-it-compare-to-the-standard-deviation
https://www.onesdr.com/dbv-to-vrms-calculator/

$V_{RMS} = 10^{dBV / 20}$

Conversion between RMS and decibels

Noise Pollution Standards

https://wsdot.wa.gov/sites/default/files/2021-10/Env-FW-BA_ManualCH07.pdf
https://nonoise.org/library/niosh/criteria.htm
https://education.nationalgeographic.org/resource/noise-pollution/
http://dangerousdecibels.org

Effects of Noise Pollution

https://www.cdc.gov/nceh/hearing_loss/what_noises_cause_hearing_loss.html
https://www.cdc.gov/ncbddd/hearingloss/data.html
https://www.cdc.gov/nceh/hearing_loss/teens/index.html
https://www.theguardian.com/environment/2022/jun/02/traffic-noise-slows-childrens-memory-development-study-finds
https://pubmed.ncbi.nlm.nih.gov/29313308/
https://www.cdc.gov/nceh/hearing_loss/public_health_scientific_info.html

Coding References

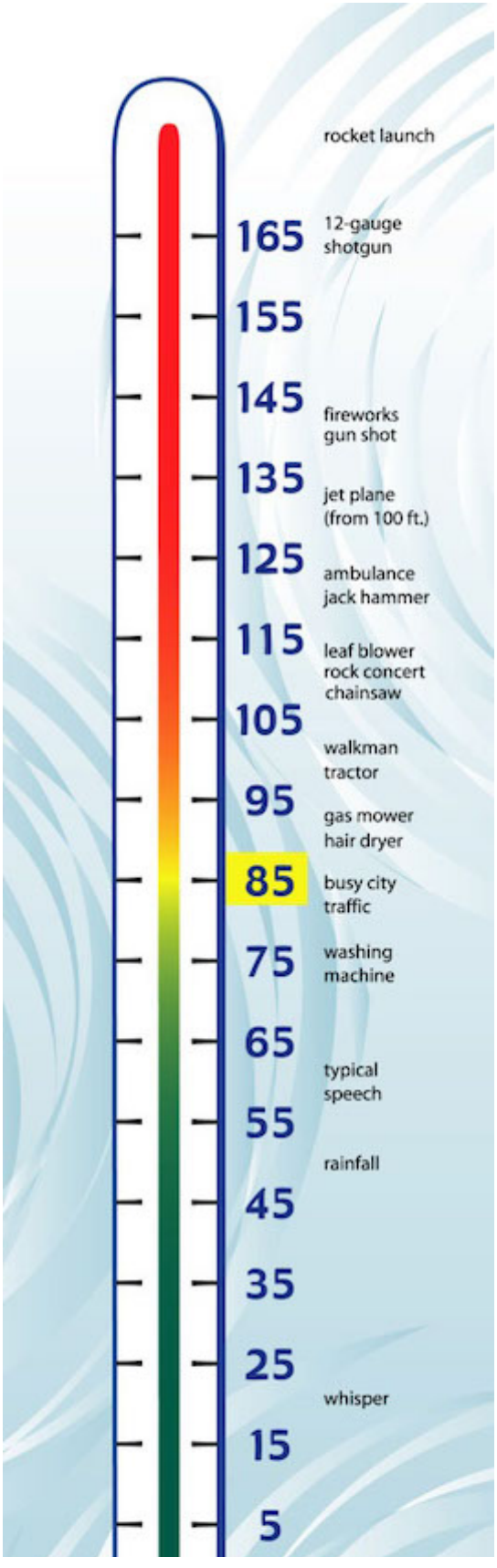
https://stackoverflow.com/questions/36799902/how-to-splice-an-audio-file-wav-format-into-1-sec-splices-in-python
https://machinelearningmastery.com/confusion-matrix-machine-learning/

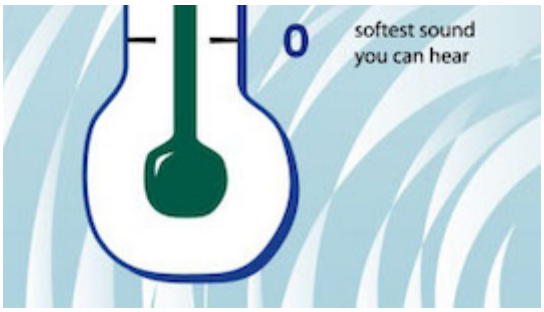
Special Thanks

A big thanks to my neighbor Joseph for the interview :)

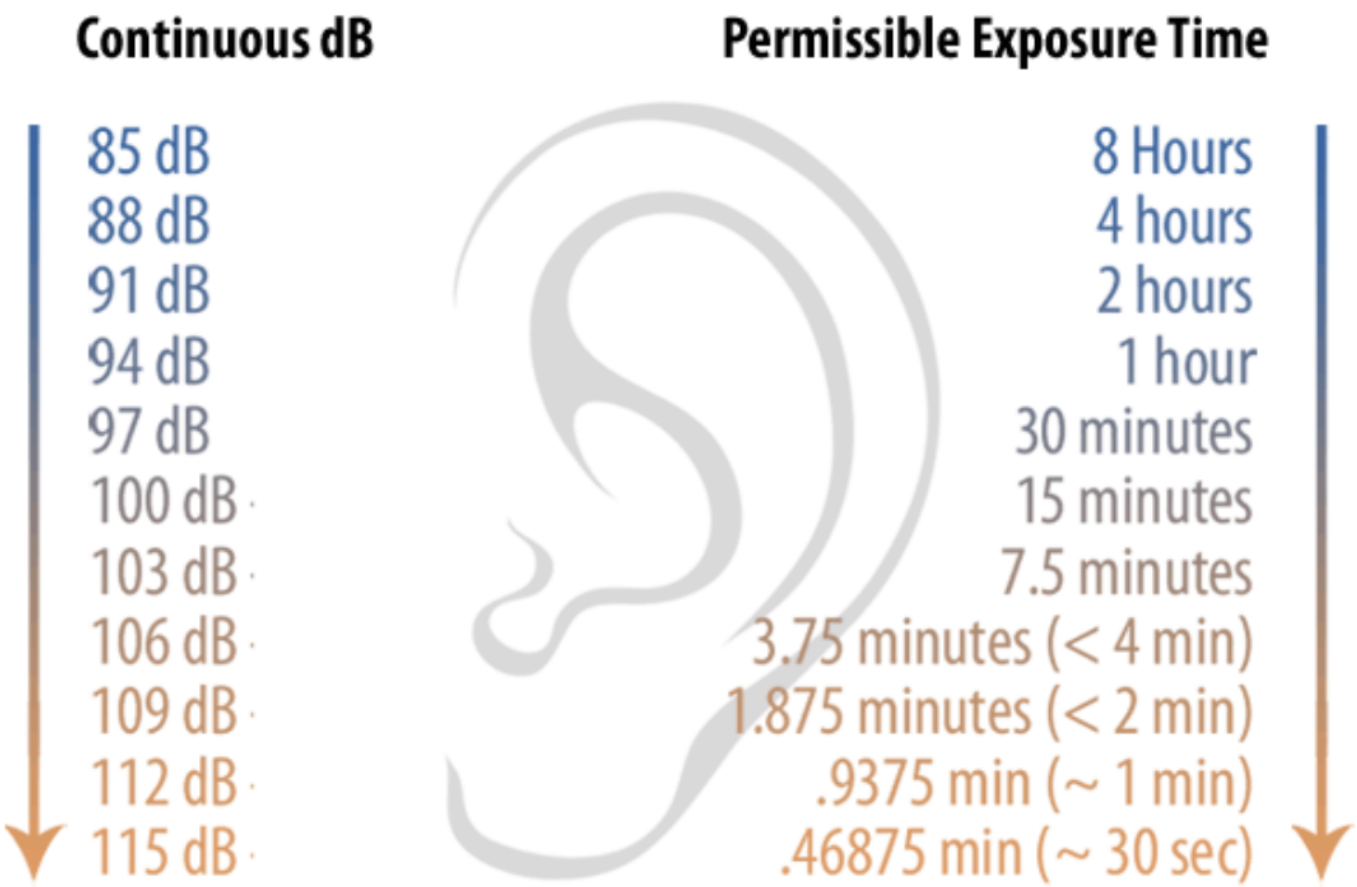
Also to my Mom for pushing me to work on this project regularly, even when I didn't want to, and to my brother for proof-reading and quite obnoxiously pointing out that my video did *not* need that much postproduction.

Charts





A decibel "thermometer" to give examples of sounds at certain decibel levels. Sourced from http://dangerousdecibels.org/education/information-center/decibel-exposure-time-guidelines/



A chart showing safe exposure times for decibel values. Sourced from http://dangerousdecibels.org/education/information-center/decibel-exposure-time-guidelines/

Source Code

Sound Splitter

Download on Github at https://github.com/apzzd/ProjectCSGirls-2023/blob/main/soundSplitter/main.py

```
from pydub import AudioSegment
from pydub.utils import make_chunks
```

```
def sliceWav(segLength, fileName):
        myaudio = AudioSegment.from_file(fileName + ".wav", "wav")
        chunk_length_ms = segLength * 1000
        chunks = make_chunks(myaudio, chunk_length_ms)

        for i, chunk in enumerate(chunks):
                chunk_name = "slicedAudio/" + fileName + "_{0}-
{1}.wav".format(i * segLength, (i + 1) * segLength)
                print("exporting", chunk_name)
                chunk.export(chunk_name, format="wav")

for fname in ["Morning1", "Morning2", "Afternoon2", "Night1", "Night2"]:
        sliceWav(4, fname)
```

Sound Recorder

Download on Github at https://github.com/apzzd/ProjectCSGirls-2023/blob/main/NoiseMeterDemo.ipynb

```
import pyaudio
import time
from math import log10 import audioop

from IPython.display import clear_output
from aiy.voice.audio import AudioFormat, play_wav, record_file

import glob
from aiy.pins import BUTTON_GPIO_PIN from gpiozero import Button
from aiy.leds import Leds, Color

buttonLight = Leds()

###

p = pyaudio.PyAudio()

WIDTH = 2
RATE = int(p.get_default_input_device_info()['defaultSampleRate' DEVICE =
p.get_default_input_device_info()['index']
rms = 1
```

```
###

def callback(in_data, frame_count, time_info, status): global rms

rms = audioop.rms(in_data, WIDTH) / 32767 return in_data, pyaudio.paContinue

stream = p.open(format=p.get_format_from_width(WIDTH),
input_device_index=DEVICE,

channels=1,
rate=RATE,
input=True,
output=False, stream_callback=callback)

###

import time recdur = 5

threshold = 0.5

stream.start_stream()

buttonLight.update(buttonLight.rgb_on((255, 100, 0)))

try:
        while stream.is_active():
                # print out the RMS and DB values
                        db = 20 * log10(rms)
                        print(f"RMS: {rms:.4f} DB: {db:.1f}")
clear_output(wait=True)

                        if (rms > threshold):
                                # red light
                                buttonLight.update(buttonLight.rgb_on((255,
0, 0)))

                                # stop stream
                                stream.stop_stream() stream.close()
```

```
                    # angrily record loud people
                                print("You're being too loud 😡 I will
record you!")

                                date = time.strftime("%Y%m%d-%H%M%S")
                                fn = "loud-" + date + ".wav"
record_file(AudioFormat.CD, filename=fn, filetype='wav', wait=lambda:
time.sleep(recdur))
                                buttonLight.update(buttonLight.rgb_on((255,
100, 0))

                                # start stream
                                stream =
p.open(format=p.get_format_from_width(WIDTH), input_device_index=DEVICE,
channels=1, rate=RATE, input=True,
        output=False, stream_callback=callback)

        # refresh every 0.3 seconds
        time.sleep(0.3)

except KeyboardInterrupt:
    print('Done')

stream.stop_stream()
stream.close()
```

Today's Loud Sounds

Note: This code was not featured in the report because it was not essential to the project. It's pretty cool though, so I decided to include it here. It provides a very basic user interface for viewing and playing loud sounds recorded that day. :)

Download on Github at https://github.com/apzzd/ProjectCSGirls-2023/blob/main/NoiseMeterDemo.ipynb

```
today = time.strftime("%Y%m%d")

global printedSnds
printedSnds = False
```

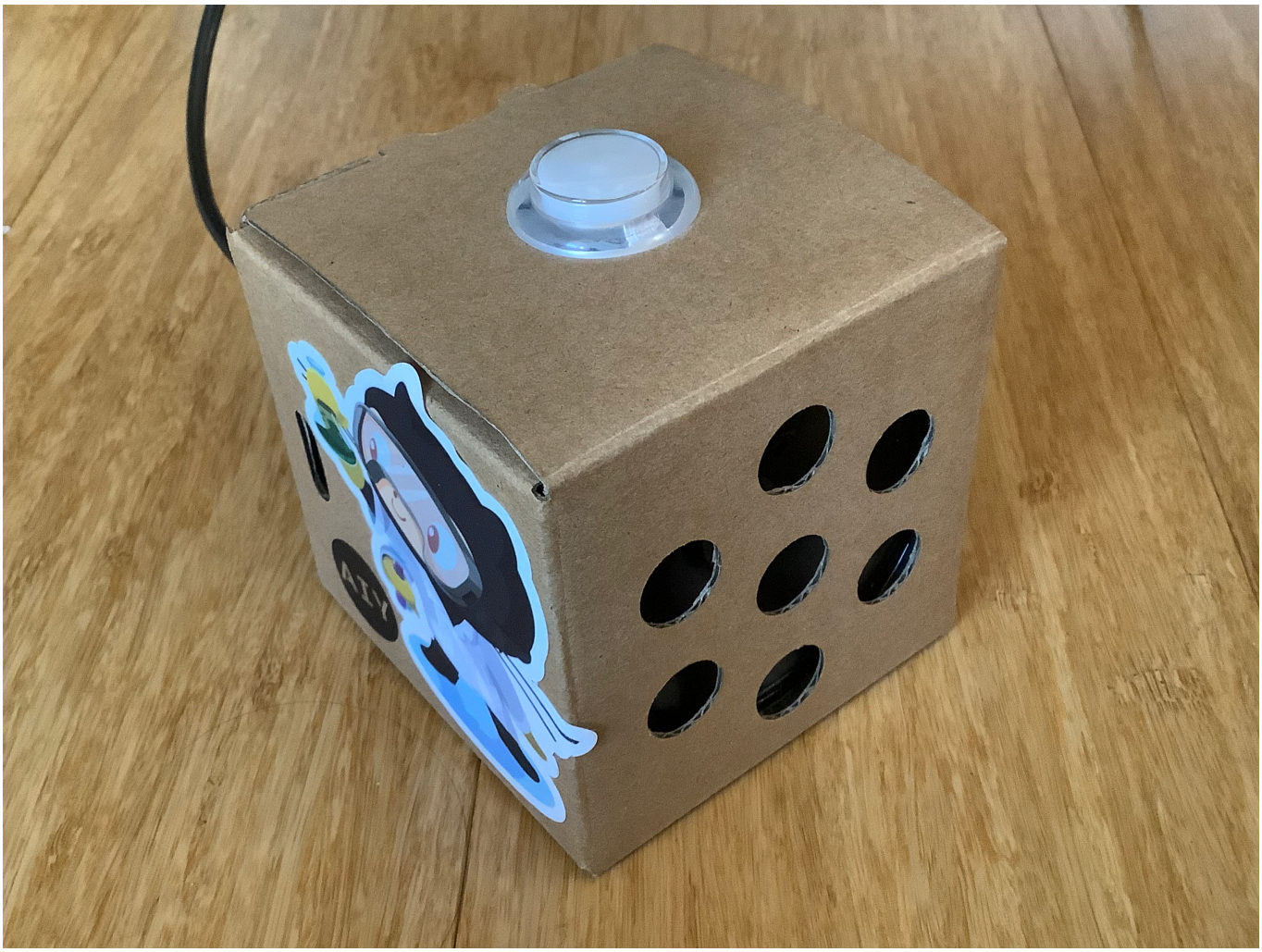
```
def playSounds():
        print("Pressed")
        global printedSnds

        files = glob.glob("loud-" + today + "*.wav")

        for file in files:
                if (not printedSnds):
                        print("Printin' it")
                        print(file)
                else:
                        print("Playin' it")
                        play_wav(file)
                        sleep(2)

        printedSnds = True

with Leds() as buttonLight, Button(BUTTON_GPIO_PIN) as button:
        if (printedSnds):
                buttonLight.update(buttonLight.rgb_on((0, 0, 255)))
        else:
                buttonLight.update(buttonLight.rgb_on((255, 0, 255)))
                while True:
                        button.when_pressed=playSounds
```

Close and Terminate Stream

```
stream.close()
p.terminate()
```

Photos

All photos taken by Ada Dyer



The NRP0, fully enclosed in its cardboard casing and decorated with an Octocat (Github's mascot) sticker