

**Name: Jason Zhou**

**Mentor: Dr. Dongjin Song**

**Status Report #: 16**

**Time Spent on Research This Week: 14**

**Cumulative Time Spent on Research: 121**

**Miles Traveled to/from Mentor This Week: 0**

**Cumulative Miles Traveled to/from Mentor: 0**

=====

**Monday, January 24th, 2022: (0.25 Hrs)**

I had a discussion with my mentor about how I would go about creating a neural network and preprocessing data. Afterward, he introduced me to a graduate student that was working under him through email. This student's name is Binghao Lu, and he is also working with sound anomaly detection.

Afterward, Binghao asked if I would be able to meet later on in the week to discuss my background in machine learning and what I did and did not know. As of right now, my Zoom meeting with Binghao is scheduled to be on Thursday (January 27th) at 2:45 PM.

**Tuesday, January 25th, 2022: (2 Hours)**

I spent this day reviewing the code I had written during December, which involved grabbing audio files from the computer's storage and storing their addresses within a variable.

Although it took a while to review, I then moved on to trying to convert my audio files into mel spectrograms. I determined that because each mel spectrograms are technically 2D arrays filled with data, in order to pass all of them through a machine learning algorithm, I would need to store all of the mel spectrograms into a 3D array (one can imagine this as a stack of papers, where each paper represents a single mel spectrogram).

To achieve this, I practiced manipulating my data and changing the shapes of arrays; however, for some reason, I was unable to stack all of the mel spectrograms together. Every time I tried, the data would mesh together to form one giant, discombobulated mel spectrogram. I will have to look further into why this was not working.

**Wednesday, January 26th, 2022: (3 Hours)**

Thinking back on the previous day, I explored different ways of adding multiple mel spectrograms into a single array. For example, I looked at multiple functions like `numpy.append()`, `numpy.insert()`, `numpy.vstack()`, and `numpy.dstack()`; however, all of these functions still created one giant mel spectrogram that was a mixture of the data, which is not what I wanted.

Looking on the bright side, I was able to gain a lot of practice with manipulating arrays and working with various functions to work with data.

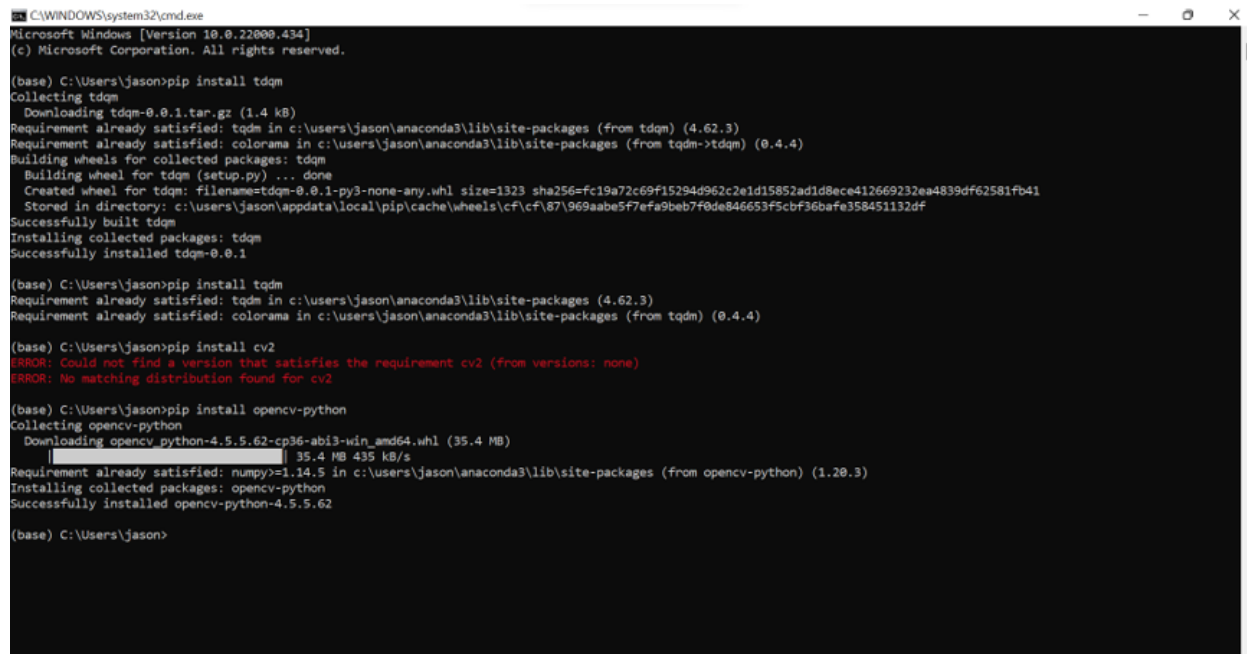
### Thursday, January 27th: (0.25 Hrs)

On this day, I met with Binghao on Zoom. We discussed topics such as coding environments, inputting data, and objected-oriented programming. After confirming that I at least knew the basics, he showed me some examples of neural networks in a coded format and also recommended that I watch a video series made by a YouTuber called sentdex to learn how to code neural networks (this was the same video series he used to learn about neural networks).

### Friday, January 28th: (1.5 Hours)

After my meeting with Binghao, I took a look at the introductory episodes of the YouTube video series he had recommended to me. The video covered the basic chemistry of a neural network and the general steps needed to construct one. Although I already knew everything that was covered in the introductory episodes, it was still served as good review material.

Additionally, I had previously been using TensorFlow (a machine learning library), but I switched over to PyTorch because that is what the video uses. As a result, I ended up needing to download the Pytorch library and some additional libraries from my command prompt.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22000.434]
(c) Microsoft Corporation. All rights reserved.

(base) C:\Users\jason>pip install tqdm
Collecting tqdm
  Downloading tqdm-0.0.1.tar.gz (1.4 kB)
Requirement already satisfied: tqdm in c:\users\jason\anaconda3\lib\site-packages (from tqdm) (4.62.3)
Requirement already satisfied: colorama in c:\users\jason\anaconda3\lib\site-packages (from tqdm->tqdm) (0.4.4)
Building wheels for collected packages: tqdm
  Building wheel for tqdm (setup.py) ... done
  Created wheel for tqdm: filename=tqdm-0.0.1-py3-none-any.whl size=1323 sha256=fc19a72c69f15294d962c2e1d15852ad1d8ece412669232ea4839df62581fb41
  Stored in directory: c:\users\jason\appdata\local\pip\cache\wheels\cf\87\969aabe5f7efa9beb7f0de846653f5cbf36baf358451132df
Successfully built tqdm
Installing collected packages: tqdm
Successfully installed tqdm-0.0.1

(base) C:\Users\jason>pip install tqdm
Requirement already satisfied: tqdm in c:\users\jason\anaconda3\lib\site-packages (4.62.3)
Requirement already satisfied: colorama in c:\users\jason\anaconda3\lib\site-packages (from tqdm) (0.4.4)

(base) C:\Users\jason>pip install cv2
ERROR: Could not find a version that satisfies the requirement cv2 (from versions: none)
ERROR: No matching distribution found for cv2

(base) C:\Users\jason>pip install opencv-python
Collecting opencv-python
  Downloading opencv-python-4.5.5.62-cp36-abi3-win_amd64.whl (35.4 MB)
    | 35.4 MB 435 kB/s
Requirement already satisfied: numpy>=1.14.5 in c:\users\jason\anaconda3\lib\site-packages (from opencv-python) (1.20.3)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.5.5.62

(base) C:\Users\jason>
```

(This is an image of me downloading libraries from my command prompt. Specifically, I am downloading the tqdm and opencv libraries.)

### Saturday, January 29th, 2022: (2 Hours)

Continuing with the video series, I began getting into the more in-depth parts of a neural network and actually coding the process of obtaining preprocessing the data. For practice, I

used a very common, publicly available dataset called MNIST that is made up of images of numbers.



(An example of what an image in the MNIST dataset would look like. Obviously, a human would know that this is a four, but the goal is to have the computer recognize this as a four)

I have actually analyzed this same dataset before with TensorFlow; however, that was a very long time ago, and I thought it would be useful to review this and see how it is done with PyTorch.

Overall, I have noticed that style of coding that is used for PyTorch is similar to how I would normally code, so it feels more natural to me than TensorFlow did.

### **Saturday, January 29th, 2022: (3 Hours)**

I quickly finished up the MNIST dataset and moved onto the next episodes of the video series. The series now focuses on analyzing a more complex dataset, which is a collection of images that are distributed between cats and dogs.

For the first part of the tutorial video, I focused on inputting the images into a variable and then stacking them all into an array. The video did a good job of showing this process, so I was able to see what was going on. Additionally, this was something I had been struggling to do earlier in the week, so it was helpful to see what I had been doing wrong (I was using a numpy array instead of a Python list). Next, along with the video instructor, I went through steps to balance the data<sup>1</sup>.

---

<sup>1</sup>In the context of the problem I was practicing, balancing the data means ensuring that there are an equal number of dog and cat pictures that the neural network can look through. This helps to make sure that the neural network is not over exposed to one class of data (For example, the goal is to prevent it from seeing more dogs than cats and vice versa).

**Sunday, January 30th, 2022: (2 Hours)**

Beyond preprocessing the data, I worked to construct the simple neural network by creating a class, defining the layers through variables, and then passing the data through it.

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 32, 5)
        self.conv2 = nn.Conv2d(32, 64, 5)
        self.conv3 = nn.Conv2d(64, 128, 5)

        x=torch.randn(50,50).view(-1,1,50,50)
        self._to_linear = None
        self.convs(x)

        self.fc1 = nn.Linear(self._to_linear, 512)
        self.fc2 = nn.Linear(512,2)

    def convs(self, x):
        x = F.max_pool2d(F.relu(self.conv1(x)), (2,2))
        x = F.max_pool2d(F.relu(self.conv2(x)), (2,2))
        x = F.max_pool2d(F.relu(self.conv3(x)), (2,2))

        if self._to_linear is None:
            self._to_linear = x[0].shape[0] * x[0].shape[1] * x[0].shape[2]
        return x

    def forward(self,x):
        x = self.convs(x)
        x=x.view(-1, self._to_linear)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
```

(A picture of the code used to create a simple class for the neural network used to differentiate between dogs and cats)

Overall, so far, the video series has been very informative, and I believe the instructor does a very good job of breaking everything down.

After going through these steps, in a separate file, I simply did some practice to make sure that I remembered what I needed to do in order to create a neural network.

While doing this, there were some aspects of the process that confused me, so I intend to ask my mentor about it when I get the chance during our next weekly Zoom meeting.

## References

sentdex. (2019, September 23). *Introduction - deep learning and neural networks with Python and Pytorch p.1* [Video]. YouTube.

<https://www.youtube.com/watch?v=BzcBsTou0C0&list=PLQVvvaa0QuDdeMyHEYc0gxFpYwHY2Qfdh>

sentdex. (2019, September 24). *Data - deep learning and neural networks with Python and Pytorch p.2* [Video]. YouTube.

<https://www.youtube.com/watch?v=i2yPxY2rOzs&list=PLQVvvaa0QuDdeMyHEYc0gx FpYwHY2Qfdh&index=2>

sentdex. (2019, September 25). *Building our neural network - deep learning and neural networks with Python and Pytorch p.3* [Video]. YouTube.

<https://www.youtube.com/watch?v=ixathu7U-LQ&list=PLQVvvaa0QuDdeMyHEYc0gx FpYwHY2Qfdh&index=3>

sentdex. (2019, September 27). *Training Model - Deep Learning and Neural Networks with Python and Pytorch p.4* [Video]. YouTube.

[https://www.youtube.com/watch?v=9j-\\_dOze4IM&list=PLQVvvaa0QuDdeMyHEYc0gx FpYwHY2Qfdh&index=4](https://www.youtube.com/watch?v=9j-_dOze4IM&list=PLQVvvaa0QuDdeMyHEYc0gx FpYwHY2Qfdh&index=4)

sentdex. (2019, October 2). *Convnet intro - deep learning and neural networks with Python and Pytorch p.5* [Video]. YouTube.

<https://www.youtube.com/watch?v=9aYuQmMJvjA&list=PLQVvvaa0QuDdeMyHEYc0gx FpYwHY2Qfdh&index=5>

sentdex. (2019, October 7). *Training convnet - deep learning and neural networks with Python and Pytorch p.6* [Video]. YouTube.

<https://www.youtube.com/watch?v=1gQR24B3ISE&list=PLQVvvaa0QuDdeMyHEYc0gx FpYwHY2Qfdh&index=6>