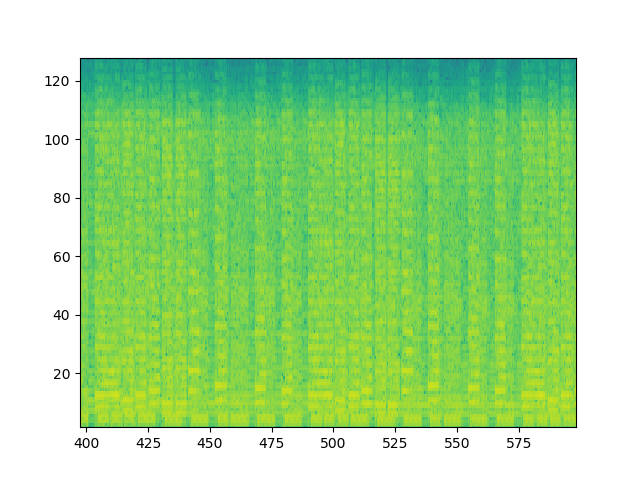
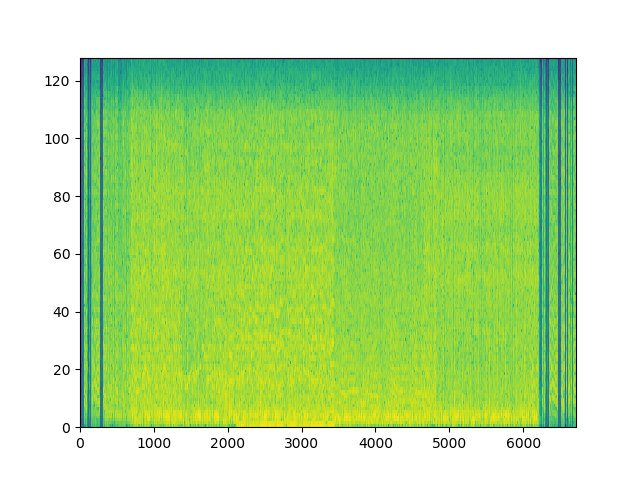
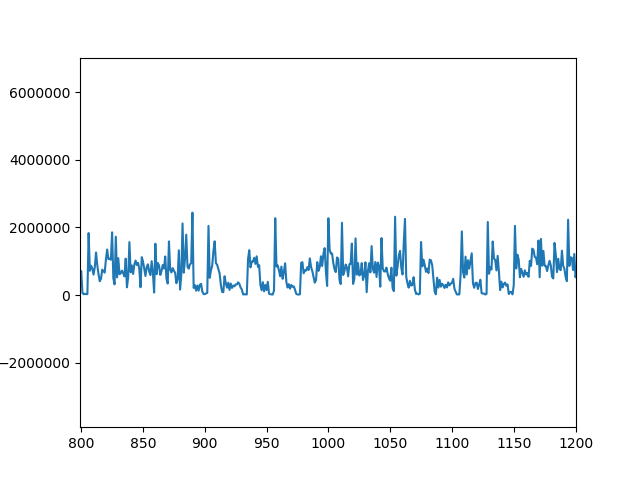
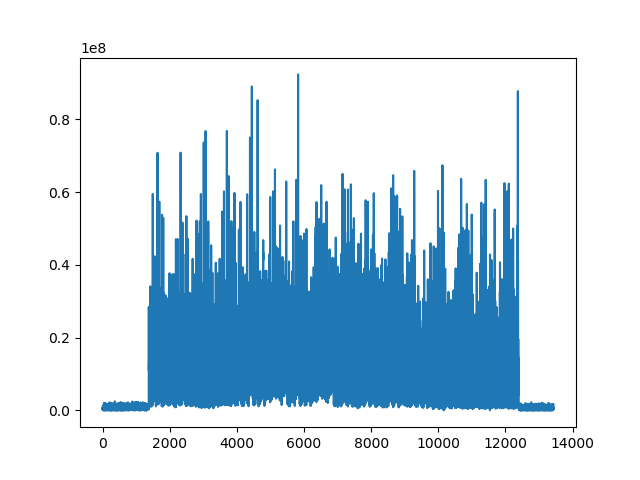
The skill of recognizing the tempo of a song is one that almost every human shares. Not only that, but, as we saw in this class, many animals, such as cockatoos, monkeys and sea lions also have the ability. It is uncertain where the dividing line is, where a larger brain size means an animal probably can recognize tempo, and a smaller brain means it probably can’t, or whether there is some specialized neural structure that deals with tempo (a question on the midterm makes me think that that structure might be the one in charge of vocal learning.) However, it is certain that recognizing tempo is something that is achievable with an intelligence that is significantly lower than human. To better understand how the recognition of tempo works, I set out to create a neural network that could classify the tempo of a song.

The bare minimum of what I set out to do was to input a song, and have the neural net output a single number for the tempo. My additional goals were to detect the actual beats within a song, and be able to give a map of tempos for a song with a variable tempo. To do this, I was primarily trying to replicate Liam Schoneveld’s work in his two blog posts. A link to them is in the works cited.

The basic model for my neural network is that of supervised learning, a method of machine learning for when the program is trying to predict a known answer, so it can get immediate feedback. This is contrasted with unsupervised learning, which is generally about finding patterns in the data. I don’t know which method humans and other animals use; it could be that we can learn how to match tempo just by listening to a bunch of songs, but it’s also possible that we need to see someone else to demonstrate how to find the beat before we can find it ourselves. In other words: did Snowball the cockatoo need to be taught to recognize tempo, or did it learn on its own? All methods for detecting tempo, that I am aware of, use supervised learning, so what version humans use could greatly affect whether the neural net is doing something analogous to a human brain, like the way low level neurons in the occipital lobe to analogous computations to the low level neurons in an image classifier.

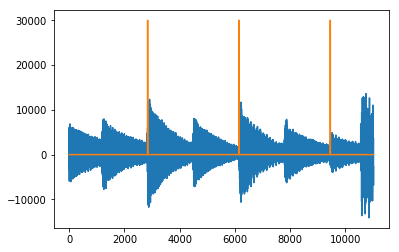
The songs I used for the input data were from the Undertale soundtrack, mostly because it was the most complete soundtrack I had readily available, and I thought if all the songs had the same composer, it might be easier for the neural network to interpret. I started out by making a spectrogram of the song. When looking at the spectrogram, I noticed a regular pattern that seemed to correlate with the tempo. Trying to capture this, I added up all the values at each time step, then measured the change between each time step, in the hopes that my crude derivative would have regular spikes. No patterns could be discerned from the resulting graph (beyond the human brain’s tendency to find patterns where there are none, that is.), however. I never actually measured the length of the seeming pattern to see if it matched the bpm, as Liam’s second blog post dealt with the raw audio, and that seemed easier to start with.





*Figure 1. The top row are the spectrograms, the bottom is the resulting ‘derivative.’ The left column is a full song, and the right is a zoomed in portion to better see what’s happening. These turned out to be completely useless.*

To get the training data, I first edited each song in audacity to start exactly on the first beat. I then took a random 4-second snippet from a random song, and did this for some number of times (usually 1000 for the training data, and 100 for the validation data. I tried increasing it to 10,000 and 1,000 respectively, but that didn’t seem to increase the accuracy). At 11025 hz, this resulted in an array of 44100 numbers, which was then converted to a 441x100 matrix. To get the tempo of each song, I found a reddit comment that listed the tempos for most songs in the Undertale soundtrack, then verified that they were correct by listening to the song and tapping along to the beat. I used [www.all8.com](http://www.all8.com) for this purpose. Next, to get the expected output, a second signal is generated. This signal has 0 for every value except for a 1-valued spike n times per minute, where n is the bpm. Each song should start exactly on a beat so that this will line up properly. 2 out of 7 songs didn’t line up regardless, so they were discarded.



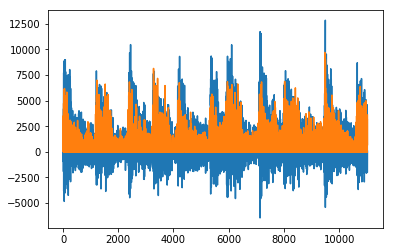
*Figure 2. A 4-second snippet of a song, and the expected output. (The output has been multiplied by 30,000 to make it more visible.)*

The neural network was made using Keras, a python API that works at a very high level. The neural network used was a fairly simple model with only two main layers. The first is a convolutional 1D layer, which I don’t understand very well, and the second is a max-pooling layer, which takes the maximum of every 4 values as a way of trimming down the output.

Generally, supervised neural networks have 3 sets. A training set, where every time it makes a mistake, it backpropagates the correction through the network, a validation set, used to judge the accuracy during training, and a testing set, for giving the final accuracy. In my case, there was no testing set, as the output was a predicted signal, not a tempo.

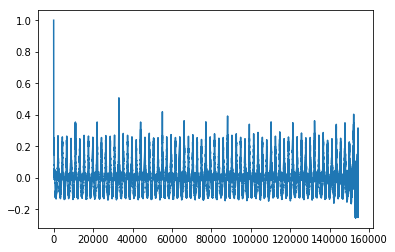
To train the network, training data and validation data are generated, then the model is updated through the fit() function. Only one ‘epoch’ is used, because more epochs are simulated by calling the fit() function again. In the next cycle, new training and validation sets are generated. Once the best accuracy of the validation set hasn’t improved in 5 cycles (usually it was five, I also tried other numbers, and having it do a set number of cycles. It didn’t improve much after the validation error dropped below 2000, however, and a 5-cycle check was usually enough for that), the training stops. This usually takes about 30 cycles.

Now the network has been trained on 4-second snippets, each song is broken up into 4-second pieces, the trained network then makes a prediction for each of them, and the predictions are pieced together to make a prediction for the whole song.



*Figure 3. First four seconds of a song (blue), and the predicted output (orange). It’s doing… something.*

Next is, I suspect, the part where it all went wrong. In order to convert the predicted output into a tempo, Liam used something called autocorrelation, which is a way of finding repeating patterns in a signal. I don’t know how to implement and autocorrelation function, and it seemed fairly complicated, so I used a method from user unutbu from stackexchange.com.

**

*Figure 4. Autocorrelation of an entire song.*

This is the part where things stopped going as planned, though it is still possible the error happened in an earlier stage. The distance between the peaks in the autocorrelation graphs is usually 15 seconds, sometimes 7.5 or 30 seconds, but always 15 times some power of 2. This holds even for tempos like 115, for which 15 doesn’t evenly divide into. Because the songs are broken up into 4-second snippets, and one beat every 4 seconds is 15 beats per minute, it seems like the length of the snippet is more important than the contents of the song.

There are a number of things that could have gone wrong. It’s possible that by only using 5 songs in the training/validation data, the network had insufficient information to work with. However, I would have thought that not enough data would make it predict the tempo too well; it would be overfitting the data so it wouldn’t work on other songs.

It’s possible that this entire model only works for electronic music, which is what Laim used, though it really seemed more general than that to me. It’s possible that the neural net’s output was correct, and I messed up the autocorrelation, though the output does not look that different from the original song, and if that was right, you would think that doing autocorrelation on the original song would be all you need. It’s also possible that even the autocorrelation is correct, and I’m just not correctly interpreting it.

If I had to bet, however, I would guess that the problem lies in the 1D convolutional layer. I changed some numbers compared to Liam’s version in order to make the shapes line up properly to get it to compile. I think it’s fairly likely that my model is doing something entirely different from Liam’s because I didn’t compensate for the change in some other way.

If you would like to run my nonfunctional algorithm, place some songs in a folder called “Songs,” and have their file names start with the tempo. (Technically, you should also edit the songs so that they start exactly on the first beat, but the program doesn’t work anyways, so I wouldn’t recommend it.) Then, run Tempo Neural Net 3.py in the same location as the “Songs” folder (not inside it). You may have to download keras, and some other python modules.

As for things I could have done differently, I should have done more work earlier. By the time I realized I had no idea what the autocorrelation function was supposed to do, it was too late to talk with someone in signal processing to figure it out, and it was outside the expertise of my AI professor. Another thing I should have done was try to run Liam’s code earlier. He used some python modules that I don’t have, and couldn’t quickly figure out how to get, so I ended up never even knowing if it could even work in principle for the songs I picked.

My main goal for this project was to pave the way for my drum doodle classifier. While interesting, beat detection and tempo recognition are things people have done before, and I was just attempting to copy them (though, I’m not sure if anyone has accomplished tempo recognition to the extent that even sea lions do it, but that would be ground breaking research, and not something I have much of a chance of impacting.) My drum doodle project, in contrast is more of a low-hanging, unpicked fruit (It doesn’t have much practical value, but that’s the case for pretty much every low-hanging fruit that hasn’t been picked yet.)

All in all, despite my failure to achieve even the most meager goal I set out for myself, I’m satisfied with how this project went. I gained a greater understanding for how neural networks are implemented in practice. My drum doodle project is less of a “I think this is probably possible, but I have no idea how to start,” and more of a “I could make some solid progress given some time,” and that’s really all I could hope for.

Works Cited

Blog posts by Liam Schoneveld (I primarily used the 2nd one.):

<https://nlml.github.io/neural-networks/detecting-bpm-neural-networks/>

<https://nlml.github.io/neural-networks/detecting-bpm-neural-networks-update/>

Autocorrelation function by unutbu:

<https://stackoverflow.com/a/14298647>