HW1.md 4.47 KB

1. @rice09:/farmshare/home/users//hw1$ git remote -v origin git@code.stanford.edu:bioe301p/22s/student//h_w_1.git (fetch) origin git@code.stanford.edu:bioe301p/22s/student//h_w_1.git (push)

2. The dataset was "Large-scale neural recordings with single neuron resolution using Neuropixels probes in human cortex " from the paper with the same name. This data is human electrophysiology data (local field potential and action potential recordings) mapped over multiple motor, language, and sensor regions in the brain in nine consenting patients using Neuropixel probes that can record multiple single unit recordings of neurons over 384 channels. These recordings were completed at Massachusetts General Hospital in Boston. There is a paper (https://www.nature.com/articles/s41593-021-00997-0) and a url (https://www.datadryad.org/stash/dataset/doi:10.5061%2Fdryad.d2547d840) for this dataset.

3. The file format for the data is in BIN files, which are binary files that have the data read into bytes and are easy to interpret and encode. The metadata includes at a specific to each recording type: such as recording duration (in seconds) , time and day of recording, and original data location among others.

4. The structure of this dataset is split into 3 types of neural recordings: action potential band (AP), local field potential band (LFP), and AP band that has been aligned after a movement artifact. There are three patients who have each type of this recording. Each patient ahs a metadata file associated with the unaligned AP and LFP data

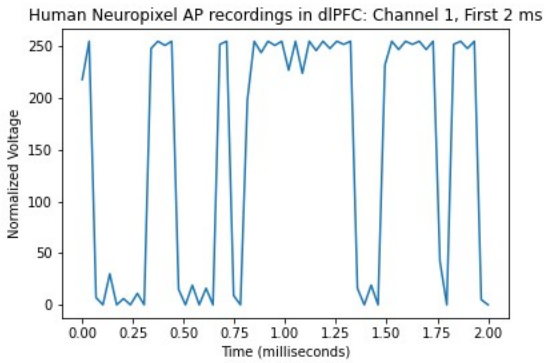
5. The size of the dataset is roughly 96GB, which is all uncompressed.

@rice09:/farmshare/user_data//humnpix$ du -ah 19K ./1268748 1.4G ./1268770 54G ./humnpix_re.h5 37K ./ch1_2ms_pt01_ap.png 11G ./1268763 19K ./1268769 58G ./humnpix.h5 19K ./1268764 13G ./example.h5 13G ./1268745 14G ./1268773 460M ./humnpix_r_2.h5 431M ./humnpix_r_3.h5 15G ./1268767 19K ./1268774 19K ./1268746 19K ./1268765 1.4G ./1268768 19K ./1268771 512 ./humnpix_real.h5 913M ./1268747 11G ./1268766 431M ./humnpix_r_grain.h5 54G ./humnpix_r.h5 15G ./1268772 259G . The files starting with 1268 (original files) all sum up to roughly 96GB. 6) I can could perform a mean of the baseline for the LFP and spiking data, calculate spike rate statistics, autocorrelations, interspike interval histogram, among others. This data set is well sutied to HDF5 format because it contains long timeseries data that can be easily binned and divided to save memory at a given time. 8) I created one group (humnpix) to store all of the data for each patient and recording type combination as well as the metadata. I recreated the initial shape of the data (channel by session length) so I could chunk each channel into a specific time bin. I save nearly half (53GB compared to 95GB) by saving it into HDF5 format, mainly due to the compression by gzip that I included in the create_dataset function. 11) I took the mean of the patient 1 action potential band for the first 3 channels (usually more but I needed to save time for the assignmetn), which is important to include so that if one where to understand the baseline signal for thresholding action potentials, this can be done. I used three chunk sizes, all of which varied in the time domain: 6000 timesteps (200ms), 12,000 (400ms), and 240,000 timesteps (8s). When I calculated the time before and after taking the mean: import time

start = time.time() print(np.mean(pt1_ap[0:2,:],axis=1)) end = time.time() print(end - start)

I found that the time to do the operation was 2.925s, 7.54s, and 24.4s. This makes sense because of how much time it takes to load the chunk to memory, especially if you are making it longer each time. The ideal chunk size would be smaller in the time domain, but not too small so you would not be able to sample your waveform of interest (1ms spike).

10.



12. Having cut against the grain (include all of the timeseries, but for only one channel) for the chunk size, the time it took was roughly 27s, which was surprisingly close to the highest chunksize for all the channels; but would not be ideal due to the fact that you would need to load each channel (384/385 of them) into disk which is very taxing.
13. Using the original size, the time it took was roughly on the order of minutes, which was expected considering that we had to load it all into memory.