CS482/682 Final Project Report Group 25

Semi-Supervised Spike Sorting

Paul A. (padkiss1), Giorgio D. S. (gdisalv1), Spencer L. (sloggia1), Jacopo T. (jtenegg1)

1 Introduction

Background Spike sorting assigns spiking activity to individual neurons from extracellular electrode recordings. This task is critical for experimental neuroscience and its applications. Traditional methods rely on humans, hence they fail with large numbers of electrodes and spiking units. We propose a solution based on autoencoders and clustering.

Related Work Traditionally, automatic spike sorting is divided into **i)** spike detection, **ii)** feature extraction, and **iii)** clustering. However, these methods are sensitive to noise and spike shape similarity. To overcome some of these shortcomings, various machine learning approaches have been explored [1, 2, 3]. Recent works also propose deep-learning based methods [4, 5].

2 Methods

Dataset Here, we will use two datasets. **alm-1**[1] comprises real single-electrode recordings from 19 behaving mice. This dataset does not contain any labels, hence we will use it for unsupervised training. **Pedreira et el.** [6] comprises 95 simulated sessions of varying difficulty with 2 to 20 spiking units. This dataset also includes the label of the spiking unit corresponding to each spike. We will use this dataset for supervised training and evaluation. We show that our method achieves performance comparable to other methods benchmarked on SpikeForest[2].

For preprocessing, we **i)** bandpass filter the voltage recordings, **ii)** identify spikes by thresholding [2], **iii)** extract a 2 ms window around each peak, and **iv)** normalize the spike traces with their amplitude. Since the two datasets are sampled at different rates, we downsample the Pedreira dataset to have the same number of samples as the alm-1 dataset. Fig. 1 shows two example traces and their reconstruction.

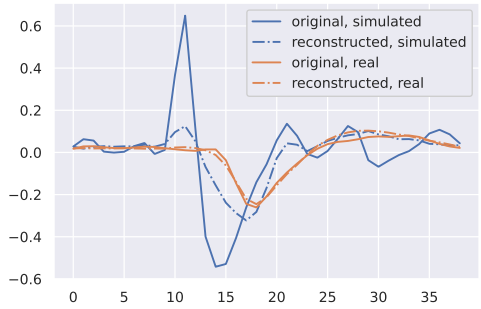
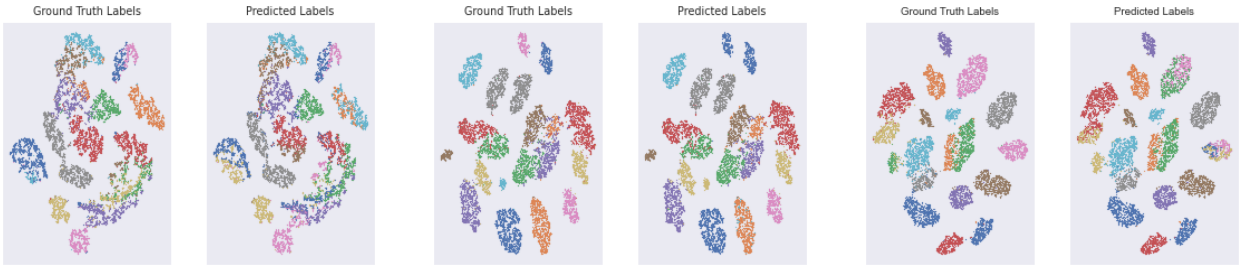


Figure 1: Two example spike reconstructions.

Setup, Training and Evaluation In this work, we expand on [5], and we try to improve upon it. We iterate various solutions. **v0**: we train an autoencoder ensemble which comprises 3 fully-connected encoders of increasing depth (2, 3, and 4 layers), and their respective fully connected decoders. Each encoder reduces the input dimension from 39 to 3. We train for 50 epochs on the Pedreira dataset using MSE loss as a measure of goodness of reconstruction and Adam optimizer; **v1**: we use convolutional encoders, and attach an MLP to the latent spike representations. We train by alternating batches from both datasets. We add the cross-entropy loss of the MLP to the reconstruction loss on the Pedreira dataset. The intuition is that adding a classification term favors representations that can be clustered more easily; **v2**: we implement an end-to-end deep clustering pipeline to train on the alm-1 dataset. We

[1] https://crcns.org/data-sets/motor-cortex/alm-1
[2] https://github.com/flatironinstitute/spikeforest2



(a) v0 (b) v1 (c) v2

Figure 2: t-SNE embedding of the latent representations of a test session with 20 spiking units, colored with ground truth labeling (left) and *auto*-GMM predictions, for SNR > 4, for all versions.

train convolutional autoencoder ensembles in parallel, replacing the classification head with a clustering head. We use a multitarget loss function, which measures how well the embeddings can be clustered and how well the input can be reconstructed. The clustering loss target is a novel function called *"Atom Loss"*. Atom Loss is defined for each point i, as $L_i = \sum_{j \neq i} \frac{.1}{dist(i,j)^3} + \sum_k (\frac{.1}{dist(i,k)^3} - \frac{1}{dist(i,k)^2})$, where *dist* is euclidean distance. The total loss is the sum of atom loss and reconstruction loss. For each parallel architecture we fit a Gaussian Mixture Model (GMM) at each epoch, which gives a new set of centroids (k), the total loss is calculated, and the encoder models are updated. At inference time the best model can be selected based on Bayesian Information Criterion (BIC). All versions are tested on the same subset of the Pedreira dataset. Performance is evaluated with precision, accuracy, and recall of correct classifications of the latent spike embeddings. We compare *gt*-GMM performance (i.e. setting the number of clusters to the ground truth value) and *auto*-GMM performance (i.e. finding the number of optimal number of clusters by picking the model with lowest BIC score). In a fashion similar to SpikeForest, we test performance as a function of minimum SNR of the spike traces, which filters out noisy spikes.

3 Results

Fig. 2 shows the t-SNE embeddings of the latent representations, colored with correct and predicted classification. Table 1 shows the performance of all the versions on the Pedreira holdout set.

Algorithm	Accuracy	Precision	Recall
v0 (gt-GMM)	72%	76%	73%
v0 (auto-GMM)	63%	60%	62%
v1 (gt-GMM)	90%	91%	88%
v1 (auto-GMM)	85%	86%	87%
v2 (gt-GMM)	82%	82%	82%
v2 (auto-GMM)	86%	87%	81%

Table 1: Average accuracy, precision, and recall for SNR > 4 for all versions. We remark that v1 is a semi-supervised solution.

4 Discussion

We have expanded on [5], confirming that an ensemble of autoencoders can perform spike sorting with performance comparable or higher to current methodologies. The addition of a convolutional layer before each encoder increased robustness of the model to different sampling rates and spike sizes in the data and the inclusion of a classification head during training on supervised data favored formation of distinct clusters of the latent features, thus increasing accuracy. The introduction of the *"Atom Loss"* during unsupervised training further improved cluster quality by introducing attractive and repulsive forces between the latent features. gt-GMM and auto-GMM performance differences indicate that the model is sensitive to clustering model selection; an automatic clustering model selection algorithm could be developed to reduce such differences. Another limitation of the current model is that it is not suitable for online processing which would be needed for brain machine interfaces (BMI).

References

[1] Michael S. Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network (Bristol, England)*, 9(4):53–78, 11 1998.

[2] R Quian Quiroga, Z Nadasdy, and Y Ben-Shaul. Unsupervised Spike Detection and Sorting with Wavelets and Superparamagnetic Clustering. *Neural Computation*, 16(8):1661–1687, 8 2004.

[3] R.J. Vogelstein, Kartikeya Murari, P.H. Thakur, Chris Diehl, Shantanu Chakrabartty, and Gert Cauwenberghs. Spike sorting with support vector machines. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 3, pages 546–549. IEEE, 2004.

[4] Kai Yang, Haifeng Wu, and Yu Zeng. A Simple Deep Learning Method for Neuronal Spike Sorting. *Journal of Physics: Conference Series*, 910:012062, 10 2017.

[5] Junsik Eom, In Yong Park, Sewon Kim, Hanbyol Jang, Sanggeon Park, Yeowool Huh, and Dosik Hwang. Deep-learned spike representations and sorting via an ensemble of auto-encoders. *Neural Networks*, 134:131–142, 2 2021.

[6] 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, 10 2012.