

A neural network-based spike discriminator

John S. Oghalai ^{a,*}, W. Nick Street ^b, William S. Rhode ^a

^a *Department of Neurophysiology, University of Wisconsin Medical School, Madison, WI 53706, USA;*

^b *Department of Computer Sciences, University of Wisconsin, Madison, WI 53706, USA*

Received 6 July 1993; revised 31 January 1994; accepted 8 March 1994

Abstract

A software routine to reconstruct individual spike trains from multi-neuron, single-channel extracellular recordings was designed. Using a neural network algorithm that automatically clusters and sorts the spikes, the only user input needed is the threshold level for spike detection and the number of unit types present in the recording. Adaptive features are included in the algorithm to allow for tracking of spike trains during periods of amplitude variation and also to identify noise spikes. The routine will operate on-line during extracellular studies of the cochlear nucleus in cats.

Key words: Multi-unit spike train; Spike discrimination; Extracellular recording; Cochlear nucleus; Neural network; ART-2

1. Introduction

In most central nervous system (CNS) extracellular recording experiments, a microelectrode is advanced into neural tissues with the hope of recording the spike train of an isolated neuron, in order to study the cell's physiologic characteristics. However, often two or more neural spike trains are superimposed on the recording. Small interneurons may be nearly impossible to isolate completely from larger cells nearby. In order to study the physiology of each of the individual cells, as well as the simultaneous multi-neuronal intercellular interactions, it is important to be able to separate out each of the units' spike trains. This is done with a spike discriminator.

Studies of the CNS using various spike discrimination methods have been quite successful in furthering the understanding of CNS functioning. Voigt and Young (1980) used the principal-component method to show inhibitory interactions in the dorsal cochlear nucleus. Gochin et al. (1989, 1990) further described excitatory interactions as well as stimulus-modulated

variations in interneuronal connectivity in the dorsal cochlear nucleus, using the same technique. Epping and Eggermont (1987) used principal-component analysis in recordings from the auditory midbrain of the grassfrog. Eggermont (1991) used template-matching software to study the intercellular interactions in the auditory midbrain of the leopard frog. Also using the template-matching technique, Reinis et al. (1992) diagrammed tentative neural network interactions in the visual cortex, inferior colliculus, rostral ventromedial medulla, and the ventrobasal complex of the thalamus. These studies demonstrate the usefulness of spike discrimination techniques in studying multiple neurons simultaneously.

There have been many different techniques used for spike train separation, including both hardware- and software-driven methods. Wheeler and Heetderks (1982) and Schmidt (1984a,b) reviewed these, and discussed the drawbacks of each technique. A frequent problem is the inability to separate spike trains on-line during an experiment. In addition, many spike separation techniques require considerable user input to define and track the different action potentials (although some work has been done to fully automate the spike separation process (Salganicoff et al., 1988; Sarna et al., 1988)). In some situations, there is a long set-up or 'training' period between the time when neuronal units are first found and when collection of actual data

* Correspondence: John S. Oghalai, Department of Otolaryngology, Baylor College of Medicine, 1 Baylor Plaza (NA102), Houston, TX 77030, USA. E-mail: Oghalai@Neurophys.wisc.edu; Telephone: (608) 262-5896; FAX (608) 265-3500.

begins. We have noticed this to be a factor with both template-matching methods (Schmidt, 1984b; Salganicoff et al., 1988; Sarna et al., 1988; Bergman and DeLong, 1992; Jansen and Maat, 1992) and traditional neural network methods (Jansen, 1990; Yamada et al., 1992).

A traditional neural network algorithmic approach to spike separation first involves training the network from a bank of wave forms which are already separated into correct groups. A neural network has the ability to learn what features of a spike wave form best distinguish it apart from spikes in the other previously defined groups. Once the slower training phase is completed, the algorithm can then quickly classify the rest of the incoming data based on those clusters. The traditional neural network technique is based on a supervised learning algorithm, using back-propagation (Rumelhart et al., 1986).

An off-line, traditional neural network-based spike discrimination algorithm has been described, and was tested on multi-unit extracellular in vivo recordings from the snail *Lymnaea stagnalis* (Jansen, 1990). After first collecting all the raw data and storing it, a bank containing samples of all of the different spike types was defined. This was done manually via a user interface. These wave forms were then used to train the neural network. Then, all the raw data was processed using the trained neural network.

Yamada et al. (1992) also used the traditional neural network spike discrimination routine for use in optical recordings (using a membrane voltage-sensitive dye) of multi-unit action potentials. In optical recordings, the shapes of different units are almost always the same, only varying in peak amplitude. This is much different than for extracellular recordings, and it allowed them to train the neural network only once, and then to use the same learning pattern for all their experiments. They did not have to implement a time-consuming training phase before each multi-unit recording was analyzed. Nevertheless, their algorithm required more user involvement than they wanted in order to sort the many unclassified spikes.

For our application of recording from the cochlear nucleus in cats, several features were essential in the design of the spike discrimination routine. Most important, it had to have accuracy that was at least equal to currently developed spike discrimination techniques, even if the spike amplitudes were nearly the same. It also had to be able to handle noise spikes appropriately. Second, it had to be simple to use, with minimal set-up time by the user. This would be especially important during situations of poor unit stability, which we commonly encounter in the cochlear nucleus. Third, it had to run in real-time, sorting the spikes as they were collected, so that plots of the individual spike time data could be displayed on-line. Fourth, the rou-

tine needed to be able to track minor shape changes in a unit with time (corresponding to a slow drift of the cell body away from the electrode tip), and it should also be able to follow sudden amplitude shifts in a wave form (which is often due to breathing artifact). Finally, we wanted it to be flexible enough to work with as many simultaneous units as needed.

The algorithm we used in our software was based on a neural network recognition system using adaptive resonance theory, called ART-2, developed by Carpenter and Grossberg (1987, 1988). It has the capability to forego the training phase where clustering occurs, and to cluster and sort incoming spikes on-line, while data collection is occurring. It does this by tailoring its decision boundaries adaptively to the particular recording situation, weighting certain features over others. This algorithm has the flexibility to form new clusters to handle previously unseen wave forms, as well as to adaptively reformulate existing cluster definitions as spike wave shapes change. New clusters can be allocated as needed and processed simultaneously. Also, every spike is normalized before clustering occurs, so that any amplitude variability between spikes coming from the same neuron will not affect the results. This allows for excellent tracking of neural spikes in difficult recording environments. We feel that our spike discriminator software will allow us to confidently record multiple units simultaneously.

2. Methods

2.1. Spike discriminator

2.1.1. Hardware

The data used for testing the spike discriminator was recorded from the cochlear nucleus of cats. Acoustical stimulation at the external auditory meatus was controlled by a VAXstation 3200 computer (Digital Equipment, Maynard, MA) and consisted of tone pips of variable frequency, volume, and duration. Extracellular multiple-unit recordings were made from glass micropipettes with impedances between 5 and 15 M Ω , when measured at 1 kHz. After 1000 \times amplification, the signal was filtered with a second-order Butterworth high-pass filter set at 600 Hz, and a second-order Butterworth low-pass filter set at 5000 Hz (A-200 multichannel amplifier, M. Walsh Electronics, 1322 W. Windsor Dr., San Dimas, CA 91773). The signal was then passed through a variable gain amplifier (Model 113 Pre-Amp, Princeton Applied Research) to achieve a full scale swing of ± 10 V, before being digitized by a 12-bit analog-to-digital (A/D) converter (DEC model ADQ-32) at a sampling rate of 40 kHz. We did not test our software using lower sampling rates.

2.1.2. Digital sampling

The A/D converter starts at the beginning of a stimulus and stops before the next stimulus starts. The digitized data collected during this period is stored in a memory buffer. A double buffered system allows one buffer to be processed while the next buffer is filling; this allows for nearly continuous sampling. Since the spikes are not sorted as soon as they occur, the software is running in 'pseudo real-time'. Ideally, one buffer has completed processing before the next buffer is collected and ready for processing, so the user is not aware of any decline in speed. The speed of the software is described in the Results section.

2.1.3. Threshold level detection

The first step in analyzing the digitized data is to set a threshold level above the baseline noise level where any rising voltage through the threshold triggers the detection of a spike. The reason threshold detection was implemented was to reduce the amount of noise that needed to be processed by the neural net, although there may still be a few noise spikes rising above the threshold level (Abeles and Goldstein, 1977). These are filtered out later.

The time at which the rising voltage crosses the threshold level is recorded as the spike time. After this, the local maximum is found from the next 12 data points (0.3 ms); this point is the spike's peak amplitude. From this peak, the 10 previous data points and the next 30 data points are extracted out from the buffer and represent a picture of the spike (40 data points or 1 ms). Because each spike wave form is aligned at its maximum amplitude, the height of the threshold level will have no impact on what is used for processing.

2.1.4. Spike analysis

The 40-point spike picture array is then sent to the neural network algorithm for analysis (described later), and a cluster number is returned. Then, the time of the last spike in that same cluster is compared with the current spike time. If the difference is less than 0.4 ms, the current spike is thrown out. This process is done to eliminate noise spikes which can cause 'spike doubling'; the explanation for this is in the Discussion section. There is no way physiologically for the same cell to discharge twice in this time period, due to the neuronal refractory period of approximately 1 ms in the cochlear nucleus.

2.1.5. Cluster reduction

After parsing through the entire data buffer and processing all the detected spikes into clusters, the total number of clusters formed by the neural network is reduced to the number of different spike types in the recording, selected previously by the user (described

later). As an example, if the neural network created 10 different clusters, but the user had entered that there were really only 2 different neural spike types present, the 2 clusters with the highest number of spikes would be kept, with the other 8 clusters being grouped together and called the 'noise' cluster.

2.1.6. Data storage

The list of spike times and their corresponding cluster numbers is kept in a large array. Simple plots of spike time data, such as a post-stimulus-time (PST) histogram for each unit, are done on-line during the experiment, while more complex plots, such as cross-correlation histograms, are done at the end of the experiment.

2.2. Neural network algorithm

The clustering was performed using the continuous version of Carpenter and Grossberg's (1987, 1988) ART-2. There are several different versions of the ART algorithm, including ART-2, ART-2A, and Fuzzy ART (Carpenter et al., 1991; personal communications) (Note: the ART-2 algorithm that we used is available by anonymous ftp to the machine cns.bu.edu, in the file pub/art2.shar). No separate training phase is needed for this algorithm, since it forms clusters automatically as spikes are collected. Each neural unit represents a cluster of spikes, and can be viewed as a prototype or average of the wave forms that were used to construct it (see Fig. 1). An incoming digitized, 40-point spike is first normalized to a unit vector and then compared to each already existing cluster. If a close match is found, the spike is added to that cluster

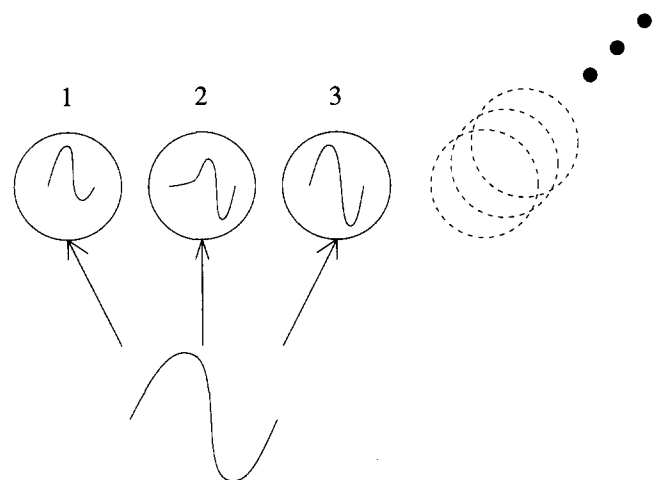


Fig. 1. The ART-2 algorithm applied to wave form classification. The scaled incoming spike is compared to each existing cluster. In this case, the wave form might be classified as belonging to cluster 3, and the prototype of unit 3 would be adjusted to more closely resemble the input. If no close match is found, a new unit (from a potentially infinite supply) is allocated.

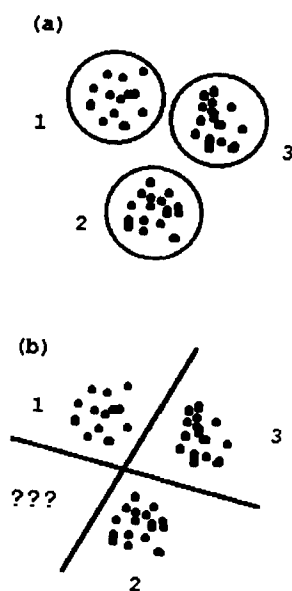


Fig. 2. Method of clustering for ART-2 versus traditional neural networks. This figure shows a 2-dimensional representation of how clustering occurs. In both of these examples, there are 3 different clusters. The dots indicate individual action potentials. In our program, these are really 40-dimensional vectors (since each spike wave form consists of 40 data points). Part (a) shows how the ART-2 algorithm places a boundary around action potentials that are close together. Any new point that falls outside of each of the 3 existing clusters, would cause a fourth boundary to be formed to accommodate it. This is how ART-2 handles noise. Part (b) shows how traditional neural networks divide up the space by placing hyperplanes between clusters. Once trained to this pattern, every point in space will be classified to 1 of 4 regions: either 1 of the 3 clusters or the indeterminate area.

and the prototype is modified, moving closer to the shape of the new spike. This is analogous to the learning phase of traditional neural networks. Other clusters are not affected; hence, this procedure is a type of 'competitive' learning, in which only the winning cluster is modified by a particular wave form. This allows the algorithm to track slow changes in spike shapes.

If the spike is not sufficiently close to any of the existing clusters, it forms a new cluster. The cluster boundaries are regulated by a vigilance parameter, ρ . The value of ρ needs to be modified to handle different spike train characteristics. For instance, a particularly noisy set of spikes would indicate the need for a lower vigilance (hence, a larger boundary around the cluster), since signals from the same neuron might vary significantly.

ART-2 differs from traditional neural networks in the way that it partitions the input space for classification. The prototypes constructed by ART-2 represent a group of input vectors which are 'close' to one another in 40-dimensional space, as shown in Fig. 2a. The boundaries of the clusters are shown as circles in the figure; the extent of the boundary is controlled by the

vigilance parameter, ρ . Traditional neural networks divide the input space by placing separating hyperplanes between the various classes, as shown by the lines in Fig. 2b. The number of these planes must be fixed before training and, once trained, the orientation of these planes cannot be changed. The ART-2 algorithm permits the addition of new boundaries as needed. Hence, wave forms never seen previously can be categorized by ART-2. Conversely, the traditional approach can only categorize wave forms based on the previously defined hyperplanes.

In order to make the selection of ρ automatic in our algorithm, two different ρ values are tried on the very first data buffer processed, to decide which one works better for the situation. One value tends to work better in low-noise environments, while the other works better with higher noise levels. The ρ value that forms the most clusters with a significant number of spikes (at least 5% of the total number of spikes) is used for the remainder of the data collection. The setting of the vigilance parameter could be considered 'training'. It does not, however, involve collecting several hundred spikes and iterating through them to form cluster templates, before any actual data begins to be collected. Selection of the vigilance parameter happens without any user input.

The second controlling parameter on the neural network is a noise threshold, θ . After the individual spike has been normalized, the algorithm disregards all activity beneath this threshold. So, this low-amplitude variation is considered to be noise, and is not used in the classification and learning steps. We always use the same noise threshold, set at a low level, so as to use almost all of the 40-point wave form in classification. Note that both neural network parameters, ρ and θ , are set without user input.

2.3. User-set parameters

Before the spike discriminator can begin to collect data during an experiment, there are 2 parameters that do need to be set by the user. The first is the amplitude trigger level. This is set by collecting one trial buffer of data, plotting the recording trace on the computer screen, and allowing the user to set the trigger level by moving a superimposed cursor.

The second parameter is the number of neuronal spike types present in the recording. Before requesting this information, the routine analyzes the first data buffer and prints out both the number of clusters the neural network formed and the number of spikes in each cluster. This data is helpful to the user in deciding how many spike types are present because, in most cases, there will only be a few clusters with many spikes in them, usually 2–3, that probably represent different units. Also, there will be several more clusters with

only a few spikes in each, usually 2–15 depending on the noise margin, that probably represent noise spikes. Since noise spike shapes tend to vary more than neuronal spike shapes, the neural network ‘spreads out’ the noise into many small clusters. So, both the number of clusters with a significant number of spikes and visual inspection of the data tracing can be used to estimate the number of neuronal units that are represented in the recording.

Once the user has selected the trigger level and entered the number of units, the program is fully automated. This user-interface is designed to take about 30 s to set up once a signal is found.

2.4. Data simulation routine

A routine was written to randomly generate simulated data for use in testing the spike discriminator objectively against different variable factors (spike shape, spike amplitude, and noise level). In order to do this, several representative spike wave forms were extracted from in vivo recordings that had been sampled at 40 kHz and stored on disk. Each extracted spike shape consisted of 40 points (1 ms). The spike peak amplitudes were then adjusted to user specification by scalar multiplication of the entire 40-point shape file.

The data simulation routine, using the extracted spikes, positioned copies of the wave forms randomly within a blank 500 ms buffer, making sure that none of

the spikes overlapped each other. Simultaneous spikes were not considered during initial testing since we wanted to concentrate on our main objective, shape discrimination. The spike rates were set at 200 spikes per second for each spike type. Randomly generated gaussian noise, filtered from 600 to 5000 Hz with a second-order digital band-pass filter, with a user selected root-mean-squared (RMS) value was added to the entire 500 ms buffer equally.

The buffer was then used by the spike discriminator program in place of actual data, for testing purposes. This method allowed us to compare the results of the spike discriminator using different spike shapes, spike amplitudes, and noise levels.

2.6. Calculation of accuracy

The output file of spike times with their corresponding cluster numbers from the spike discriminator was compared with the ‘answers’ (known spike times) from the data simulation routine. If the difference between any discriminator output spike time and any answer spike time was less than 0.3 ms, a match was made. Any spikes found in the discriminator output file, but not in the answer file, were called noise spikes. Noise spikes occurred randomly based on the noise level added to the generated data.

Table 1 demonstrates how the accuracy of the spike discriminator was calculated, using the generated 2-unit

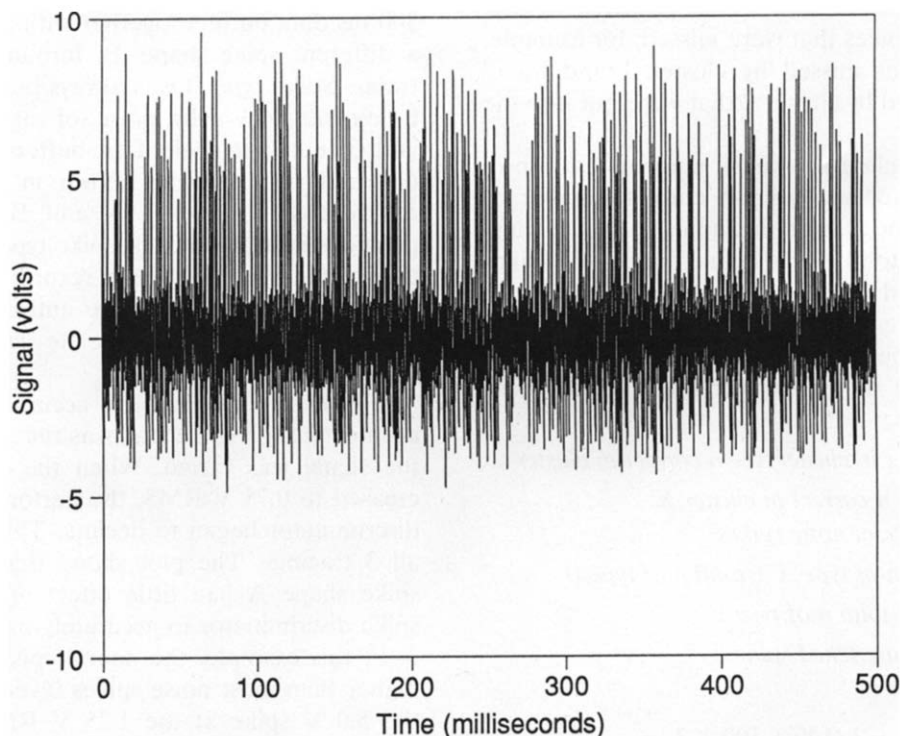


Fig. 3. Randomly generated 2-unit data. This is an example of a 500 ms data buffer generated in software. Two different spike types are present. The noise level is 0.75 V RMS. This data is analyzed in Table 1.

Table 1
Spike discriminator accuracy calculation

	Cluster 1	Cluster 2	Noise	Answer totals
Spike type A	73	22	5	100
Spike type B	0	93	7	100
Unmatched spikes	0	0	2	2
Cluster totals	73	115	17	

This is the output from our spike discrimination software after comparing its results (cluster totals) with the answer file (answer totals) for the data in Fig. 3. There are 2 different spike types (A and B); there are 100 of each in the data. The cluster 1, cluster 2, and noise columns show the breakdown of what types of spikes are in each cluster. There were 2 spikes picked up by the discriminator that were not neural spikes, and so they were called unmatched spikes. These are true noise spikes. Please see the text for a complete explanation of the breakdown.

data shown in Fig. 3. This was a particularly tough case with many noise spikes and poor spike discriminability. Cluster 1 represents type-A spikes, and cluster 2 represents type-B spikes. Our goal was to formulate a single, overall value that represented the accuracy of the spike train separation. The accuracy must take into account the true positives, true negatives, false positives, and false negatives.

A true positive is a correctly clustered spike; in the example, there were 73 in cluster 1, and 93 in cluster 2. A true negative is a noise spike that was correctly sorted into the noise cluster. There were 2 in the example. A false positive is a spike put into a group that it did not belong to. In the example, the 22 type-A spikes sorted into cluster 2 were false positives. A false negative includes spikes that were missed; for example, the 5 type-A spikes missed by cluster 1 and the 7 type-B spikes missed by cluster 2 that were put into the noise cluster.

One way to calculate an overall number that represents accuracy is to sum up the correct number of spikes in each of the clusters including the noise cluster, and divide the total by the actual number of spikes that were supposed to be present, according to the answer file. X is the number of clusters selected by the user; Y is the number of spike types actually present in the data according to the answer file. Ideally, these should be the same.

$$\begin{aligned}
 \text{Accuracy} = & n \text{ correct in cluster 1} + n \text{ correct in cluster 2} \\
 & + \dots + n \text{ correct in cluster } X \\
 & + n \text{ correct noise spikes} \\
 & / \text{total } n \text{ of type A} + \text{total } n \text{ of type B} \\
 & + \dots + \text{total } n \text{ of type Y} \\
 & + n \text{ unmatched spikes.}
 \end{aligned}$$

In the example,

$$\begin{aligned}
 \text{Accuracy} &= 73 + 93 + 2/100 + 100 + 2 \\
 &= 168/202 = 83\%
 \end{aligned}$$

This method of calculating accuracy includes both the number of spikes that are supposed to be picked up (the answer totals) as well as the actual number of spikes that the discriminator found (the cluster totals). Note that as the number of correctly sorted noise spikes (the true negatives) goes up, the accuracy goes up as well, even though the discrimination of the neural spikes (the true positives) might be unchanged. This effect is minimized by keeping the threshold well above the noise margin so that the number of triggered noise spikes is small.

2.7. Technical note

All the software was compiled and run on both a VAXstation 3200 computer and a VAXstation 4000-60 computer, using the VMS operating system. The main body of the software is written in FORTRAN, while the neural network algorithm is written in C.

3. Results

3.1. Accuracy in sorting two spike types

The spike discriminator was first tested using generated data containing 2 different spike shapes (see Fig. 4). The first spike shape, A, was scaled to 3 different peak amplitudes: 10 V, 7.5 V, and 5 V. Each of these 3 amplitude shapes was distributed randomly into three 500 ms data buffers together with an equal number of a different spike shape, B, forming 3 different spike trains. Spike type B was always used at a peak amplitude of 7.5 V. Finally, noise (of varying amplitude) was added equally to each of the buffers. This created the 3 different data tracings shown in Figs. 4a–c. The 2 different spike shapes, A and B, are in the same positions in all 3 tracings; spike type A is slightly wider than spike type B. These recordings each show the same 10 ms segment of the entire 500 ms generated data buffer, and only show the data using a 0.125 V RMS noise level.

Fig. 4d is a graph of the accuracy results of sorting each of these 3 spike trains as the noise level added to the signal was varied. When the noise level was increased to 0.75 V RMS, the performance of the spike discriminator began to decline. This was the same for all 3 tracings. The plot shows that the amplitude of spike shape A had little effect on the ability of the spike discriminator to accurately sort the spikes.

In this example, the neural spikes were seen to be higher than most noise spikes (even in the worst case, the 5.0 V spike at the 1.25 V RMS noise level), so noise spike detections by the threshold detector were not the reason for lower performance. A signal-to-noise

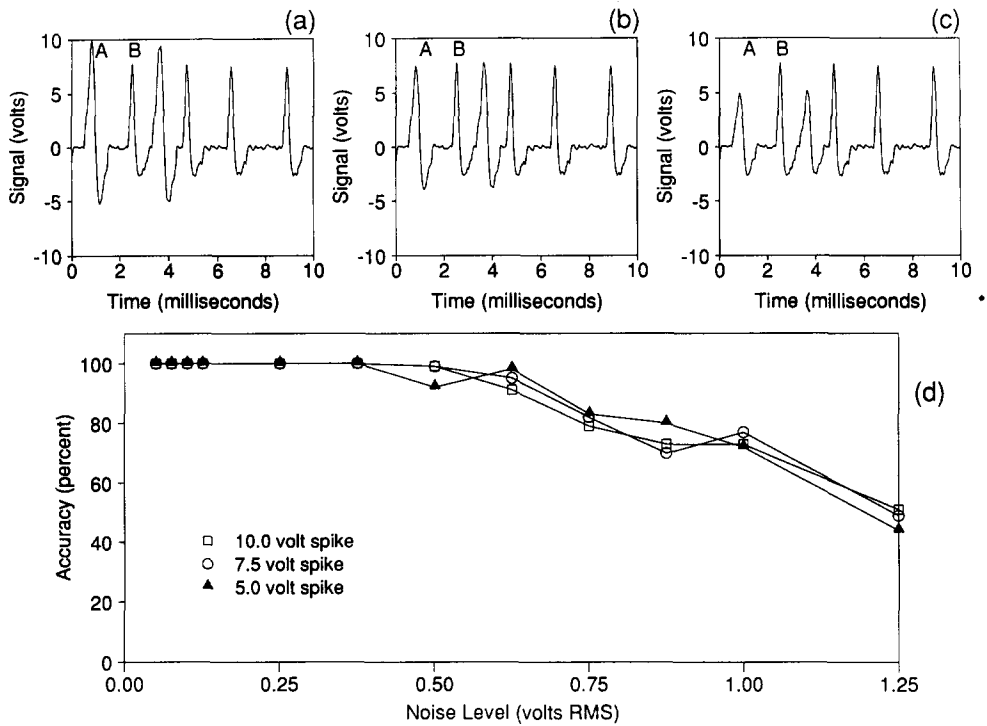


Fig. 4. Example of generated 2-unit data. Ten millisecond extractions of the 3 data tracings are shown in parts (a), (b), and (c). Spike type B was kept at a constant amplitude, while spike type A was scaled to 3 different amplitudes: 10 V, 7.5 V, and 5 V. The noise level was 0.125 V RMS in these figures. Part (d) shows the accuracy of the spike discriminator while separating the units for each of the 3 recordings, with increasing noise levels. Note that the accuracy is quite good until the decline beginning at a noise level of 0.75 V RMS. There is no difference in discriminability between the 3 different amplitudes of spike type A, at any noise level.

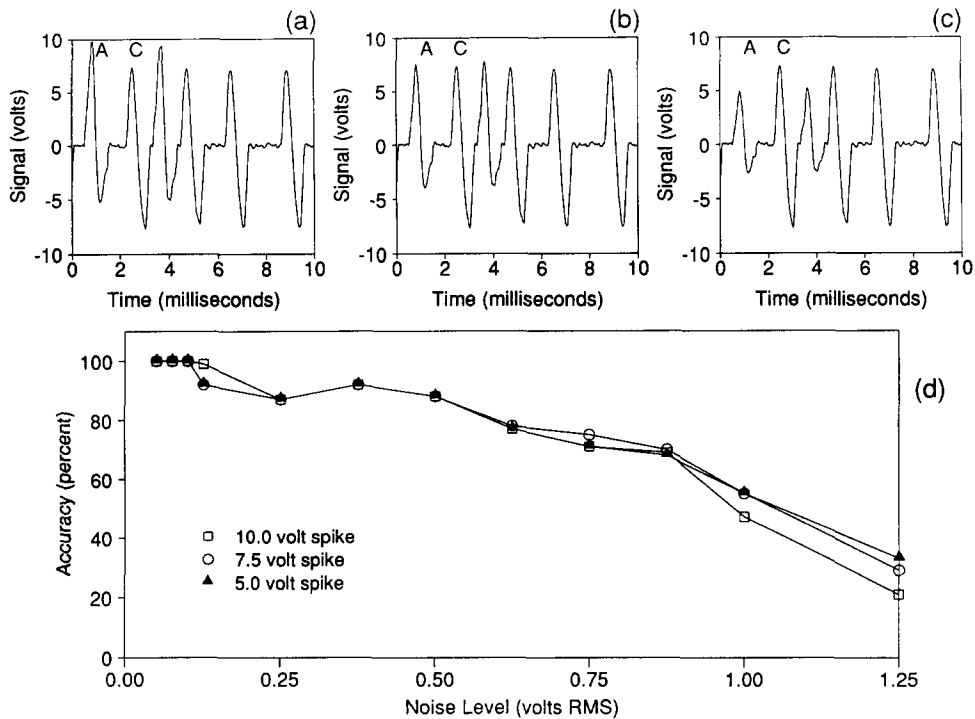


Fig. 5. Another example of generated 2-unit data. This time, the 3 different amplitude type-A spikes were mixed with a different spike wave form, labeled type-C spikes. The noise level was 0.125 V RMS in parts (a), (b), and (c). Part (d) shows the accuracy results as the noise level was increased. Again, note the similar decline in accuracy for all 3 tracings, in this instance beginning at a noise level of 0.25 V RMS.

ratio (SNR) can be calculated for this data by dividing the spike's peak amplitude by the RMS value of the noise. This is not a true SNR, for which the RMS value of the signal would need to be used; but, for neuronal spike detection and discrimination, this is a more useful value, since the first step in the spike identification process is threshold detection. Once spike amplitudes rise above the noise margin, they can start to become identified and sorted. In this and the other examples where there are different spike peak amplitudes, the 7.5 V spike was always used in the calculation of the SNR, although as shown in results section, the 5.0 V spike gave identical results, and would calculate out to a lower SNR.

The drop-off in accuracy in Fig. 4d occurred at a SNR of 10 (noise level of 0.75 V RMS). This would be a high SNR for a decline in accuracy in a single-unit recording, but not in a multi-unit recording. The cause of the drop was because the noise margin became larger than the difference between the 2 spike shapes, so the discriminator had difficulty distinguishing between the 2 different spike types.

In another example of neural spike discriminability, the same spike shape, A, was again used at each of the 3 peak amplitudes versus a different spike shape, C, used only at 7.5 V. Fig. 5a–c show 10 ms segments from each of the 3 generated data tracings. Spike C

appears to be about the same width as spike A, but has a much deeper trough. Again, these tracings are displayed using a noise level of 0.125 V RMS added to the data.

Fig. 5d, like Fig. 4d, is a graph of the accuracy results from the spike discriminator after sorting each of these 3 tracings. All 3 tracings have nearly equivalent results as the noise level is increased, showing that spike amplitude is not a factor in the ability of the software to cluster effectively. In this example, the noise has an effect on accuracy at a lower level (about 0.25 V RMS, or a SNR of 30, compared with a SNR of 10 in the previous example). Therefore, at a lower noise level, spike types A and C were undifferentiable, while types A and B were still quite separable.

Also of interest in this plot is the slight rise in accuracy around noise levels of 0.375 V RMS; the reason for this is unknown. The neural network software builds a new learning pathway with each different data tracing used, and it is possible that at noise levels just below this point, the neural network formed stricter clustering requirements than at noise levels just above it. This could cause some neural spike wave forms with a particularly bad noise spike superimposed on it to get mistakenly clustered into the noise group. Since the neural network 'self-learns' by using repetitive algorithms without programmer input, it is extremely diffi-

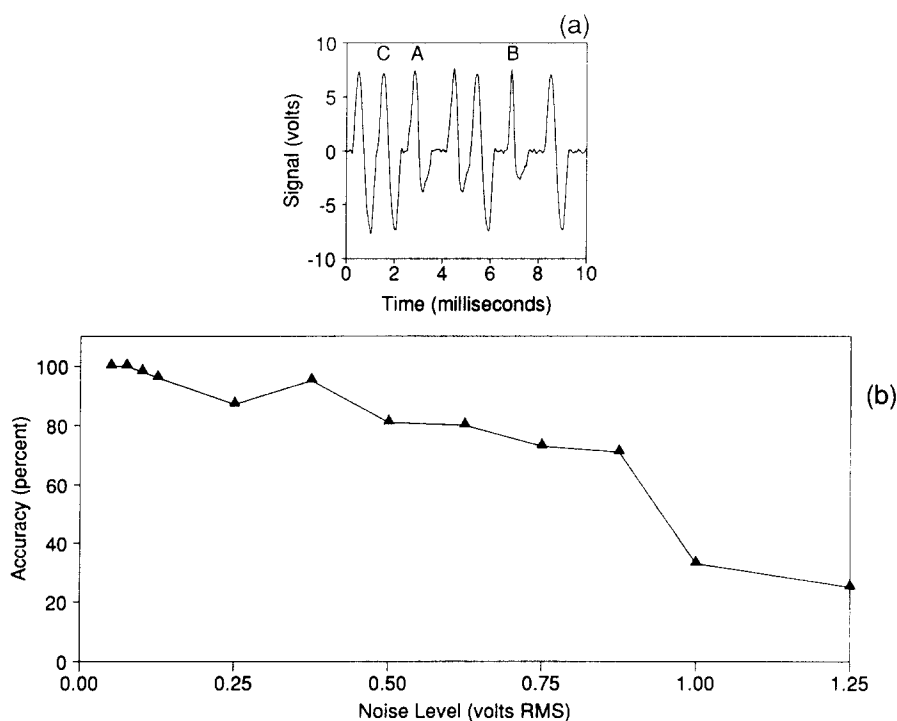


Fig. 6. Example of generated 3-unit data. Three different spike wave forms, A, B, and C, all at 7.5 V amplitudes, were equally distributed within the 500 ms data buffer. Part (a) shows a 10 ms segment of the buffer at a noise level of 0.125 V RMS. Part (b) shows the accuracy results as the noise level was increased.

cult to analyze what features of the neural spike wave forms it considers important, so the cause for this rise in performance is not known.

3.2. Accuracy in sorting three spike types

Fig. 6a is a generated data tracing of the 3 different spike shapes, A, B, and C, all with similar amplitudes (7.5 V), and at a low noise level (0.125 V RMS). Fig. 6b shows the performance of the spike discriminator while clustering the 3 spikes as the noise level increases. Similar to the previous example, there was a rise in accuracy at 0.375 V RMS noise. Presumably, these rises occurred for the same reason.

3.3. Amplitude Tracking

An important feature of this spike discriminator is the ability of the algorithm to track a spike shape as its amplitude changes, especially if it changes rapidly. To test this, a single spike shape, A, was scaled to amplitudes of 5 V to 10 V in steps of 0.5 V. Then, eleven generated data tracings were created, using the 7.5 V spike and each of the scaled versions of the same spike. Each of the spike trains was then run on the spike discriminator; this was done at 3 different noise levels:

0.125, 0.500, and 0.750 V RMS. Fig. 7a–c show 3 of the 11 tracings of the generated data buffers, with the 5, 7.5, and 10 V spikes mixed with the 7.5 V spike, using the 0.125 V RMS noise level. Obviously, in Fig. 7b both spikes are exactly the same and could never be distinguished.

In Fig. 7d, the performance of the spike discriminator is shown, comparing the 7.5 V spike with each of the different amplitude spikes. It is important to remember that while the spike amplitudes are different, the spike shapes are the same, so the point of this plot is to show that different amplitude spikes with the same shape get clustered into the same group. The number of spike types present in the data was entered as 2, so that the algorithm would try to separate the 2 different amplitude spikes into 2 different clusters. When the algorithm sorted both spikes to the same cluster, the accuracy result was 50%, since it could only classify the spikes into 1 cluster, not 2 as had been requested. When sorted into different clusters, the accuracy was 100%.

This plot shows that a wide range of amplitude variability can be present without causing separation of the different amplitude spikes into different clusters (i.e., a calculated accuracy of 50%, but actually the result was as desired). A spike could vary anywhere

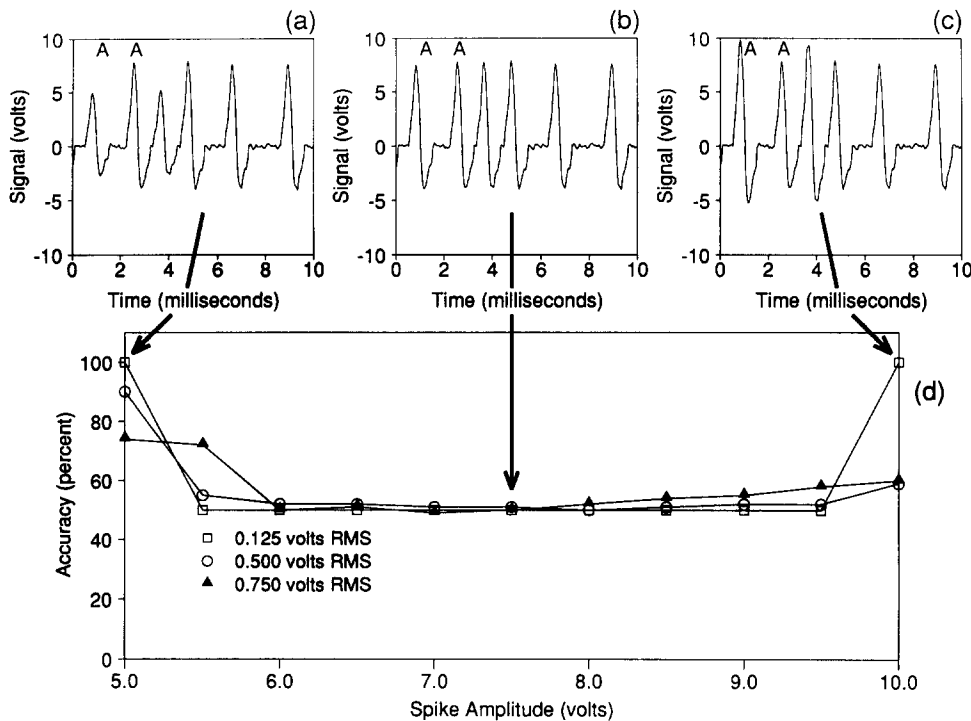


Fig. 7. Discriminability of the same wave form, scaled to different amplitudes. One neural spike wave form, type A, was scaled from an amplitude of 5 V to 10 V in steps of 0.5 V, and each of them randomly distributed in a buffer with the 7.5 V type-A spike to prove that no matter what the amplitude difference, both spikes would be put into the same cluster. This was done at 3 different noise levels: 0.125, 0.5, and 0.75 V RMS. Parts (a), (b), and (c) show 3 of the buffers using the 5, 7.5, and 10 V spikes versus the 7.5 V spike, all at the 0.125 V RMS noise level. Part (d) displays the accuracy results at each of the 3 noise levels. Note that from 6.0 to 9.5 V, the accuracy was 50%, meaning that both spike types were sorted to the same cluster.

from 73 to 127% of its peak amplitude without any misclassification. This test is more difficult than a test of tracking slow changes in spike amplitude (i.e., over minutes) because, in that situation, both the cluster learning process and the spike normalization process are occurring (as opposed to only the normalization process, as in this case).

At both the low and high extremes, the spike discriminator did sort the different amplitude spikes into their own clusters. Small differences between the spikes, possibly formed during the normalization process, could cause fractionation of the spikes into 2 different clusters.

3.4. Discrimination of real data recordings

Next, 2 *in vivo* recordings were analyzed. The first tracing is shown in Fig. 8. The noise level is relatively low at 0.09 V RMS. The threshold level was user-set to midway between the noise margin and the peak of the small spike. Accuracy was determined by comparing the spike times from the discriminator output with the actual data tracing. The spike discriminator sorted the spikes with 100% accuracy forming 2 clusters with 16 and 9 spikes correspondingly. This was a relatively easy example.

The second *in vivo* example was more difficult, as seen in Fig. 9, since the noise floor was higher (0.343 V RMS). Once again, the trigger level was set about midway between the noise margin and the peak of the small spike. The results of the clustering showed 53 total spikes, with 2 clusters of 32 and 11 spikes, and a noise cluster of 10 spikes. The accuracy, calculated by

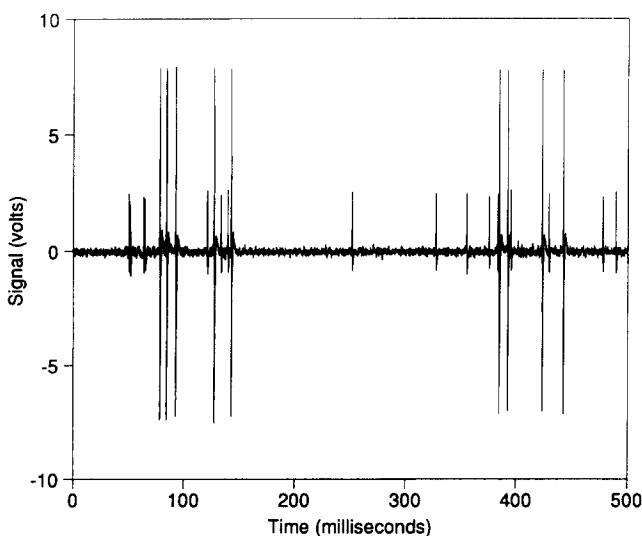


Fig. 8. Actual 2-unit data recorded from cat cochlear nucleus. The noise level was measured to be 0.09 V RMS. The accuracy of the spike discriminator on this data was 100%.

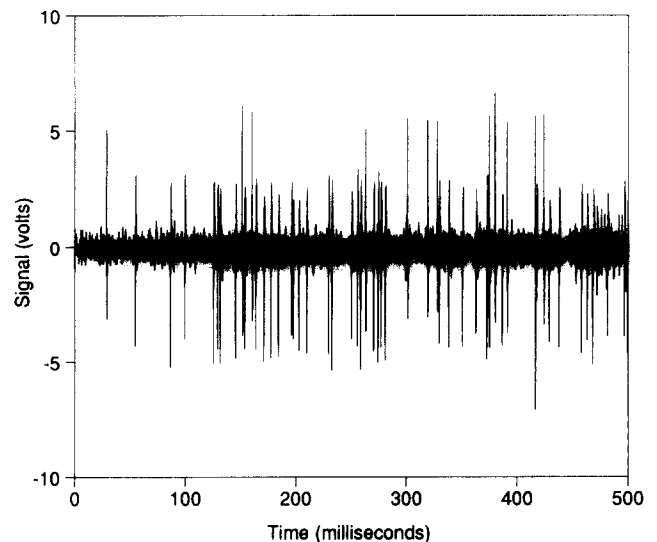


Fig. 9. Another example of actual 2-unit data recorded from cat cochlear nucleus. The noise level was measured to be 0.343 V RMS. The accuracy of the spike discriminator on this data was 86%.

comparing the discriminator output spike times with the actual data tracing, was determined manually to be 86%. The spike discriminator mistakenly classified some of the small neuronal spikes and large neuronal spikes as noise spikes, causing the mild loss of performance.

3.5. Performance testing

Finally, the speed of the spike discriminator algorithm was determined by finding the time it took from when a 500 ms data buffer was ready to be processed, until the time all the spikes were clustered and the number of clusters was reduced (as explained in the Methods section). The buffer was made up of generated data, containing 400 spikes, one-half type A, and one-half type B. Both had amplitudes of 7.5 V. Two different noise levels were used: 0.05 V RMS and 0.625 V RMS. Also, both of these trials were run on two different computers, a VAXstation 4000-60 and a VAXstation 3200.

On the VAXstation 4000-60, the low-noise data was processed in 9.16 s, and the high-noise data in 14.14 s. This was calculated to give a processing time of 44 spikes/s and 28 spikes/s correspondingly. On the VAXstation 3200, the low-noise data was processed in 38.38 s, and the high-noise data in 58.46 s. This was calculated to give processing times of 10 spikes/s and 7 spikes/s. As seen, the older VAXstation 3200 was about a fourth as fast as the newer model. The data shows that as noise levels increase, the performance decreases.

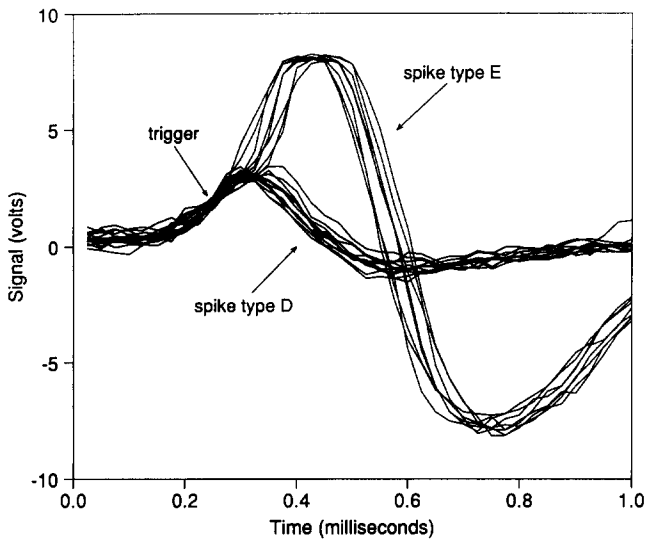


Fig. 10. Typical spike wave forms sent to the neural network for classification. This plot shows several overlapping 1 ms spike wave forms, all triggered at the same point. There are 2 spike types present, D and E. Each of these spike wave forms are classified by the neural network algorithm, and are easily sorted to their correct clusters.

4. Discussion

4.1. Errors in spike discrimination

Before a discussion of the results, it is important to understand the theory of where errors in spike discrim-

ination originate. In all the following examples, the errors come from actual recordings and were discovered during testing. Fig. 10 shows several 1 ms spikes seen by the spike discriminator after threshold detection at the trigger level. In this data, two obviously different shapes were present, D and E, and all of these spikes could be discriminated easily.

Fig. 11 shows what happened to 1 type-D spike with a small downgoing noise spike superimposed upon it. The spike was first detected at trigger 1 and sorted into the correct cluster containing all the other type-D spikes. Then, the negative noise spike dip caused the signal to go below the threshold and then back above it again, causing a second detection to occur at trigger 2. A second 40 point extraction of the spike was taken and was close enough in shape, although shifted in time slightly, so that it was sorted into the same cluster, causing a false positive (i.e., a noise spike being sorted into a neuronal spike cluster). We call this source of error 'spike doubling'; it was eliminated in our current algorithm by not allowing 2 detected spikes within 0.4 ms of each other into the same group.

In spike doubling, often the second triggering caused by a noise spike is delayed in time enough so that the neural network decides to form a new cluster to accommodate the new spike. This is how many noise clusters with very few spikes in each are formed. When this happens, a noise spike is correctly sorted into a noise cluster, and a true negative occurs.

A similar situation can occur, especially in higher

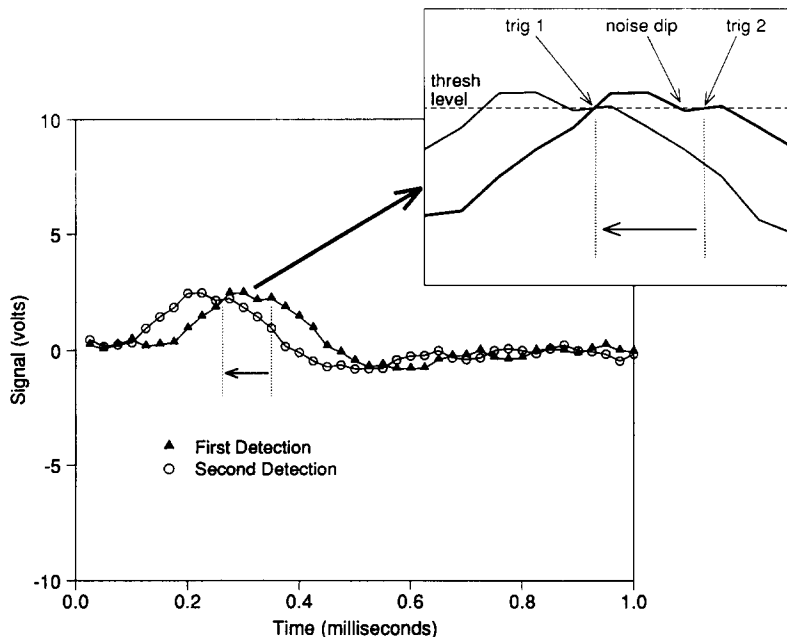


Fig. 11. Example of spike doubling. This tracing shows a type-D spike that was detected twice, with both the original spike wave form and a shifted version of the same spike wave form sent to the neural network for classification. The first detection occurred when the wave form rose above the threshold level at trigger 1. The second detection occurred at trigger 2, just after a small downgoing noise spike on top of the neural spike had caused dip in the signal below the threshold level.

noise environments, where a neural spike is detected only once, but the wave form is not similar enough in shape to what the neural network considers correct for that cluster, so a new noise cluster is formed for it. This means the neural spike is misclassified as a noise spike, causing a false negative.

Fig. 12 shows 2 near simultaneous spikes: a type D and a type E. The type-D spike was detected at trigger 2, and the type-E spike was detected at trigger 1. Both spikes were correctly sorted into 2 different clusters in this example.

When 2 spikes were closer together in time, as the questionable spike in Fig. 13, only one triggering occurred since the signal did not drop below threshold between the 2 spikes. In this case, the single detected spike could have been sorted into either one of the neural spike clusters, or its own noise cluster. So, at a minimum 1 spike is missed, either the type D or the type E; in the worst scenario, a new noise cluster is formed for this spike, causing both spikes to be missed. One way to control this is to remember that the higher the threshold level, the closer 2 near-simultaneous spikes can become and yet still be detected as 2 individual spikes (of course, if the threshold level is set too high, the risk of missing a spike increases).

Once 2 spikes run together and are only sensed once by the threshold detector, it is very difficult to do anything further to separate them. Our spike discriminator does not have a way to deal with this problem. One possible technique that could be used for separat-

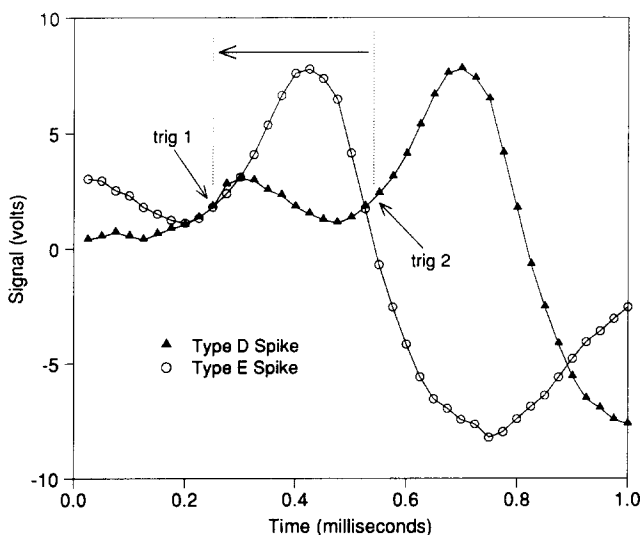


Fig. 12. Example of near simultaneous spikes. In this tracing, a type-E spike closely followed a type-D spike. Both were triggered independently and clustered correctly because the signal fell below the threshold level between the spikes. Both 1 ms wave shapes are shown, shifted in time, as seen by the neural network processing algorithm.

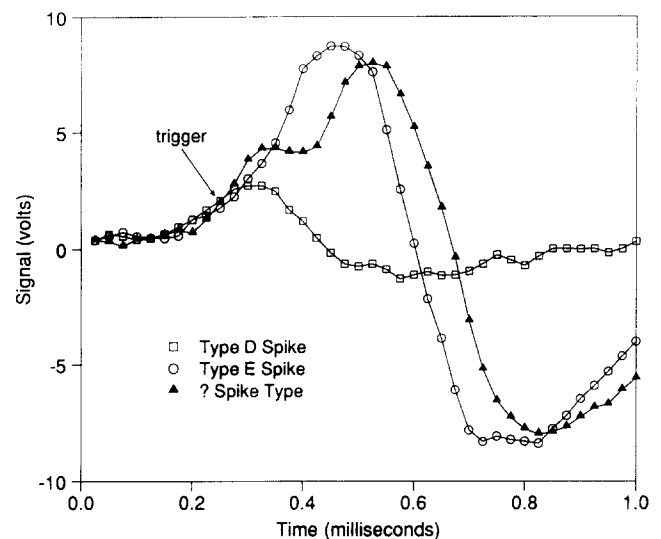


Fig. 13. Example of superimposed spikes. Here, the type-E spike overlapped with the type-D spike, creating a new wave form. Normal type-D and -E spikes are overlaid for comparison. This questionable spike was detected only once, but contained 2 spikes within it. Depending on which cluster this spike is sorted into, at least 1 of the overlapping spikes will be lost, and possibly both if it is put into a noise cluster.

ing overlapped spikes would be to subtract out known 40-point spike wave forms from the 40-point overlapping spike wave form. The differences could then be run through the neural network to see if it was close enough to any existing cluster. This would involve a large amount of calculation, since the peak of an overlapping spike could occur at any time in the sampled wave form, meaning that the known wave form to be removed would have to be shifted through the 40 points, and tried individually at each step.

4.2. Accuracy

The accuracy of the spike discriminator was shown using 2-unit generated data, 3-unit generated data, and 2-unit actual data. The neural spike shapes were quite similar in shape, yet the spike discriminator was very good in separating them. This suggests that even though spike amplitude is an important feature often used to distinguish between different spikes, it is very possible to throw out this variable and still get excellent sensitivity based solely on spike shape. The noise did have an effect at lower levels than if only 1 spike shape were present in the tracing, in which case only simple threshold detection would be needed, and not spike discrimination as well. In other words, during multi-unit spike discrimination it is usually the difference between the spike wave forms that predicts accuracy, and not so much the difference between the spikes and the noise level (i.e., the SNR).

4.3. Amplitude tracking

Amplitude changes had basically no effect on the accuracy of clustering spikes. This is useful for circumstances of either slow or rapid drift of spike amplitude.

4.4. Performance

The performance goal of a spike discrimination system is real-time operation. Right now, using the VAXstation 4000–60 system, we notice a slight delay between sequential stimuli while spikes are being processed (depending on data conditions); this slows down on-line data collection a little bit. Using an ALPHA workstation (Digital Equipment, Maynard, MA), there should be no delay, since it should be at least ten times faster than our current computer.

4.5. Conclusion

Overall, our neural network-based spike discriminator met the criteria that we set. Its accuracy is good, even while sorting nearly similar spikes shapes. It has the ability to work in noisy environments and to eliminate triggered noise spikes. Also, the software tracks changes in spike shape and amplitude, both slow and fast variations. Its speed is adequate so that it can be used in real-time. Since there is very little user set-up and no training phase required, using the spike discriminator should not interfere with data collection under circumstances of sub-optimal unit stability, when there is not much time to record from the cell before it is lost. This software could easily be run by anyone familiar with single-unit neural recording techniques.

Our neural network algorithm is limited by the speed of the latest computer system in our lab right now, and when more advanced algorithms are developed, the hardware will need to be updated. We feel that on-line spike train separation is a necessity in order to see the physiological response of the cells to different stimuli, as we record from them. The use of spike discrimination in multiple-site electrode recordings will also be important in the future, as the push to record from as many neurons as possible simultaneously continues.

Of course, the mathematical theory needed to analyze all of this data, once separated into different clusters, has barely been developed. Most of the techniques used today were developed by Perkel et al. (1967a,b, 1975). These include cross-correlation histograms, 2-dimensional scatter plots, and 3-dimensional scatter plots. More recently, the joint-peristimulus-time histogram has been introduced (Aertsen et al., 1989). Reinis et al. (1992) has also been developing techniques for creating block diagrams of neuronal interactions. Furthering this work is critical to under-

standing the simultaneous multi-unit data made available by the advancing spike discrimination techniques.

Our software, including the ART-2 subroutine, is available upon request to the authors.

Acknowledgements

This work was supported by NIH Grant DC00116. The authors are grateful to Dr. Daniel Geisler, Dr. John Brugge, Ravi Kochhar, and Jane Sekulski, Department of Neurophysiology, University of Wisconsin-Madison; Dr. Rick Jenison, Department of Psychology, University of Wisconsin-Madison; Dr. Kristin Bennett, Mathematical Sciences Department, Rensselaer Polytechnic Institute; and Dr. Paolo Guadiano, Department of Cognitive and Neural Systems, Boston University.

References

- Abeles, M. and Goldstein, M.H., Jr. (1977) Multispike train analysis, *Proc. IEEE*, 65: 762–773.
- Aertsen, A.M.H.J., Gerstein, G.L., Habib, M.K., Palm, G., Gochin, P.M., and Kruger, J. (1989) Dynamics of neuronal firing correlation: Modulation of 'Effective Connectivity', *J. Neurophysiol.*, 61: 900–917.
- Bergman, H., and DeLong, M.R. (1992) A personal computer-based spike detector and sorter: implementation and evaluation, *J. Neurosci. Methods*, 41: 187–197.
- Carpenter, G.A. and Grossberg, S. (1987) ART2: Self-organization of stable category recognition codes for analog input patterns, *Applied Optics*, 26: 4919–4930.
- Carpenter, G.A. and Grossberg, S. (1988) The ART of adaptive pattern recognition by a self-organizing neural network, *Computer*, March 1988: 77–88.
- Carpenter, G.A., Grossberg, S., and Rosen, D.B. (1991) ART-2A: An adaptive resonance algorithm for rapid category learning and recognition, *Neural Networks*, 4: 493–504.
- Eggermont, J.J. (1991) Neuronal pair and triplet interactions in the auditory midbrain of the leopard frog, *J. Neurophysiol.*, 66: 1549–1563.
- Epping, W.J.M. and Eggermont, J.J. (1987) Coherent neural activity in the auditory midbrain of the grassfrog, *J. Neurophysiol.*, 57: 1464–1483.
- Gochin, P.M., Gerstein, G.L., and Kaltenbach, J.A. (1990) Dynamic temporal properties of effective connections in rat dorsal cochlear nucleus, *Brain Res.*, 510: 195–202.
- Gochin, P.M., Kaltenbach, J.A., and Gerstein, G.L. (1989) Coordinated activity of neuron pairs in anesthetized rat dorsal cochlear nucleus, *Brain Res.*, 497: 1–11.
- Jansen, R.F. (1990) The reconstruction of individual spike trains from extracellular multineuron recordings using a neural network emulation program, *J. Neurosci. Methods*, 35: 203–213.
- Jansen, R.F. and Maat, A.T. (1992) Automatic wave form classification of extracellular multineuron recordings, *J. Neurosci. Methods*, 42: 123–132.
- Perkel, D.H., Gerstein, G.L., and Moore, G.P. (1967a) Neuronal spike trains and stochastic point processes. I. The single spike train, *Biophys. J.*, 7: 391–418.

- Perkel, D.H., Gerstein, G.L., and Moore, G.P. (1967b) Neuronal spike trains and stochastic point processes. II. Simultaneous spike trains, *Biophys. J.*, 7: 419–440.
- Perkel, D.H., Gerstein, G.L., Smith, M., and Tatton, W.G. (1975) Nerve-impulse patterns: a quantitative display technique for three neurons, *Brain Res.*, 100: 271–296.
- Reinis, S., Weiss, D.S., McGaraughty, S., and Tsoukatos, J. (1992) Method of analysis of local neuronal circuits in the vertebrate central nervous system, *J. Neurosci. Methods*, 43: 1–11.
- Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986) Learning Internal Representations by Error Propagation. In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing*, Vol. 1, Chapt. 8, Cambridge, MA.
- Salganicoff, M., Sarna, M., Sax, L., and Gerstein, G.L. (1988) Unsupervised wave form classification for multi-neuron recordings: a real-time software-based system. I. Algorithms and implementation, *J. Neurosci. Methods*, 25: 181–187.
- Sarna, M., Gochin, P., Kaltenbach, J., Salganicoff, and Gerstein, G.L. (1988) Unsupervised wave form classification for multi-neuron recordings: a real-time software-based system. II. Performance comparison to other sorters, *J. Neurosci. Methods*, 25: 189–196.
- Schmidt, E.M. (1984a) Instruments for sorting neuroelectric data: a review, *J. Neurosci. Methods*, 12: 1–24.
- Schmidt, E.M. (1984b) Computer separation of multi-unit neuroelectric data: a review, *J. Neurosci. Methods*, 12: 95–111.
- Voigt, H.F., and Young, E.D. (1980) Evidence of inhibitory interactions between neurons in dorsal cochlear nucleus, *J. Neurophysiol.*, 44: 76–96.
- Wheeler, B.C., and Heetderks, W.J. (1982) A comparison of techniques for classification of multiple neural signals, *IEEE Trans. Biomed. Eng.*, 29: 752–759.
- Yamada, S., Kage, H., Nakashima, M., Shiono, S., and Maeda, M. (1992) Data processing for multi-channel optical recording: action potential detection by neural network, *J. Neurosci. Methods*, 43: 23–33.