EEE 511 - Final Project
Automated Spike Sorting

Xiao Huang, Suhas Lohit, Rakshit Raghavan
1206424709, 1206451918, 1207751060

***Abstract*—Neural spike sorting is a challenging unsupervised machine learning problem where neural spikes are to be clustered based on the similarity of their shapes. In this project, we have discussed various solutions to this problem and demonstrated the performance of our algorithms on the datasets provided. We look at different feature extraction techniques such as Principal Component Analysis, wavelets etc. and at different clustering algorithms. Of the algorithms considered, it was determined through the testing results that the combination of Haar wavelet features and Gaussian Mixture Models yielded the best clustering accuracy and we discuss in detail the results obtained. We also identify areas for further improvement of the algorithms considered.**

***Index Terms*—Spike sorting, Haar Wavelet Features, Gaussian Mixture Models**

I. Introduction

SPIKE sorting has been a widely used technique in modern as well as classical neurophysics and biology. This technique essentially is used for identifying and distinguishing neuron activity in the brain cells. A basic principle that is widely used here is to partition the neurons into clusters based on their stimuli or firing rate. When there is simultaneous activity in the brain, the interactions produce a plethora of waveforms together. Thus, the detection, sorting and separation of these neural spikes becomes important and is hence considered a challenging machine learning problem. In general, spike sorting can be partitioned into four steps in order. The first step is raw data collection where data is acquired from the human or animal. This process involves acquisition of data from a number of sources or nodes simultaneously into one single dataset. The second step is filtering of this huge data source into usable dataset. Various techniques such as windowing, thresholding, lowpass/bandpass filters etc. are used to initially extract the data [1]. Filtering ensures that data is well separated from background and other noise sources. This is followed by peak detection and alignment. The third step is the extraction of features from the detected data and the final step now is to separate the featured data into their clusters. Since there is a high probability of these newly formed clusters overlapping, distinct boundary formation and classification is an important part of spike clustering

In this project, we assume that the spikes have been detected, aligned and converted into snippet form. Hence, we focus only on the final part of the process - separating the waveforms into clusters such that the each cluster contains spikes belonging to a single neuron. This can be split into two steps (1) feature extraction and (2) clustering.

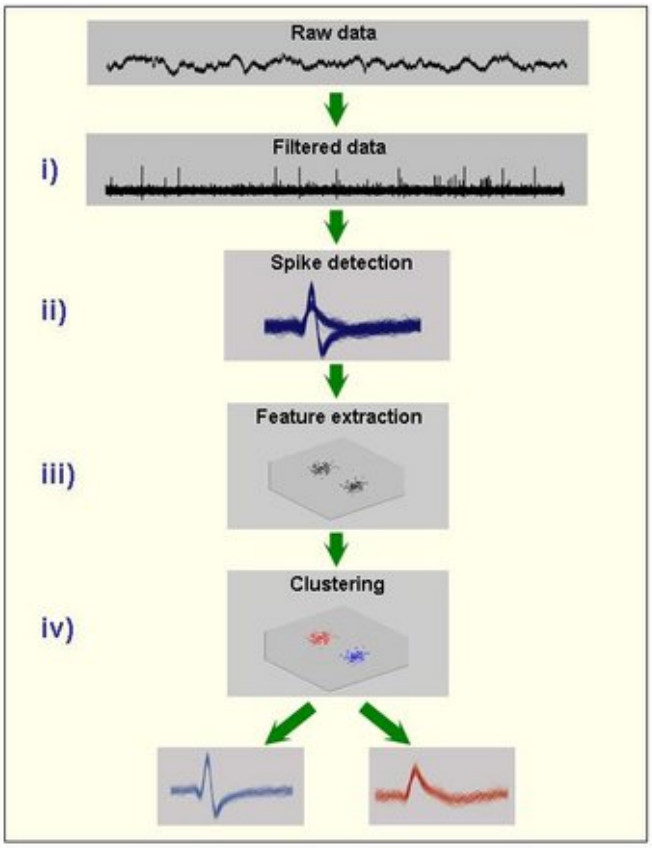


Fig. 1: Main steps in spike sorting. [1]

In Section II, we discuss the related work. A brief detail about the datasets and an initial visual inspection is given in Section III . In Section IV, we investigate various methods of feature extraction suitable for spike sorting. Once the features are extracted, the next step is clustering the spikes in the feature space. We explore different clustering algorithms for this purpose in Section V . It is possible that the best clustering method for a particular set of features may not be the optimal for another set. It is for this reason that we use the training data for different combinations of features and clustering algorithms. The results obtained are detailed in Section VI.

II. Background and Related Work

Spike sorting is an unsupervised clustering problem. Experimental data collection in itself is a time consuming and tedious activity. This raw data collected is not ready to use and

a number of preprocessing and filtering algorithms needs to be applied on this raw data before proceeding with any analysis.

There is a large amount of literature available discussing various strategies and combinations for feature extraction and clustering [3] for spike sorting.

Simple features such as peak amplitude, width [4], Fourier features etc. have been utilized in the past, with limited success. Principal Components Analysis (PCA) has been used in spike sorting with limited success. Another option is wavelet analysis [5]. This has been to shown to be better than PCA features [1]. It is also possible to use the raw snippets as features themselves and proceed to the next step in spike sorting – clustering.

As in the case of feature extraction, a host of clustering algorithms developed in machine learning and data mining can be employed for spike sorting. An obvious choice of a clustering algorithm is the k-means algorithm, which approximately solves the NP-hard problem of finding clusters such that the average distance of points to the respective cluster centers is minimized.Other popular options for clustering include Gaussian Mixture Models (GMM). This has been applied to clustering neural spikes [6]. Here, the number of Gaussian components must be specified to the algorithm. In [1], a clustering algorithm that utilizes nearest-neighbor interaction called super-paramagnetic clustering was proposed for spike sorting and was shown to be very successful.

Milestone-1: As a part of the first milestone for this project, we found publicly available neural spike datasets. In the case where only raw data is available, we demonstrated how we can obtain the snippets required for sorting. This helped us understand the basic processes involved in spike sorting and its challenges. A major part of our work was to first apply the principles on the *Wave_Clus* datset developed by Quiroga et al. [1] and get a feel of snippets generation and alignment. We also found two other neural data sources - the first one being *hc-1* dataset and the second one, the *pvc-5* from the **CRCNS** - *Collaborative Research in Computational Neuroscience* website [7]. These datasets gave a considerable idea of our approach and helped us proceed with this project further.

III. Spike Dataset Details

The training and testing datasets provided for this project contain 1500 spikes each. By ”training data”, we mean the data that was used to build the model that achieves the best accuracy. The testing data is used only once to determine the clustering accuracy. The pre-processing (data detection and filtering) has been achieved and the spikes are aligned at the 16th dimension of each data sample. Each spike has total of 48 samples. The final data provided for feature extraction and clustering is a 1500 X 48 matrix. It is observed that each dataset has a variety of spike shapes and is difficult to differentiate one spike from the other, even visually, especially for datasets 3 and 4. Visual inspection of these plots also suggest that the presence of combinations may have a single peak, multiple peaks, a trough followed by a peak, a peak followed by a trough etc. A few of these variations are

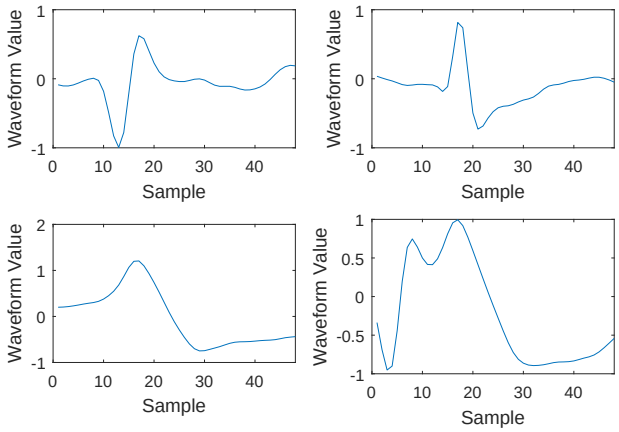


Fig. 2: Various spike shapes present in training dataset 1. This indicates how challenging the problem is. For example, the double peak may cause unexpected results after feature extraction and clustering. The algorithm designed must be robust to such variations.

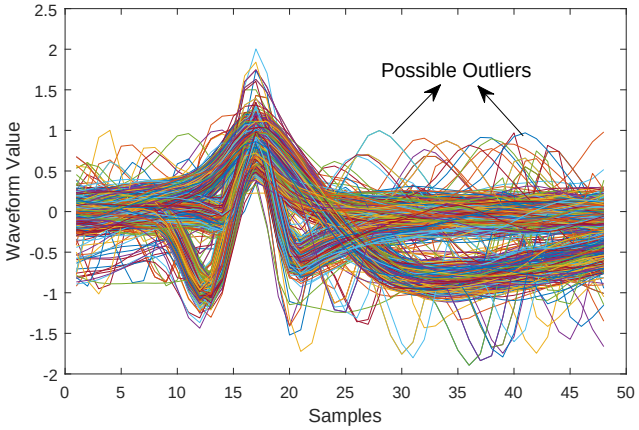


Fig. 3: An overlaid plot of all the snippets in training dataset 1. We clearly see the presence of outliers which are hard to cluster compared to the ”well-behaved” snippets.

illustrated in the Figure 2. The presence of such outliers and noisy waveforms can also be observed by overlaying the plots of the entire dataset as shown in Figure 3.

IV. Feature Extraction Algorithms

The following methods were explored for extracting features for the spike datasets provided.

A. Principal Components Analysis

Simple features such as peak amplitude, width [4], Fourier features etc. have been utilized in the past, with limited success. A simple machine learning technique that can be easily employed is the well known dimensionality reduction technique - principal component analysis (PCA). Here, the data are projected onto the principal directions, which are orthogonal vectors accounting for maximum variation in the data. The number of principal components is less than or equal to the number of original variables. The resulting vectors

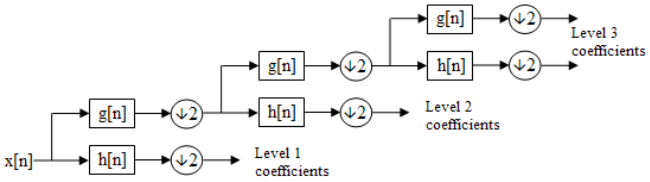


Fig. 4: Generating multi-level Haar wavelet coefficients. $\mathbf{x}(n)$ is the snippet.

are an uncorrelated orthogonal basis set. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. PCA is designed with optimal reconstruction as the objective and as such, cannot always be expected to preserve the cluster distances. We compute and use the principal components for the spike datasets and use these as the feature descriptors. These are then clustered in stage two.

B. Wavelet Features

The spikes generated by different neurons have different shapes and thus, are also very likely to have different frequency signatures. We extract wavelet features which capture this information, based on [1]. Fourth level decomposition of the signal using Haar wavelets is used for this purpose. Each snippet is convolved with two signal $g(n) = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$ and $h(n) = [\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}]$. This gives us the first level coefficients. Then, this convolution step is repeated on the coefficients obtained from the convolution of the snippet and $g(n)$ to get the second level and so on. This is shown in Figure 4.Since the snippets are 48 dimensional, we get 48 wavelet coefficients for each snippet. The multi-level decomposition thus obtained contain both the high and low frequency components at different scales and has been shown to be useful in spike sorting. We experiment with both the full set of features as well as a subset of features selected as in [1] where we only use the top wavelet coefficients which are the most discriminative. These features are expected to have multimodal distributions and are selected using Lillifors test. Using this test, those coefficients that are farthest from being distributed as a Gaussian are selected and input to the clustering algorithm in the next stage. Through experiments, it was determined that using a smaller subset of features selected in this manner did not yield the best results.

C. Shape Context Features

There is a vast amount literature in computer vision that deals with characterizing shapes of objects in images such as handwritten digits. We found that spike sorting could also be posed as clustering of objects based on their shape. Here, the objects are the plotted waveforms of the snippets. That is, each waveform is first plotted and saved as an image. Thus, we have one image per snippet. Then, each image of a snippet is used to extract shape features of the curve that is in the image. For extracting shape characteristics, we use very well known shape descriptor in computer vision – shape contexts [8].

The details of this feature are described as follows. The first step is extracting only the contours of the image using an edge detector. In our case, this corresponds to the snippet itself. The contour can be stored as a set of edge pixel locations $S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$. This set of points is sampled to reduce final feature dimension as well as computational complexity. Usually, about 100 samples are used in the case of hand written digit recognition. In our case, we use 50 samples. By making the assumption that the snippet waveforms are smooth, this number of samples from the contour is sufficient to approximate the entire waveform. For each point (x_i, y_i) in the S, we generate the *shape context* vector by computing a histogram h_i of the relative coordinates of the remaining points in S. The bin centers of the histogram are the distances from the point (x_i, y_i) and the element $h_i(k)$ contains the number of points in S that are at a distance represented by the bin (see [8] for more details). The concatenation of all the histograms for all the points in S gives shape context feature vector for a snippet. This is done for all snippets in the dataset before inputing them into a clustering algorithm in the second stage. To the best of our knowledge, this type of shape descriptor has not been employed in spike sorting before.

V. Clustering Algorithms

We have experimented with various clustering algorithms to cluster the features extracted in the previous section. These are described briefly below:

A. k-means

Once the features are obtained, we use k-Means as the first clustering method. For a given dataset, k-Means returns k vectors, called the cluster centers, such that the dataset is divided into k clusters. Each point in the dataset is assigned to a single cluster center that is "closest" to it in some sense.

More formally, given a set of n datapoints $\{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n\}$, we want to obtain k disjoint clusters $C = (C_1, C_2, ...C_k)$ with cluster centers, $\{\mu_1, \mu_2, ...\mu_k\}$ where each μ_i is the mean of the points in cluster i, such that the total sum of intra-cluster distances over the entire dataset is minimized. Mathematically, we would like to find

$$argmin_C \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mu_i||^2 \quad (1)$$

This is an NP-hard problem and the algorithm that is widely used to approximately solve this problem is called the k-means algorithm or Lloyd's algorithm. Although it is simple to implement, it is sensitive to initialization and the no. of clusters, k should be known apriori. Usually, k is determined using cross-validation. The distance function is usually the Euclidean distance although other variants exist. The algorithm is shown in Algorithm 1.

B. Gaussian Mixture Models

The second clustering algorithm that we have implemented in this project is the Gaussian Mixture Model. This has been

Algorithm 1 k-Means for clustering

Input: Training Data: $X = \{\mathbf{x}_i\} i = 1, 2, ..n,$,
No. of Clusters = k
Output: Cluster Centers $\{\mu_i\}, i = 1, 2, ...k$
Initialize $\{\mu_k\}$ randomly by choosing k points from the dataset
while (Cluster Assignments Change) **do**
for $i = 1 : n$ **do**
Cluster($\mathbf{x}_i$) = $argmin_i ||\mathbf{x} - \mu_i||^2$
end for
for $i = 1 : k$ **do**
$\mu_i = \frac{1}{|C_i|} \sum_{\mathbf{x}_i \in C_i} \mathbf{x}_i$
end for
end while

applied to clustering neural spikes [9]. Here, the data is assumed to be distributed as a weighted sum of K Gaussians [10] and each cluster is represented by a Gaussian:

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{K} w_i g(\mathbf{x}|\mu_i, \Sigma_i) \tag{2}$$

where $\mathbf{x}$ is the D-dimensional input vector, the spike snippet for example. $\lambda = \{w_i, \mu_i, \Sigma_i\}, i = 1...K$, w_i is the weight for the i^{th} mixture component such that $\sum_{i=1}^{K} w_i = 1$ and

$$g(\mathbf{x}|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp(-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)) \tag{3}$$

The parameters λ are learnt using the Expectation - Maximization (EM) [11] algorithm, such that each Gaussian corresponds to a cluster. It has been shown empirically that GMMs generally produce better clusters than k-means. k-means can be viewed as a special case of the EM algorithm where the hard assigning of points to clusters is used instead of probabilities and the clusters are assumed to be spherical i.e. the covariances of the Gaussians are multiples of the identity matrix. The EM algorithm is general and is applicable to a variety of problems. For the GMM, the algorithm is shown in Algorithm 2.

Initializing the EM algorithm: Determining the optimal parameters of the GMM based on the data is a non-convex problem. In addition to this, it is unbounded and multi-modal, which means that the optimization problem is hard to solve. Thus, the EM is only a heuristic and is likely to get stuck in a local optimum. In such a case, initialization of the algorithm becomes an important factor in the final results. It has been shown that running k-means and using the k-means solution to initialize the parameters of the GMM generally leads to better solutions. However, k-means is also only an approximate solution to the NP-hard problem of clustering. Thus, in order to initialize GMM with k-means, we run k-means multiple times and use the best solution to initialize the parameters of the GMM.

It is worth mentioning that generally, GMMs are superior to k-means because they allow for more flexible cluster shapes. Also, if the initialization of the GMM is good, the effect of outliers on the parameters is less in the case of GMMs than in k-means.

Algorithm 2 EM for GMM

Input: Training Data: $X = \{\mathbf{x}_i\} i = 1, 2, ..n,$,
No. of Clusters = K
Output: $\{\mu_i\}, \Sigma_i, w_i, i = 1, 2, ...K$
Initialize parameters randomly or using k-means
while (Not Converged) **do**
Expectation: $\forall j \in \{1, 2, ..., K\}$

$p(j|\mathbf{x}_n) = \frac{w_j g(\mathbf{x}|\mu_j, \Sigma_j)}{\sum_{i=1}^{K} w_i g(\mathbf{x}|\mu_i, \Sigma_i)}$

Maximization: $\forall j \in \{1, 2, ..., K\}$

$\mu_j = \frac{\sum_{n=1}^{N} p(j)\mathbf{x_n}}{\sum_{n=1}^{N} p(j)}$

$\Sigma_j = \frac{\sum_{n=1}^{N} p(j)(\mathbf{x_n} - \mu_j)(\mathbf{x_n} - \mu_j)^T}{\sum_{n=1}^{N} p(j)}$

$w_j = \frac{\sum_{n=1}^{N} p(j)}{N}$
end while

C. Self-Organizing Maps

A self-organizing map (SOM) is an artificial neural network that maps the input data to a smaller dimensional space (typically two-dimensional) such that topological properties are preserved. Self-organizing maps are different from other artificial neural networks as they apply competitive learning as opposed to error-correction learning, such as back propagation with gradient descent, and in the sense that they use a neighborhood function to preserve the topological properties of the input space. This means that the weights of the network are learned in such a way that the input vectors that are close to together are mapped to close output neurons. Thus a k-output SOM can be employed to perform clustering that produces k clusters. This idea has been shown to be applicable in spike sorting [12].

D. DBSCAN

DBSCAN stands for Density Based Spatial Clustering of Applications with Noise, this algorithm mainly uses density based clustering to separate local clusters from noise. It is also one of the better algorithms to handle arbitrary shape clusters. As against other algorithms, DBSCAN does not require apriori information about the number of clusters to be formed and can create its own separation if a certain set of parameters are well defined. It is also efficient in terms of handling outliers and its robustness in separating noise points from the actual data is well known [13].

In order for DBSCAN to work, it is important to define two essential parameters. Initially, we choose $MinPts$ that is greater than the number of features in the dataset. Secondly, a neighborhood function ϵ needs to be defined so that an optimal part of the data is covered.

$$\epsilon(p) = \{q|d(p, q) \leq \epsilon\} \tag{4}$$

where, p and q can be defined as density objects and which form a set of points called as core or non-core. The clusters formed contain one or more of these core points and non-core points form the boundaries of these clusters. These set of points thus define the clusters and outliers [13].

Even though DBSCAN has been very effective for a variety of clustering applications with noise, it is very sensitive to the settings of its parameters. Also, unlike k-means, the number of clusters and the outliers is determined by $MinPts$ and ϵ, which makes it very unintuitive and hard to use.

VI. Experimental Results

A. Clustering Accuracy

The clustering accuracy obtained on all the datasets are shown in Table I. Clearly, considering the performance over all the datasets, **Haar features + GMM performs the best**.

TABLE I: Accuracy (%) for each dataset using different methods (feature extraction + clustering). These results are obtained using the training datasets.

Method	Dataset			
	I	II	III	IV
PCA + k-Means	96.33	92.86	83.80	68.33
PCA + GMM	96.27	91.07	55.60	57.26
Haar features + k-Means	96.27	93.80	93.13	87.20
Haar features + DBSCAN	96.06	92.06	84.87	50.66
Haar features + GMM	**96.20**	**94.13**	**94.60**	**94.33**
Shape context features + k-Means	95.60	N/A	N/A	34.67

B. Time Complexity

In order to determine the computation time of our method, we averaged the computation time over 5 runs on a 2.7 Ghz i7 processor. All the programs were run on Matlab. The results are shown in Table II. It can be seen that GMM with k-means initialization converges faster than GMM with random initialization. It is also important to note that even though we use 48 wavelet features, the algorithm converges quickly in about 2.5 seconds for all the datasets.

TABLE II: Time complexity (in seconds) with different initializations for each dataset. It can be seen that GMM with k-means initialization converges faster than GMM with random initialization.

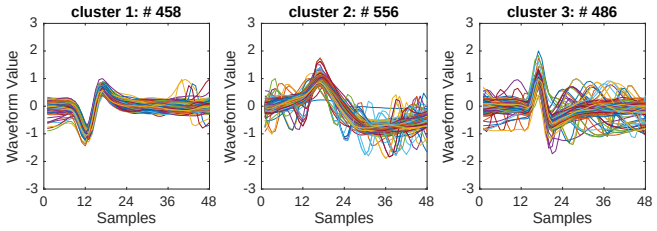
Method	Initialization	Dataset			
		I	II	III	IV
GMM	k-Means	2.05	2.26	2.07	2.55
GMM	Random	2.29	2.87	2.77	2.87
k-Means	Random	0.009	0.011	0.011	0.016

VII. Final Model Selection and Results

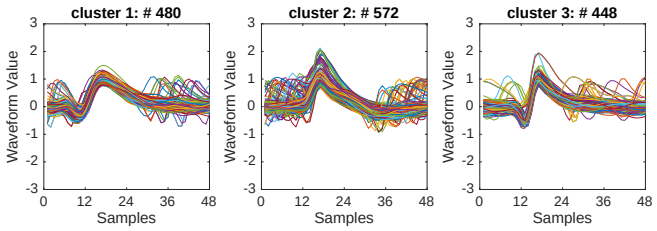
From the accuracy results on the training data, we determined that, over all the training datasets, the method that performed the best was fourth level **Haar wavelet features + GMM**. We found, through visual inspection and accuracy, that all the datasets have 3 clusters.

We have used 48 wavelet features for clustering. It is not easy to reduce the dimensionality of wavelet features unlike say, PCA features, since there is no sure way to find out a priori which features are important. Statistical tests such as Kolmogorov-Smirnoff and Lillifors can be performed to see which wavelet features have multi-modal distributions and choose those features. But, this may reduce the performance of clustering significantly since, when clustering is performed, all features are considered together, which is different from considering the features one at a time. We used all the features because the main focus of the algorithm was improve clustering accuracy.

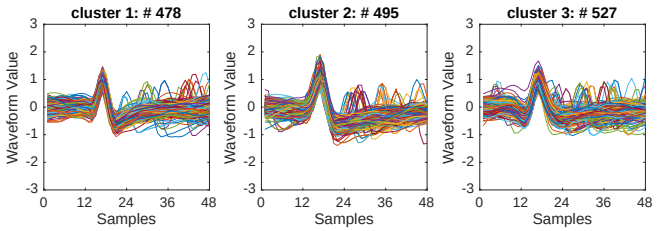
Also, we have shown that 48 features is *not* a large number for spike sorting since the algorithm converges in less than 2.6 seconds. There are also precedents in this regard. For example, in [14], the authors use 20 - 100 random projections for the MNIST dataset and cluster them using GMMs. The number of



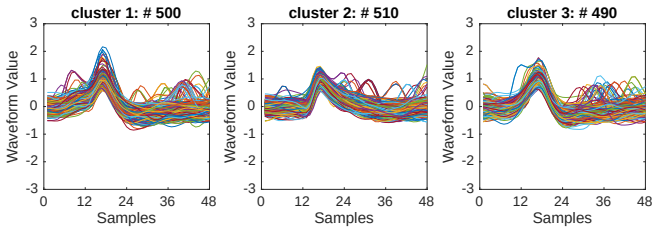
(a) Cluster assignments for Dataset-1



(b) Cluster assignments for Dataset-2

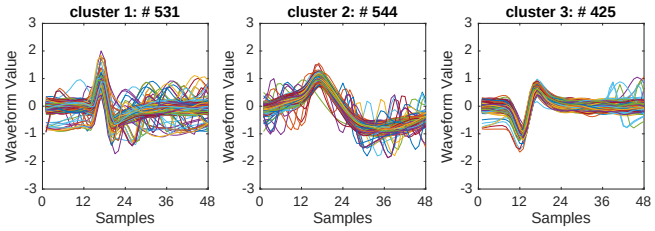


(c) Cluster assignments for Dataset-3

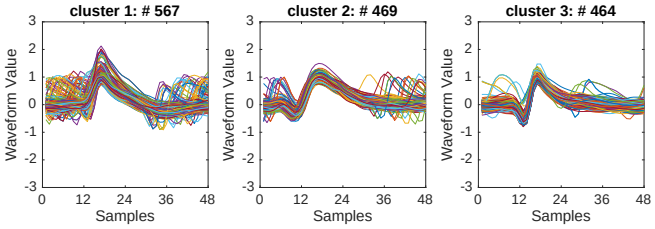


(d) Cluster assignments for Dataset-4

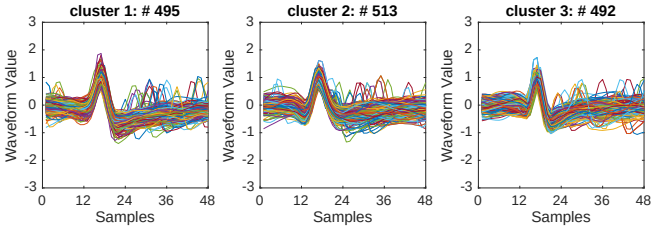
Fig. 5: Clustering results using Wavelet Feature Extraction and GMM clustering on training data. The number above each plot indicates the number of snippets assigned to that cluster.



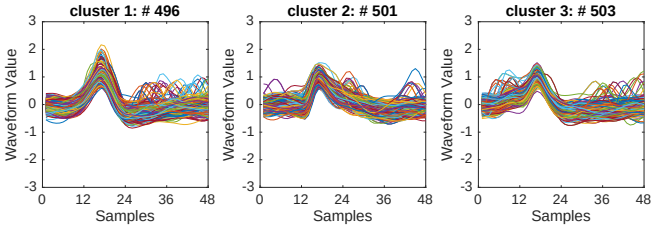
(a) Cluster assignments for Dataset-1



(b) Cluster assignments for Dataset-2



(c) Cluster assignments for Dataset-3



(d) Cluster assignments for Dataset-4

Fig. 6: Clustering results using Wavelet Feature Extraction and GMM clustering on testing data. The number above each plot indicates the number of snippets assigned to that cluster.

features will be an issue only when the datasets are extremely large – millions of datapoints. In such a case, we can select a subset of features using statistical tests mentioned above. The final results for the training and testing datasets are mentioned in Table III below.

TABLE III: Final clustering accuracy (%) for each dataset

Dataset	I	II	III	IV
Training Data	96.20	94.13	94.60	94.33
Testing Data	96.53	95.53	96.06	95.00

VIII. Conclusion

In this project, we have implemented a robust, efficient yet simple method for spike sorting. The datasets provided were fairly complex. For datasets 1 and 2 the clustering was simple and the final result plots show that the spikes can be well distinguished. The final result plots for datasets 3 and 4 indicate that at least two among the three shapes have striking similarity to each other. We have explored different feature extraction and clustering methods and compared their outcomes to choose the best performing one. Haar wavelet features with Gaussian mixture models for clustering yielded the best clustering accuracy.

Our algorithm is reliable and extremely fast considering the complexity of this task. With k-means initialization, the average run-time is less than 2.6 seconds for each dataset. Comparing the final result plots for the training and testing datasets, we observe that number of clusters assigned for both are close to each other. We have also achieved at least 94% or more accuracy in classification for all datasets. One drawback of GMMs is that we need to input the number of clusters. One possible way of determining this automatically from data is by using the Bayesian Information Criterion (BIC). However, this did not produce the desired results and will be part of future work.

References

[1] Quiroga, R. Quian, Zoltan Nadasdy, and Yoram Ben-Shaul. *Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering.* Neural computation 16.8 (2004): 1661-1687.

[2] Hernan Gonzalo Rey, Carlos Pedreira, Rodrigo Quian Quiroga. *Past, present and future of spike sorting techniques* Brain Research Bulletin, 2015

[3] Rey, Hernan Gonzalo, Carlos Pedreira, and Rodrigo Quian Quiroga. "Past, present and future of spike sorting techniques." Brain research bulletin (2015).

[4] Lewicki, Michael S. "A review of methods for spike sorting: the detection and classification of neural action potentials." Network: Computation in Neural Systems 9.4 (1998): R53-R78.

[5] Hulata, Eyal, 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 (2002): 1-12.

[6] Pouzat, Christophe, Ofer Mazor, and Gilles Laurent. "Using noise signature to optimize spike-sorting and to assess neuronal classification quality." Journal of neuroscience methods 122.1 (2002): 43-57.

[7] *Collaborative Research in Computational Neuroscience- Data sharing* , Link.

[8] Belongie, Serge, Jitendra Malik, and Jan Puzicha. *Shape matching and object recognition using shape contexts.* Pattern Analysis and Machine Intelligence, IEEE Transactions on 24.4 (2002): 509-522.

[9] Pouzat, Christophe, Ofer Mazor, and Gilles Laurent. *Using noise signature to optimize spike-sorting and to assess neuronal classification quality.* Journal of neuroscience methods 122.1 (2002): 43-57.

[10] Reynolds, Douglas. *Gaussian mixture models.* Encyclopedia of Biometrics. Springer US, 2009. 659-663.

[11] Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. *Maximum likelihood from incomplete data via the EM algorithm.* Journal of the royal statistical society. Series B (methodological) (1977): 1-38.

[12] Hermle, Thomas, Cornelius Schwarz, and Martin Bogdan. *Employing ICA and SOM for spike sorting of multielectrode recordings from CNS.* Journal of Physiology-Paris 98.4 (2004): 349-356.

[13] Ester, Martin and Kriegel, Hans-Peter and Sander, Jorg and Xu, Xiaowei *A density-based algorithm for discovering clusters in large spatial databases with noise.*, Kdd. Vol. 96. No. 34. 1996.

[14] Dasgupta, Sanjoy. "Experiments with random projection." Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 2000.