Supervised Spike Sorting Using Deep Convolutional Siamese Network and Hierarchical Clustering

Yinghao Li [α,γ] **Shuai Tang** [α,β] **Virginia R. de Sa** [α,β]
[α] Department of Cognitive Science, UC San Diego
[β] Halıcıoğlu Data Science Institute, UC San Diego
[γ] Computational Neurobiology Laboratory, Salk Institute for Biological Studies

Abstract

Engineering advances have enabled the simultaneous recording of large numbers of neurons across multiple channels. Determining which spikes are from which neurons (spike sorting) is a critical first step in analyzing these electrophysiological data. However, the most widely used methods are primarily semi-automatic and not suitable for a large number of recording sites. Most existing spike sorting algorithms are unsupervised or semi-supervised in nature, while little has been explored with a supervised approach. We propose a new fully automated and supervised spike sorting algorithm composed of deep similarity learning and hierarchical clustering. First, a convolutional Siamese network is trained on the simulated data with ground truth labels to learn the pairwise similarity of spikes, and then a similarity matrix is constructed and clustered with hierarchical clustering with stopping criteria to determine the number of clusters. Furthermore, once the deep Siamese network is trained, we can fine-tune the fully connected layers with a small labeled dataset and transfer the learned model to other cell types and recording devices. With the labeled information, our proposed method outperforms other state-of-the-art algorithms on both simulated and real neuronal data in terms of the number of hits, misses, and false positive neurons.

1 Introduction

With recent advances in electrode probe development [1, 2, 3, 4, 5, 6, 7] , simultaneous recordings across a large number of channels and recording sites are becoming a primary tool for studying the complex network ensembles and dynamics in the brain. However these extracellular micro-electrodes pick up action potentials from several nearby cells, giving rise to the problem of "spike sorting" - determining which action potential or spike belongs to which neuron. Each neuron has a unique characteristic spike shape determined by the dendritic morphology, the density and expression of ionic channels of the cell, the spatial location of the electrode relative to the neuron, and the resistivity of the extracellular medium [8]. Spike sorting is the task determining the number of neurons in the multiunit recording and assigning the time of occurrence of each spike to its firing neurons [9]. This is usually considered as an (unsupervised) problem in cluster analysis, and has been used for both *in vivo* and *in vitro* studies, along with applications in brain-computer interfaces such as development for new prosthetic control devices and neurorehabilitation techniques [10].

1.1 Related Work

Spike sorting algorithms have been extensively studied. Some commonly used methods are principal component analysis (PCA) [11], wavelet decomposition [12, 13], Bayesian approaches with Gaussian models [14, 15, 16] and Kalman filter based models [17]. The recent progress in automated spike

Preprint. Under review.

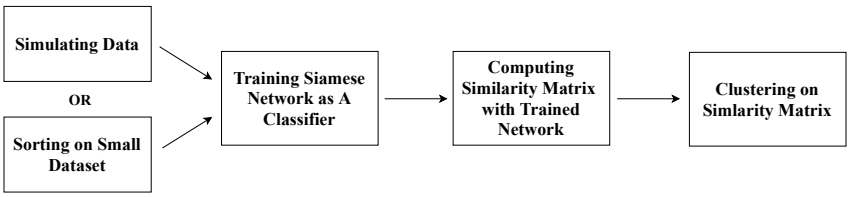

Figure 1: Illustration of the sorting pipeline.

sorting algorithms includes diffusion maps [18], t-distributed stochastic neighbor embedding (t-SNE) [19], and heuristic improvement on semi-automatic methods [20, 21].

1.2 New Supervised Sorting Pipeline

In the studies mentioned above, only [19] uses simulated data for parameter optimization, while other methods are fully unsupervised, where the simulated data were only used for performance evaluation. Here, we present a new supervised automatic spike sorting pipeline that takes advantages of the labeled information in simulated or expert confirmed real data. The pipeline for our algorithm is illustrated in Figure 1. First, we use the normal spike detection method described in section 2.1 to extract spikes from the raw data and pick several spikes as the templates for simulation. Then we simulate the recording data using the method in section 2.2 and train the deep convolutional Siamese network shown in section 2.3. Lastly, we compute the similarity matrix for target spikes and feed the matrix to a hierarchical clustering algorithm for sorting. Once the general model with simulated data in a given cell type and recording device is trained, we can fine-tune the fully connected layers with a small labeled dataset and transfer the learned model to other cell types and recording devices. Hence our method can be modified efficiently with a small amount of labeled data for new datasets.

2 Materials and Methods

2.1 Filtering and Spike Detection

General spike sorting pipelines include filtering, spike detection, feature extraction, and clustering. The first step is to apply a bandpass filter on the recorded data to remove low-frequency background activity and high-frequency noise that could be due to non-neural sources. The bandpass filter range for this study was between 300 Hz and 3,000 Hz for spike detection and 300 Hz to 1,000 Hz for spike snipping. The second step, spike detection, is to snip putative spikes from filtered recording. We employed the spike detection method proposed in [13]. The detection threshold is set to be the standard deviation σ_n of the background noise of filtered data, which is assumed to follow a Gaussian distribution and thus estimated by the median absolute deviation. Spikes are snipped with length 2 ms at the location where the absolute value of (filtered data) exceeds five times σ_n. That is, $Thr = 5\sigma_n$ and $\sigma_n = \text{median}\left\{\frac{|x|}{0.6745}\right\}$ where x is the band-passed signal. As in [13], we used the standard deviation of x to estimate σ_n.

2.2 Data Simulation

We followed the data simulation approach in [22]. One simulated recording consists of three components: background activity, multiunit activity, and single unit activity. Unlike in [22] where templates were obtained by averaging previously sorted putative spikes, we simply picked those presumably not from the same neurons as the templates. This is because we want to start from scratch where we possibly do not have sorted datasets similar to those of our interest in sorting.

The target sampling rate was 30,000 Hz, and the templates were from data used in [23], recorded with a Utah Array in the medial temporal lobe in four human epilepsy subjects from the Massachusetts General Hospital. We identified 488 templates in total, where the first 70% were used for training and the rest were split evenly between test and validation. The background activity was generated by superimposing millions of units from all 488 templates and then normalizing to have a mean 0 and standard deviation 1. The multiunit activity is generated from 20 to 30 randomly chosen templates, each of which has an amplitude ranging from 0.5 to 1.5 times the detection threshold. The multiunit

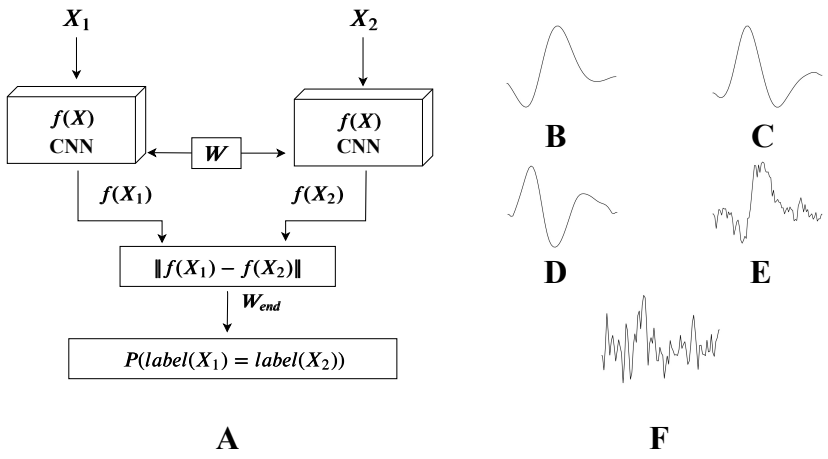

Figure 2: Siamese network architecture and sample inputs. (A) Siamese network architecture: The network has two copies of the same underlying network $f(x)$ while processing different input X_1, X_2. The difference of outputs from each copy is taken and passed to the last layer with W_{end} and sigmoid function that outputs the probability that two spikes belong to the same neuron. (B-F) Sample input features: (B) Filtered spike waveform. (C) First order derivative of (B). (D) Second order derivative of (B). (E) Unfiltered waveform of (B). (F) First order derivative of (E).

activity follows a Poisson process with mean firing rate 20 Hz. The single unit activity is generated by selecting N templates different from those used for multiunit activity, with each assigned an amplitude ranging from 1.5 to 4 times the detection threshold. Each unit in the single unit activity follows a Poisson process with a firing rate ranging from 0.5 to 5 Hz.

2.3 Convolutional Siamese Network

Siamese networks can learn pairwise similarity and the contrastive loss function is usually used to minimize the distance for samples within the same group and maximize the distance otherwise [24]. For this study, we employed the discriminative model for Siamese networks proposed in [25]. The architecture of the Siamese network used for this study is given in Figure 2A. A fully connected layer before the output is added as indicated by the parameter W_{end}. The inputs X_1, X_2 to the network are two spikes (along with their features); if their labels are the same $y = 1$ else $y = 0$.

2.3.1 Loss Function

Instead of contrastive loss, we used pairwise loss in order to push the outputs for positive and negative pairs to two extreme sides of the distributions

$$\mathcal{L}(p_+, p_-) = \max\left(0, \delta - (p_+ - p_-)\right) \tag{1}$$

where p_+ is output probability for positive pair from the network, p_- is the output probability for negative pair from the network, and δ is the margin between the two distributions.

Different choices of δ in the training stage affect the threshold ϑ to use later in the clustering stage. Though smaller δ makes training faster and produces a slightly higher testing accuracy towards the end, it could be painful to find the optimal ϑ used for classifying whether a given pair is positive or negative. Hence, all results reported here are trained with $\delta = 0.5$, with which that testing accuracy of the classifier is stable for $\vartheta \in [0.4, 0.75]$ (See Table 1 in Suppl. Materials for the effect of δ on ϑ).

2.3.2 Convolutional Networks

The "sister networks" of Siamese networks used for this study are two later-fused copies of convolutional neural networks with identical architectures but different weights. One takes the raw spikes (with length 2.5 ms) along with their first order derivatives, and the other takes the filtered spikes along with their first and second order derivatives. Derivatives are treated as different channels of the same input and are provided as input features as it has been shown that first and second order

derivatives can be used to differentiate similar units [26, 27]. Indeed we found that removing the derivative features results in a drop in the classification accuracy by nearly 5%.

The outputs from the convolutional layers of raw and filtered spikes are then concatenated, along with the first 14 principal components of both raw and filtered spikes. Sample inputs are given in Figure 2B-2E, and the architecture of the convolutional network used is given in Fig. 1 in Suppl. Materials.

2.3.3 Training Dataset and Algorithm

The training dataset was simulated with the method in section 2.2 with 340 templates chosen for the training set (80% out of the total 488 templates). We only used the single unit activity for each template, and we set a balanced number of spikes for each template to be around 3,000, and therefore the final training set consists of 1,2205,656 spikes in total. However, the set of positive and negative pairs are highly imbalanced, and hence, we used a balanced sampler to sample both positive and negative pairs with a chance of 50%.

The network was trained through mini-batch stochastic gradient descent with adaptive learning rate. In particular, Adabound [28] with initial learning rate 0.001, final learning rate 0.1, batch size 128, and weight decay parameter (equivalent to the L2 regularization factor) 0.0001 was used. Training was stopped after the validation loss had plateaued for at least 5 epochs.

2.3.4 Transfer Learning

The above training set is enormous, which poses a severe obstacle in practical use if the trained model was not able to generalize to other datasets with different cell types and recording devices. However, we are able to fine-tune this heavily trained model with a small labeled dataset to transfer the performance to other cell types and recording devices. We specifically tested this transfer learning scheme on one simulated dataset and one real dataset.

We test our algorithm with the simulated dataset presented in [29] [1]. It consists of 95 channels in total, with the number of units in each channel ranging from 2 to 20. The templates used for these simulated datasets are 594 different averaged spike waveforms from recordings of monkey basal ganglia and neocortex, which are different from the cell type used to train the original model (recorded in the human temporal lobe). Here we picked the first channel with only 16 units and 9,612 spikes detected for training. This is considerably smaller than the one used for training the general model.

The real dataset used was from https://crcns.org/data-sets/hc/hc-1/about, which was simultaneous intracellular and extracellular recordings in the hippocampus of rats [30]. Extracellular recordings were processed partly with wire tetrodes for about half of the data set, while the remaining part was done with six-site linear silicon probes. This is drastically different from our original setting, as both the recording device and species are completely different. The labeling was done by snipping the spikes around the time window where the intracellular recording indicates that the neuron has fired. Since each dataset has only one intracellularly recorded unit, we picked four datasets with a total of 4,621 spikes in the training set (See Table 2 in Suppl. Materials for details). Note that since the general model was trained on the single-channel data, the network was not able to take more than one channel for its input, and therefore we had to get the corresponding spikes from all recorded channels and regard them as from a single channel in the training and testing procedure.

2.4 Hierarchical Clustering

For the clustering algorithm, we employed an elementary variant of average-linkage hierarchical clustering [31] with stopping criteria. After we finish training the Siamese network, we compute the probability that a given pair of spikes were fired by the same neuron for all pairs in the input spikes. We then perform average-linkage clustering on the similarity matrix. In the original agglomerative clustering algorithms, the clustering tree is cut at different levels to obtain different numbers of clusters from which we pick the desired one. Here, we used a stopping criterion based on threshold, ϑ, (also used in the classifier) to decide whether a given pair is positive or negative. If the output from the Siamese network $p > \vartheta$, we classify this pair as positive, otherwise negative. This ϑ can also be used for clustering as additional information to get the number of clusters. If the pairwise similarity for all clusters in the clustering process is less than ϑ, we stop merging.

[1] Available at https://www135.lamp.le.ac.uk/hgr3/

Table 1: Performance of the model and algorithm under different conditions. Five models were trained with different initial weights, and Mean (SD) are shown for each measure. Definitions of "F_1-precision" and "F_1-recall" are in section 2.6, and the testing accuracy was assessed with the optimal ϑ by cross-validation on validation sets, sampled from balanced pairs in the dataset. S-10 dataset is one of our own simulated recordings with 10 neurons, and C-3 is the third simulated recordings from [29] with 12 neurons. (1) $\delta = 0.5$ for training, no parallelization in clustering. (2) $\delta = 0.1$ for training, no parallelization in clustering. (3) Same model as in (1), with parallelization in clustering ($K = 5$). (4) no fine-tuning on different distribution. (5) model fine-tuned (section 2.3.4.)

conditions	datasets	testing acc (%)	# of hits	# of missed	# of FP	F_1-precision (%)	F_1-recall (%)
default	S-10	84.21 ± 0.23	9 ± 0	1 ± 0	0 ± 0	69.22 ± 0.65	62.46 ± 0.53
$\delta = 0.1$	S-10	85.87 ± 0.24	S/A	S/A	S/A	69.50 ± 0.60	62.62 ± 0.68
parallelized	S-10	same as default	S/A	S/A	S/A	69.20 ± 0.87	61.16 ± 0.74
no fine-tuning	C-3	70.38 ± 2.11	7.8 ± 0.83	4.2 ± 0.55	1.6 ± 0.89	50.24 ± 5.62	54.46 ± 5.33
fine-tuned	C-3	91.90 ± 0.08	10 ± 0	2 ± 0	1 ± 0	83.53 ± 0.27	86.02 ± 0.32

Many clusters obtained by this process have only a few spikes. Thus we set another threshold, E_{min} (the minimum number of spikes in a cluster), to classify those with elements less than E_{min} as multiunit activity. For this study, we set $E_{min} = 30$ to accommodate sparse firing neurons.

To reduce the false positive clusters, we compute the within-cluster similarity for each cluster C

$$Sim_{in}(C) = \frac{1}{|C|} \sum_{c \in C} \left(\frac{1}{|C| - 1} \sum_{\substack{k \in C \\ c \neq k}} P(\text{label}(c) = \text{label}(k) | c, k) \right) \tag{2}$$

If $Sim_{in}(C)$ is less than ϑ_{in}, it is considered multiunit activity. Heuristically, we found $\vartheta_{in} =$ Acc $- 0.1$ gives the least false positive clusters on our validation set, where Acc is the classification accuracy with ϑ_{in} as the classification threshold.

2.5 Time Complexity

The time complexity of pairwise computing similarity is $\Theta(n^2)$, and that of average-linkage clustering is $\Theta(n^2 \log n)$ [32]. Since computing the pairwise similarity is not iterative, one may use GPUs to perform parallel computation at this step, so the tightest bound is $\Theta(n^2 \log n)$.

Parallelization To reduce the time complexity to be able to handle large numbers of spikes, we parallelize the process by dividing the target spikes into K chunks, where each chunk contains roughly the same number of spikes, which are then processed in parallel. After clusters are formed for all K chunks, we take the average of spike waveforms in each cluster as the representatives and run our algorithm on these waveforms. That is, the averaged waveform of spikes in each cluster is fed into the network to decide whether to merge these clusters or not.

The parallelization significantly reduces the runtime to $\Theta\left(\frac{n^2}{K^2} \log n\right) + \Theta(T^2 \log T)$, where T is the total number of clusters identified from K chunks. Since usually $T \ll n$, the complexity is $\Theta\left(\frac{n^2}{K^2} \log n\right)$. With parallelization, the runtime for $n >$ 100,000 is reduced from hours to minutes - faster than most other clustering algorithms that use K-means when n is not too large $\left(n < 1 \times 10^8 \text{ with } K = 5, \text{ for example}\right)$, which has complexity $O(n^2)$ [33].

2.6 Performance Evaluation

We evaluate our algorithm on three different distributions and compare to the state-of-the-art algorithms proposed in [19] and [21]. The datasets used for evaluation are simulated datasets out of the templates reserved for testing, all 95 simulated recordings in [29], and nine selected channels (see Table 2 in Suppl. Materials) in real neural data from [30]. Since there is only one neuron intracellularly recorded, we only have ground truth for one cluster for each dataset in [30]. To test the performance, we have picked nine datasets to get nine clusters and we split these clusters into

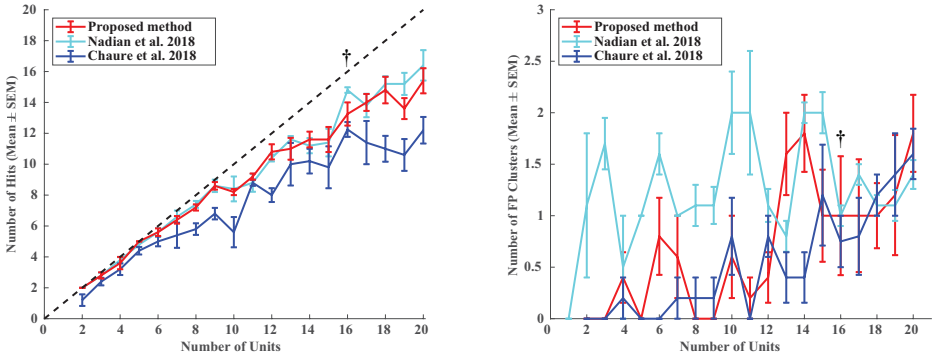

Figure 3: Performance of different algorithms on datasets from [29]. (A) True positive clusters with number of neurons ranging from 2-20, Mean ± SEM plotted. (B) False positive clusters, Mean ± SEM plotted. †: The first channel that has 16 units was used for fine-tuning, so we removed that dataset when testing our algorithm and [21]. (Better view in color.)

two, three, four clusters mixed with 10% noise spikes to generate three testing sets, with two, three, and four neurons respectively. Note that in [34], however, they made combinations on these datasets, while we found no actual difference with respect to performance between different combinations and hence only tested on three splits.

We evaluate the performance based on the number of hit (true positive), false positive and missed clusters along with F_1-precision and F_1-recall scores. The precision, recall and F_1 score of C_r from the results clusters and C_g from ground truth clusters are defined as

$$\text{recall}(C_r, C_g) = \frac{|C_g \cap C_r|}{|C_g|} = \text{precision}(C_g, C_r) \tag{3}$$

$$F_1(C_r, C_g) = \frac{2 \cdot \text{recall}(C_r, C_g) \cdot \text{precision}(C_r, C_g)}{\text{recall}(C_r, C_g) + \text{precision}(C_r, C_g)} \tag{4}$$

and the overall F_1 score for clusters $\mathcal{P}$ and $\mathcal{Q}$ are defined as

$$F_1^{\text{clus}}(\mathcal{P}, \mathcal{Q}) = \frac{1}{|\mathcal{P}|} \sum_{C_i \in \mathcal{P}} \max_{C_j \in \mathcal{Q}} \{F_1(C_i, C_j)\} \tag{5}$$

As suggested in [29] and [35], a selected cluster $C \in \mathcal{R}$ is counted as a hit if both $\max_{C_i \in \mathcal{G}} \text{recall}(C, C_i) > 50\%$ and $\max_{C_i \in \mathcal{G}} \text{precision}(C, C_i) > 50\%$, where $\mathcal{R}$ are clusters produced by sorting algorithms and $\mathcal{G}$ are ground truth clusters. Clusters that are not a hit are counted as false positives. Multiunit activity is not considered in hit calculation but is counted as false positive if it contains less than 50% of multiunit spikes from the ground truth. To assess the quality of each cluster, we also computed the F_1-precision and F_1-recall scores defined in [36], where F_1-precision $= F_1^{\text{clus}}(\mathcal{R}, \mathcal{G})$ and F_1-recall $= F_1^{\text{clus}}(\mathcal{G}, \mathcal{R})$. The F_1-precision measures the quality of selected clusters, while F_1-recall measures the quality of the matches to the ground truth clusters. If the selected clusters $\mathcal{R} \subseteq \mathcal{G}$, then F_1-precision $= 1$, while if $\mathcal{R} \supseteq \mathcal{G}$, F_1-recall$= 1$. In practice, we usually want the algorithms to identify as many high quality clusters that match with the ground truth as possible and later decide whether to use them or not, which makes F_1-recall a better measure with respect to this goal.

Note that we did not employ the mutual information (MI) measure used in [34] because it allows a high MI even if none of the selected clusters are hits as they subdivide a real cluster - this would be bad for the goal of separating neurons. Since the code for [19] is not publicly available, we compare our numbers to readouts from their reports. Since [19] only tested on datasets from [29] for number of hit, missed and false-positive clusters, we only compared to [19] in section 3.3 with these measures.

Table 2: Performance of our algorithm and [21] on real datasets from [30]. Note that the method in [21] finds few clusters for these datasets, while our method gets a decent number of correct clusters.

Dataset with 2 Neurons ($n = 2,291$)					
Algorithms	# of hits	# of missed	# of FP	F_1-precision	F_1-recall
Proposed method	2	0	1	63.11%	75.02%
Chaure et al. 2018	0	2	2	15.02%	8.67%
Dataset with 3 Neurons ($n = 2,479$)					
Algorithms	# of hits	# of missed	# of FP	F_1-precision	F_1-recall
Proposed method	3	0	0	70.10%	70.10%
Chaure et al. 2018	1	2	1	74.41%	30.26%
Dataset with 4 Neurons ($n = 1,761$)					
Algorithms	# of hits	# of missed	# of FP	F_1-precision	F_1-recall
Proposed method	2	2	1	60.30%	51.64%
Chaure et al. 2018	0	4	3	42.40%	32.75%

3 Results

All results shown below, except in section 3.1 where different conditions are specified, were run in the default setting, where $\vartheta = 0.7$, $E_{min} = 30$ and $\vartheta_{in} = \text{Acc} - 0.1$ without parallelization, and tests with [21] were run under default parameters published on GitHub [2].

3.1 Experiments

We ran several experiments to test our model and algorithm under different conditions, shown in Table 1. Under each setting, five models with varying initialization of weights were trained and clustering performance was assessed for each model.

We note that smaller δ would result in a relatively better testing accuracy if optimal ϑ was chosen out of cross-validation. This selection, however, is unnecessary because it does not improve clustering performance. Hence, by setting a large δ value, we can train models with similar performance for any $\vartheta \in [0.4, 0.75]$ without cross-validation. In order to reduce the number of false positive clusters, we set $\vartheta = 0.7$ for our subsequent results. We also note that parallelizing the clustering process does not cause any significant performance loss, but it does accelerate the clustering process as tested with $n = 10,773$. The runtime without parallelization was $2,578.60 \pm 98.58$ s, while that with parallelization was 89.00 ± 22.64 s. Because datasets from [29] are easier than our own simulated datasets, the fine-tuned performance was even better than on its original training distribution.

3.2 On Simulated Data: Ours

We first ran our algorithm and that suggested in [21] on our own simulated datasets. Each recording consists of 2-10 neurons with duration 10 minutes, and for each number of neurons, five recordings were simulated for testing. Our model have achieved 84.33% classification accuracy on the testing set. We obtained $192/270$ hits and $28/220$ false-positive clusters with F_1-precision $76.60\% \pm 16.79\%$ and F_1-recall $54.25\% \pm 12.42\%$ averaged across all 45 recordings with our algorithm, while [21] produced $102/270$ hits and $140/242$ false-positive clusters with F_1-precision $49.21\% \pm 14.23\%$ and F_1-recall $43.58\% \pm 15.28\%$. The plots with detailed results are given in Fig. 2-3 in Suppl. Materials. The figures clearly reveal that we outperform [21] if we train the models with the cell types and recording devices of interest from scratch.

3.3 On Simulated Data: C. Pedreira et al. 2012 [29]

We also compared the performance of our algorithm to [21] and [19] on datasets from [29]. These are datasets tested both in [21] and [19]. Note that since we have no code for [19], the results shown in Figure 3 of [19] are eyeballed from reports in [19].

Our fine-tuned model achieved 92.11% classification accuracy on the testing set. As shown in Figure 3, our algorithm obtained a comparable number of true-positive clusters as in [19] but much less false

[2] Downloadable at https://github.com/csn-le/wave_clus

positives than [19]. We note that in [19], the authors used genetic algorithms to optimize parameters for t-SNE and DBSCAN with the simulated datasets from [29]. Also it appears that they fine-tune parameters to optimize test performance, while we only fine-tuned our model with one out of the total 95 recordings. Our algorithm had significantly more hits with comparably low false positives compared to [21]. Our tests on the code of [21] had slightly fewer hits than reported in their paper but also consistently lower false-positive clusters than in their reports.

We have also computed the F_1-precision and F_1-recall which are shown in Fig. 4 in Suppl. Materials. The overall F_1-precision and F_1-recall of our algorithm are $81.23\% \pm 8.68\%$ and $81.84\% \pm 7.97\%$, respectively, and the overall F_1-precision and F_1-recall of the algorithm suggested in [21] are $86.46\% \pm 11.14\%$ and $75.63\% \pm 13.84\%$, respectively. Note that our algorithm had a lower F_1-precision score than [21] because our algorithm produced more false positives than [21], while the F_1-recall of our algorithm is higher than [21] because we have more hits. Also note that our algorithm produced a relatively stable F_1-precision and F_1-recall trade-off and less standard deviation across all 95 simulations than [21] did. We believe that F_1-recall is the more meaningful measure because we want to have as many high quality hit neurons as possible.

These results show that our models can be readily used through fine-tuning with a small labeled dataset skipping the tedious simulation process. As shown in section 3.2, though, training the model with simulated data from scratch is likely to achieve much better performance.

3.4 On Real Biological Data

Lastly, we tested our algorithm and algorithm from [21] on the real data provided in [30]. Due to the noisy nature of the real datasets and multi-electrode recordings (see Fig. 5 in Suppl. Materials for sample clusters), our fine-tuned model was only able to get the classification accuracy of 71.38% on the testing set. The results on the three datasets described in section 2.6 are shown in Table 2.

The method suggested in [21] had almost failed for these datasets, while our method was able to identify a reasonable number of correct clusters and produced acceptable F_1-precision and F_1-recall scores, given that the datasets are extremely hard, the classification accuracy is relatively lower, and the multi-electrode data recorded by tetrodes and six-site probes were treated as single-channel data.

Finally, we note that [34] have very recently tested on the same datasets and would have failed these tests as well if they had employed our hits, false positive and F-1 measures instead of the MI measures. This is because they had barely captured any true positive neurons (due to splitting real neurons into multiple clusters), even though they had fair MI values for their selected clusters. They also present an automated feature selection method rather than a complete automatic spike sorting pipeline.

4 Discussion

We presented a novel supervised and fully automatic spike sorting pipeline that has achieved the state-of-the-art performance in terms of both the number of hits, misses, and false positive neurons. Unlike other state-of-the-art algorithms such as [19], which rely heavily on supervised training for finding optimal parameters, our model was able to generalize its performance by fine-tuning with only a small labeled dataset, while achieving similar to better performance. Also, our algorithm has consistently beaten the algorithm proposed in [21] in all three different datasets.

In this study, we only trained the model to consider spikes recorded from one channel. This is true for the templates used in our training set. In future work, we will extend the model to better handle datasets such as [30], where neurons are simultaneously recorded by four or more channels. We also believe our supervised pipeline will work for other clustering problems where realistic data simulation is possible and is more straightforward than clustering itself, or where big datasets with definite ground truth are more readily accessible.

The development of automatic and reliable spike sorting algorithm is becoming crucial as the development of new electronic probes is greatly increasing the number of sites and resolutions recorded [37]. Researchers are now expected to handle data recorded from thousands of sites, which are too large to be processed manually in a supervised way [4, 38]. With the help of deep learning, we were able to shift the burden for supervision from humans to artificial neural networks, allowing more efficient and repeatable processing.

References

[1] Luca Berdondini, Kilian Imfeld, Alessandro Maccione, Mariateresa Tedesco, Simon Neukom, Milena Koudelka-Hep, and Sergio Martinoia. Active pixel sensor array for high spatio-temporal resolution electrophysiological recordings from single cell to large scale neuronal networks. *Lab on a Chip*, 9(18):2644–2651, 2009.

[2] 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(1):11–17, 2012.

[3] J Müller, M Ballini, P Livi, Y Chen, A Shadmani, U Frey, IL Jones, M Fiscella, M Radivojevic, DJ Bakkum, et al. Conferring flexibility and reconfigurability to a 26,400 microelectrode cmos array for high throughput neural recordings. In *2013 Transducers & Eurosensors XXVII: The 17th International Conference on Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS & EUROSENSORS XXVII)*, pages 744–747. IEEE, 2013.

[4] A Paul Alivisatos, Anne M Andrews, Edward S Boyden, Miyoung Chun, George M Church, Karl Deisseroth, John P Donoghue, Scott E Fraser, Jennifer Lippincott-Schwartz, Loren L Looger, et al. Nanotools for neuroscience and brain activity mapping, 2013.

[5] Gabriel Bertotti, Dmytro Velychko, Norman Dodel, Stefan Keil, Dirk Wolansky, Bernd Tillak, Matthias Schreiter, Andreas Grall, Peter Jesinger, Sebastian Röhler, et al. A cmos-based sensor array for in-vitro neural tissue interfacing with 4225 recording sites and 1024 stimulation sites. In *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*, pages 304–307. IEEE, 2014.

[6] Vijay Viswam, Jelena Dragas, Amir Shadmani, Yihui Chen, Alexander Stettler, Jan Müller, and Andreas Hierlemann. 22.8 multi-functional microelectrode array system featuring 59,760 electrodes, 2048 electrophysiology channels, impedance and neurotransmitter measurement units. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 394–396. IEEE, 2016.

[7] Kevin A White, Geoffrey Mulberry, Jonhoi Smith, Manfred Lindau, Bradley A Minch, Kiminobu Sugaya, and Brian N Kim. Single-cell recording of vesicle release from human neuroblastoma cells using 1024-ch monolithic cmos bioelectronics. *IEEE Transactions on Biomedical Circuits and Systems*, 12(6):1345–1355, 2018.

[8] Carl Gold, Darrell A Henze, Christof Koch, and Gyorgy Buzsaki. On the origin of the extracellular action potential waveform: a modeling study. *Journal of neurophysiology*, 95(5):3113–3128, 2006.

[9] Rodrigo Quian Quiroga. Spike sorting. *Current Biology*, 22(2):R45–R46, 2012.

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

[11] Dimitrios A Adamos, Efstratios K Kosmidis, and George Theophilidis. Performance evaluation of pca-based spike sorting algorithms. *Computer methods and programs in biomedicine*, 91(3):232–244, 2008.

[12] Kyung Hwan Kim and Sung June Kim. A wavelet-based method for action potential detection from extracellular neural signal recording with low signal-to-noise ratio. *IEEE Transactions on Biomedical Engineering*, 50(8):999–1011, 2003.

[13] 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.

[14] Tatsuya Haga, Osamu Fukayama, Yuzo Takayama, Takayuki Hoshino, and Kunihiko Mabuchi. Efficient sequential bayesian inference method for real-time detection and sorting of overlapped neural spikes. *Journal of neuroscience methods*, 219(1):92–103, 2013.

[15] Cyrille Rossant, Shabnam N Kadir, Dan FM Goodman, John Schulman, Maximilian LD Hunter, Aman B Saleem, Andres Grosmark, Mariano Belluscio, George H Denfield, Alexander S Ecker, et al. Spike sorting for large, dense electrode arrays. *Nature neuroscience*, 19(4):634, 2016.

[16] Jin Hyung Lee, David E Carlson, Hooshmand Shokri Razaghi, Weichi Yao, Georges A Goetz, Espen Hagen, Eleanor Batty, EJ Chichilnisky, Gaute T Einevoll, and Liam Paninski. Yass: Yet another spike sorter. In *Advances in Neural Information Processing Systems*, pages 4002–4012, 2017.

[17] Ana Calabrese and Liam Paninski. Kalman filter mixture model for spike sorting of non-stationary data. *Journal of neuroscience methods*, 196(1):159–169, 2011.

[18] Thanh Nguyen, Asim Bhatti, Abbas Khosravi, Sherif Haggag, Douglas Creighton, and Saeid Nahavandi. Automatic spike sorting by unsupervised clustering with diffusion maps and silhouettes. *Neurocomputing*, 153:199–210, 2015.

[19] MH Nadian, S Karimimehr, J Doostmohammadi, A Ghazizadeh, and R Lashgari. A fully automated spike sorting algorithm using t-distributed neighbor embedding and density based clustering. 2018.

[20] 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.

[21] Fernando J Chaure, Hernan G Rey, and Rodrigo Quian Quiroga. A novel and fully automatic spike-sorting implementation with variable number of features. *Journal of neurophysiology*, 120(4):1859–1871, 2018.

[22] Juan Martinez, Carlos Pedreira, Matias J Ison, and Rodrigo Quian Quiroga. Realistic simulation of extracellular recordings. *Journal of neuroscience methods*, 184(2):285–293, 2009.

[23] Adrien Peyrache, Nima Dehghani, Emad N Eskandar, Joseph R Madsen, William S Anderson, Jacob A Donoghue, Leigh R Hochberg, Eric Halgren, Sydney S Cash, and Alain Destexhe. Spatiotemporal dynamics of neocortical excitation and inhibition during human sleep. *Proceedings of the National Academy of Sciences*, 109(5):1731–1736, 2012.

[24] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[25] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005.

[26] Zhi Yang, Qi Zhao, and Wentai Liu. Improving spike separation using waveform derivatives. *Journal of neural engineering*, 6(4):046006, 2009.

[27] Sivylla E Paraskevopoulou, Deren Y Barsakcioglu, Mohammed R Saberi, Amir Eftekhar, and Timothy G Constandinou. Feature extraction using first and second derivative extrema (fsde) for real-time and hardware-efficient spike sorting. *Journal of neuroscience methods*, 215(1):29–37, 2013.

[28] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.

[29] Carlos Pedreira, Juan Martinez, Matias J Ison, and Rodrigo Quian Quiroga. How many neurons can we see with current spike sorting algorithms? *Journal of neuroscience methods*, 211(1):58–65, 2012.

[30] 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. *CRCNS. org*, 2009.

[31] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The computer journal*, 26(4):354–359, 1983.

[32] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, pages 385–388, 2010.

[33] James MacQueen. 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.

[34] Bryan C Souza, Vítor Lopes-dos Santos, João Bacelo, and Adriano BL Tort. Spike sorting with gaussian mixture models. *Scientific reports*, 9(1):3627, 2019.

[35] Johannes Niediek, Jan Boström, Christian E Elger, and Florian Mormann. Reliable analysis of single-unit recordings from the human brain under noisy conditions: tracking neurons over hours. *PloS one*, 11(12):e0166598, 2016.

[36] Stephan Günnemann, Ines Färber, Emmanuel Müller, Ira Assent, and Thomas Seidl. External evaluation measures for subspace clustering. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1363–1372. ACM, 2011.

[37] Kenneth D Harris, Rodrigo Quian Quiroga, Jeremy Freeman, and Spencer L Smith. Improving data quality in neuronal population recordings. *Nature neuroscience*, 19(9):1165, 2016.

[38] Ian H Stevenson and Konrad P Kording. How advances in neural recording affect data analysis. *Nature neuroscience*, 14(2):139, 2011.