Spike Sorting Algorithm Design Using Simulated Data

13807

December 15, 2021

1 Introduction

Spike sorting algorithms extract the timing and identity of neural action potentials (APs) and are an essential component for gaining meaningful information from neural recordings. This report details the implementation of a spike sorting algorithm applied to neural recordings from the brain trained using a data set of labeled neuron APs from a simulation. The classifier is not designed to be deployed off-line and therefore focuses on maximising accuracy.

The spike sorting algorithm pipeline used here is shown in figure 1, which demonstrates the operations applied to neural time series data to extract the time of neuron firing and neuron identity. First, filtering is applied to remove noise from the signal. Second, spike detection is performed which outputs a set of M clips, one for each spike, which are each n samples long and centered around their respective spike peaks. These clips are attributed time (or sample) stamp (t_m), which can further be refined during classification. Next the clips are all classified and assigned a neuron identify (k_m). The final output is then a series of time stamps and neuron labels which can be used for higher-level analysis of the neural system. Lastly, the output can also be compared to known data to evaluate certain performance criteria.

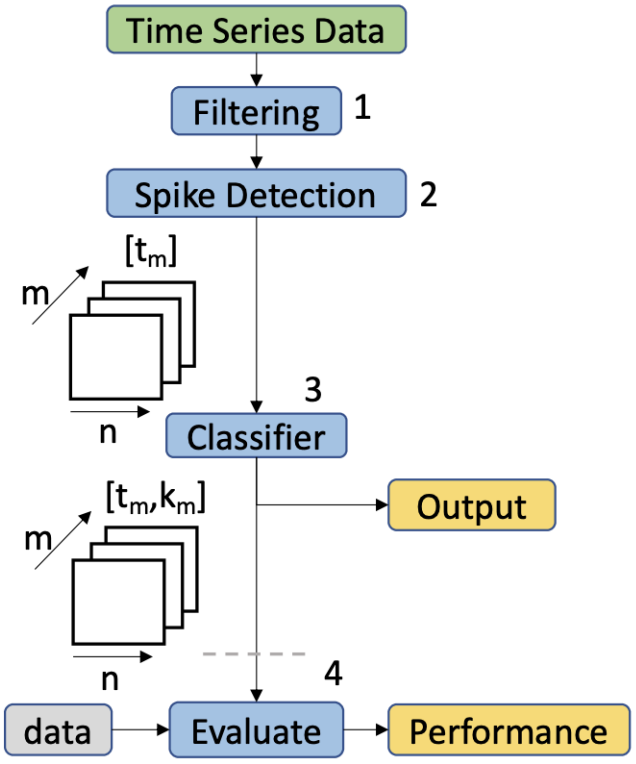


Figure 1: Spike sorting pipeline used to classify new data into neuron AP time stamps and identities (t_m, k_m).

2 Performance Metrics

2.1 Confusion Matrix

The concept of confusion matrix will be useful for quantifying the performance of various elements of the spike sorting algorithm. For a set of neuron labels $\mathbf{k} = \{1, ..., K\}$ (in this case K=5), two classifications are performed and a confusion matrix $Q^{(K+1)\times(K+1)}$ is formed, shown in figure 2. Its rows n represent the categories of the first classification (c_1) and the columns m represent the same categories for the second classification (c_2). The entries (n,m) represent the sum of the classifications labelled $\mathbf{k}$(n) in c_1 and labelled $\mathbf{k}(m)$ in c_2. Spikes are associated through their spike times (within a tolerance). Spikes detected and classified in c1 which have no associated spike in c2 and vice versa are placed in their respective null ($\emptyset$) cells. Thus a confusion matrix with only diagonal terms has 0 misclassifications, and non-diagonal terms indicate misclassifications.

2.2 Filtering and Spike Detection

The main purpose of filtering is to remove low frequency drifting and high frequency noise to facilitate subsequent spike detection. Due to this dependence, the performance of the filtering and spike detection are performed together. A good first test is to assess the number of spikes detected on the training set since the number of spikes are known. In general, slightly more spikes detected than present in the data is advantageous since false positive spikes can later be discarded [1]. However, the in-vivo dataset on which the spike sorter is to be applied has substantially more noise, and therefore the spike detector must also be designed with the submission data set in mind. Although the real spike count is unknown, a sensitivity analysis can be performed on the number of spikes detected with respect to a change in threshold of the spike detector. The detection sensitivity S_d can be estimated using equation 1

$$S_d = \frac{\Delta n_d}{\Delta th}\bigg|_{n_d=3343} \tag{1}$$

where n_d is the number of detected spikes and th is the threshold value of the spike detector. Equation 1 is taken

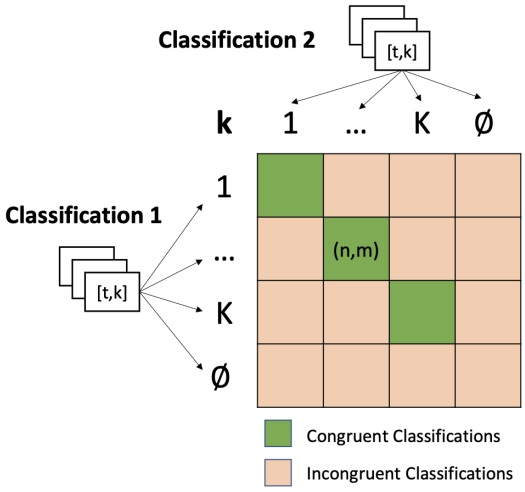


Figure 2: Confusion matrix which shows the congruent and non-congruent classifications between two classification results.

about $n_d = 3343$ since this is the number of spikes in the training data, which is assumed to be similar in the submission data. A spike detector with low sensitivity is more robust since the number of spikes detected depends less on the threshold value, and is therefore likely more generalisable to new data with more noise.

2.3 Classifier

A confusion matrix Q can be created using the output of the classifier (here c1) and the labeled data (here c2). In this case, the associated labels are known and thus clips are not associated via their indices. This also means the classifier performance, evaluated alone, does not contain false positives/negatives (i.e. no null row or column) by definition and thus only classifies true clips. Its accuracy can be determined for each neuron type a_k using equation 2. Alternatively, the total accuracy can be determined by clip type (merged, connected, or isolated - defined in 3.3.1) by dividing the sum of diagonal terms of Q by the sum of all Q entries, with one Q matrix for each clip type.

$$a_k = \frac{Q(k,k)}{\sum Q(:,k)} \tag{2}$$

2.4 Spike Sorter

The complete spike sorter performance is also assessed using a confusion matrix Q. Unlike the confusion matrix in 2.3, the associations between detected spikes and the labeled spikes are not known *a priori*, and therefore they must first be associated by the proximity of their indices before being placed in the confusion matrix. This is done by associating the closest labeled spike to one from the spike sorter, within a tolerance of 10 indices. Spikes that are detected from the spike sorter but not present in the labeled data are considered false positives, and are placed in their respective row in the null column. Spikes undetected by the spike sorter but present in the labeled data are false negatives, and placed in their respective column in the null row.The number of false positives can be found by summing the null column and likewise for false negatives by summing the null row. Also, the total accuracy of the spike sorter can be found by dividing the sum of diagonal terms by the sum of all Q entries. Lastly, the classifier accuracy can be calculated by removing the null column and row and performing the same analysis as above in 2.3.

2.5 Stability Using Self-Blurring

The spike sorting algorithm will eventually be applied to new unlabelled data for which the above accuracy metrics cannot be evaluated. It would therefore be useful to have a performance metric for spike sorting algorithms on unlabeled data to estimate the performance on new data. Some methods quantify feature space isolation of clusters or attempt to identify non-uniform variance in detected spike shapes for individual categories [1], but these methods are either classifier specific or make assumptions about the noise in the signals which may not be applicable [2]. Therefore, a technique called *self-blurring* is used [2] which requires no ground-truth of the signals. Given a disturbance, it assesses the stability of each neuron classification, thus providing an estimate of the robustness of the classifier.

2.5.1 Stability

For a confusion matrix created from two classifications c_1 and c_2 on the same data, the *stability* of each classification category is defined as

$$f_k = \frac{2Q_{k,k}}{\sum Q_{k,:} + \sum Q_{:,k}}, \; k = 1, ..., K. \tag{3}$$

The stability for each neuron category approaches 1 if all values in the row k and column k are concentrated in the diagonal term (k,k), and approach 0 if all values lie outside the diagonal term. A stability of 1 indicates that both classifications are identical, and thus stable.

2.5.2 Self-Blurring

The self blurring process is applied to each classified clip so that a second classification can be performed on the self-blurred clips and the neuron categories' stability assessed. Let x_k be the set of signals that have been classified as belonging to the neuron k. A new set of signal values $\tilde{x}_k$ can be formed using equation 4:

$$\tilde{x}_k = x_k + \gamma(\hat{x_k} - W_k), \tag{4}$$

where W_k is the mean waveform of all x_k, $\hat{x_k}$ is a randomized vector of x_k, and $\gamma > 0$ is a scaling factor that increases the blurring amount. In this way blurring is added through noise present in the classified clip's category. The stability of each neuron category is found using the following steps

1. Classify unlabeled clips using the classifier.
2. Perform self-blurring in each category using equation 4.
3. Asses stability f_k between the original classification and self-blurred clips using equation 3.

Although self-blurring stability is not a measure of *absolute* accuracy, it can provide an indication of a poor classifier on unlabeled data by indicating badly separated clusters in feature space. In other words, it can indicate whether a solution is bad, but not necessarily if it is good.

3 First Implementation

3.1 Filtering

Filtering was originally implemented using a 0.3-3kHz second order section Butterworth band-pass (BP) filter, which are standard values for spike sorters [1]. The filtering was performed using forward-backward filtering to avoid inducing phase shift in the filtered signal. However, when applied to the submission data it was observed that the spike detector described in 3.2 was sensitive to changes in threshold. This is problematic since the spike detection risks creating either large amounts of false positives or false negatives. Therefore, an alternative to a simple linear filter was explored. Wavelet denoising was implemented using the sym4 mother wavelet and 1st level decomposition followed by a 50Hz high-pass Butterwoth forward-backward filter to remove electrode drift. This showed substantially better performance, as shown in figure 3. Importantly, the wavelet denoising filter is almost 4x less sensitive based on the sensitivity criterion in equation 1.

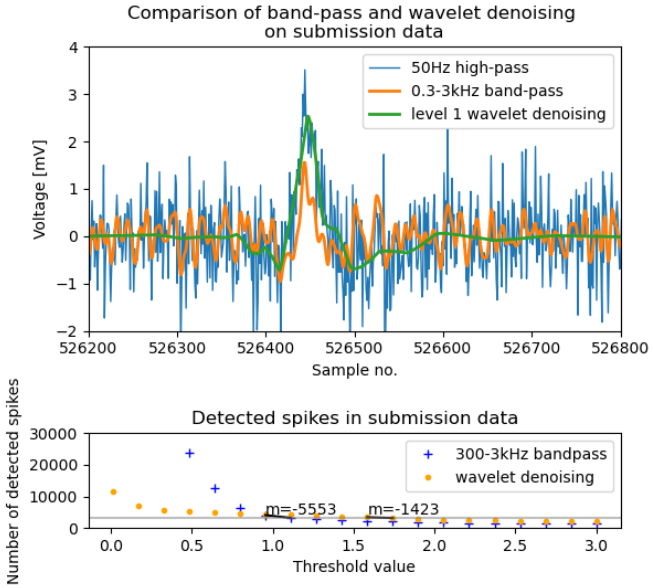


Figure 3: Comparison of a 0.3-3kHz bandpass filter to a first level wavelet denoising filter using a sym4 wavelet. Top: time domain filtering showing better fidelity and noise rejection of the wavelet denoiser. Bottom: Almost 4x lower sensitivity of the wavelet denoiser.

3.2 Spike Detection

3.2.1 Peak Detection by Thresholding

Spike detection is performed using peak detection above a certain calculated threshold. The equation to find the threshold value th, shown in equation 5, was taken from [1], and is found by using an estimate of the variance in the filtered signal based on a scaled value k of the median of the absolute value of the filtered time series X. The value of k is generally between 3 and 5 [1].

$$th = k * \frac{\text{median}(|X|)}{0.6745} \tag{5}$$

A minimum spike distance of 20 samples was arbitrarily selected to ensure a spike is not counted multiple times from small signal fluctuations about a peak, but this parameter that can later be optimised.

3.2.2 Clip Creation

Following peak detection, clips were generated using a number samples of a multiple of 2 centred around each detected peak because of the binary requirements of the discrete wavelet transform (or DFT, etc.). A 64 sample size clip is large enough to capture the bulk of the spike, but likely misses some of the refractory period information. Nonetheless, the 64 sample clip size was selected over 128 samples to avoid overlapping signals sharing too much signal, which would likely make them more difficult to distinguish. The window size can also later be optimised.

3.3 Classifier

3.3.1 Generating Training Clips

First, training clips need to be generated from the labeled time series which are also 64 samples long and centred around the spike peak. Each spike index in the labeled data was assigned the next peak above the detection threshold as its respective peak. Additionally, clips were assigned a clip type based on the smallest distance between its own and both neighbouring spike indices ds. Clip types are arbitrarily defined as merged ($ds < 20$), connected ($20 \leq ds < 100$), and isolated ($ds \geq 100$). Although the clips will not be treated differently during classification, this information will aid the improvement of subsequent classifiers by allowing the performance of isolated, connected, and merged spikes to be assessed separately. Also, the mean spike t_s to peak time t_p of all five neuron categories were computed ($\bar{t}_k$), and will be used for computing the spike time of classified clips according to their peak times and neuron category.

Spike vs. Peak Time

Spike time is defined as the start of an AP, just like in the labeled training data indices.
Peak time is defined as the time of the peak of an AP.
N.B. In practice, index values are used instead of time

One encountered problem was that some labeled spikes had no distinguishable peak, as shown in figure 4, which shows some undetected peaks with their respective labeled spike time. These spikes were therefore assigned a peak time $t_p = t_s + \bar{t}_k$ using $\bar{t}_k$ of their labeled neuron type. The training clip generation yielded 64% isolated, 28% connected, and 8% merged clips. Also, to validate the generated labeled clips, all clips had their spike times estimated $\hat{t}_s = t_p - \bar{t}_k$ and compared to their labeled spike times to ensure that they were within 10 samples of the true labeled spike times (i.e. $t_s - 10 < \hat{t}_s < t_s + 10$). Only 5 out of 3343 (0.25%) of labeled clips were found to be erroneous and eliminated using this criterion.

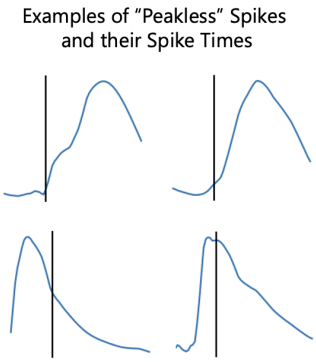


Figure 4: Some examples where a spike could not be associated to its own peak, and thus lead to faulty labeled clip creation

3.3.2 Feature Extraction

The discrete wavelet transform (dwt) was used as a new basis for each clip. Because useful information is contained in both the time and frequency domain, it is assumed that using either the time series or a DFT basis alone to represent the data will be sub-optimal. Although this can be accomplished using a spectrogram, a wavelet transform, in a sense, optimises the information content in the time and frequency domain, and thus will likely serve as a superior basis. Additionally, dwt coefficients will likely serve as a better basis for classifying connected spikes. This is because both time and frequency content are needed to correctly classify the central clip by rejecting frequency content of spikes on the edges of the clip. Because a purely frequency domain basis has no time information, it will likely be a poorer basis for classifying connected spikes (although windowing would partly mitigate this). Also, dwt coefficients have already been used in spike sorting algorithms with success [3]. Implementation was performed using a dwt from the `PyWavelets` package using the 'sym4' wavelet due to its similarity to the AP waveform (this can later be optimised). The decomposition level was computed using the in-built `dwt_max_level` function which depends on the input data length and window size. A Principal component analysis (PCA) was then performed for dimensionality reduction, with a number of components selected explained in section 3.3.4.

3.3.3 SVM Classification

Support vector machines (SVMs) classification works by separating classifications directly in feature space according to a kernel type (e.g. linear or polynomial). If clusters are relatively well seperated in the feature space, this approach is likely more efficient and accurate compared to more "brute force" methods like artificial neural networks (ANNs) which start training from a randomised state and rely on large volumes of training data. Also, ANNs can arbitrarily fit any function (with enough size) and therefore run the risk of over fitting the data, in particular with small data sets (like the training data set). This is particularly relevant because the spike sorter will be trained to classify in-vivo recordings which will likely have a greater variability in spike waveforms compared to the low-noise training data. Because SVMs are based on optimal separation of categories in feature space, it will likely be less prone to over-fitting and therefore generalise better to new data. The `svm.SVC` class from the `scikit_learn` package was used using the default parameters (radial basis function kernel, or rbf), which can later be optimised.

3.3.4 K-Fold Cross-Validation

Cross-validation was performed on the SVM using 80% of the labeled clips as training data and 20% as test data. This was performed 20 times with a shuffled deck of clips using 2-6 principal components (PCs). The total accuracy of the test data as well as the accuracy by clip type for all 20 samples is shown in a whisker plot in figure 5. For the isolated spikes, classification is close to perfect even for 2 PCs, which shows they form highly distinct clusters. Connected clips also show good classification accuracy starting at 5 PCs of 97%, showing that despite two spikes being within 100 samples of each other, they are still largely distinguishable in the wavelet basis. Merged clips show significantly lower accuracy at a little over 50%. Because the clips of two merged spikes are likely nearly identical, both clips are probably labeled as the same neuron type. The fact that merged clips still have over 50% accuracy indicates that at least one of the spikes is correctly labeled, which is surprisingly good given their merged morphology. The total accuracy still remains high at around 95% because merged clips only make around 8% of total clips and therefore only induce around 4% error. Five PCs were selected as classification features since more did not increase classification performance.

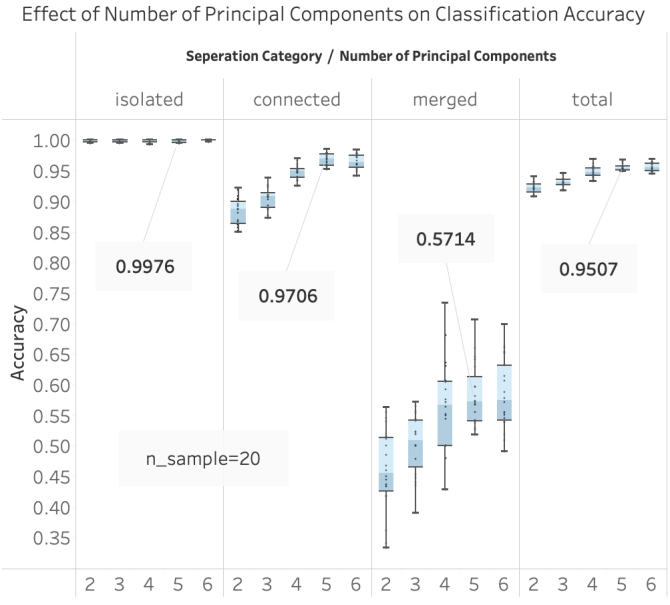
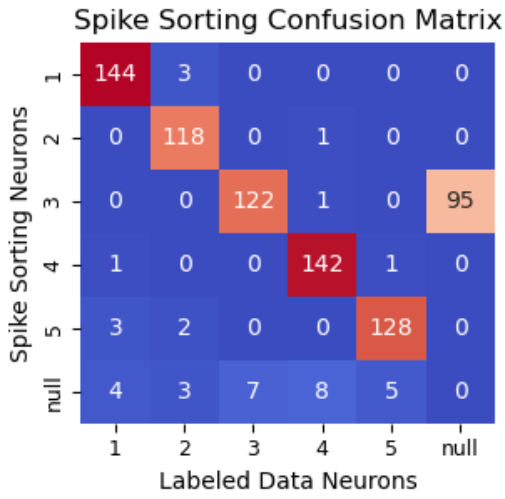


Figure 5: Total accuracy of the SVM from 20 randomised cross-validations (right), as well as the accuracies by clip type, left to right: isolated, connect, and merged clips

3.4 Spike Sorting Performance

The classifier was trained on the first 80% of the time series, and the entire spike sorting algorithm was applied to the last 20% for validation. In this case, k-fold cross validation is not as simple since the time series being used should be continuous, and therefore cannot be cut

and joined together at random. Therefore, simple cross-validation on the final 20% of the training data is used. The detected, then classified spikes were compared to the labeled data to form a confusion matrix from which performance metrics were extracted, shown in figure 6. Total performance was around 83%, with most of the error occurring due to misclassified false positives (70% of induced error). Also, notice that the error summed from misclassification and false negatives are approximately equal to the classifier error (5%) found in 3.3.4. This is likely because half of the merged spikes were simply not detected at the spike detection step and thus are false negatives in the spike sorter and not categorised as misclassifications.



Metric	Value	Induced Error
Total Accuracy	0.83%	N/A
Correctly Classified	654	N/A
False Positives	95	**12%**
False Negatives	27	3.5%
Misclassified	12	1.5%

Figure 6: Confusion matrix between the spike sorting algorithm and the labeled data in addition to its tabulated performance metrics. False negatives are the null row and false positives are the null column

4 An Improved Classifier

4.1 Analysis of Spike Sorter Error

4.1.1 Reducing False Positives

False positives created by the spike detector that reach the classifier are necessarily mislabeled with the current architecture. Interestingly, they are all mislabeled as neuron type 3, likely due to the false positives having a small spikes size which are most similar to neuron type 3. A simple way to reduce the number of false positives is to increase the spike detection threshold. Table 1 shows the effect of increasing the detection threshold by sweeping k in the threshold equation 5. As expected, increasing the threshold reduces the number of false positives. However, with k values above 7, false negatives start to increase. A naive approach would therefore be to select a threshold with $k = 7$. However this would be unwise since although it would be the optimal threshold for the training data, there is no guarantee that it would be optimal in the submission data, particularly since it contains more noise. A potentially better strategy would be to use a lower threshold and design a classifier to reject false positives. Also, this approach seems wise in general since it must be assumed that a great deal of false positives can occur for a given neural recording, which would drastically reduce the spike sorter's overall accuracy. In this light, false positive rejection is likely the most important improvement to be made to the current spike sorter.

k value	False Positives	False Negatives
1	328	27
3	175	27
5	32	27
7	5	28
10	3	51

Table 1: Effect of increasing the threshold value (k coefficient in equation 5) on the number of false positives and false negatives

4.1.2 Reducing False Negatives

As mentioned in 3.4, the false negatives are largely due to one of the merged spikes not being detected and are therefore counted as a false negative. However, even if they were to be detected separately, they only achieve around 50% accuracy in the current classifier (see figure 5). Visual inspection of the false negative clips (n=27) from cross-validation indicated three main types of false negatives, shown in figure 7. *Superposed spikes* have a normal morphology, but larger amplitudes which are due to two highly coincident APs. *Silent spikes*, usually from neuron 3 which has the smallest amplitude, show no noticeable trace on the time series data. Finally, *missed peaks* are spikes which are undetected due to their peaks being closer to each other than the peak detector minimum separation of 20 samples.

Missed Peaks

The number of missed spikes could be reduced by reducing the minimum detection distance between peaks. However, a quick test of changing the peak minimum peak distance to 10 resulted in far more false positives. Because the undetected peaks are a small portion of the dataset (<1%), the downside of reducing the peak distance has a higher chance of outweighing the upside. However, the minimum peak distance can nonetheless be optimised in the final section to identify if some small improvement can be gained by modifying its value.

Silent Spikes

Silent spikes are challenging, first to identify, and then to classify. To detect them in the first place, the usual thresholding peak detector would have to be changed since threshold peak detection will not work. A possibility would be to use a sliding window approach, running the classifier on every window and identifying spikes by their extracted features, not by their peaks. It is, however, doubtful that there is enough information content in hidden spikes to even classify them in this manner.

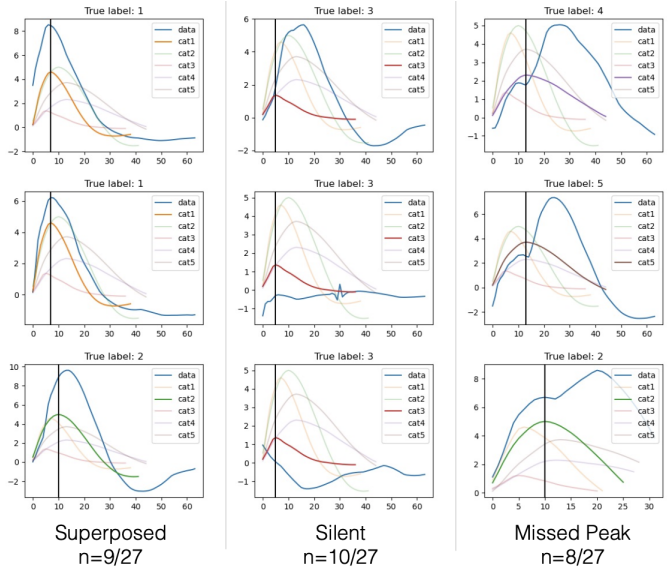


Figure 7: Three types of categories of false negatives were found: superposed, silent, and missed spikes. They all occur in roughly equal frequencies

Additionally, classification of this kind on low information content windows would require a large training set, which is lacking for hidden spikes, with only around 50 of them in the entire training data set. These false negatives are therefore taken as an acceptable error.

Superposed

Of all the false negative categories, superposed spikes have the best chance of being classified due to their often substantially higher amplitude. It is therefore conceivable that a classifier could distinguish an isolated AP from a merged AP. However, each superimposed pair of spikes could be any number of the 10 possible combinations of the 5 neuron types. This, combined with the low number of superposed training clips (approx. 45), would mean that all categories may not be represented, and even if they were, would only have around 2-3 samples per category. Training a classifier on this small a dataset would likely lead to poor performance with a high chance of overfitting. For this reason, only optimisation of the minimum detected peak distance (to fix *missed peaks*) will be implemented to reduce false negatives.

4.1.3 Improving Classification

The misclassification of true spikes accounts for a small portion of overall error (1.5% of the total 17% error). Although this could be further improved, it is likely that accuracies approaching 100% would simply be over-fitting the training data and reduce generaliseability to the in-vivo data. Because the classifier performs very well already, improvements should focus more on the robustness of the overall spike sorting algorithm rather than optimising the classifier.

4.2 New Proposed Classifier

A new classifier is therefore proposed which adds a new false positive category to the SVM classifier, shown in figure 8. In addition to the usual "true" labeled spikes, false positive clips, labeled "0" are used to train the classifier. These are generated by applying the spike detector to the training data time series with a low detection threshold ($\tilde{th} = th/4$). From the detected clips, all associated labeled clips are subtracted and the remaining set of clips are considered false positives.

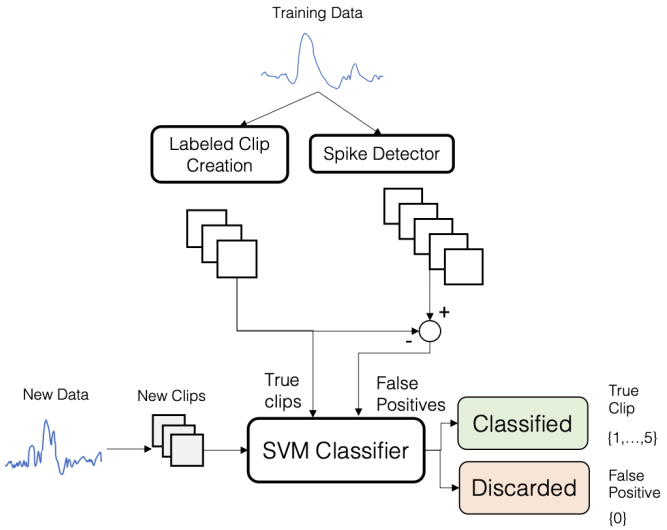


Figure 8: The SVM classifier is trained on the labeled spikes in addition to false positives found by the spike detector. In this way the classifier can distinguish false positives from true action potentials.

4.3 Results of New Classifier

4.3.1 Cross-Validation on Training Data

The new classifier was tested using the same cross-validation as the previous spike sorter. Both the old and new spike sorters were run for a low threshold of $k = 1$ value as an extreme case to demonstrate the false positive rejection capabilities of the new classifier, shown in table 2. The new classifier was able to reject 312 out of 314 false positives and only increase the misclassifications by 1. In practice, such a low threshold would not be used and therefore the vast difference in accuracy is not a fair representation of the accuracy improvement. However, it does demonstrate that the new classifier works even for very low thresholds, which makes the spike sorted far less dependant on finding a perfect threshold for a given set of data. This is particularly important since it will be applied to unknown data with more noise. With the new classifier, a low threshold can be used with confidence that it will not induce large errors from a high number of false positives.

Parameter	New	Old
False Positives	2	314
False Negatives	27	27
Misclassified	13	12
Accuracy	94%	65%

Table 2: Comparison of the new false positive rejection spike sorter to the old spike sorter using a low detection threshold ($k = 1$). The new classifier can reject most false positives leading to a higher accuracy

4.3.2 New Classifier on Submission Data

Both the new and old classifier were applied to the submission data with varying threshold coefficients $k = [1, 2, 3, 4, 5, 6]$ and their respective number of classified spikes are shown in table 3. The new classifier has a far more consistent number of detected spikes compared to the old classifier. In addition, the number of detected spikes is quite close to the number of spikes in the training data (3343). For both these reasons, there can be greater confidence placed in the new classifier due to its ability to reject false positives, even in the submission data.

k	No. Labeled (New)	No. Labeled (Old)
1	3416	6896
2	3306	5560
3	3199	5109
4	3109	4842
5	3017	4504
6	2934	4162

Table 3: Comparison of the number of classified spikes in the submission data with the new and old classifiers. The new classifier is able to reject many false positives and is therefore less sensitive to changes in threshold value

5 Optimisation

A genetic algorithm (GA) was selected to optimise the spike sorter hyper-parameters. The GA method was selected due to it being applicable to non-numerical parameters, and has already been implemented by the author and thus is easily adaptable to this specific problem. Because the entire process of instantiating the spike sorter class, training, and then evaluation takes approximately 30s, around 5 possible values were set for each parameter to ensure the GA could complete in a reasonable time.

5.1 Optimisation Parameters

The parameter values were set to cover a range which were estimated to cover the optimal range of values, found from previous experimentation. The parameters and their respective values are:

1. **High-pass frequency**: The high-pass frequency value of the initial data filtering. Its possible values are 20, 50, 100, 150, and 200Hz
2. **Denoising wavelet and dwt wavelet**: Both wavelets used for denoising and the dwt can take any of the following wavelets: haar, db4, bior1.5, coif2, rbio3.3, and sym4. The wavelets were selected by taking the wavelet from each wavelet family that appeared morphologically closest to an AP. The Haar wavelet was also added since it is the "conventional" wavelet.
3. **Denoising Threshold Coefficient**: This is the coefficient to scale the thresholding value of the wavelet coefficient during wavelet denoising. It can have values of 0.4, 0.6, 0.8, 1, 1.2. The default is 1, but lower values seemed to give better results, and therefore the range is more extended <1.
4. **Threshold** k: The coefficient to find the peak detection coefficient and can take values of 3, 4, 5, or 6. These were selected since this value is generally between 3-5 [1].
5. **Clip Size**: This could have one of 3 values that are powers of 2: 32, 64, 128.
6. **Minimum Peak Distance**: This value is used by the peak detector to set the minimum sample count between two adjacent peaks. It can take values of 15, 20, 25, and 30.
7. **SVM Kernel**: This is the kernel type used by the SVM classifier and can take a value of linear, poly, rbf (radial basis function), or sigmoid.

5.2 Fitness Function

The fitness function is a combination of the total accuracy on the training data and stability on the submission data. This is because optimising for either alone could lead to a degradation of the other, which would lead to poorer performance. For example, if only accuracy on the training data were optimised, the spike sorter could overfit the training data. On the other hand, if only stability were optimised, the accuracy of the spike sorter could be compromised since stability does not necessarily imply accuracy. Because both accuracy and stability are measured from 0 to 1, with 1 being the best score, they are simply combined using a mean squared equation, shown in equation 6, where F is the fitness given an accuracy a and stability s. Both terms are squared to penalise large values more.

$$F(a, s) = \frac{a^2 + s^2}{2},\ F \in [0, 1] \tag{6}$$

As before, total accuracy is calculated using a confusion matrix on cross-validation data and stability is computed using the self-blurring technique (equation 4) with $\gamma = 1.5$. This value of γ was selected since it yielded a stability of around 0.8 for most randomly generated hyper-parameters and was therefore enough to give room for improvement for stability.

5.3 GA Parameters

Because the fitness function was quite slow, the GA parameters were set to balance the total simulation time with adequate solution space exploration.

5.3.1 Population Size and Survival Parameters

The population size was set to 20. If the population size were to be larger, each generation would take longer to generate, and there would therefore be less time to run multiple generation, defeating the purpose of the genetic algorithm. Due to the small population size, a survival ratio of 0.8 was selected to ensure that genetic diversity was preserved. If too many individuals were culled between generations, the population would soon loose much of its genetic diversity. Also, a small "random select" value of 0.01 was selected to occasionally enable culled individuals to reproduce.

5.3.2 Elitism

One elite per generation was allowed to pass to the next generation directly. This was to ensure that the best solution would always pass to the next generation and thus hopefully improve the consistency of improvement between generations

5.3.3 Mutation Rate

A mutation rate of 0.5 was selected, which means that approximately half of children will have one gene (of the 8 parameters) randomly modified. This high mutation rate was selected because it will allow a smaller population to better explore the solution space given enough generations.

5.3.4 Termination Condition

The GA was run until the solution stagnates for over 10 generations and was satisfactory (i.e. gave results that were at least as good as the non-optimised solution), or had run for 100 generations.

5.4 GA Results

After 50 generations, the GA stagnated for 10 generations on a solution, whose fitness is compared to the unoptimised spike sorter in table 4. The GA was able to improve the stability of the solution by 2.5%, and only marginally increase the accuracy by 0.5%. Although the GA used is not guaranteed to have found the optimal solution, this shows that the original solution used was generally quite good, in particular in terms of accuracy. A explained in section 4, around 4-5% error is inevitable using the current classifier architecture and thus the accuracy is caped at around 95-96%.

Spike Sorter	Accuracy	Stability	Fitness
Un-Optimsied	0.939	0.960	0.902
Optimised	0.944	0.985	0.930
Abs. Improvement	+0.5%	+2.5%	+2.8%

Table 4: Improvement of the solution found with the GA in accuracy and self-blurring stability. The GA was only able to noticeably improve the stability

The new solution changed 3 parameters from the unoptimised solution:

1. **Threshold** k: its value was changed from 3 to 4. This likely leads to slightly fewer false positives and therefore better performance. Although false positive rejection is implemented, some may still be misclassified. Therefore minimising their occurrence at the spike detector is still potentially beneficial overall.

2. **Clip Size**: was changed from 64 to 32, which shows that capturing the entirety of the AP does not seem to be necessary for good classification. Most of the information must therefore be concentrated around the peaks. An additional benefit of a smaller clip size could also be its greater spacial selectivity and therefore better ability to classify connected spikes.

3. **DWT Mother Wavelet**: was changed form Sym4 to Haar. This was unexpected since the Haar wavelet is quite a crude wavelet, and differs from the AP morphology far more than the sym4 wavelet. However, the goal of the dwt is not to maintain the morphology of the AP (as would be with wavelet denoising for instance), but rather to extract meaningful information. Because most of the improvement in the optimised solution is in stability, it could be hypothesised that the dwt using the Haar wavelet is able to generate a reduced feature set that better distinguishes the neuron categories compared to the sym4 wavelet. This is because self-blurring stability, in effect, blurs the neuron-category clusters in the feature space, and therefore better stability indicates better separation of clusters in the feature space.

5.5 Fine Tuning of Solution

A plot of all classified waveforms on the submission data by label are shown in figure 9. In addition, the mean waveform of the classified submission spikes (yellow) and mean waveform from the training data (red) are superimposed for comparison. There are some obvious false positives which are mislabeled, in particular for labels 3,4, and 5.

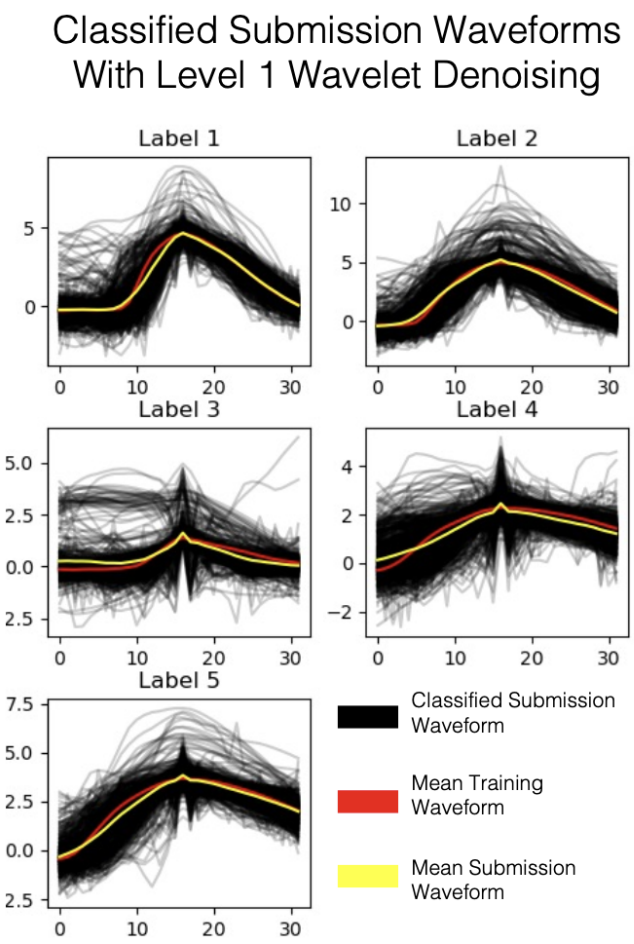


Figure 9: All classified waveforms in the submission data by label. Notice the spurious noise spikes which noticeably occur in many of the classified clips, and can be assumed to be false positives

These spurious spikes were identified to be caused by the wavelet denoising filter, and shows that filter char-

acteristics are important to consider for the submission data as it can potentially behave differently. By increasing the wavelet denoising decomposition level from 1 to 4, the classified waveforms from the submission data showed fewer spurious noise spikes, shown in figure 10. The APs of label 3, being the smallest of all in amplitude, still contains many spurious spikes. However, the overall mean waveforms for each label in the submission data are much closer to that in the training data, thus giving greater confidence.

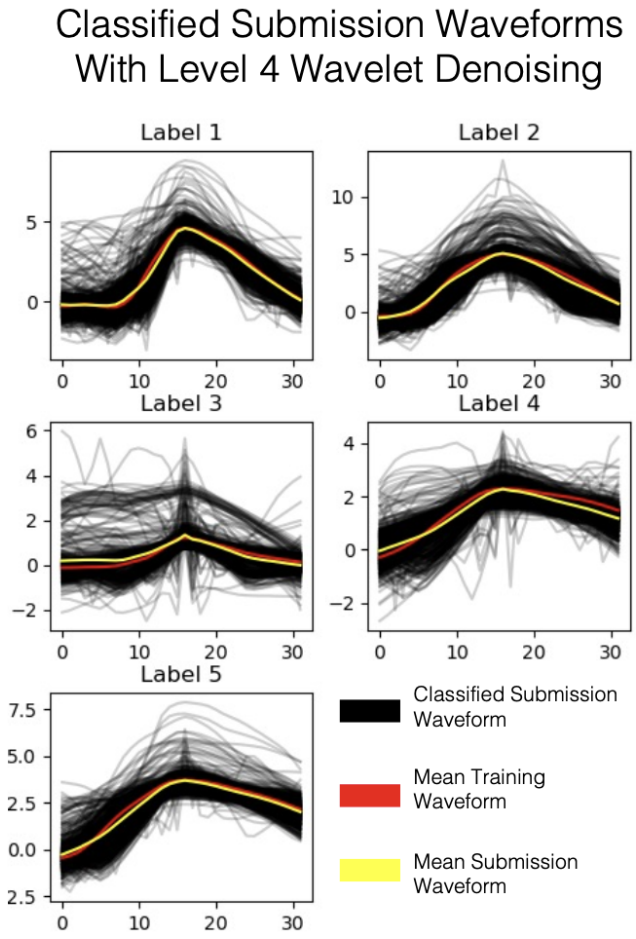


Figure 10: Increasing the wavelet denoising decomposition level from 1 to 4 lead to fewer spurious noise spikes being detected and misclassified

6 Confidence and Conclusions

6.1 Estimation of Accuracy on Submission Data

The confidence in the final solution applied to the submission data can be found by estimating the occurrence of error due to (1) merged clips, (2) false negatives/positives, and (3) misclassifications.

6.1.1 Error from Merged Clips

As already justified in section 4.1.2, it was decided not to attempt to improve the classification of merged clips, who currently have approximately 50% classification accuracy. Because merged clips account for approximately 8% of total clips, this will automatically lead to 4% error in the submission data, assuming roughly equal statistical distribution of APs between the training and submission data sets.

6.1.2 Error from False Positives

As shown by visual inspection in figure 10, some false positives still remain despite tuning of the wavelet denoising filter and false positive rejection in the SVM classifier. This will add additional error, which is difficult to quantify, but is liberally estimated to be between 5-15%.

6.1.3 Error from Misclassifications

Misclassifications of true clips was shown to be consistently low (<1.5%) in the training data. It is assumed that similar misclassifications in the submission data will occur slightly more, yet still remain relatively small. A maximum of 3% induced error (2x) is therefore estimated for misclassifications of true clips.

6.1.4 Final Accuracy Estimate

Combining the estimates above, the overall accuracy of the spike sorting algorithm on the submission data is estimated to be 80% with a range between 70% and 90%.

6.2 Conclusions

The classification of *isolated* spikes was shown to be trivial, since it can be achieved with a near 100% accuracy using only 2 dwt PCs and a SVM classifier (see figure 5). The neuron types therefore have highly differentiated characteristics in the dwt feature space. The challenge in robust spike detection thus principally lies in two aspects: (1) spike detection and (2) merged clip classification.

6.2.1 Improving Filtering

Robust spike detection first requires filtering to remove noise while simultaneously maintaining all required information for classification. Although the wavelet denoising filter showed far better performance compared to the linear band-pass filter in the training data, it still lead to some spurious spikes in the submission data. Further tuning of the filter and/or the exploration of new filtering techniques tuned to the in-vivo data should therefore be explored. It is estimated that this would lead to the biggest improvement in the current spike sorting algorithm.

6.2.2 Improving Merged Spike Classification

The undetected or misclassified merged spikes also posed a challenge due to their low count in the training data. More training data would be required, perhaps by using multiple simulations, to create a classifier capable of detecting (1) if a spike is merged and if so, (2) which two (or more) neurons compose it.

References

[1] H. G. Rey, C. Pedreira, and R. Q. Quiroga, "Past, present and future of spike sorting techniques," *Brain Research Bulletin*, vol. 119, pp. 106–117, 10 2015.

[2] A. H. Barnett, J. F. Magland, and L. F. Greengard, "Validation of neural spike sorting algorithms without ground-truth information," *Journal of Neuroscience Methods*, vol. 264, pp. 65–77, 8 2015.

[3] R. Bestel, A. W. Daus, and C. Thielemann, "A novel automated spike sorting algorithm with adaptable feature extraction," *Journal of Neuroscience Methods*, vol. 211, pp. 168–178, 10 2012.