**Jacob Edwards**
**Multivariate Statistical Methods**
**Final Project**
**4/27/2021**

# A multiple classifier system for robust data cleaning in voltage recording experiments

## INTRODUCTION

Understanding the neural code, or the patterns of electrical impulses that occur in the brain, is a central pursuit of neuroscience. These impulses, commonly called "spikes", are how neurons transmit information, and they underlie perception, cognition, and behavior. They are a fundamental unit of analysis for researchers spanning a range of disciplines.

Among the most common methods for measuring spikes are extracellular electrical recordings (or, voltage recordings). In these designs, an electrode, often with many conductive sites, is lowered into the brain of an animal or human. The electrode allows researchers to "eavesdrop" on the activity of neurons while presenting stimuli of interest to the study aims. Other techniques include imaging with genetic markers, or changes in blood flow, such as in functional MRI. Voltage recordings with an electrode, however, have superior temporal and spatial resolution of neural activity, and are often preferred when possible.

One of the major drawbacks of voltage recordings is the substantial post processing required. Spike signals must be isolated from the background, and are easily contaminated by electrical noise or slight motion of the recording subject, as illustrated in **Figure 1**. Further, a continuous recording may contain spikes from more than one neuron, in which case they must be assigned to distinct classes. Nearly all modern algorithms use some form of an unsupervised clustering approach, searching for and naming distinct peaks in multidimensional space - e.g. "neuron 1", "neuron 2", "noise" – and assigning each spike its nearest cluster label.

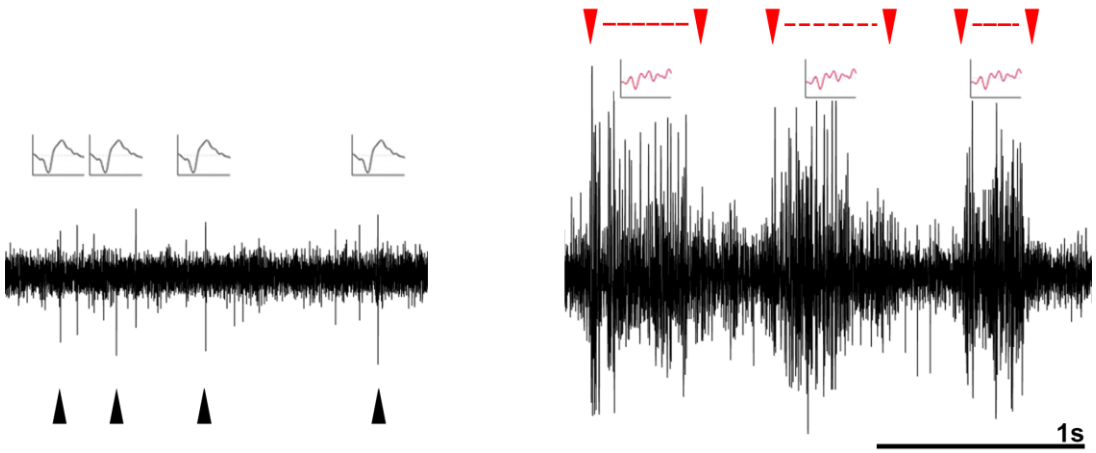


**Figure 1.** *Left*, a recording of electrical potential (voltage, y-axis; time, x-axis) from a songbird brain. Sharp deflections (black arrowheads) are spikes. *Right*, another recording segment, showing several movement artifacts (areas between red arrowheads). Y-axis scale is the same in both panels.

But, assigning spikes to neurons is not by necessity the same job as discerning spikes from noise. Spikes are constrained by biology. For example, their voltages go quickly negative, then slowly positive, then back to baseline. *How* quickly and slowly depends on the neuron, but severe deviations from this overall pattern should be obvious even to an untrained eye. Such events can be removed, resulting in a cleaner dataset for the sorter – human or computer. **Figure 1** highlights a common problem in voltage recordings: both spikes (left) and noise (right) are sharp voltage deflections that cross a threshold and are stored as events (panel insets).

Here, I developed and tested a classification system for discerning spikes from non-spikes, based on data labeled previously by two experts in our lab. I trained and evaluated Random Forest (RF) models, nonlinear Support Vector Machines (SVM), and a 4-dimensional kernel density estimate (KDE) on their ability to classify new data as "spike" or "non-spike". For the RF and SVM, I trained models on 5 feature mappings of spike data across two large datasets, and used cross-validation to estimate performance. Finally, I applied the models to new spikes that I have recently recorded. Combining the output of each model's prediction resulted in exceptional and robust noise removal despite variation in signal-to-noise ratio across recordings. While full automation of spike post-processing still lies far ahead, partitioning the job into spike-noise separation before spike-spike sorting is a promising approach for reducing labor hours and subjectivity, and may improve the discriminability of spike classes for cluster algorithms further down the analysis pipeline.

All feature extraction, modelling, and evaluation was performed in MATLAB 2020b. All relevant code is attached alongside this report.

## MATERIALS & METHODS

### *TRAINING DATASETS*

I investigated two training datasets, each containing events labeled "spike" (label 1), or "non-spike" (label 0). Both datasets came from our lab at Columbia and were published in the last two years (So, Edwards, & Woolley 2019; Moore & Woolley 2019). A summary of class breakdown is provided in **Table 1**. Each observation/row in each dataset is a 51-length vector, with the first column as the class label. Visualizing the time samples in series results in the signature "spike" shape, and plotting many spikes and non-spikes in a heatmap results in distinct "textures", with a rougher appearance as shown in **Figure 2**. Note the dark valley in the image, which is the threshold-crossing event to which all samples are aligned.

**Table 1.** Datasets investigated in this project, summarized.

| *So Dataset* | | % of total dataset |
|---|---|---|
| # of events/rows | 156,237,257 | |
| # of true spikes (1) | 145,798,650 | 93.32 % |
| # of rejected events (0) | 10,438,607 | 6.68 % |
| Size on disk | 55.9 GB | |
| *Moore Dataset* | | |
| # of events/rows | 255,163,319 | |
| # of true spikes (1) | 228,843,900 | 89.69 % |
| # of rejected events (0) | 26,319,419 | 10.31 % |
| Size on disk | 99.6 GB | |

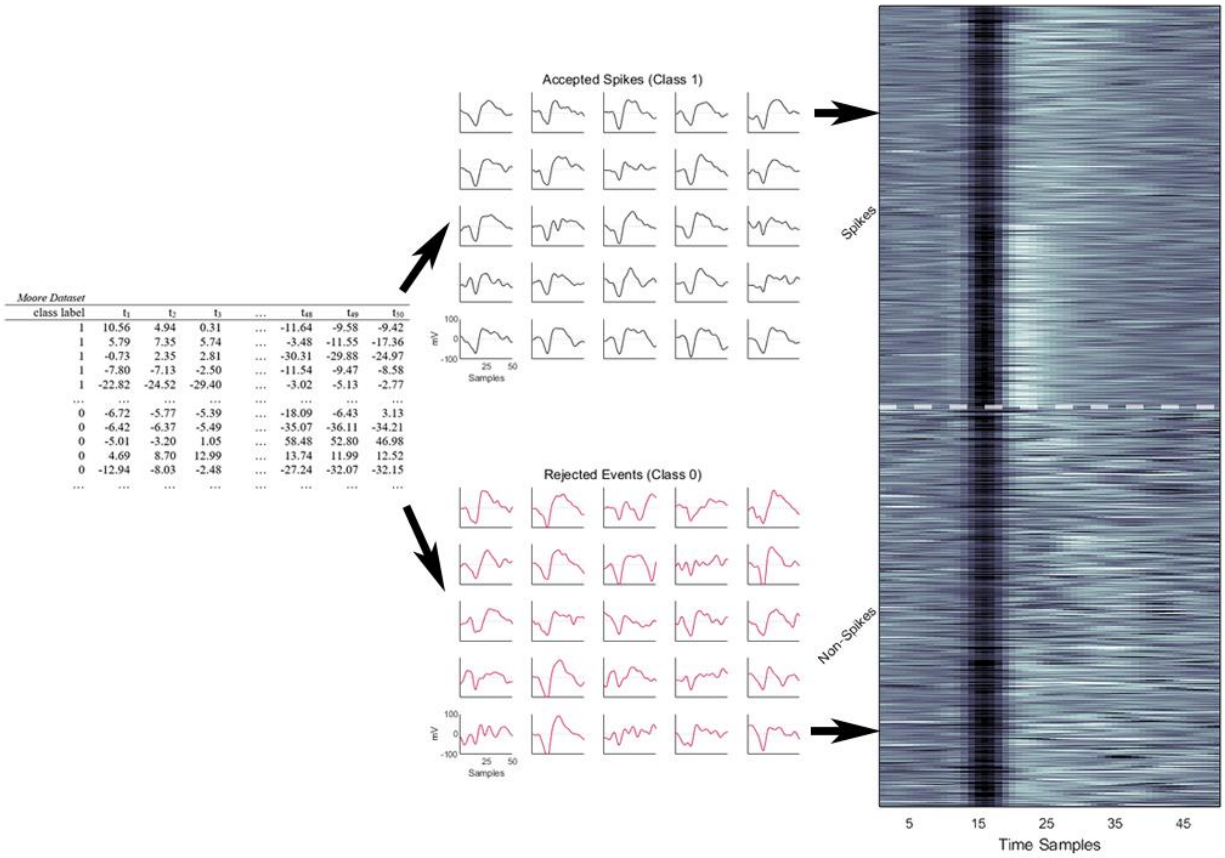


**Figure 2.** Representations of the raw data. ***Left***, the raw data matrix, with columns for class label, and 50 time steps corresponding to the voltage reading at each time. ***Middle***, displaying each row as a time series produces either a spike-like shape or a noisy waveform. ***Right***, viewing a subset of the data table as a heatmap reveals the "texture" difference between spikes (top half) and non-spikes (bottom half). Note the subtle difference in the spikes subset – the bottom half has a brighter peak abutting the dark valley. There are likely two real neurons in this sample.

Of important note: While these data were labeled by experts, they are by no means perfect, and the subjective element of labeling is a key limitation to the success of all learning algorithms. Carefully looking at the middle panel of **Figure 2**, which is a random sample of "spikes" and "non-spikes", one can quickly find "spikes" that look like "non-spikes", and vice versa. In fact, despite extensive experiments on the So Dataset, no algorithm reached greater than 68% cross-validated classification accuracy (data not shown). **For this reason, the remaining RF and SVM analyses are based solely on a subsample of the Moore Dataset**.

Finally, training the models on the full datasets proved to be too time-intensive for the scope of this project. Further, a simple random subset led to a drastic underrepresentation of non-spikes that resulted in poor performance (in the RF case, most of the weak learners never encountered a non-spike). For these reasons, the RF and SVM models were trained on balanced random subsets sampled without replacement, with equal numbers of spikes and non-spikes.

*FEATURE EXTRACTION*

To maximize the discriminability between classes, I tested classifier performance on the raw event waveforms (as in **Figure 2**) and features extracted using four methods described below. I keep the discussion brief, as for both RF and SVM, the raw event waveforms outperformed all feature extraction methods.

*Discrete Wavelet Transform.* The discrete wavelet transform is a one-dimensional analog to the Viola-Jones algorithm for object recognition. It convolves non-overlapping filters of successively smaller sizes against the signal, returning a large coefficient when the wavelet matches a feature in the signal. The Haar wavelets in the wavelet transform are thus analogous to Haar features in the Viola-Jones algorithm, and register changes in the signal at many frequencies and positions. **Figure 3A** shows a schematic of the shape of the Haar wavelets and their resulting coefficients. The wavelet coefficients in **Figures 3A & 4** are ordered from the largest (lowest-frequency) on the left, to smallest (highest-frequency) on the right. In general, the largest differences in spikes vs. non-spikes are in the low-frequency range, corresponding to broad, sweeping changes in the signal rather than fast changes. The wavelet transform has been used successfully in a spike-sorting application by Souza et al. (2019).

*Wavelet Scattering Network Transform.* This feature extraction method was developed by Mallat and others (e.g. 2011) as an extension of the Wavelet transform with two key properties: it is stable to small deformations of the signal, and is invariant to translation of the signal. I was interested in trying this method because of the stability to small time warpings, and its successful applications in music and speech classification. However, because it resulted in the same performance as the "vanilla" wavelet transform, I won't elaborate on it here. My intuition is that the precise alignment of all spike/non-spike events in the dataset precluded any advantage this method might offer. Note the repeating pattern in **Figure 4**, which corresponds to the four scattering paths of the network, each one appears to pick up changes in the signal at the beginning, the end, and towards the middle where the threshold-crossing event happens.

*Principal Components.* I attempted to find the axes that maximized the variance in the data using principal components analysis. I first normalized the data by *z*-scoring, and converted each waveform to a vector of PC loadings. **Figure 4** shows that the largest loading values are in the first ~20 PCs, as expected, however I passed the full vector as training data to the models.

*Hand-engineered Features.* In the past, our lab has measured four key features of the spike waveform: the duration of the negative peak, the duration of the positive peak, the duration of the peak-to-peak distance, and the voltage ratio between the two peaks (**Figure 3B**). These features require some definition of baseline, a clear minimum, and a clear maximum, which are not obvious to determine from non-spike events. Because of this, I expected this feature extraction method to perform poorly. From **Figure 4**, it is difficult to discriminate between spikes and non-spikes by eye based on these features.

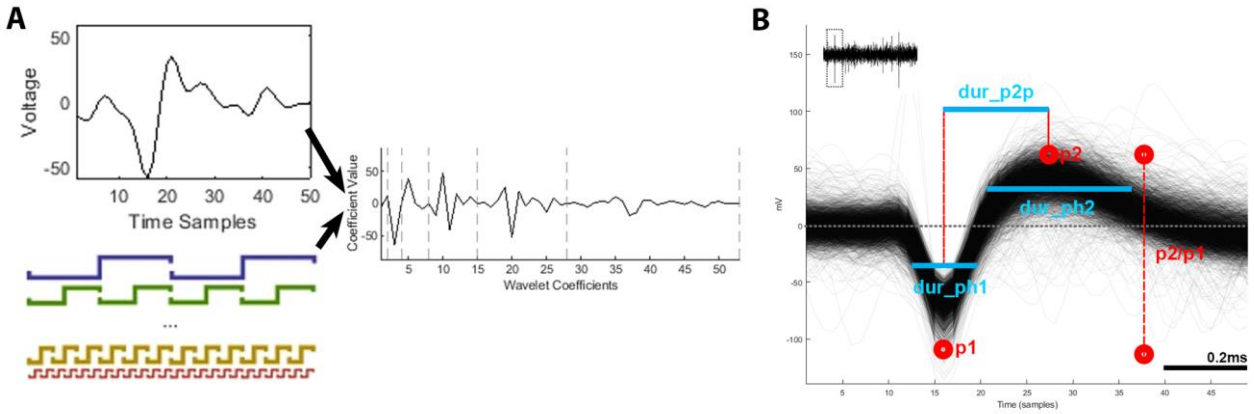


**Figure 3.** Two feature extraction methods. **(A)** Discrete wavelet transform convolves a signal (top) with successively smaller, non-overlapping waveforms, derived from a "mother wavelet". Here, a Haar wavelet is used. The operation returns a series of coefficients, each responding to a wavelet. **(B)** Hand-engineered features are chosen based on the known shapes of real spikes.

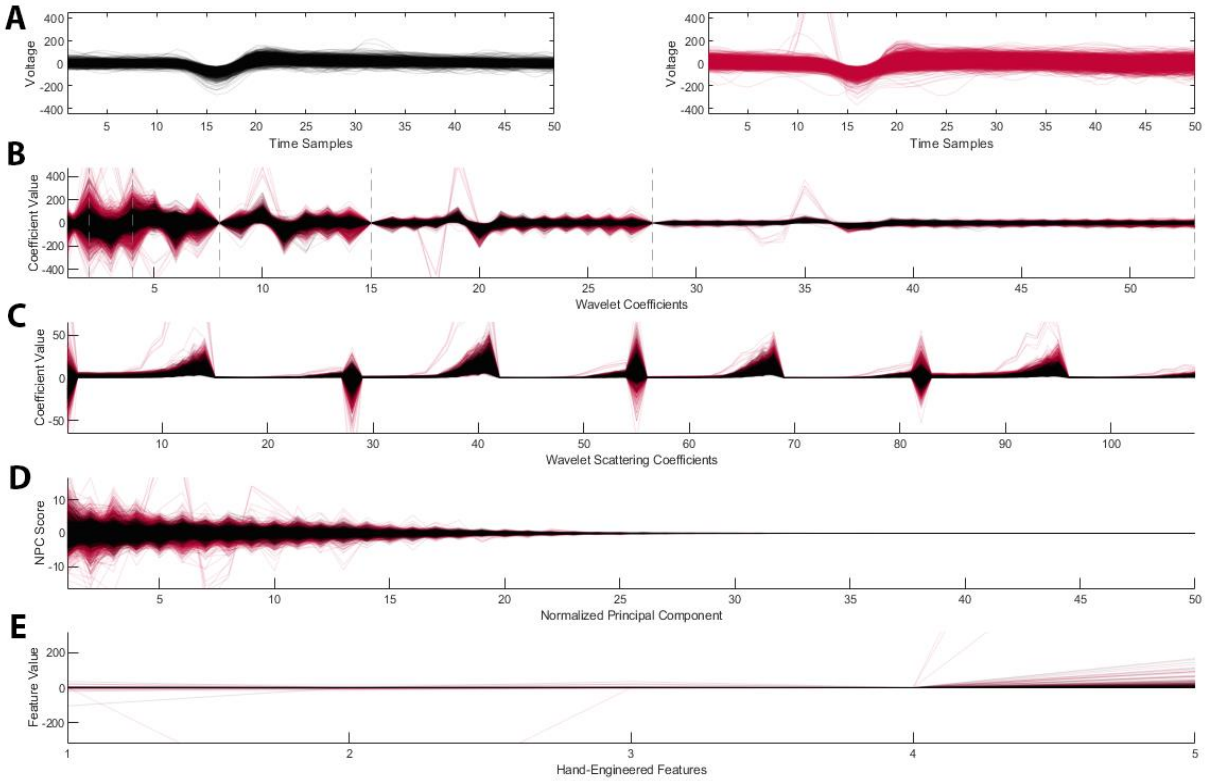


**Figure 4.** Feature extraction results on a small subset (~5000 samples) of the data. All "spikes" are in black, all "non-spikes" are in red across plots. **(A)** *Left*; Raw data (x-axis, time; y-axis, voltage) waveforms for events labeled "spikes". *Right*; Raw data (same axes) waveforms for events labeled "non-spikes". **(B-E)** All plots are parallel coordinate plots, with x-axis labeling increasing index. **(B)** Wavelet transform features. Note highest spike/non-spike separation at smaller indices, corresponding to slower frequency changes. **(C)** Wavelet scattering network features. Note repeating pattern as the mother Morlet wavelet is shifted slightly in time. **(D)** PC features. Note the values flatline after ~20 PCs, as expected. **(E)** Hand-engineered features. Note minimal variability in all features except perhaps at index 5 (max peak to negative peak ratio).

*CLASSIFIER ALGORITHMS*

All classifiers were trained and cross-validated (except KDE) using built-in MATLAB functions: *fitcsvm* for SVM training and optimization; *TreeBagger* for RFs; and *mvksdensity* for KDE. Both SVMs and RFs were trained on the raw spike/non-spike waveforms, and each of the four extracted feature mappings.

*Kernel Density Estimation (KDE).* My first approach to this problem was to model the feature distributions of real spikes, determine a threshold, and reject any event as non-spike that exceeded the threshold. Because KDE's do not scale well to high-dimensional space, all KDE analyses were done on the hand-engineered features described above. Along several dimensions, the raw data spanned the full range of possible values, rendering a density estimate meaningless. So, instead I used measurements of the average spike waveform of each neuron across the two datasets (9,071 neurons) to fit the KDE. I selected a cutoff threshold based on extensive testing, but with more time, a better approach would be to search for the cutoff value that minimizes the misclassification rate on the labeled data.

*Support Vector Machines (SVM).* I fit nonlinear SVMs to the training data using a Gaussian radial basis function (*rbf*) as the kernel. SVMs were evaluated and optimized using 10-fold cross-validation, and a hyperparameter grid search of kernel scale (width of the Gaussian kernel) and box constraint (cost of misclassification).

*Random Forests (RF).* I trained RFs up to sizes of 150 trees. Each decision tree was trained on a bootstrapped data sample, and a subset of training features. Therefore, the algorithm provides a built-in cross-validation metric and measure of feature importance by evaluating performance on samples not used for training (out-of-bag), and which features led to splits that produced the lowest misclassification rates. Trees were not pruned.

*MULTIPLE CLASSIFIER SYSTEM – APPLICATION TO NEW DATA*

The goal of this project was to develop a classifier that could separate real spike signals from noise on new data. Because each classifier (KDE, SVM, RF) might offer a different outlook (though they are not truly independent), I combined the predictions of each into a multiple classifier system. Similar to the RF approach, I allowed each of the three classifiers to cast a 'vote' as to whether each new event was a spike or non-spike, and took the majority vote (best 2-out-of-3). Having no ground truth, I assessed the classifier performance on new data visually, both by inspecting the waveforms of events deemed spike and non-spike, and by inspecting which events were classified over time. Non-spike noise events tend to occur in clusters over time, and span a range of voltage magnitudes within those clusters, whereas spikes are comparatively consistent over time.

## RESULTS

*KDE*

The KDE classifier rule amounts to setting a threshold value, above which events are classified as spikes, and non-spike otherwise. This model is different from the RF and SVM, in that the density was estimated on the hand-engineered features of average waveforms from each neuron in both datasets (9,071 neurons). The advantage is that it captures the diversity of spike shapes, however it does not capture the variability in individual spikes or noise. **Figure 5** shows

scatter plots in two of the four dimensions (phase 1 duration and peak-to-peak duration), with contours fit to the raw data, and results of the decision process on new data. While I did not optimize the threshold parameter using the labeled data, the value chosen (0.00000006893) performed well after extensive testing. Overall, the KDE classifier performed best when spikes were easily discernible from non-spikes, i.e. the signal-to-noise ratio was high. It performed worse, and tended to reject more real spikes when the signal-to-noise ratio was low (not shown).

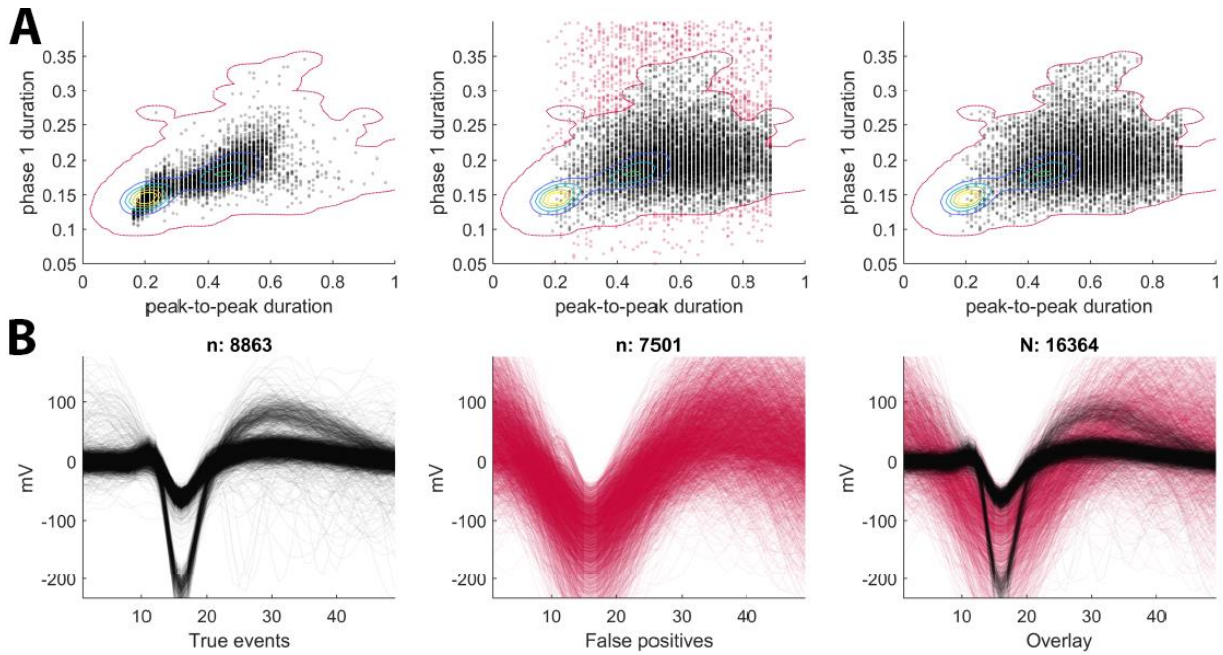


**Figure 5.** Illustration of the KDE classifier. The KDE performed well when the signal-to-noise ratio was high. (**A**, *left*) Bivariate scatter plot of average waveform data on hand-engineered features from 9,071 neurons, from which the KDE was fit. Contours show peaks in the density; red contour is threshold value. (**A**, *middle*) Same axes and contours, but with new data. Red points are classified as non-spikes, *right*, with non-spikes removed. **(B)** Raw waveforms of events classified as spikes (*left*), non-spikes (*middle*), and their overlay (*right*).

*SVMs*

Nonlinear SVMs were trained and cross-validated using the 10-fold misclassification rate. Hyperparameters were optimized using a grid search along the box constraint (cost of misclassification) and kernel scale (width of the Gaussian kernel; left column in **Figure 6**). At small training samples (1-2,000 events) during prototyping, the best SVMs achieved nearly 80% cross-validated accuracy (middle of **Figure 6**). Unfortunately, on larger samples (200,000+ events), the models either failed to converge, or could not perform better than random guessing (50% misclassification rate; data not shown). At small samples, the raw waveform data resulted in best performance, and hand-engineered features yielded the lowest (65% accuracy). Confusion matrices showed that only PCA- and hand feature-trained models were biased in misclassification: non-spikes were more often misclassified as spikes, than were spikes to non-spikes (right column of **Figure 6**).

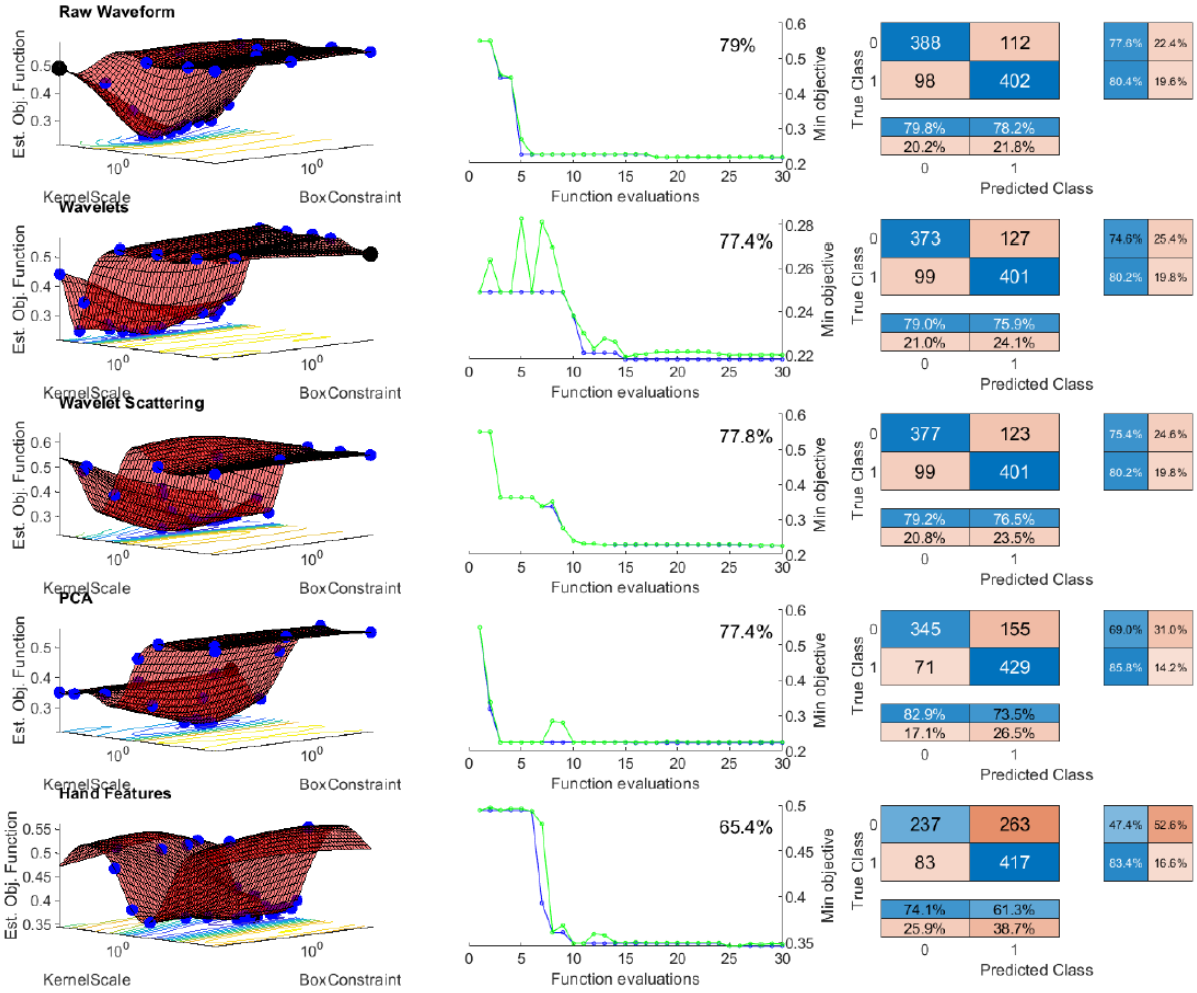


**Figure 6.** Results of SVM training and hyperparameter optimization. Each row is a model trained on a different feature mapping. ***Left***, grid search for best hyperparameters. Width of the Gaussian kernel (KernelScale) appeared more important for optimizing than the cost of misclassification (BoxConstraint). ***Middle***, observed (blue) and estimated (green) misclassification rates. Percent accuracy is labeled in top right of each plot. SVMs trained on the raw waveform gave the highest accuracy. ***Right***, confusion matrices for each model (with row and column summaries). Note that the PCA and hand feature models tend to mistake non-spikes for spikes (false positives), whereas the other feature mappings produce no bias in error rates. Also note, these models were trained on a very small subset (1,000 events) of the data. The models could not perform better than random guessing on large training samples.

*Random Forests (RFs)*

Random forest models were trained and evaluated using the out-of-bag misclassification rate. One advantage of RFs in practice was that they maintained good performance, even on the largest subset of training data I tested (2,000,000 events). Accuracy ranged between 76-79% on training data sizes of 1,000; 10,000; 100,000; and 2,000,000 (not shown). RFs are designed to overcome the high variance of individual learners, and the robust performance here appears to reflect that. The results of training and evaluation are shown in **Figure 7**. Of note, regardless of feature mapping, nearly all features were useful for classification, making a case against dimensionality reduction. Again, the raw event waveforms yielded the best accuracy. All feature mappings resulted in a slight tendency to misclassify non-spikes as spikes, but this was most

severe in the hand-engineered feature set, which also produced the lowest accuracy. Misclassification rates quickly converged after a forest size of ~20 trees.

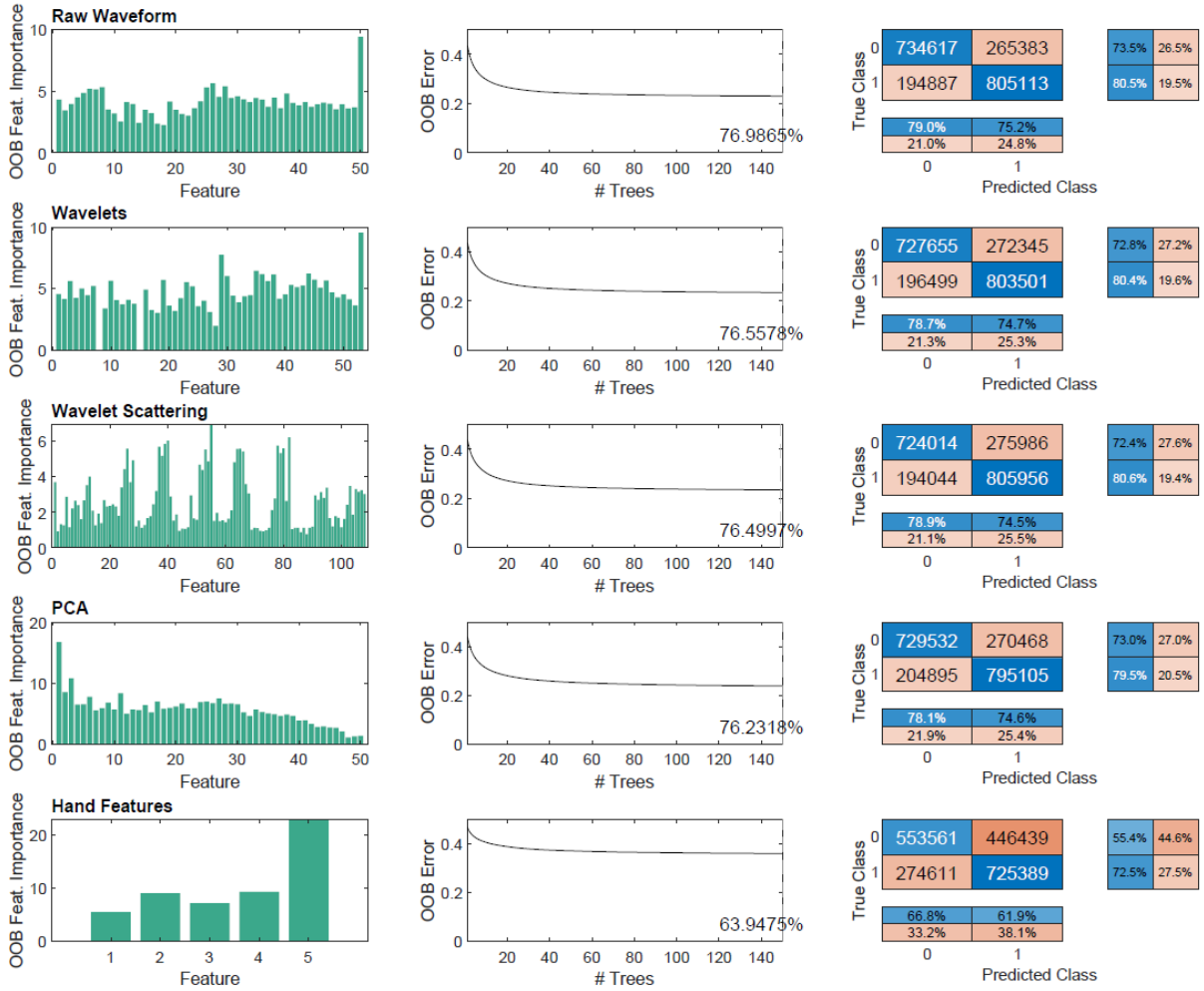


**Figure 7.** Results of RF training, evaluation, and feature importance. Each row is a model trained on a different feature mapping. ***Left***, feature importance scores, ordered by index. Note that, in general, all features for a given mapping are useful for classification. ***Middle***, cross-validation (out-of-bag) misclassification rate as a function of forest size, with average accuracy annotated. Note that misclassification rates quickly converge after about a size of 20 trees. ***Right***, confusion matrices for each model (with row and column summaries). Note that there is a slight tendency to misclassify non-spikes as spikes, with a severe bias for the hand-engineered feature mapping.

*Multiple Classifier System – Application to new data*

Because the SVM and RF models performed best on the raw event waveforms, I used these models to evaluate classification on new data that I have recently recorded. The KDE was fit using hand-engineered features, so I measured these for each event in the new data set, evaluated the density at their positions, and compared the value to the cutoff threshold to determine a class label. **Figure 8** displays an exemplar recording with full summaries of the multiple classifier output. In general, the system successfully removes noise events, with varying levels of disagreement between classifiers.

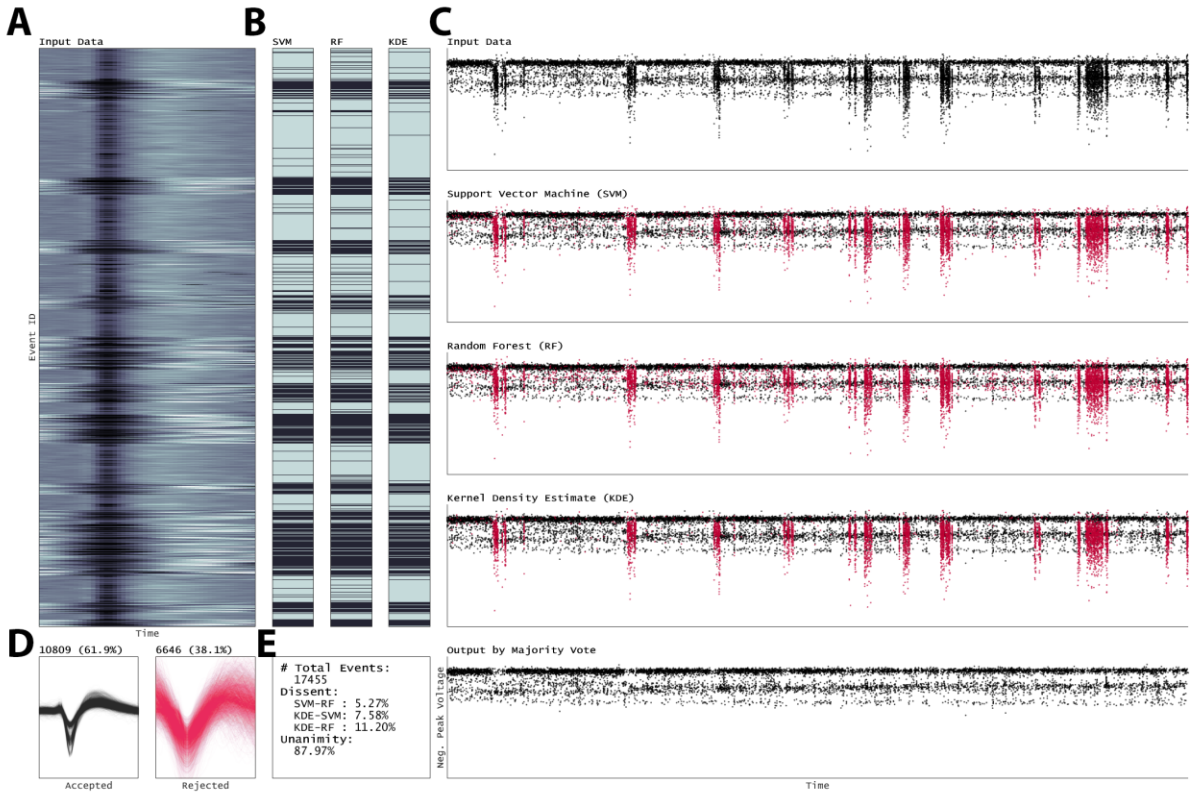


**Figure 8.** Multiple classifier predictions on new data. The system successfully removes noise events and retains real spikes. **(A)** Input data represented as a heatmap like in Figure 2. Rows with wide fluctuations are noise. **(B)** Predictions from each model aligned to the input heatmap. Light gray/blue rows are predicted spikes, black rows are predicted non-spikes. Note the black segments tend to align with the noise fluctuations in (A). **(C)** Raw input data as recorded over the actual time of the recording session. X-axis spans roughly 75 minutes. Y-axis is peak negative voltage, which reveals separate bands corresponding to different neurons. Noise clusters are visible as streaks down the y-axis. Models effectively isolate these noise streaks, colored in red, while retaining spikes in black. Bottom panel shows final output by majority vote, with removal of noise. **(D)** Waveforms of accepted spikes (black) and rejected non-spikes (red) along with total counts for each and percentages of all events. Panel **(E)** summarizes agreement and disagreement between models. In this example, all three models agreed on 87.97% of predictions.

Unanimous agreement among the models ranged from 74-92% across 32 experiments on new data. In cases with a good signal-to-noise ratio, SVM and RF tended to agree with each other most, while KDE was more lenient with what it accepted as spikes (note the fewer small black bands in **Figure 8B** above). In more difficult cases, RF and KDE tended to agree more, with SVM overly misclassifying spikes as non-spikes (not shown). The majority rule of the multiple classifier thus provides a more robust classification by considering model agreement.

Finally, to confirm that de-noising from the classifier did not degrade the stimulus response of neurons (spikes aligned in time to the stimulus), I inspected “raster plots” for consistency before and after applying the classifier system. Across all 32 experiments, the actual signal (spike responses to the stimulus) was nearly perfectly preserved. An example is presented in **Figure 9**. The top panel shows the audio waveform (sound stimulus) of a bird song. The bottom two panels are the same recording data, before and after spike/non-spike classification. The rows in each panel correspond to a trial/repetition – the neuron was presented the same

sound stimulus 10 times (10 rows). Each vertical black bar is a spike. Visible columns of black bars down the raster plot indicate a response of the neuron to that time point in the stimulus. The two panels are nearly identical, showing that the stimulus response of the neuron (the real signal of interest) is preserved.

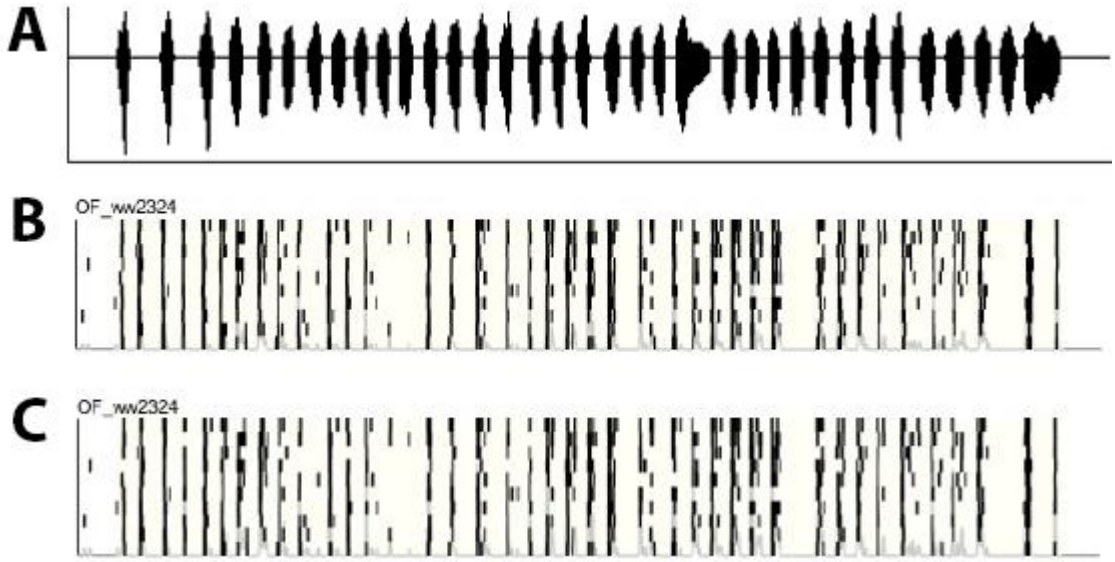


**Figure 9.** Raster plots indicate the classifier preserves the spike response to the stimulus. (A) The audio stimulus waveform. A bird song is presented to the neuron here. (B) Raster plot before classification and noise removal. Each row corresponds to a trial/repetition (10 stimulus presentations total). Each vertical black bar is a spike. Spikes aligned down the columns indicate a response to the stimulus. (C) Raster plot after classification and noise removal. The response to the stimulus is nearly perfectly preserved.

## CONCLUSIONS

In this project I explored three multivariate classification techniques and four feature extraction methods with the aim of easing the burden of processing voltage recordings from the brain. Post processing is really two steps that are often treated as one: separating spikes from noise, and separating spikes from other spikes. Automating the spike-noise separation phase has high potential for reducing labor hours and subjectivity, and improving data quality for downstream analyses. Using expertly labeled data on massive datasets allowed for supervised model training, which achieved reasonable cross-validation accuracy. Even imperfect labeling will greatly streamline the processing workflow. On new data, the aggregate decision of the three models provided exceptional de-noising.

Because no model achieved greater than 80% accuracy across many experiments of sample size, hyperparameters, feature extraction, etc., my belief is that 80% may be the optimal limit for spike/non-spike discriminability in this dataset – *i.e.* the same human expert labeling a new dataset would incorrectly classify 20% of the time! This highlights the need for a rigorous solution to the spike-noise and spike-spike sorting problem that minimizes subjectivity.

Combining the predictions of multiple learners, even if not truly independent, provided robust classification across a range of signal-to-noise ratios. Random Forests handled massive training sets and had consistent cross-validation accuracy, whereas SVMs suffered from convergence problems on larger training sets. For both methods, performance was maximal

using the raw input data, however features extracted using wavelets were close behind, and may be improved with more careful parameter adjustment. In all cases, hand-engineered features provided the least discriminability between spikes and non-spikes. Random Forests, which allow for feature selection, indicated that nearly all features were useful in discriminating classes. These results are in line with Parikh (2018) who showed that Random Forests trained on raw spike waveforms were among the top performing algorithms for sorting spikes from motor cortex. Finally, de-noising preserved the real spike response to the stimulus, which is the critical data that allows us to make inferences about how the brain encodes information and generates perceptions of the outside world.

This project was a great learning opportunity, and looks to be already a useful tool for my ongoing data collection and processing. Thanks for all your input and advice!