# Shallow Neural Networks Classify Extracellular Spike Waveforms With Greater Accuracy Than Traditional Methods

Peter Hebden

22 May 2019

## Contents

# Abbreviations

- ANN: artifical neural network
- CMOS: complementary metal-oxide semiconductor
- EC: extracellular
- GT: ground truth
- IC: intracellular
- k-NN: k-nearest neighbors
- KS: Kolmogorov-Smirnov (test)
- MAD: median absolute deviation
- NN: neural network
- PCA: principal components analysis
- SNR: signal to noise ratio

**Abstract**

Classic spike sorting pipelines cluster spike waveforms into groups that represent their source neurons, but they suffer from inaccuracy and incomplete automation. Artificial neural networks (ANNs) are class of machine learning methods that are remarkably flexible and well suited to learning from large data sets. This report presents empirical evidence that neural network enhanced pipelines can provide higher classification accuracy than pipelines that rely exclusively traditional feature extraction and clustering methods.

# 1 Introduction

Electrophysiological methods have traditionally used a small numbers of electrodes to record from a small number of neurons. They are the gold standard in neuroscience because they can record single neuron activity at almost any location with high temporal and spatial resolution. However, microelectrode arrays based on CMOS technology, such as Neuropixels, provide ∼1000 recording sites. They can deliver low-noise recordings from hundreds of neurons [65]. Large scale electrophysiology requires new computational methods, and associating spikes with single neurons is the main challenge [65].

Historically, due to data-handling and computational considerations, simple online methods detected the presence of action potentials, and only a few milliseconds of data per action potential was saved for future analysis. With the advent of CMOS probes and ever cheaper storage, it is now becoming standard practice to record the entire the wide-band (0Hz - 30kHz) signal originating from each channel for off-line analysis. This massive increase in data requires spike detection and classification methods that are automated and more complex than traditional methods. Better methods would not only increase the quantity and quality of information gained from recordings, they would also reduce the number of experimental animals used and the time taken to collect data.

Artificial neural networks (ANNs) are class of highly parameterized machine learning methods that have attracted more and more attention in recent years. They are flexible and can accurately model relatively small irregularities in functions. Even a network with a single hidden layer can model any continuous function if the hidden layer contains enough nodes (artificial neurons) [28].

The aim of this project is to investigate the utility of ANNs for spike detection and classification in the context of spike sorting. The classic spike sorting pipeline proceeds in four basic stages: data filtering, threshold spike detection, feature extraction, and clustering spike waveforms into groups that represent their source neurons. My hypothesis is that a pipeline that includes a neural network will classify spikes with greater accuracy than pipelines that rely exclusively on traditional methods.

To test the impact of ANNs on performance, I used data sets that included ground truth in the form of paired intracellular and extracellular recordings, i.e. the hippocampus data set hc-1 from the Buzsáki Lab [29, 31, 32]. As shown in the results section of this report, pipelines with ANNs significantly outperformed the competition, especially after the spike waveform data was augmented.

# 2 Background

In contrast to intracellular recordings, extracellular recordings do not give a direct access to the neuronal activity. They usually include contributions from multiple neurons and noise, so they must be "spike sorted" before being converted into spike trains [17, 29, 45, 57]. Spike sorting is the process of assigning experimentally recorded voltage waveforms to single neurons when the number and identity of the spiking neurons is not known [11, 34].

Figure 1 shows an overview of a spike sorting pipeline from start to finish that incorporates the use of templates (exemplary waveforms). Ideally, all spike waveforms from the same the source neuron will end up in the same cluster (classification).

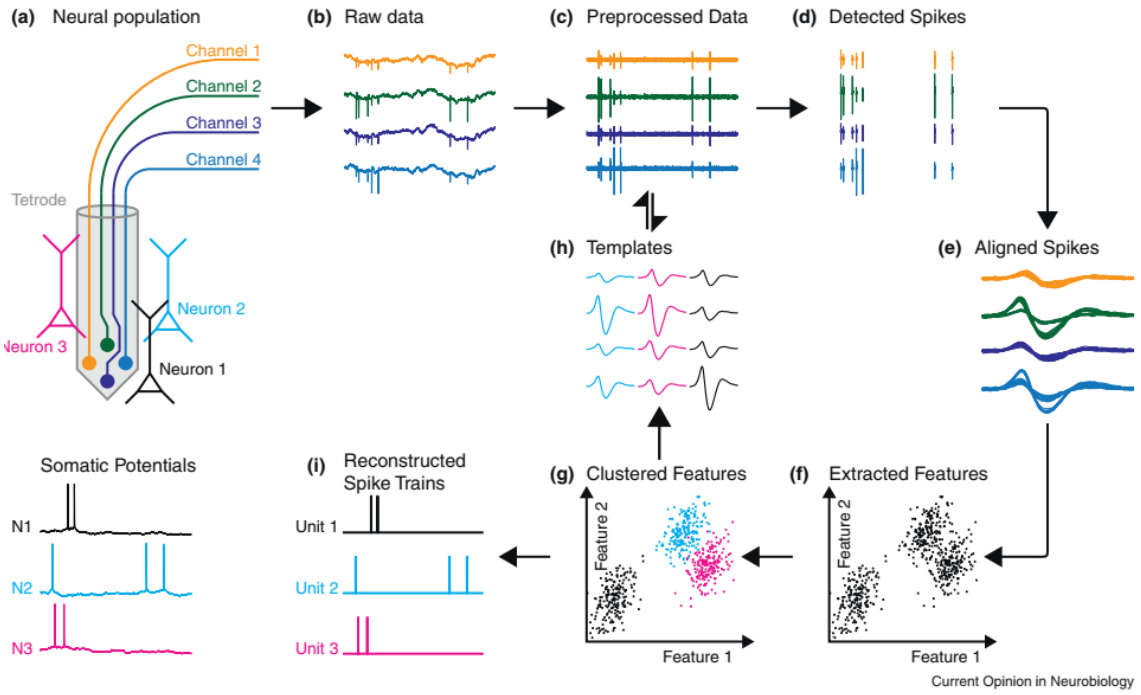



Figure 1: Overview of spike-sorting process: (a,b) record changes in the extracellular electrical potential caused by nearby spikes, (c) filter, (d) detect, (e) peak align waveforms, (f) extract features, (g) cluster, (h) use templates, (i) construct spike trains. Image: [17].

## 2.1 Introduction

Extracellular recordings reflect the sum of all transmembrane currents for all neurons in the vicinity of the electrode tip weighted by the distances [37]. In Figure 2, the concentric circles indicate various distance estimates; extracellular electrodes can record spikes from single neurons less that ∼50 to 100 $\mu$m away from the tip of the electrode in the inner circle, neurons further away can be detected but not separated, while neurons in the outer area, more than ∼150 $\mu$m away, contribute to background noise and their spikes cannot be detected [32, 57, 58].

Recording from multiple neurons may be more useful if spike trains can be constructed by associating each spike with its source neuron. Neuron spike times transmit most of the information [66]. Therefore spike trains are the main carrier of information in mammalian neural systems [17], and may even be the sole information carrier between brain regions [2, 3].

### 2.1.1 Spike Detection

The first steps are critical for accurate spike sorting. While bandpass filtering makes spike detection easier and is relatively straightforward, spike detection itself requires picking a good threshold. If the threshold is too high, spike sorting is likely to result in too many false negatives; if too low, then too many false positives.

Consequently, some sorting algorithms include an extra transformation to the filtered data before spike detection. For example, apply an energy operator [7, 62], the continuous wavelet transform [51], or fuzzy and probability theories [5].

More recently, Rossant *et al.* [60] used the program SpikeDetekt for spike detection and feature extraction. First they highpass filtered the raw data to remove the slow local field potential signal. Then spikes were detected using a double-threshold

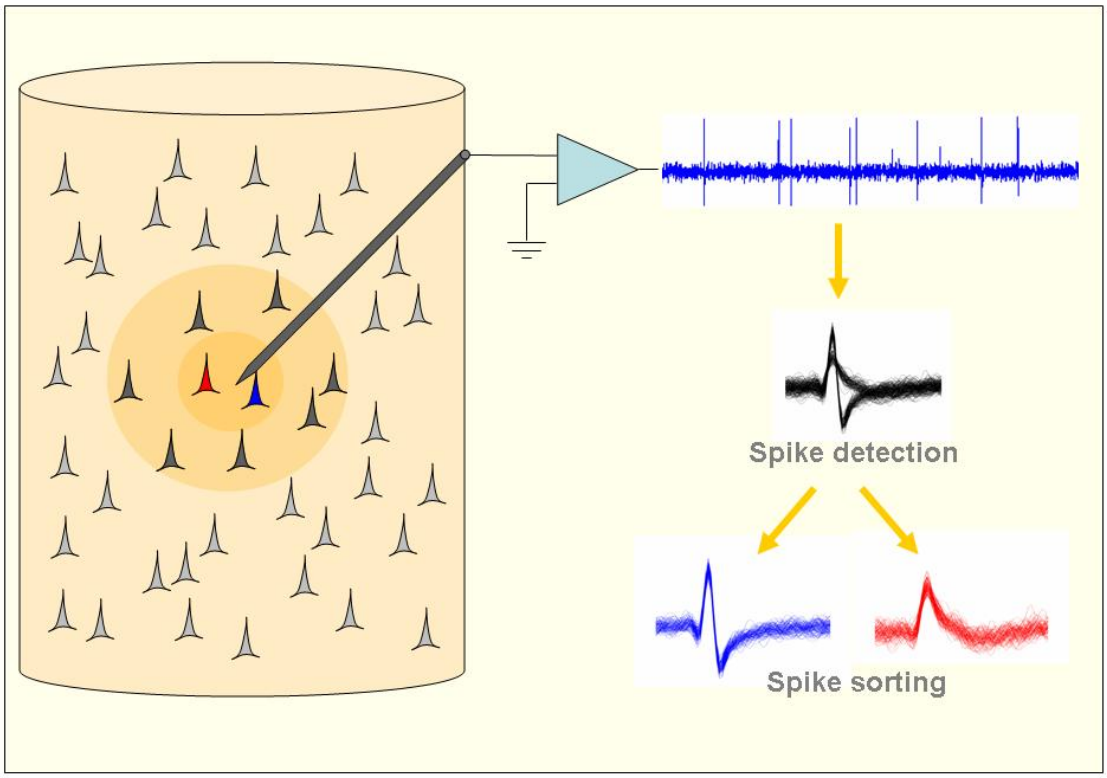



Figure 2: For neurons close to the electrode tip in the inner circle, ∼50 to 100 microns, the signal to noise ratio (SNR) is good enough to distinguish the activity of each single unit. For neurons in the outer circle, up to about 150 microns away, spikes can be detected but their shapes cannot be distinguished. Spikes from neurons further away in the outer area contribute to background noise. Image: [57].

flood fill algorithm where spikes were detected as spatio-temporally connected components. Specifically, a spike is detected when the filtered signal exceeds a weak threshold $\theta_w$ for every point and at least one point exceeds a strong threshold $\theta_s$ [60].

In addition, the noise distribution might be nonstationary, so it may be useful to compute the noise estimate (and therefore the threshold) with a sliding block of data (perhaps a few minutes long). Indeed, some data might have a high median value during the first half of the recording, for example, followed by a low median value during the second half.

After detection, each putative spike waveform (∼2 to 3 ms long) is stored in a matrix, one row for each waveform, which can be fed into the sorting algorithm. The extracted spike waveforms may be peak aligned to improve the classification process.

### 2.1.2 Feature extraction and clustering

Waveform shapes can be important, e.g. where "wide" is characteristic of an excitatory neuron and "narrow" is characteristic of an inhibitory neuron [63]. An example of feature extraction and clustered waveforms is shown in Figure 3. However, in practice, such features do not work well for spike sorting [45, 56].

Most traditional spike sorting methods use wavelets [56] or principal components

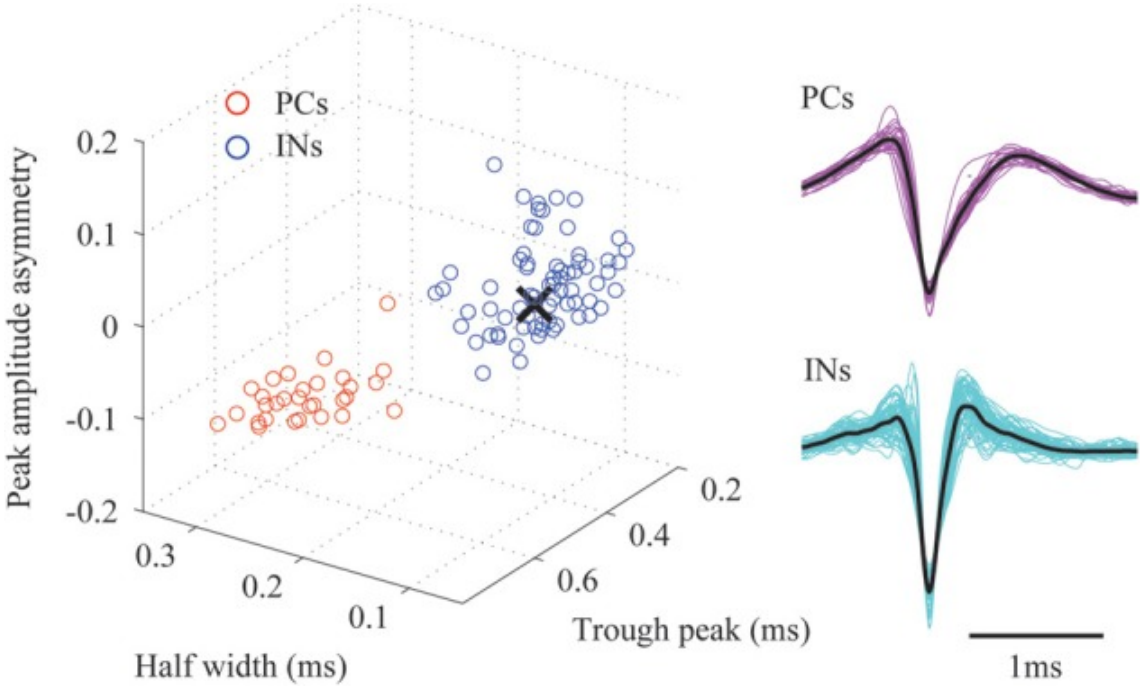



Figure 3: Excitatory pyramidal neurons and interneurons occupy different regions in feature space. Image: [59]

analysis (PCA) for feature extraction, and K-means for clustering [45]. Despite major progress in algorithms and software, spike sorting is still labor-intensive during the clustering and quality assessment phases [6].

## 2.2 Electrode Technology

The hardware for recording *in vivo* has shown accelerating improvement over the last 50 years or so, with recording sites per electrode shank increasing from 1 to about 1000 [65]. This flood of data requires better computational methods, i.e. completely automated methods.

### 2.2.1 Tetrodes

The hc-1 extracellular recordings analyzed in this report were collected in the late 1990s with wire tetrodes (4 channels per shank). It may be worth be noting that the recordings from the four extracellular channels was remarkably similar for all of the tetrode data files used in this report. In general, the distance between the four electrodes would have caused them to receive signals that were not virtually identical [Personal communication: Dr Caswell Barry, UCL]. In any event, one half of the hc-1 data set was recorded by tetrodes, the other half by six-site linear silicon probes [23].

### 2.2.2 Neuropixels Probes

Neuropixels are shown in Figure 4. They are based on the same complementary metal-oxide-semiconductor (CMOS) technology that is used for constructing silicon integrated circuits. This technology reduces wire width (to 130nm in the probes) and produces a probe that contains all the active circuits needed for amplification, digitization, and multiplexing [65].

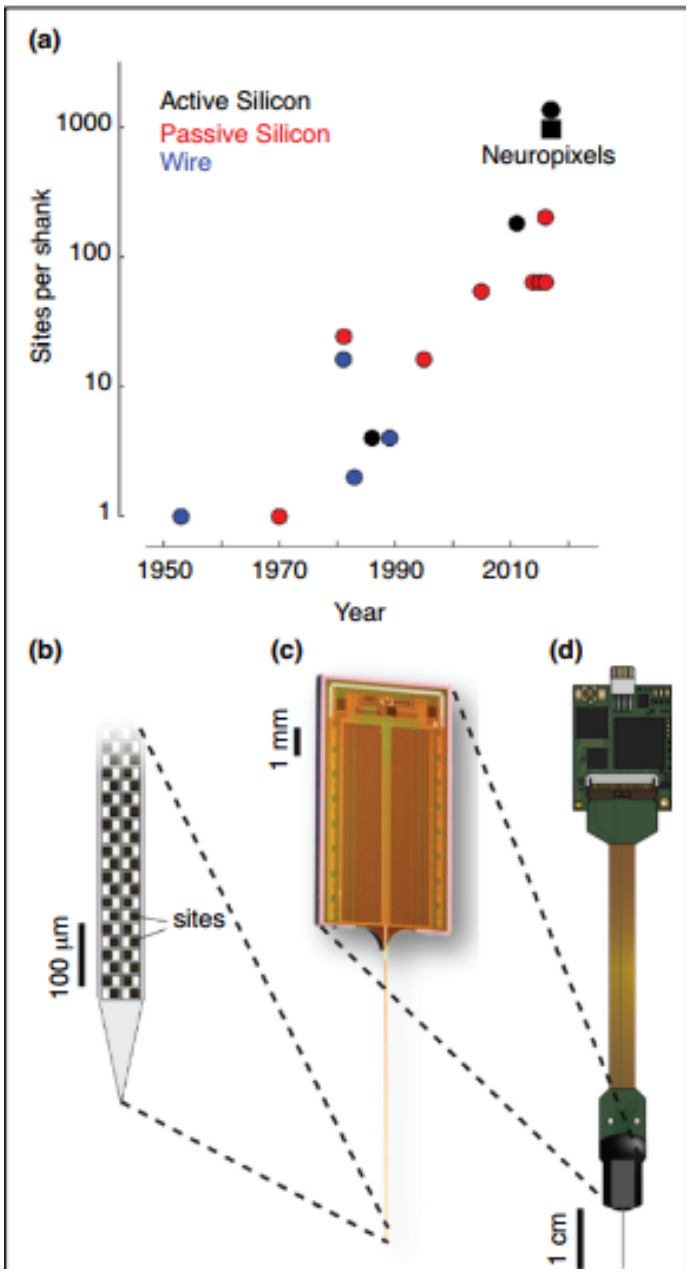



Figure 4: Electrode technology has advanced from simple wires to passive silicon to active silicon as exemplified by the Neuropixels probe: (a) Sites per shank. Blue dots for wire, red dots for passive silicon, black dots for active silicon. (b)–(d) Neuropixels probe. (b) Schematic of tip. (c) The printed CMOS element, including the shank, circuitry for amplification, multiplexing and digitization. (d) The packaged device with flex cable, and head stage for interfacing and further multiplexing. Image: [65].

Neuropixels sample signals densely. As a result, they can isolate single neuron activity better than previous technologies and, in some cases, detect spikes from individual neurons at over a dozen sites [41]. And they produce large amounts of data. The data acquisition rate (∼1 GB/min for 384 channels at 30 kHz) is on par with imaging [65], but this data needs to be processed to assign spikes to individual neurons [17]. In response, new spike sorting algorithms exploit GPUs and incorporate novel algorithmic steps [12, 14, 33, 42, 54, 67].

Kilosort [54] is a spike sorting algorithm that uses a novel approach to dimensionality reduction. It also uses GPU-acceleration to enable processing speeds on the scale of datasets from *in vivo* recordings with Neuropixels probes. Kilosort com-

bines spike detection with clustering and template matching to resolve overlapping spikes and was found to provide superior performance when tested against prior algorithms [54].

Nonetheless, no spike sorting algorithms are yet truly automatic, they require manual supervision to improve results. This is partly due to the movement of brain tissue relative to the probe, a problem known as electrode drift [65].

## 2.3 Ground truth neuronal data

The greatest barrier to developing automatic spike sorting algorithms is the difficulty of ground truth validation [65]. One way is through detailed simulations of spiking neurons, which produce synthetic voltage traces with predetermined spike times [27].

However, the best form of ground truth, is actual simultaneous recording of spikes with another method such as a juxtacellular electrode [65]. This provides a set of known spike times for the extraction of waveforms from adjacent extracellular recordings.

# 3 Methods

I used Matlab R2018b, [`https://uk.mathworks.com`], for all computation and plots. Matlab function names and parameters are in this font: `function_name`.

## 3.1 The hc-1 data set

I used the hc-1 paired-recording data set from the Buszáki Lab [29, 31, 32]. The extracelluar data was recorded from the hippocampus by tetrodes while an intracellular probe recorded "ground truth" directly from a nearby cell. As a result, spikes recorded simultaneously by the intracelluar (IC) and extracelluar (EC) probes were almost certainly from the same neuron. If the signal to noise ration (SNR) is high enough, the simultaneous extracelluar spikes should be detectable and, ultimately, grouped together in the same cluster by the clustering algorithm.

### 3.1.1 Tetrode recordings overview

Most of the tetrode recordings include 4 EC and 1 IC channel. If available, I used the IC channel to detect ground truth (GT) spike times for EC waveform extraction.

- Sampling rate: 10000.
- EC channels: 1 to 4 or 2 to 5.
- IC channel: 6 or 7.
- Channel 7: membrane potential, varies depending on the acquisition system.
- Channel 8: almost always injected current.

## 3.2 Data filtering and spike detection

Most algorithms first use bandpass filtering to pre-process the data, then amplitude thresholding to detect spikes [57]. Amplitude thresholds are based on an estimate of the standard deviation of the background noise. The standard deviation of the data and median absolute deviation of the data are two common methods [56]. Other spike detection methods are presented in [24, 50, 51].

### 3.2.1 Data was filtered with a bandpass filter

Electrodes record signals due to synaptic currents and action potentials [25]. Synaptic currents are slow and action potentials are fast, so action potentials can be separated with a high bandpass filter.

To make it easier to detect high frequency neural spikes the first step is to "flatten" the signal by filtering out the low frequency content. In this report, the bandpass was set to 300 to 3000Hz for spike detection, the default used Wave_Clus [56]. The best cut-off frequencies depend on the application.

Causal filters usually introduce phase distortions which may seriously change the spike shapes, but distortions can be greatly reduced by using a non-causal filter that removes noise without introducing phase lag [57]. The Signal Processing Toolbox in Matlab includes the function `filtfilt` for implementing a non-causal filter. The `filtfilt` function performs zero-phase digital filtering by processing the input data in both the forward and reverse directions [53]. Algorithm 1 shows Matlab code for implementing the bandpass filter input data X.

```
fs  = 10000;   % Sampling Frequency
fc1 = 300;     % First Cutoff Frequency
fc2 = 3000;    % Second Cutoff Frequency
order = 2;

fnorm = [fc1 fc2]/(fs/2);
[b1,a1] = butter(order, fnorm, 'bandpass');

filtered_data = filtfilt(b1, a1,  X);
```

**Algorithm 1:** Bandpass filter: this non-causal bandpass filter is based on a second order Butterworth filter, [ `https://uk.mathworks.com/help/signal/ref/butter.html` ].

Details are provided in the Appendix, Section 7.1.

### 3.2.2 Spikes were initially detected by amplitude threshold

Spikes were detected by two user defined parameters: a threshold amplitude (a multiple pf the median absolute deviation), and the required number of consecutive data points above the threshold. After detection, a number data points before and after the time of detection were peak aligned, extracted, time stamped, and stored in a matrix of vectors.

A threshold can be set manually, but the detection trade-off (false positives versus false negatives) depends on the signal to noise ratio of the recording. One possibility is an automatic threshold calculated as a multiple $K$ of the estimated

standard deviation $\sigma$ of background, i.e. threshold $= K \times \sigma$, where K is a constant typically between 3 and 5 [58] and noise is assumed to be normally distributed.

As proposed in [56], the median absolute deviation (MAD) provides a more robust estimate of the standard deviation of background noise [16, 56].

Details are provided in the Appendix, Section 7.2.

## 3.3 Feature Extraction

Manual feature extraction requires time consuming user input and often yields poor results [45]. Partially automated computational methods have been shown to generally outperform manual methods [29].

Features can be extracted automatically with wavelet decomposition [35, 36, 44], or with principal component analysis (PCA). [20, 21, 22],

For example, given 50 spikes, 21 data points each, both wavelet decomposition and PCA will generate 50 transforms, 21 coefficients each. The waveforms can be clustered using a small number of coefficients.

### 3.3.1 Wavelet decomposition

Wavelet decomposition generates vectors of coefficients from waveforms such that each coefficient characterizes the spike shapes at different scales and times [56]. The Matlab function `wavedec` can be parameterized to perform four-level multiresolution decomposition with Haar wavelets (a member of the Daubechies family of wavelets).

Haar wavelets are well suited to spike sorting. Their properties allow the discriminative features of the spikes to be expressed with a small number of wavelet coefficients and regardless of spike shape [56].

As shown in Algorithm 2, each waveform is a short vector of data points, e.g. about 2 ms of data. Wavelet decomposition is applied to each waveform to compute 21 coefficients per waveform, one wavelet coefficient for each data point.

```
waveforms(1..N);

for ii=1:N
    W(ii) = wavedec( waveforms(ii), 4, 'haar');
end
```

**Algorithm 2:** Pseudocode for wavelet decomposition of N waveforms.

Details are provided in the Appendix, Section 7.3.1.

### 3.3.2 The Kolmogorov-Smirnov test

The one-sample Kolmogorov-Smirnov test [40, 48] was used to determine whether the waveform features came from a population with a standard normal distribution. I used this test to select three wavelet coefficients that were not normally distributed and, hopefully, would best separate the different waveform classes. However, if the features are normally distributed, then PCA is likely to result in better classification performance.

### 3.3.3 Principal component analysis (PCA)

PCA can be used to extract features for clustering/classifying waveforms. The aim of principal component analysis (PCA) is to represent the d-dimensional data in a lower-dimensional space that best expresses the variation in a sum-squared error sense. For example, PCA might reduce 21 features to 3 principal components that account for over 90% of data variance. I used Matlab's `pca` function and the first three principal components. More details are provided in the Appendix, Section 7.3.4

## 3.4 Classification of Spike Waveforms

### 3.4.1 Neural Networks

Neural networks that operate on two or three layers of connected artificial neuron layers are known as shallow neural networks. Deep learning networks can have many layers, even hundreds. Both are machine learning techniques that learn directly from input data. To avoid confusion, artificial neurons in an ANN will be referred to as "nodes".

**3.4.1.1 The training function** In general, a training function will try to minimize the sum of squared errors between the correct output and predicted values by following the steepest descent in weight space. This is a sequence of steps, known as back-propagation, in which the weights are updated by working from each output node back to the input nodes.

The default training function for `patternnet` is `trainscg`; it uses the *scaled conjugate gradient* algorithm [49]. It can train any network as long as its weight, net input, and transfer functions have derivative functions.

Backpropagation is used to calculate derivatives of performance with respect to the weight and bias variables. However, whereas standard backpropagation always proceeds down the steepest gradient of the error function, a conjugate gradient method (CGM) will go in a direction that is conjugate to the directions of the previous steps. As a result, minimization done in one step is not partially undone by the next [26].

A line search at each iteration is computationally expensive; it requires that the network response to all training inputs be computed several times for each search. The scaled conjugate gradient algorithm developed by Moller uses a step size scaling mechanism to avoid time-consuming line search [49].

**3.4.1.2 The pattern recognition network** Algorithm 3 shows how to use Matlab's `patternnet` to create, train and test a pattern recognition network, i.e. a two layer feedforward network with one hidden and one output layer, [`https://www.mathworks.com/discovery/pattern-recognition.html`]. The input data contains spike waveforms from four known classes. The target data contains column vectors of class labels for each corresponding waveform, e.g. [1,0,0,0] indicates class 1 and not classes 2, 3 or 4.

First, set the training function to be `trainscg`. Set the hidden layer to 10 nodes (default is 10), then create the net object. Set the neural network GUI window to not show. Set the division of data for training, validation, testing to 70, 15, and 15 percent. Train and test the network. Calculate the percent error.

```
x = input_data;
t = target_data;

training_function = 'trainscg';
hidden_layer_size = 10;
network = patternnet(hidden_layer_size, training_function);
network.trainParam.showWindow = false;

network.divideParam.trainRatio = 70/100;
network.divideParam.valRatio   = 15/100;
network.divideParam.testRatio  = 15/100;

[network,tr] = train(net,x,t);
predictions  = network(x);

target_idx = vec2ind(t);
pred_idx   = vec2ind(predictions);
percent_error = sum(target_idx ~= pred_idx)/numel(target_idx)*100;
```

**Algorithm 3:** Matlab code for the pattern recognition neural network.

Figure 5 shows the Matlab generated diagram for a two layer feedforward network with a sigmoid hidden layer and softmax output nodes (their output consists of positive numbers that sum to one). In theory it can classify waveforms arbitrarily well given enough nodes in its hidden layer. The input to this network is a matrix of waveforms, 21 features each, a hidden layer with 10 nodes, and an output layer with 4 nodes (one for each class). For each input waveform, the network's softmax output layer outputs a value in the range [0,1] for each possible class. This output sums to one for each waveform and indicates, in a sense, how confident the network is that this waveform is a member each class.

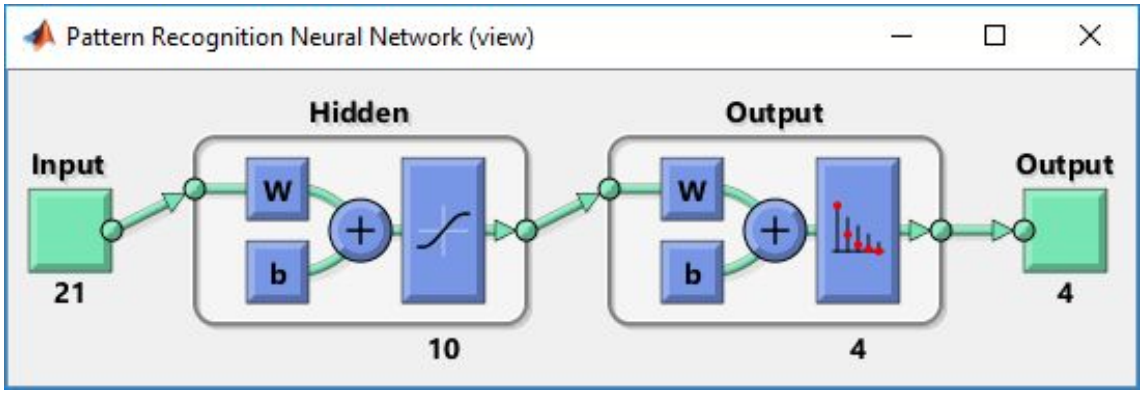



Figure 5: Diagram of a pattern recognition neural network with 21 nodes in its input layers, 10 nodes in its hidden layer, and 4 nodes in its output layer. Image: see Matlab `nprtool`, `nnstart`.

### 3.4.2 K-means

K-means [47] is an algorithm for finding structure in multidimensional data sets [45]; it is a form of unsupervised learning [39]. Although simple and proposed over 50 years ago, it is still popular and widely used [38]. Essentially, K-means clusters

"similar" objects together based on a distance metric.

If the clusters are well separated, then ideally all or nearly all waveforms in a cluster will be from the same neuron. However, this is challenging because each electrode can record from many neurons, the number of neurons in the neighborhood is not known, and waveforms from different neurons may overlap to some extent in feature space. This means that the SNR might not be high enough to accurately partition all of the data into separate clusters.

After PCA, the N principal components (PC) of each waveform were clustered by `kmeans` into k clusters using a distance metric, for example `c_numbers = kmeans(PC, k, 'Distance','sqeuclidean', 'Replicates', 10)`, where the best set of clusters out of 10 replicates were returned, and each replicate used different initial random centroids.

In general, compact and well separated clusters score high for quality [38]. The quality of the clusters depends partly on the initial random centroids, so `kmeans` can be set to run multiple times ('Replicates'), then it chooses the set of clusters than maximized the chosen quality metric. For this report, `kmeans` generated 10 replicates.

More details are provided in the Appendix, Section 7.4.

### 3.4.3 K-means accuracy

K-means is an unsupervised technique and quality metrics can be applied to determine the quality of its clusters, but the cluster numbers it generates are not associated with class numbers and may change during every replication. Indeed, K-means is not usually assessed for accuracy.

However, k-means can be assessed for accuracy using Algorithm 4 if the true class numbers for each waveform are known. Given four classes and cluster numbers 1 to 4, try all possible mappings of k-means cluster numbers to class numbers, and use the maximum score found as a measure k-means accuracy for comparison with supervised and semi-supervised classification algorithms.

```
max_acc=0;
maps=perms([1,2,3,4]);  % 24 possible permutations

for map_idx=1:24
    map=maps(map_idx,:)';
    idx_mapped=map(idx);

    temp=100*(sum(idx_mapped==cluster_targets)/length(cluster_targets));
    if max_acc < temp
        max_acc=temp;
    end
end
fprintf('maximum score = %0.2f', max_acc);
```

**Algorithm 4:** Pseudocode for finding the mapping from k-means cluster numbers to class numbers that maximizes the assessed accuracy.

### 3.4.4 K-medoids

K-medoids is similar to k-means. It partitions data into k clusters based on distance to the nearest medoid instead of the nearest centroid. It is based on medians so the center of a cluster, the medoid, is a member of the subset. Therefore it may be appropriate for domains that require robustness to outlier data, arbitrary distance metrics, or in cases where the mean does not have a clear definition [43, 55].

### 3.4.5 K-Nearest Neighbors (k-NN)

Intuition suggests that new objects may be classified based on their proximity in feature space to previously classified objects. The nearest neighbor (NN) rule is simple: classify a new object in the category of its nearest neighbor [15]. The first formulation of the k-nearest neighbor (k-NN) rule appears to have been in the 1950s by Fix and Hodges [18, 19]. Even when k=1, as the size of the sample set approaches infinity, the rule has a probability of error less than twice the Bayes error rate (the theoretical minimum) and, therefore, of any other decision rule [15]. This approach can also be used to build clusters. An iterative algorithm was proposed circa 1978 [46] that assigns each new object to the cluster of its nearest neighbor if that distance is below a threshold.

k-NN has some interesting properties. Firstly, the function is approximated locally and, therefore, sensitive to the local structure of the data. Secondly, it is "lazy", which means that all computation is deferred until classification.

k-NN has several advantages. No optimization or training is required. Accuracy often compares well with more complex methods such as ANNs. Also, it allows for easy implementation of a *reject option* for cases where confidence about the predicted classification is not high enough [28].

Although the basic premise is simple, many variants of k-NN are possible. For example, neighboring objects can be weighted by some function of distance. Consequently the user is burdened with having to choose a weight function, distance distance metric and, not least of all, k. Choice of k can be made empirically by running k-NN on a representative test set with a range of values for k and choosing the value that maximized accuracy. In general, the best choices depend on the data set and the research question.

# 4 Results

Spike waveforms were extracted from five files in the hc-1 data set, d533101.dat, d561103.dat, d561104.dat, d1453103.dat, and d14521.001.dat where each file included one IC channel and four EC channels. The five files correspond to five classes of spike waveform.

To compare the classification accuracy of a shallow neural network versus traditional methods, the training set was built by extracting 100 ground truth spikes per EC channel from the five hc-1 data files, 400 spike waveforms per class.

## 4.1 NN classification accuracy

Figure 6 shows the five classes of waveform in rows one to five. In some preliminary tests, the NN was able to distinguish the GT EC waveforms from non GT EC

waveforms with 100% accuracy ... the two classes were obviously different. In contrast, the waveforms shown in Figure 6 are much more challenging.

The NN was trained on the examples from the first four rows (four classes) in Figure 6, i.e. 1600 waveforms. The bar chart in row six shows that the NN correctly classified class 1 and class 4 waveforms as class 1 and class 4 respectively with very high probability ("confidence") on average, but could not easily distinguish classes two and three.

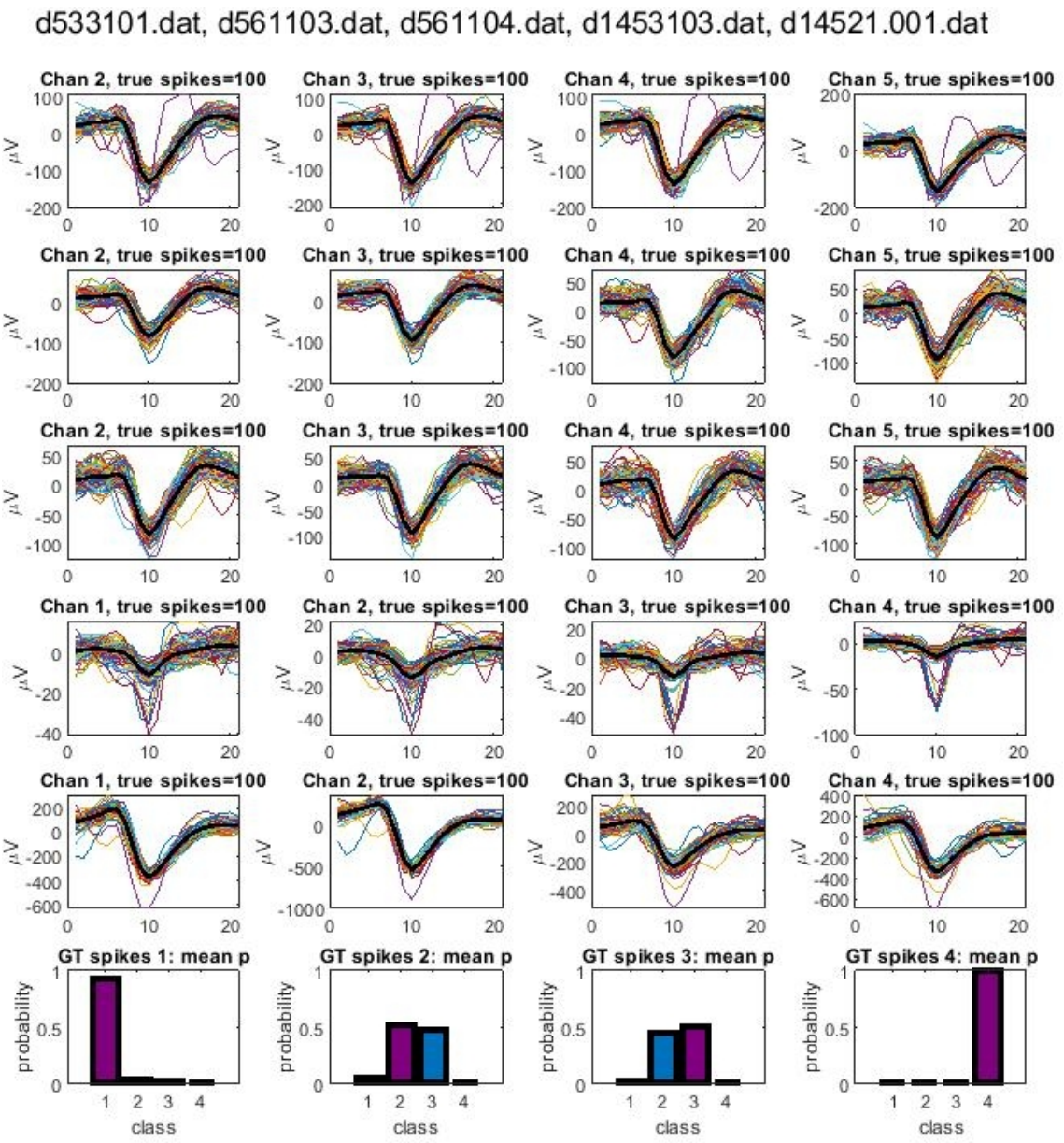
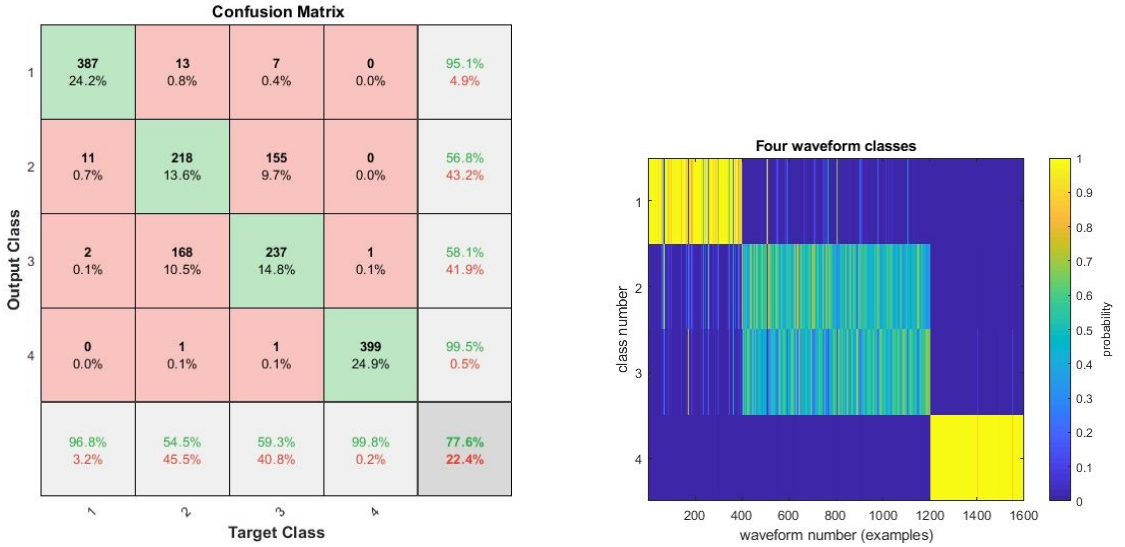



Figure 6: The data set was extracted from EC recordings taken during five separate experiments. Rows 1 to 5 correspond to five classes of EC waveforms and their 5 neuron sources. Row 6 shows a bar chart of NN predictions. Prediction accuracy was high for classes 1 and 4, but not for the more similar waveforms in classes 2 and 3. The NN was trained on examples from classes 1 to 4 only. Note: the y axis scales in rows 1 to 5 vary to reveal the waveform shapes.

Figure 7 shows a confusion matrix and an image of the probabilities. The confusion matrix in Figure 7a shows that the NN classified class 1 and class 4 waveforms with an accuracy of 95.1% and 99.5% respectively. Although classes 2 and 3 are very

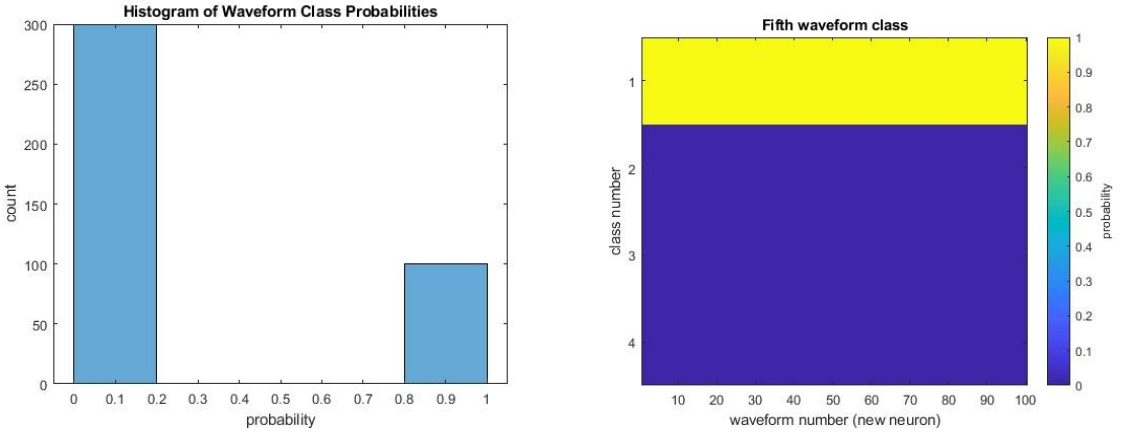
similar, the NN did better than chance, 56.8% and 58.1%. Overall accuracy was 77.6%. Figure 7b shows the same data as an image, where bright yellow indicates, in a sense, high confidence predictions.




(a) (b)

Figure 7: NN classification of waveforms from four classes: (a) Confusion matrix shows an overall accuracy of 73.6%, but relatively low accuracy for classes 2 and 3. (b) Image shows that waveforms classified as class 1 or class 4 were made with high probability (yellow), classifications for classes 2 versus 3 were made with lower probability (less confidence).

Figure 8 shows the NN's predictions when given class 5 waveforms. It predicted that all of them are class 1 waveforms, i.e. with a probability $\geq 0.90$.




(a) (b)

Figure 8: NN classification of new waveforms: (a) this bar chart shows low probability classifications in 3 out of 4 cases, and high probability classifications in 1 out of 4 cases, (b) the image plot shows that all class 5 waveforms were classified as class 1 waveforms with a probability of $\geq 0.90$.

## 4.2 NN classification accuracy versus traditional methods

As Table 1 shows, mean k-NN accuracy over 20 trials was relatively high with moderate standard deviation, and was almost as fast as k-means. However, the 20 trials for k-NN was done by setting k equal to values from 1 to 20. For any given value of k, k-NN accuracy is constant. The best k-NN accuracy was (74.69%, k=20), (74.44%, k=16), and (74.62%, k=13) using `wavedec` + KS, `pca`, and 'none' feature extraction respectively.

k-medoids was very slow and delivered poor accuracy regardless of feature extraction method or distance metric. PCA feature extraction improved classification accuracy for k-means, k-means-C, and k-medoids, but not for k-NN.

`kmeans` used squared Euclidean distance and the k-means++ algorithm [4] for centroid initialization (default settings). However, the initial centroids can set manually, by the (name, value) pair ('Start', C_init), where `C_init` is a matrix of the average features for each class, as follows: `kmeans(spk_feats, K, 'Distance', d_metric, Start', C_init)`. This variant of k-means is listed in Table 1 as k-means-C; it started with the true centroids.

Giving k-means the true centroids to start with ("k-means-C") did not increase accuracy if features were extracted using wavelets or PCA. This was because k-means already uses k-mean++ to find good initial centroids (not purely random centroids) and it returned the best result out of 10 replications. However, providing the true centroids to start with did improve accuracy for the case of no feature extraction, i.e. k-means 63.35%, k-means-C 71.63%.

As Table 2 shows, the NN performed better than the competition regardless of whether the hidden layer contained 10 or 20 nodes. Concatenation of waveforms from the four channels significantly improved accuracy over single channel waveforms (21 data points each).

| classifier | feature extraction | distance metric | mean acc (%) | std | time (s) |
|---|---|---|---|---|---|
| k-NN | wavedec + KS | Euclidean | 72.99 | 8.96 | 3.34 |
| k-NN | PCA | Euclidean | 73.08 | 8.74 | 3.14 |
| k-NN | none | Euclidean | 73.01 | 9.08 | 2.95 |
| k-means | wavedec + KS | squared Euclidean | 58.01 | 0.04 | 20.70 |
| k-means | PCA | squared Euclidean | 71.83 | 0.18 | 0.73 |
| k-means | none | squared Euclidean | 63.35 | 0.03 | 0.96 |
| k-means-C | wavedec + KS | squared Euclidean | 58.19 | 0 | 19.64 |
| k-means-C | PCA | squared Euclidean | 71.81 | 0 | 0.17 |
| k-means-C | none | squared Euclidean | 71.63 | 0 | 0.11 |
| k-medoids | wavedec + KS | squared Euclidean | 51.72 | 6.95 | 135.26 |
| k-medoids | PCA | squared Euclidean | 58.85 | 15.62 | 109.24 |
| k-medoids | none | squared Euclidean | 47.41 | 4.45 | 131.96 |
| k-medoids | wavedec + KS | Mahalanobis | 49.95 | 1.90 | 139.10 |
| k-medoids | PCA | Mahalanobis | 51.07 | 5.98 | 113.72 |
| k-medoids | none | Mahalanobis | 46.10 | 1.42 | 126.62 |

Table 1: Classification accuracy: traditional methods. 20 trials.

| classifier | nodes in hidden layer | data augmentation | mean acc (%) | std | time (s) |
|---|---|---|---|---|---|
| ANN | 10 | none | 76.97 | 0.79 | 5.65 |
| ANN | 20 | none | 77.20 | 0.87 | 5.15 |
| ANN | 10 | concatenation | 80.92 | 2.25 | 4.86 |
| ANN | 20 | concatenation | 81.20 | 2.95 | 4.58 |

Table 2: Classification accuracy: neural networks. 20 trials.

# 5 Discussion

The development of automatic computerized spike-sorting methods has been an active research topic for several decades [1, 45] and, after great effort, a completely automated solution has not been found. The spike-sorting problem is similar to the "cocktail-party problem" of tuning into one of the conversations at a busy party [9, 30], but if the SNR is too low, even the most scintillating gossip will be lost in the noise forever.

The waveforms in this report were sampled at 10 kHz and only 21 data points long, i.e. 10 before and 10 after the peak. This is a relatively small number of data points for a neural network to learn from, or for wavelets and PCA to extract features. Those methods are often used for instances composed of hundreds or thousands of data points.

Nonetheless, the classification accuracy of ANNs was significantly higher than the traditional methods considered in this report. ANNs are quite flexible, robust, and can learn from examples in various forms. For example, the ANN performed better when waveforms from the four channels were concatenated. The methods that relied on dimensionality reduction techniques and could not exploit this higher dimensional data format.

Although ANNs are highly flexible and parameterizable, they offer another advantage. They are not very sensitive to the number of nodes in their hidden layer, i.e. both 10 and 20 node hidden layer had similar accuracy. But a shallow ANN trained on a few classes of examples is far from ideal.

ANNs depend heavily on the training set. If a NN with 1 hidden layer and softmax output is trained on examples from 4 classes, as in this report, and then asked to predict the classification of samples from a 5th class, it is likely to classify all of them as one of the four classes that it was trained on with a probability $p \geq 0.90$, e.g. $p =$(0.95, 0, 0, 0.05) for the four classes respectively Essentially, this NN will not say, "I don't know" ... I don't recognize the new samples as being similar enough to anything I was trained on."

And yet this ANN can be even worse. A test input of all zeros was classified as a class 1 waveform. Therefore the ability to classify an input as unknown is essential. Otherwise the ANN might generate rational classifications most of the time and complete nonsense when fed pathological data.

ANNs can be optimized, but ultimately a neural network is only as good as its training set. Therefore, the network will be most useful after it has been trained on a database of labeled waveforms of interest, including negative examples. Then after new waveforms are classified, they can be checked against a distance metric. If the distance is greater than a defined threshold, then the waveform is not recognized at all and can be discarded.

# 6 Future Work

This report has presented empirical evidence that the addition of ANNs to a spike sorting pipeline may improve classification accuracy. However, ANN methods could also be tested on several other data sets that include some form of ground truth.

A more recent ground truth data set is available from the Kampff Lab. Their procedure for precisely aligning two probes for in vivo "paired-recordings" is described in [52]. Optically-calibrated mechanical manipulators positioned two probes at the same location in the brain. To obtain ground truth data, the spiking activity of a single neuron was monitored with a dense extracellular silicon polytrode and a micropipette [52].

Synthetic data sets, where spike times and classes are known, and Neurocube software [13], which uses detailed neuron models to simulate realistic EC recordings, can be obtained from The Centre of Systems Neuroscience at the University of Leicester.

Use Neurocube to generate a database of examples for an ANN to learn from, including examples of background noise and artifacts, then assess whether its classification of new data is potentially useful.

The performance of ANN pipelines can be compared with state of the art software such as Kilosort [54] or Kilosort2, on ground truth data. And, perhaps as the ultimate test, the performance of ANN enhanced pipelines can be compared with Kilosort2 on data recorded by Neuropixels by domain experts where ground truth is not available.

Links to the data sets and software:

- Cortex Lab: `https://www.ucl.ac.uk/cortexlab/`
- CRCNS datasets, hippocampus: `https://crcns.org/data-sets/hc`.
- Buzsaki Lab, description of CRCNS datasets: `https://buzsakilab.com/wp/resources/datasets/`
- Kampff Lab: `http://www.kampff-lab.org/validating-electrodes`, [52].
- Centre for Systems Neuroscience, data sets and software: `https://www2.le.ac.uk/centres/csn/software`.
- Kilosort: `https://github.com/cortex-lab/KiloSort`
- Kilosort2: `https://github.com/MouseLand/Kilosort2`

# References


[1] M. Abeles and M. H. Goldstein. Multispike train analysis. *Proceedings of the IEEE*, 65(5):762–773, May 1977.

[2] E D Adrian and Y Zotterman. The impulses produced by sensory nerve endings: Part 3. impulses set up by touch and pressure. *The Journal of physiology*, 61:465–483, August 1926.

[3] E D Adrian and Y Zotterman. The impulses produced by sensory nerve-endings: Part ii. the response of a single end-organ. *The Journal of physiology*, 61:151–171, April 1926.

[4] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[5] Hamed Azami, Javier Escudero, Ali Darzi, and Saeid Sanei. Extracellular spike detection from multiple electrode array using novel intelligent filter and ensemble fuzzy decision making. *Journal of neuroscience methods*, 239:129–138, 2015.

[6] Alex H Barnett, Jeremy F Magland, and Leslie F Greengard. Validation of neural spike sorting algorithms without ground-truth information. *Journal of neuroscience methods*, 264:65–77, May 2016.

[7] Robert Bestel, Andreas W Daus, and Christiane Thielemann. A novel automated spike sorting algorithm with adaptable feature extraction. *Journal of neuroscience methods*, 211:168–178, October 2012.

[8] Giovanni Bianchi and Roberto Sorrentino. *Electronic filter simulation & design*. McGraw-Hill New York, 2007.

[9] G D Brown, S Yamada, and T J Sejnowski. Independent component analysis at the neural cocktail party. *Trends in neurosciences*, 24:54–63, January 2001.

[10] Stephen Butterworth. On the theory of filter amplifiers. *Wireless Engineer*, 7(6):536–541, 1930.

[11] Gyorgy Buzsaki, Costas A Anastassiou, and Christof Koch. The origin of extracellular fields and currents–eeg, ecog, lfp and spikes. *Nature reviews. Neuroscience*, 13:407–420, May 2012.

[12] Ana Calabrese and Liam Paninski. Kalman filter mixture model for spike sorting of non-stationary data. *Journal of neuroscience methods*, 196:159–169, March 2011.

[13] Luis A Camuñas-Mesa and Rodrigo Quian Quiroga. A detailed and fast model of extracellular recordings. *Neural computation*, 25:1191–1212, May 2013.

[14] Jason E Chung, Jeremy F Magland, Alex H Barnett, Vanessa M Tolosa, Angela C Tooker, Kye Y Lee, Kedar G Shah, Sarah H Felix, Loren M Frank, and Leslie F Greengard. A fully automated approach to spike sorting. *Neuron*, 95(6):1381–1394, 2017.

[15] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[16] DL Donoho and IM Johnstone. Ideal spatial adaptation by wavelet shrinkage. biometrika 81 425–455. *Mathematical Reviews (MathSciNet): MR1311089 Zentralblatt MATH*, 815, 1994.

[17] Gaute T Einevoll, Felix Franke, Espen Hagen, Christophe Pouzat, and Kenneth D Harris. Towards reliable spike-train recordings from thousands of neurons with multielectrodes. *Current opinion in neurobiology*, 22:11–17, February 2012.

[18] Evelyn Fix and JL Hodges. Discriminatory analysis, nonparametric discrimination. 1951.

[19] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination: Small sample performance. Technical report, CALIFORNIA UNIV BERKELEY, 1952.

[20] George L Gerstein, Marc J Bloom, Ismael E Espinosa, Stephen Evanczuk, and Mark R Turner. Design of a laboratory for multineuron studies. *IEEE Transactions on systems, man, and cybernetics*, 1983.

[21] EDMUND M Glaser and WB Marks. Separation of neuronal activity by waveform analysis. *Advances in biomedical engineering*, 1:77–136, 1971.

[22] EM Glaser and WB Marks. On-line separation of interleaved neuronal pulse sequences. *Data Acquisition Process Biol Med*, 5:137–156, 1968.

[23] Carl Gold, Darrell A Henze, Christof Koch, and György Buzsáki. On the origin of the extracellular action potential waveform: A modeling study. *Journal of neurophysiology*, 95:3113–3128, May 2006.

[24] Shai N Gozani and John P Miller. Optimal discrimination and classification of neuronal action potential waveforms from multiunit, multichannel recordings using software-based linear filters. *IEEE Transactions on Biomedical Engineering*, 41(4):358–372, 1994.

[25] C M Gray, P E Maldonado, M Wilson, and B McNaughton. Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. *Journal of neuroscience methods*, 63:43–54, December 1995.

[26] M.T. Hagan, H.B. Demuth, and M.H. Beale. *Neural Network Design*. Boston, MA: PWS Publishing, 1996.

[27] Espen Hagen, Torbjørn V Ness, Amir Khosrowshahi, Christina Sørensen, Marianne Fyhn, Torkel Hafting, Felix Franke, and Gaute T Einevoll. Visapy: a python tool for biophysics-based generation of virtual spiking activity for evaluation of spike-sorting algorithms. *Journal of neuroscience methods*, 245:182–204, April 2015.

[28] David J. Hand, Padhraic Smyth, and Heikki Mannila. *Principles of Data Minin*. MIT Press, Cambridge, MA, USA, 2001.

[29] K D Harris, D A Henze, J Csicsvari, H Hirase, and G Buzsaki. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *Journal of neurophysiology*, 84:401–414, July 2000.

[30] Simon Haykin and Zhe Chen. The cocktail party problem. *Neural computation*, 17:1875–1902, September 2005.

[31] DA Henze, KD Harris, Z Borhegyi, J Csicsvari, A Mamiya, H Hirase, A Sirota, and G. Buzsáki. Simultaneous intracellular and extracellular recordings from hippocampus region ca1 of anesthetized rats, 2009.

[32] Darrell A Henze, Zsolt Borhegyi, Jozsef Csicsvari, Akira Mamiya, Kenneth D Harris, and György Buzsáki. Intracellular features predicted by extracellular recordings in the hippocampus in vivo. *Journal of neurophysiology*, 84(1):390–400, 2000.

[33] Gerrit Hilgen, Martino Sorbaro, Sahar Pirmoradian, Jens-Oliver Muthmann, Ibolya Edit Kepiro, Simona Ullo, Cesar Juarez Ramirez, Albert Puente Encinas, Alessandro Maccione, Luca Berdondini, et al. Unsupervised spike sorting for large-scale, high-density multielectrode arrays. *Cell reports*, 18(10):2521–2532, 2017.

[34] Daniel N Hill, Samar B Mehta, and David Kleinfeld. Quality metrics to accompany spike sorting of extracellular signals. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 31:8699–8705, June 2011.

[35] Eyal Hulata, Ronen Segev, and Eshel Ben-Jacob. A method for spike sorting and detection based on wavelet packets and shannon's mutual information. *Journal of neuroscience methods*, 117(1):1–12, 2002.

[36] Eyal Hulata, Ronen Segev, Yoash Shapira, Morris Benveniste, and Eshel Ben-Jacob. Detection and sorting of neural spikes using wavelet packets. *Physical review letters*, 85(21):4637, 2000.

[37] Donald R Humphrey and Edward M Schmidt. Extracellular single-unit recording methods. *Neurophysiological techniques: Applications to neural systems*, pages 1–64, 1990.

[38] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[39] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[40] Frank J. Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[41] James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Çağatay Aydın, Mladen Barbic, Timothy J Blanche, Vincent Bonin, João Couto, Barundeb Dutta, Sergey L Gratiy, Diego A Gutnisky, Michael Häusser, Bill Karsh, Peter Ledochowitsch, Carolina Mora Lopez, Catalin Mitelut, Silke Musa, Michael Okun, Marius Pachitariu, Jan Putzeys, P Dylan Rich, Cyrille Rossant, Wei-Lung Sun, Karel Svoboda, Matteo Carandini, Kenneth D Harris,

Christof Koch, John O'Keefe, and Timothy D Harris. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551:232–236, November 2017.

[42] James Jaeyoon Jun, Catalin Mitelut, Chongxi Lai, Sergey Gratiy, Costas Anastassiou, and Timothy D Harris. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *bioRxiv*, page 101030, 2017.

[43] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[44] J C Letelier and P P Weber. Spike sorting based on discrete wavelet transform coefficients. *Journal of neuroscience methods*, 101:93–106, September 2000.

[45] M S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network (Bristol, England)*, 9:R53–R78, November 1998.

[46] S. Lu and K. S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *and Cybernetics IEEE Transactions on Systems, Man*, 8(5):381–389, May 1978.

[47] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[48] George Marsaglia, Wai Wan Tsang, and Jingbo Wang. Evaluating kolmogorov's distribution. *Journal of Statistical Software, Articles*, 8(18):1–4, 2003.

[49] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[50] Nhamoinesu Mtetwa and Leslie S Smith. Smoothing and thresholding in neuronal spike detection. *Neurocomputing*, 69(10-12):1366–1370, 2006.

[51] Zoran Nenadic and Joel W Burdick. Spike detection using the continuous wavelet transform. *IEEE transactions on Biomedical Engineering*, 52(1):74–87, 2005.

[52] Joana P Neto, Gonçalo Lopes, João Frazão, Joana Nogueira, Pedro Lacerda, Pedro Baião, Arno Aarts, Alexandru Andrei, Silke Musa, Elvira Fortunato, Pedro Barquinha, and Adam R Kampff. Validating silicon polytrodes with paired juxtacellular recordings: method and dataset. *Journal of neurophysiology*, 116:892–903, August 2016.

[53] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[54] Marius Pachitariu, Nicholas A Steinmetz, Shabnam N Kadir, Matteo Carandini, and Kenneth D Harris. Fast and accurate spike sorting of high-channel count probes with kilosort. In *Advances in Neural Information Processing Systems*, pages 4448–4456, 2016.

[55] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert Syst. Appl.*, 36(2), March 2009.

[56] R Quian Quiroga, Zoltan Nadasdy, and Yoram Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004.

[57] Rodrigo Quian Quiroga. Spike sorting. *Scholarpedia*, 2(12):3583, 2007.

[58] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. Past, present and future of spike sorting techniques. *Brain research bulletin*, 119:106–117, 2015.

[59] Jorge J Riera, Takakuni Goto, and Ryuta Kawashima. A methodology for fast assessments to the electrical activity of barrel fields in vivo: from population inputs to single unit outputs. *Frontiers in neural circuits*, 8, 2014.

[60] Cyrille Rossant, Shabnam N Kadir, Dan F M Goodman, John Schulman, Maximilian L D Hunter, Aman B Saleem, Andres Grosmark, Mariano Belluscio, George H Denfield, Alexander S Ecker, Andreas S Tolias, Samuel Solomon, Gyorgy Buzsaki, Matteo Carandini, and Kenneth D Harris. Spike sorting for large, dense electrode arrays. *Nature neuroscience*, 19:634–641, April 2016.

[61] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[62] Ueli Rutishauser, Erin M Schuman, and Adam N Mamelak. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of neuroscience methods*, 154:204–224, June 2006.

[63] Shuzo Sakata and Kenneth D Harris. Laminar structure of spontaneous and sensory-evoked population activity in auditory cortex. *Neuron*, 64(3):404–418, 2009.

[64] N Schmitzer-Torbert, J Jackson, D Henze, K Harris, and A D Redish. Quantitative measures of cluster quality for use in extracellular recordings. *Neuroscience*, 131:1–11, 2005.

[65] Nicholas A Steinmetz, Christof Koch, Kenneth D Harris, and Matteo Carandini. Challenges and opportunities for large-scale electrophysiology with neuropixels probes. *Current opinion in neurobiology*, 50:92–100, June 2018.

[66] Steven P Strong, Roland Koberle, Rob R de Ruyter van Steveninck, and William Bialek. Entropy and information in neural spike trains. *Physical review letters*, 80(1):197, 1998.

[67] Pierre Yger, Giulia Lb Spampinato, Elric Esposito, Baptiste Lefebvre, Stéphane Deny, Christophe Gardella, Marcel Stimberg, Florian Jetter, Guenther Zeck, Serge Picaud, Jens Duebel, and Olivier Marre. A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo. *eLife*, 7, March 2018.

# 7 Appendix

## 7.1 Signal processing: bandpass filters

The Signal Processing Toolbox in Matlab includes the function `butter`. This function was used to implement a second order Butterworth bandpass filter to prepare the data for spike detection. The ideal aim of this filter is to have uniform sensitivity to the wanted frequencies, and zero sensitivity to unwanted frequencies, i.e. completely reject unwanted frequencies [10]. In practice the frequency response of the Butterworth filter is maximally flat in the passband, and attenuates rapidly in the stopband (less than and greater than the passband) [8].

The choice of filters is important. Finite Impulse Response (FIR) filters are preferred over Infinite Impulse Response (IIR) filters. FIRs work with a convolution of the signal, while IIRs work with convolution and recursion. IIRs are more computationally efficient, but they can cause distortions of phase and waveform shape [57].

## 7.2 Spike Detection: median absolute deviation

This threshold is defined as a multiple of the median absolute deviation (MAD)

$$Thr = K \times \sigma_n; \qquad \sigma_n = \text{median}\left(\frac{|X|}{0.6745}\right) \tag{1}$$

where $K$ is the multiple, $X$ is the filtered data (mean is zero), and the denominator 0.6745 is derived from the inverse of the cumulative distribution function for the standard normal distribution evaluated at 0.75 [58]. The area under this curve left of 0.6745 standard deviations equals 0.75.

The median is less sensitive than the mean to the firing rate, large spike amplitudes, and artifacts. For example, as Figure 9 shows, the traditional standard deviation estimate of noise increases rapidly with the firing rate, while MAD remains relatively flat.

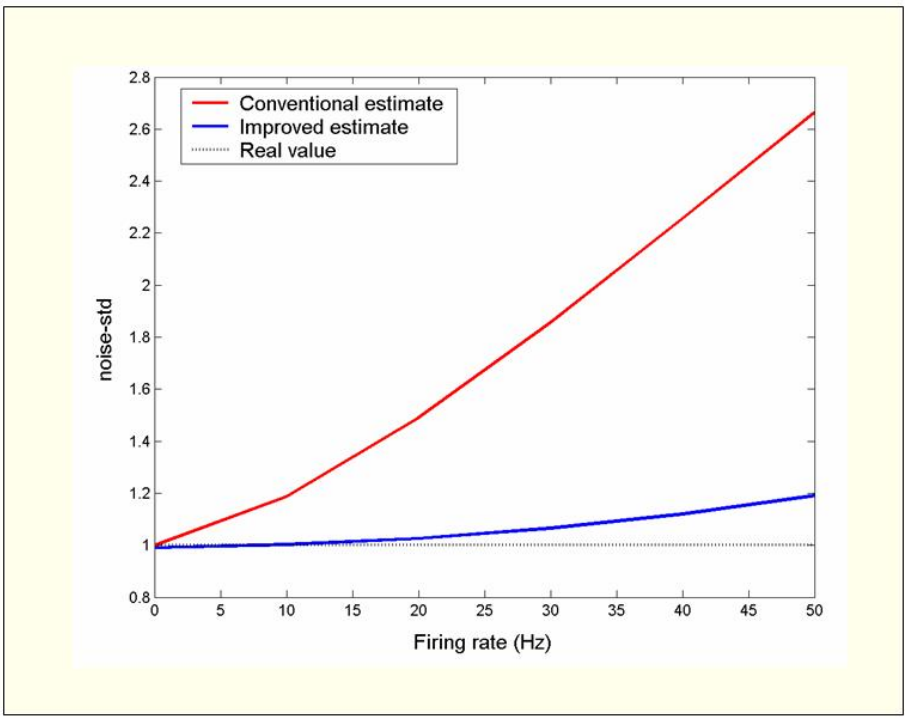



Figure 9: Median absolute deviation (MAD) versus standard deviation. The conventional mean based estimate of the standard deviation of background noise (red) increases rapidly with firing rate, while the median based estimate (blue) increases slowly. Image [57].

In general, setting the threshold for spike detection involves a trade-off between false positives and false negatives [7]. While a higher threshold may prevent more false positives, more true spikes may be discarded. However, at some point, waveforms that are too far from their centroid, nearest neighbors, or some other class landmark, can be discarded as outliers, i.e. tossed in a bin of unrecognized waveforms. So in this case a threshold that is slightly too low may be preferred.

Although outside the scope of this report, picking the right threshold is less important when using templates. The SpyKING CIRCUS algorithm uses a clustering step followed a template match step [67].

## 7.3 Feature extraction

### 7.3.1 Waveform features can be extracted by wavelet decomposition

Wavelets can be used to reduce the dimensionality of data. This section provide only a very brief summary of wavelets focused on discrete wavelets.

A wavelet is a sharp wave-like oscillation with a zero mean and a corresponding scaling signal. Unlike sinusoids, which extend to infinity, a wavelet exists for a finite duration. Wavelets come in different sizes and shapes. Families of discrete wavelets include Coiflets, Daubechies (includes Haar), and Symlets. Well known wavelets such as Mexican hat and Meyer are continuous, while Morlet wavelets are complex-valued. [https://en.wikipedia.org/wiki/Wavelet]

The discrete wavelet transform (DWT) convolves the data (a spike in this case) with the wavelet to get detailed coefficients (cD), and the scaling signal to get the approximate coefficients (cA), as shown respectively in Figure 10.

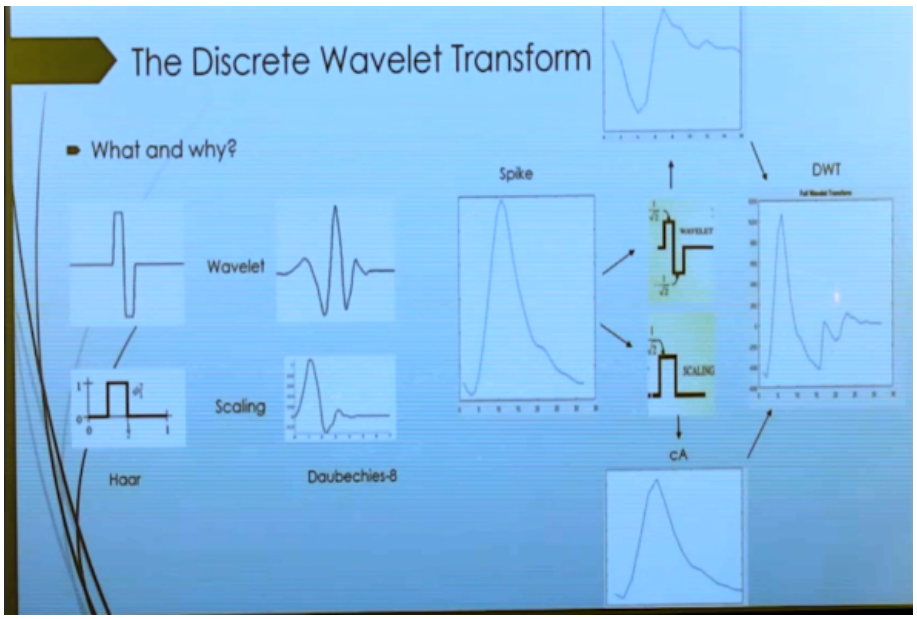



Figure 10: The Discrete Wavelet Transform (DWT). Image: Bhagat and Moore-Kochlacs, An introduction to spike sorting, Center for Brains Minds + Machines, MIT, video, 24 March 2017.

### 7.3.2 Scaling and shifting can be applied to wavelets.

Scaling refers to stretching or shrinking the signal in time

$$\Psi\left(\frac{t}{s}\right), s > 0$$

where the scale factor $s$ is a positive value and corresponds to how much a signal is scaled in time. It is inversely proportional to frequency. Scaling a 10Hz sine wave by 2 results in a 5Hz sine wave (reduces frequency by 1 octave, i.e. $\log_2(10/5) = 1$ octave).

There is a reciprocal relationship between scale and frequency with a constant of proportionality called the "center frequency" of the wavelet. Unlike the sine wave, the wavelet has a bandpass characteristic in the frequency domain. The equivalent frequency is defined using this equation

$$F_{eq} = \frac{C_f}{s\delta t}$$

where $C_f$ is center frequency of the wavelet, $s$ is the wavelet scale, and $\delta t$ is the sampling interval. A larger scale factor results in a stretched wavelet, which corresponds to a lower frequency. A smaller scale factor results in a shrunken wavelet, which corresponds to a high frequency. A stretched wavelet helps in capturing the slowly varying changes in a signal; a compressed wavelet helps in capturing abrupt changes.

Shifting a wavelet delays or advances the onset of the wavelet along the length of the signal. A wavelet represented with this notation

$$\phi(t - k)$$

is shifted and centered at k. The wavelet is can be shifted to align with the feature of interest in a signal.

Key applications of the continuous wavelet analysis are: time frequency analysis, and filtering of time localized frequency components. The key applications for Discrete Wavelet Analysis are denoising and compression of signals and images.

### 7.3.3 Discrete wavelet transforms (DWT)

DWT helps represent many naturally occurring signals and images with fewer coefficients. This enables a sparser representation. The base scale in DWT is set to 2.

Dyadic scaling and shifting. This kind of sampling eliminates redundancy in coefficients. Scaling:

$$2^j, \quad (j = 1, 2, 3, 4...)$$

Translation:

$$2^j m \quad (m = 1, 2, 3, 4...)$$

The detailed coefficients contain the high frequency components of the spike and, therefore, contain most of the energy. The approximate coefficients ("rough coefficients") contain the low frequency components of the spike and far less energy.

### 7.3.4 Principal components analysis

Algorithm 5 shows pseudocode for PCA. One approach is to set N equal to a fixed number, construct the covariance matrix with `cov`, do eigenvalue decomposition with `eig`, sort eigenvalues in descending order, extract the first N eigenvectors, and project the data matrix M (waveforms, in this report) onto a lower dimensional space using matrix multiplication.

**Data:** a matrix of waveforms
**Result:** dimension reduction
N = 3;
C = cov(M);
[V, D] = eig(C);
[eigen_values, eigen_value_indices] = sort(diag(D), 'descend') ;
V_low = V(:,eigen_value_indices(1:N)) ;
PC = M * V_low;

**Algorithm 5:** Pseudocode for principal component analysis. The length of each waveform depends on the sampling rate. In this report they were 21 data points long (about 2 milliseconds). PCA can usually reduce this down to about 3 principal components.

## 7.4 Clustering metrics

Matlab's `kmeans` function uses the cluster quality metric `silhouette` by default. It provides a graphical representation of how well each data object has been classified [61], and a score. Essentially, it measures how similar each object (vector of waveform features) is to its cluster (cohesion) relative to other clusters (separation). So, for each object $i$ in its own cluster $C_i$, let

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

be the average distance between $i$ and all other objects in $C_i$, where $d(i,j)$ is the distance between $i$ and $j$ in this cluster. The average dissimilarity of $i$ to objects in another cluster is just the mean distance of $i$ to those objects. The cluster with the smallest dissimilarity $b(i)$, as calculated by the following equation,

$$b(i) = \min_{i \neq j} \frac{1}{|C_j|} \sum_{j \in C_j} d(i,j),$$

is the *neighboring cluster* of $i$. The silhouette value of object $i$ is

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{if } |C_i| > 1$$

and

$$s(i) = 0, \text{if } |C_i| > 1$$

The mean $s(i)$ for all objects is a measure of how well the data has been clustered. This value ranges from -1 to +1, where a high value indicates high cohesion and separation, and the quality of the clustering is high. Low or negative values indicate poor quality and, therefore, too many or too few clusters. The silhouette value can be calculated with any distance metric, including squared Euclidean distance, the Manhattan distance, or the Mahalanobis distance [43].

### 7.4.1 Distance metrics for clustering

Given some number of waveforms, each represented by a feature vector of length $n$. The distance between two feature vectors $p$ and $q$ in $n$ dimensional space may be calculated by many methods [ `http://www.mathworks.com` ], including:

the squared Euclidean distance:

$$d(p,q) = \sum_{i=1}^{n} (q_i - p_i)^2,$$

the Euclidean distance (straight line distance):

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2},$$

the Manhattan distance (city block distance):

$$d(p,q) = \sqrt{\sum_{i=1}^{n} |q_i - p_i|},$$

the Mahalanobis distance:

$$d(p,q) = \sqrt{(p-q)M^{-1}(p-q)^T}$$

where $M$ is the covariance matrix, and $T$ indicates the transpose.

If $M$ is the identity matrix, the Mahalanobis distance reduces to the Euclidean distance. If $M$ is diagonal, then the resulting distance measure is called a standardized Euclidean distance

$$d(p,q) = \sqrt{\sum_{i=1}^{n} \frac{(p_i - q_i)^2}{s_i^2}}$$

where $s_i$ is the standard deviation of $p_i$ and $q_i$ over the sample set. [ `https://en.wikipedia.org/wiki/Mahalanobis_distance` ]

### 7.4.2 Cluster quality metric options

Matlab function `evalclusters` provides several metrics for the evaluation of clusters.

The following creates a clustering evaluation object containing data used to evaluate the optimal number of data clusters:

```
eva = evalclusters(x,clust,criterion)
```

where `x` is the data to be clustered, `clust` is the clustering algorithm, such as `kmeans`, and the criterion is `CalinskiHarabasz`, `DaviesBouldin`, `gap`, or the default `silhouette`. Additional options can specified by one or more name-value pair arguments. For example

```
eva = evalclusters(x,'kmeans','silhouette','KList',[1:6]),
```

would evaluate the resulting clusters with the silhouette criterion for k=1,2,3,4,5,6 and identify the optimal value for `k`. [ `https://www.mathworks.com/help/stats/evalclusters.html` ]

How can we assess unit quality? See *Quantitative measures of cluster quality for use in extracellular recordings* [64].

Given data point P and data distribution D, the Mahalanobis distance is a measure of the distance between P and D. In this report, P is vector of waveform features, D is a cluster of waveforms, and the mean of D is the cluster's centroid.

The Mahalanobis distance is a multi-dimensional generalization of measuring the distance between a point and a center in terms of standard deviations. This distance is zero if P is at the mean of D, and increases along each principal component axis as P moves away from the mean. If each of these axes is re-scaled to have unit variance, then the Mahalanobis distance corresponds to standard Euclidean distance in the transformed space. The Mahalanobis distance is unitless and scale-invariant. It takes into account the correlations of the data set, whereas Euclidean distance ignores the properties of a distribution (cluster) [ `https://en.wikipedia.org/wiki/Mahalanobis_distance` ].

The Mahalanobis distance $d^2(i,C)$ of spike $i$ from the center of the cluster $C$ is defined by the formula

$$d^2(i,C) = (X_i - \mu_C)^T M_C^{-1} (X_i - \mu_C)$$

where $X_i$ is the feature vector for spike $i$, $C$ is the mean of the values of the spikes in cluster $C$, and $M_C$ is the covariance matrix of the spikes in cluster $C$ [64].