**Data Preparation (Using MATLAB)**

For a code example and further function descriptions, see “example.m” Example folders and files in “dataprep”.

Our neural network trains off prelabeled waveforms. For spike sorting and creating pre-labeled data from waveforms, one can use our platform found at: <https://github.com/smithlabvision/spikesort>. Data should be in the following format:

A matrix of labels and waveforms of 1 x *s* in length, 1x53 (labels+wave) if using SmithCNN. Labels should be placed in the first column before the waveform, and should be formatted into spikes (1-254) and noise (0 or 255). This matrix should be stored as a .mat or .nev file within a variable named ‘waveData’. Waveforms should be converted into the same frequency and length, such that all waveforms are 1 x *s* in each file to be loaded. If using premade .mats, make sure waveforms are saved under ‘waveData’ using the ‘-v7.3’ flag. waveData should have the following format:

[0 or 255 wave; ... for noise, converted to 0 with “createNetworkFiles”

1-254 wave;]; for spike, converted to 1 with “createNetworkFiles”

Once all waveforms are selected, one can use the function in “createNetworkFiles.m” to load all waveforms in a specified folder and splitting into training and testing files. This process with relabel all spikes and noise in the previous section to the binary labels of 1 (spikes) or 0 (noise). Loaded waveforms will be split into the specified number of training files, ideally 1 if your hardware allows the entire file to be loaded into memory. If your hardware has less memory available, one is suggested to split into more files. Waveforms will be loaded and saved within the limits of memory of your computer, however the final output files for training and testing should have a load size less than your device’s total RAM. All waveforms loaded will be saved either saved into 100% training files (no validation testing) or into 80% training files, and 20% testing files, with folders created by the function. This function follows the format: createNetworkFiles(inputDir,outputDir, numSplits, trainandtest)

inputDir : where **all nevs and mats are located in subfolders**

--subfolder: Named with channels desired from nevs eg. "1-96" for channel 1-96. Any name can be used if folder contains .mats.

numSplits : splits waveforms the into this number of training files. If using a large amount of data on one machine, 2-3 may train without exceeding memory.

outputDir : places a split of training and testing data into two folders within this directory "training\_files" and "testing\_files" folders will be created.

trainandtest : 1 / creates both training and testing (validation) files

0 / creates only training data

For creating files for classification, one can use “createFilesforClassification.m”. One can use their own files, but data must follow the format for waveData above, even if all labels are 0. This function adds blank labels to the files and saves them as .mats for labeling with their original name + ”\_labeled.mat”.

The function follows the following format. createFilesforClassification(inputDir,outputDir)

InputDir: where nevs and mats are located in subfolders

--subfolder: Named with channels desired from nevs eg. "1-96" for channel 1-96. Any name can be used if folder contains .mats.

outputDir : saves all waveform files here. Ready to be classified by the network.

**Training and Testing the Network (Using PYTHON)**

For a code example and further function descriptions, see “runner.py”

Python (ver 3.8.10) running environment should have the following installed: Numpy(https://numpy.org/install/), Tensorflow 2.0 (https://www.tensorflow.org/install), Keras(https://keras.io/getting\_started/), Scipy (<https://scipy.org/install/>), pandas(https://pandas.pydata.org/), h5py (<https://docs.h5py.org/en/stable/build.html>), and datetime(https://docs.python.org/3/library/datetime.html). Anaconda(https://www.anaconda.com/products/individual) was used for running files through prompt or Spyder.

Once data is placed in folders, training and/or testing, the network can be trained or applied to the data. Matrices should be in .mat format with the name ‘waveData’. For an example of how to train and classify using our CNN, see runner.py. Functions available are (further defined in runner.py):

SmithCNN.trainnet(netname,ntimepts, trainset,testset) # for training networks from dataset

gamma = netFunctions.gammasweep(netname,modeldir, testdir) #returns the best gamma value

predicts = netFunctions.getPredicts(netname,modeldir, testdir, savedir) #to get network predictions for input waveforms\*

labels = netFunctions.getLabels(netname,modeldir, testdir, gamma, savedir) # returns predicted binary labels at a given gamma threshold\*

\* contains runtime options for saving predictions or labels into either .csv or .mat files with waveforms

Functions can be run using a runner file by:

In command prompt>> python runner.py

Or in any python IDE with runtime controls (eg. Spyder, part of Anaconda)

Or individually in command prompt such as

python -c "import SmithCNN; SmithCNN.trainnet(netname,ntimepts, trainset,testset)” # for training a network

python -c "import netFunctions; netFunctions.getLabels('SmithCNNnet','', 'D:/SmithSaves/test3/', 0.48, 'labels/' )" # For labeling waveforms

**For Reproducing Paper figures**

By using “figureprep.py,” one can reproduce the figures from the paper.

A pdf is included of the output of each code section in “figureprep\_notebook.pdf.”

Comments on the document delineate the code section for each resulting figure result. Figures generated from this document were (in order programmed:

Supplemental Figure 2, where each test set from each cortical area was swept in gamma value. This data was then averaged and summated using excel to produce Figures 2 and 3.

Figure 4.A-C, showing the differences in classification prediction (Pspike) across a channel of V4 data.

Figure 4.D-F, showing the differences in classification prediction across M1 data with networks trained on removed data.

Figure 5.A-B, showing a histogram of relative accuracy across waveforms binned by their peak max.

Figure 5.C-D, showing a histogram of relative accuracy across only spike waveforms binned by their peak max.

Subsets of the data (to keep within data storage limits) have been included in train\_mini and test\_mini folders. However, the pdf output was generated using the original files.