SimSort: A Data-Driven Framework for Spike Sorting by Large-Scale Electrophysiology Simulation

Yimu Zhang[1*] **Dongqi Han**[2†] **Yansen Wang**[2] **Zhenning Lv**[1]
Yu Gu[1†] **Dongsheng Li**[2]
[1] Institutes of Brain Science, Fudan University
[2]Microsoft Research Asia
{yimuzhang21, znlv23}@m.fudan.edu.cn, {guyu_}@fudan.edu.cn,
{dongqihan,yansenwang,dongsli}@microsoft.com

Abstract

Spike sorting is an essential process in neural recording, which identifies and separates electrical signals from individual neurons recorded by electrodes in the brain, enabling researchers to study how specific neurons communicate and process information. Although there exist a number of spike sorting methods which have contributed to significant neuroscientific breakthroughs, many are heuristically designed, making it challenging to verify their correctness due to the difficulty of obtaining ground truth labels from real-world neural recordings. In this work, we explore a data-driven, deep learning-based approach. We begin by creating a large-scale dataset through electrophysiology simulations using biologically realistic computational models. We then present **SimSort**, a pretraining framework for spike sorting. Trained solely on simulated data, SimSort demonstrates zero-shot generalizability to real-world spike sorting tasks, yielding consistent improvements over existing methods across multiple benchmarks. These results highlight the potential of simulation-driven pretraining to enhance the robustness and scalability of spike sorting in experimental neuroscience.

1 Introduction

Understanding the complex computations performed by the brain requires insight into the activity of individual neurons [1, 2], which is crucial for exploring how information is encoded, processed, and transmitted within neural circuits, as well as decoding the brain's functional dynamics [3, 4, 5]. Recent advances in neural recording technologies have enabled capturing the activity of neurons across multiple regions of the brain with high spatial and temporal precision [6, 7, 8, 9]. Notably, extracting meaningful information from recordings relies on a critical step known as **spike sorting**.

Spike sorting is the process of extracting and identifying neural activity from extracellular recordings. It involves two main steps (Fig. 1): **spike detection**, which extracts spike events from background noise, and **spike identification**, which assigns these detected spikes to individual neurons [10]. Spike sorting is indispensable for transforming raw electrical signals into interpretable data that reveal the firing patterns of individual neurons. Accurate spike sorting is essential for linking neural activity to behavior, understanding the functional organization of neural circuits [11], and uncovering mechanisms underlying various sensory and cognitive processes [12]. Furthermore, its role extends to translational applications, such as improving neural prosthetics and developing closed-loop BCIs, where spike sorting is critical for achieving precise neural decoding and control [13, 14].

[*]The work was conducted during the internship of Yimu Zhang at Microsoft Research Asia.
[†]Corresponding authors.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

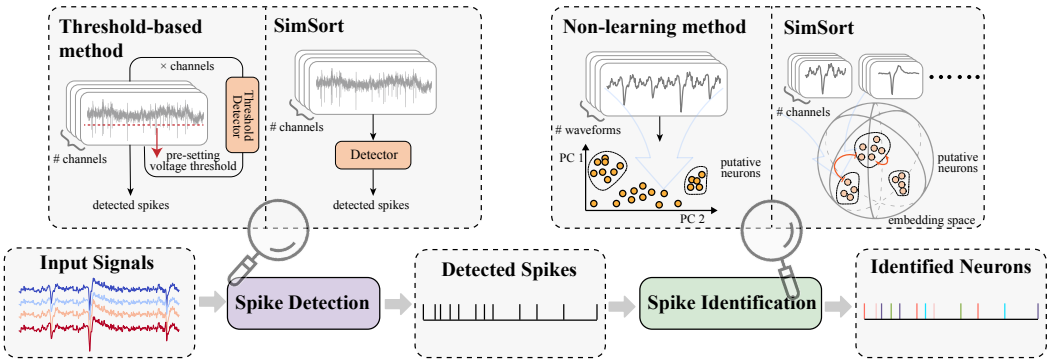

Figure 1: Pipeline of **spike sorting**, which consists of two main steps: **spike detection** and **spike identification**. For spike detection, typical spike sorting algorithms (left) utilize threshold-based detector relies on fixed voltage thresholds on each channel, and use non-learning method like performing PCA-based clustering on concatenated waveforms for spike identification. Those approaches are sensitive to noise and require manual parameter tuning. In contrast, our proposed framework SimSort (right) use a neural network-based detector replaces the threshold method for spike detection, enhancing robustness and generalization. For spike identification, the feature embeddings of multi-channel waveforms learned from contrastive learning to improve clustering accuracy.

For a long time, spike sorting predominantly relied on heuristic statistics and machine learning approaches [1, 15, 16, 17, 6, 18, 19]. While these methods have promoted neuroscience research, they exhibit several key limitations. First, their sorting results are sensitive to parameter settings and post-processing, which depend on the experimenter's expertise and must be customized for each dataset. Moreover, these approaches lack a data-driven foundation, limiting their scalability and adaptability across diverse experimental settings, particularly in low signal-to-noise ratio (SNR) and high-variability scenarios, where their performance declines significantly. Recently, deep learning-based methods, such as YASS [20] and CEED [21], have attempted to improve spike sorting by adopting data-driven approaches to enhance detection and clustering accuracy. However, their generalizability and practical use are hindered by the limited training datasets.

To address these limitations, we highlight the importance of using massive training data with ground-truth annotations to achieve more reliable and robust spike sorting. Furthermore, we aim to design a deep learning framework optimized for spike sorting and pretrain models for practical usage.

Therefore, in this work, we first generated a large-scale labeled dataset to address the scarcity of ground-truth data for spike sorting. Building on this, we present **SimSort**, a framework that utilizes data-driven approaches to enable fully automated spike sorting. By pretraining a spike detection model on the large-scale dataset, SimSort achieved notable improvements in detection accuracy and adaptability compared to commonly used threshold-based methods. Additionally, SimSort incorporated contrastive learning to enhance waveform feature representations, achieving better robustness against noise. We evaluated SimSort in zero-shot settings on publicly available datasets without fine-tuning, showing its effectiveness in spike sorting tasks compared to existing methods.

The key contributions of the SimSort framework include:

1) A publicly available, large-scale labeled dataset to help address the scarcity of ground-truth data, supporting the development and evaluation of learning-based spike sorting methods.

2) A pretraining paradigm for spike sorting with a large-scale simulation dataset. The results, for the first time, demonstrate successful zero-shot transfer from simulated to real-world spike sorting tasks.

3) An off-the-shelf pretrained model for neuroscientists to use*, enabling fully automated spike sorting on Tetrode recordings without dependence on hand-tuned parameters.

2 Preliminaries

The goal of **spike sorting** (Figure 1) is to extract single-neuron spiking activity from extracellular recordings, where the activities of an unknown number of neurons are mixed together. Mathematically,

*See `https://SimSortTool.github.io` for the model, code and usage instructions.

spike sorting is a blind source separation problem [22] and can be formally defined as follows: Let $\mathbf{V} \in \mathbb{R}^{T \times C}$ represent the extracellular recording (voltage) across C electrode channels over T time points. The goal is to infer a set of spike times $\{t_k\}_{k=1}^{K}$ and corresponding neuron labels $\{y_k\}_{k=1}^{K}$, where $y_k \in \{1, \ldots, N\}$ represents the neuron identity, and N is the total number of detected neurons.

3 Related Work

Early spike sorting pipelines typically used thresholds to detect spike events, followed by dimensionality reduction (e.g., PCA or t-SNE) and clustering [23, 24, 25]. Although this pipeline was straightforward, it often suffered from noise susceptibility, inaccurate detection of low SNR events, and reliance on manual parameter tuning.

Deep learning approaches have been adopted to replace or enhance certain steps in traditional pipelines, aiming to improve robustness, automation, and adaptability to diverse experimental conditions. Autoencoders have been explored as a method for dimensionality reduction [26, 27, 28]. YASS [20] employed a convolutional neural network for spike detection and waveform cleaning, thereby mitigating clustering errors caused by distorted waveforms. CEED [21] applied a contrastive learning framework to enforce invariances to amplitude fluctuations, noise, and channel subset changes in the extracted embeddings. However, these methods were limited by their reliance on restricted training datasets, potentially reducing their generalization across diverse experimental conditions.

In particular, YASS depended on high-quality prior training data and assumed a consistent experimental setup with validated sorting results, limiting its utility in scenarios where training data were scarce or recording conditions varied significantly.

Similarly, CEED had its limitations: (1) its invariance assumptions may not generalize to datasets with unaccounted variability; (2) it relied on KiloSort2.5-processed data, potentially inheriting inaccuracies from these analyses; (3) its training and testing datasets were narrowly scoped, originating from similar experimental conditions, which may constrain broader applicability; and (4) it did not optimize spike detection or provide a complete spike sorting pipeline.

To overcome these limitations, we propose a data-driven approach to develop a fully automated spike sorting pipeline that does not rely on manually defined parameters, demonstrating improved generalization across diverse datasets and recording conditions.

4 Methods

4.1 Dataset Creation

To address the scarcity of labeled spike data, we generated a large-scale synthetic dataset using biophysically detailed neuron models from the Blue Brain Project (BBP) [29, 30, 31], covering 206 neuron models from layers 1–6 of the juvenile rat somatosensory cortex. These models capture diverse morphologies and electrophysiological dynamics across 30 neuronal types.

Electrophysiology Simulation Based on these models, we first simulated intracellular activity by injecting noise currents into multi-compartment neurons using the NEURON simulator [32], producing realistic spiking responses. Then, extracellular potentials were computed via volume conductor theory, summing transmembrane currents across compartments to obtain virtual recordings from tetrode electrodes randomly placed near the neurons. Each simulation trial included 5 neurons of varying types, placed in a $100 \times 100 \times 100$ μm^3 volume, with randomized positions for both cells and electrodes. (Figure 2; full technical details in Appendix A).

Dataset Preparation We generated a large-scale dataset consisting of 8192 simulated recording trials, representing continuous neuronal activities from more than 40,000 individual neurons. Intracellular spike timestamps were recorded as ground-truth for extracellular spikes, providing precise annotations for model training and evaluation.

The dataset was structured into two primary subsets:

Continuous Signal Dataset: This subset comprised full-length extracellular recordings across all electrode channels. It was designed for training spike detection models and evaluating the overall performance of spike sorting pipelines.

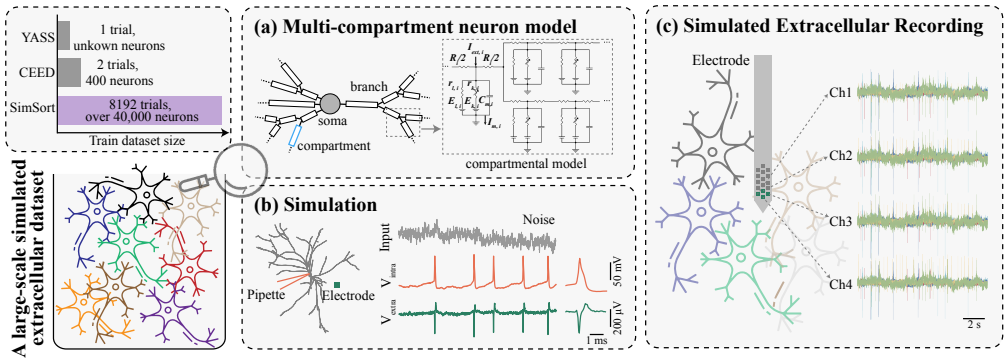

Figure 2: Overview of the large-scale simulated extracellular dataset generation process. (a) Multi-compartment neuron models simulate detailed neuronal morphologies and electrophysiological properties, incorporating realistic ion channel dynamics. (b) Noise generated by stochastic process is injected into the somatic compartment to induce stochastic intracellular action potential firing, and extracellular signals are recorded using virtual electrodes placed near the neurons in a simulated environment. (c) The resulting multi-channel extracellular recordings capture diverse and realistic neural activity.

Spike Waveform Dataset: This subset contained spike waveforms extracted from each ground-truth units. It was used for training and evaluating the spike identification model.

We used signals simulated from BBP layers 1–5 neuron models for training and validation, reserving signals from BBP layer 6 neuron models exclusively for evaluation, ensured that the test set comprises previously unseen neuron types and configurations, allowing us to effectively assess the model's generalization capabilities.

4.2 Spike Detection

We formulate the spike detection task as identifying the temporal segments $\{t_k\}$ corresponding to putative neural spikes from the raw input $\mathbf{V} \in \mathbb{R}^{T \times C}$, where T represents the number of time points and C denotes the number of electrode channels. The raw signal $\mathbf{V}$ first underwent two preprocessing steps: bandpass filtering and spatial whitening. Then, the processed signal was subsequently provided as input to the spike detection model.

We utilized a Transformer-based architecture [33] for spike detection, trained on the simulated continuous signal dataset detailed in Sec. 4.1. For robustness and generalization, data augmentation was integrated into the training process, referring to Fig. 3. We employed a binary cross-entropy loss for the spike detection model:

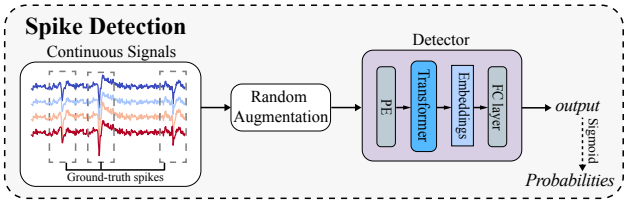

Figure 3: Spike detection model in SimSort.

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{T}\sum_{t=1}^{T}\Big(w_p \cdot y_t \log \hat{y}_t + (1-y_t)\log(1-\hat{y}_t)\Big), \tag{1}$$

where the binary label y_t denoted spike events, $\hat{y}_t$ was the predicted probability, T was trial length, and w_p was a hyperparameter to reweigh the positive predictions given the general scarcity of spikes.

Data Augmentation: To improve robustness and generalization in spike detection, data augmentation was applied to each input signal segment $V' \in \mathbb{R}^{T \times C}$ (Fig. 3). A subset of channels $C' \subseteq \{1, 2, \ldots, C\}$ was randomly selected with a probability p to apply augmentation. Various augmentations, such as adding noise, amplitude scaling, and temporal jitter, were applied to enhance the diversity of training data. The detailed implementation of these augmentations is provided in Appendix A.

4.3 Spike Identification

We formulate spike identification as embedding each identified spike waveform $X_i \in \mathbb{R}^{L\times C}$, the voltage trace (waveform) of spike i, into a latent space to learn robust representations using contrastive learning. Then the learned embeddings are clustered into several groups, indicating putative neurons.

The spike identification model was trained on the simulated spike waveform dataset (Sec. 4.1). The model consisted of several key components, as illustrated in Fig. 4 and described as follows.

Contrastive Learning: We leveraged contrastive learning to learn representations that capture the relationships between spike waveforms (Fig. 4). Contrastive learning sought to map neural data into an embedding space where related examples were positioned nearby, while unrelated examples were placed further apart [34, 35]. For each waveform X_i, we constructed a triplet by selecting X_i^+ with the same label and X_i^- with a different label. The objective was [36]:

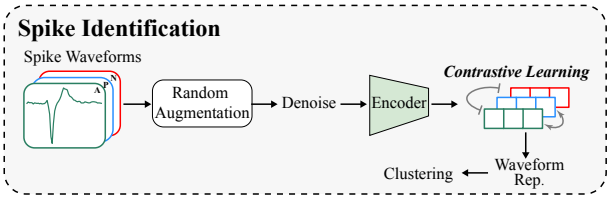

Figure 4: Spike identification model in SimSort.

$$\mathcal{L}_{\text{triplet}} = \max\left(0, \|f(X_i) - f(X_i^+)\|_2^2 - \|f(X_i) - f(X_i^-)\|_2^2 + \alpha\right), \tag{2}$$

where $f(X_i)$ denoted the embedding of X_i, and α was a margin enforcing separation between positive-negative pairs. Additionally, we uniformly applied data augmentation to each triplet waveforms. The augmentation methods followed those in Sec. 4.2, with adjusted parameters (Appendix A) for better waveform representation, ensuring robustness to noise, amplitude variations, and temporal jitter.

Denoising: The augmented waveform $X_i' \in \mathbb{R}^{L\times C}$ then underwent a denoising step using Singular Value Decomposition (SVD) [37, 38]. The waveform was first reshaped and decomposed as $X_i' = U\Sigma V^\top$, where U, Σ, and V represented the singular vectors and singular values. The first k components were retained to reconstruct the waveform.

Encoder: The denoised waveform $X'_{i,\text{recon}}$ was fed into a Gated Recurrent Unit (GRU) encoder [39] (Fig. 4), which processed the input sequentially. The final hidden state h_T served as the learned representation $f(X_i)$, encapsulating the spatiotemporal features of the waveform for subsequent clustering tasks.

Clustering: After obtaining the waveform representation, we applied dimensionality reduction using UMAP [40] before clustering (Fig. 4). We tested two clustering approaches: a parametric algorithm, Gaussian Mixture Model (GMM) [41], and a non-parametric one, Mean Shift (MS) [42].

4.4 Overall Spike Sorting Pipeline

We established a cascaded spike sorting pipeline (Fig. 1). During inference, raw multi-channel signals were processed through spike detection, waveform extraction, embedding generation, and clustering, yielding distinct neuronal units and their spike timestamps.

5 Results

With the large-scale dataset and a pretraining framework, we try to answer the following research questions (RQ).
RQ1: Can the model trained on our simulated dataset generalize to real-world spike sorting tasks?
RQ2: If the model performs well, what are the crucial underlying components?
RQ3: Does data scale hold the key of effective spike sorting?

5.1 Benchmarks

Since the models of SimSort has been trained on our simulated dataset, readers must be curious about how well it can generalize to real-world spike sorting tasks. To empirically investigate this problem, we respectively evaluate SimSort in the following benchmarks:

- **Simulated test dataset** (with ground truth): BBP L6 dataset from our simulation (using different neuronal models from the training dataset).
- **Real-world data-based synthetic datasets** (with label): Hybrid [43], WaveClus [44], IBL Neuropixels [21]. We evaluate both zero-shot (using only our simulated dataset) and fine-tuned (using also the dataset) performance of SimSort.
- **Real-world neural recording** (w/o ground truth, indirect validation): extracellular recordings collected from the mouse primary visual cortex during drifting grating stimulation.

The detailed information of benchmark datasets, hyper-parameters of models, and formulation of the evaluation metrics can be found in Appendix A.

Evaluation Metrics. To evaluate the performance of the overall spike sorting and spike detection tasks, we used Accuracy, Recall, and Precision as the evaluation metrics. Note that these metrics are specifically defined for the spike sorting context, where they measure the alignment between detected spikes and ground truth spike events, rather than their conventional definitions in classification tasks. For the spike identification task, we used the Adjusted Rand Index (ARI) [45] to measure clustering accuracy relative to the ground truth.

5.2 Spike Sorting Results on simulated dataset

We first examined Simsort on a simulated test set (BBP L6). In Table 1, we evaluated SimSort alongside widely used algorithms, including KiloSort [17], KiloSort2, MountainSort4 [6], and MountainSort5, implemented via the SpikeInterface pipeline [46], on the simulated BBP L6 dataset. SimSort achieved the best performance across all metrics, demonstrating its robustness and strong zero-shot generalization ability under idealized, simulated conditions.

Table 1: Spike sorting results on **BBP L6 dataset**. The results for other methods were obtained through SpikeInterface [46]. Values are as mean ± S.E.M.

Methods	BBP L6 (20 trials)		
	Accuracy	Recall	Precision
KiloSort	0.49 ± 0.05	0.51 ± 0.05	0.53 ± 0.05
KiloSort2	0.51 ± 0.06	0.53 ± 0.06	0.53 ± 0.06
MountainSort4	0.84 ± 0.03	0.84 ± 0.03	**0.93 ± 0.03**
MountainSort5	0.66 ± 0.06	0.68 ± 0.06	0.79 ± 0.07
SimSort	**0.85 ± 0.02**	**0.90 ± 0.01**	**0.93 ± 0.02**

Table 2: Spike sorting results on **Hybrid dataset**. Data for other methods were sourced from SpikeForest [43]. Values are mean ± S.E.M. Best results are in **bold**. Note that SimSort did not use this dataset for training, indicating a zero-shot generalization.

Methods	Hybrid-static (SNR>3, 9 recordings)			Hybrid-drift (SNR>3, 9 recordings)		
	Accuracy	Recall	Precision	Accuracy	Recall	Precision
HerdingSpikes2	0.35 ± 0.01	0.44 ± 0.02	0.53 ± 0.01	0.29 ± 0.01	0.37 ± 0.02	0.48 ± 0.02
IronClust	0.57 ± 0.04	**0.81 ± 0.01**	0.60 ± 0.04	0.54 ± 0.03	0.71 ± 0.02	0.65 ± 0.03
JRClust	0.47 ± 0.04	0.63 ± 0.02	0.59 ± 0.03	0.35 ± 0.03	0.48 ± 0.03	0.57 ± 0.02
KiloSort	0.60 ± 0.02	0.65 ± 0.02	0.72 ± 0.02	0.51 ± 0.02	0.62 ± 0.01	**0.72 ± 0.03**
KiloSort2	0.39 ± 0.03	0.37 ± 0.03	0.51 ± 0.03	0.30 ± 0.02	0.31 ± 0.02	0.57 ± 0.04
MountainSort4	0.59 ± 0.02	0.73 ± 0.01	0.73 ± 0.03	0.36 ± 0.02	0.57 ± 0.02	0.61 ± 0.03
SpykingCircus	0.57 ± 0.01	0.63 ± 0.01	0.75 ± 0.03	0.48 ± 0.02	0.55 ± 0.02	0.68 ± 0.03
Tridesclous	0.54 ± 0.03	0.66 ± 0.02	0.59 ± 0.04	0.37 ± 0.02	0.52 ± 0.03	0.55 ± 0.04
SimSort	**0.62 ± 0.04**	0.68 ± 0.04	**0.77 ± 0.03**	**0.56 ± 0.03**	**0.63 ± 0.03**	0.69 ± 0.03

Table 3: Spike sorting results on **WaveClus dataset**. The results for other methods were obtained through SpikeInterface [46]. Values are mean ± S.E.M. Note that SimSort did not use this dataset for training, indicating a zero-shot generalization.

Methods	Easy (12 recordings)			Difficult (8 recordings)		
	Accuracy	Recall	Precision	Accuracy	Recall	Precision
KiloSort	0.54 ± 0.12	0.46 ± 0.15	0.62 ± 0.13	0.17 ± 0.07	0.23 ± 0.10	0.18 ± 0.07
KiloSort2	0.61 ± 0.09	0.66 ± 0.09	0.67 ± 0.09	0.10 ± 0.07	0.12 ± 0.09	0.10 ± 0.07
MountainSort4	0.73 ± 0.08	0.79 ± 0.07	0.79 ± 0.08	0.48 ± 0.16	0.49 ± 0.16	0.52 ± 0.17
MountainSort5	0.71 ± 0.09	0.74 ± 0.08	0.79 ± 0.10	0.42 ± 0.15	0.46 ± 0.15	0.46 ± 0.16
SimSort	**0.75 ± 0.06**	**0.84 ± 0.04**	**0.85 ± 0.05**	**0.71 ± 0.10**	**0.81 ± 0.07**	**0.81 ± 0.09**

5.3 Spiking Sorting results on real-world data-based synthetic datasets

zero-shot generalization Given SimSort's strong performance on simulated data, we extended our evaluation to real-world data-based datasets to further test its zero-shot generalizability and robustness (RQ1). On the Hybrid dataset, we compared SimSort against several algorithms via SpikeForest, including HerdingSpikes2 [18], IronClust, JRClust, KiloSort [17], KiloSort2, MountainSort4 [6], SpykingCircus, and Tridesclous. The results are summarized in Table 2 and Fig. S1. On this dataset, SimSort demonstrated the highest Accuracy and Precision in static recordings and achieved the best Accuracy and Recall under drift conditions, although it slightly trailed KiloSort in Precision.

For WaveClus dataset, we used SpikeInterface pipeline to run KiloSort, KiloSort2, MountainSort4, and MountainSort5. Results are provided in Table 3 and Fig. S1. SimSort performed reliably across both easy and difficult subsets, showing a pronounced advantage on the challenging "difficult" set. This performance was likely due to its ability to handle the unique challenges of this subset, such as lower SNRs and reduced inter-class waveform variability, which resulted in significant overlaps between units. These results underscored SimSort's effectiveness in noisy and complex scenarios.

Fine-tuning Beyond demonstrating SimSort's zero-shot transfer ability on unseen datasets, we also investigate its performance after fine-tuning on a small amount of data from the target domain (Table 4). While the overall improvements in sorting metrics are modest, we observe notable changes in spike detection performance, particularly in the balance between Precision and Recall. These changes suggest that fine-tuning helps the model adapt to dataset-specific waveform and noise characteristics, enhancing its practicality with minimal labeled data.

Table 4: Performance metrics of SimSort after fine-tuning on different datasets.The hybrid-static and hybrid-drift datasets were fine-tuned on recordings 1-3 and tested on recordings 4-9 (train/test set split). The waveclus -easy and -difficult datasets were fine-tuned on subsets easy1/diff1 and tested on easy2/diff2, respectively. The IBL Neuropixels dataset was fine-tuned on 100 neurons and evaluated on 50 test sets, each with 10 randomly selected neurons.

Dataset	Model	Spike Detection (Acc/Rec/Prec)	Spike Identification (ARI)	Spike Sorting (Acc/Rec/Prec)
Hybrid-static	SimSort	0.71±0.02 / 0.83±0.02 / 0.83±0.02	0.90±0.02	0.59±0.03 / 0.65±0.03 / 0.76±0.02
	SimSort-FT	0.75±0.03 / 0.89±0.02 / 0.82±0.02	0.92±0.02	0.58±0.03 / 0.65±0.03 / 0.73±0.02
Hybrid-drift	SimSort	0.67±0.02 / 0.80±0.02 / 0.81±0.01	0.88±0.03	0.54±0.09 / 0.61±0.09 / 0.67±0.07
	SimSort-FT	0.73±0.03 / 0.85±0.03 / 0.83±0.01	0.88±0.02	0.56±0.04 / 0.63±0.03 / 0.69±0.03
Waveclus-easy	SimSort	0.59±0.02 / 0.99±0.00 / 0.59±0.02	0.91±0.01	0.70±0.11 / 0.79±0.07 / 0.84±0.09
	SimSor-FT	0.94±0.01 / 0.96±0.00 / 0.98±0.01	0.91±0.01	0.73±0.08 / 0.79±0.05 / 0.91±0.04
Waveclus-difficult	SimSort	0.66±0.08 / 0.99±0.00 / 0.67±0.08	0.88±0.01	0.76±0.14 / 0.84±0.09 / 0.84±0.12
	SimSort-FT	0.95±0.01 / 0.96±0.00 / 0.99±0.01	0.87±0.03	0.77±0.14 / 0.82±0.10 / 0.86±0.13
IBL Neuropixels	SimSort	N/A	0.49±0.09	N/A
	SimSort-FT	N/A	0.53±0.09	N/A

5.4 Real-World Neural Recording

Beyond quantitative benchmarks, we further assessed SimSort in a real experimental setting. Specifically, we applied it to extracellular recordings (collected in our wet lab, see Appendix A.1.2) from the mouse primary visual cortex obtained using tetrodes during presentation of full-field drifting gratings in 8 orientations (Fig. 5a). From a representative recording session, SimSort identified six putative single units (Fig. 5b). These units exhibited distinct spike waveforms and their autocorrelograms showed clear refractory period structure(Fig. 5c), suggesting good isolation. We further analyzed the orientation tuning of these units. Several exhibited selective responses, with global orientation selectivity index (gOSI) values exceeding 0.2 (Fig. 5d; Appendix A for definition). Across 10 tetrode recordings, SimSort isolated a total of 40 units, with a mean gOSI of 0.27 ± 0.14 (mean ± S.D., Fig. S17). These results demonstrate the practical applicability of SimSort to real neural recordings.

5.5 Detailed Analysis

To explain why our framework achieved higher overall spike sorting accuracy (RQ2), we dissected SimSort and analyzed each component, i.e., spike detection and spike identification models. Furthermore, we presented some case studies of the latent embedding of spikes from multiple neurons.

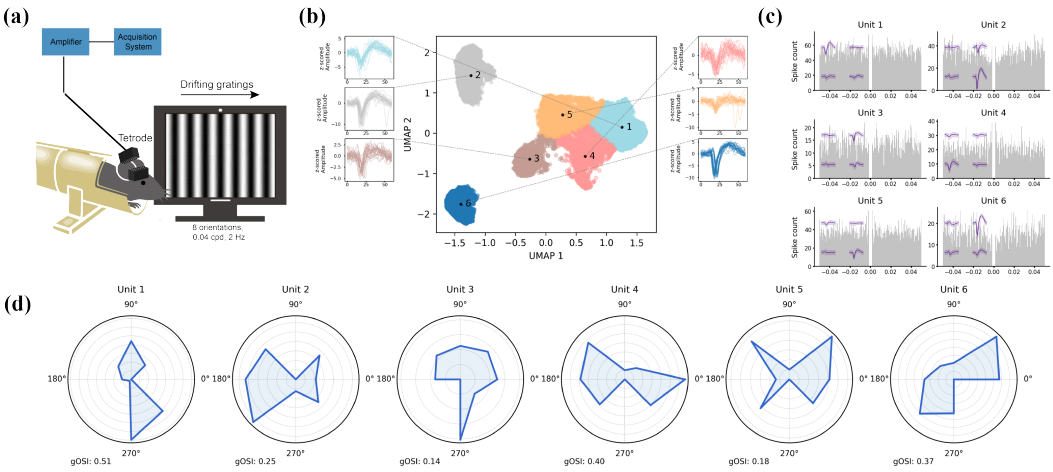

Figure 5: Evaluation of SimSort on real experimental data (See Fig. S17 for additional recording results). (a) Experimental setup: extracellular recordings from the mouse primary visual cortex during drifting gratings in eight directions. (b) Visualization of SimSort's sorting results. (c) Autocorrelograms and average waveform of each unit. (d) Orientation tuning curves and global orientation selectivity index (gOSI) for each unit.

Spike Detection We first compared our detection model to a commonly used threshold-based detection method (see Appendix A for details). The results are shown in Table 5a. SimSort's spike detector consistently outperformed the threshold-based method across all metrics on the Hybrid dataset. In the static condition, SimSort improved accuracy by $\sim$18%, recall by $\sim$18%, and precision by $\sim$1%. Under the drift condition, the improvements were $\sim$13% in accuracy, $\sim$17% in recall, and $\sim$1% in precision. It is important to note that the performance of the Threshold detector was highly sensitive to the choice of threshold, with results varying significantly across different settings. The results shown here were obtained under the best-performing thresholds. In contrast, SimSort achieved its superior performance without manual tuning, demonstrating its robustness and adaptability to varying recording conditions. These results indicated the effectiveness of our detection model, providing a solid foundation for the overall performance of SimSort in spike sorting tasks.

Table 5: Spike detection and identification results on the **Hybrid dataset**. Values are mean $\pm$ S.E.M.

(a) Spike detection results. The threshold detector used the best-performing threshold by grid search.

Dataset	Method	Accuracy	Recall	Precision
Static	Threshold detector	0.61±0.03	0.71±0.02	0.81±0.02
	SimSort's detector	**0.72±0.03**	**0.84±0.02**	**0.82±0.02**
Drift	Threshold detector	0.60±0.03	0.70±0.03	0.80±0.02
	SimSort's detector	**0.68±0.03**	**0.82±0.02**	**0.81±0.02**

(b) Spike identification results. All results averaged across 20 repeats of GMM.

Dataset	PCA	*t*-SNE	UMAP	Ours
Hybrid-static	0.79 ± 0.04	0.88 ± 0.02	0.88 ± 0.02	**0.91 ± 0.02**
Hybrid-drift	0.74 ± 0.04	0.85 ± 0.02	0.85 ± 0.01	**0.89 ± 0.03**

Spike Identification After detecting spikes, our spike identification model extracted waveform representations in the latent space, enabling subsequent clustering. To effectively demonstrate the superiority of our identification model, we assembled a waveform dataset by gathering waveform data from each neuron in every hybrid recording. Using this dataset, we compared the 2D UMAP embedding of our model's inferred representations with features extracted directly from waveforms using widely adopted dimensionality reduction techniques, including **PCA** [23], ***t*-SNE** [24], and **UMAP** [40]. For evaluation, we applied a parametric clustering method, **GMM** [41], to compute the Adjusted Rand Index (ARI), providing a quantitative measure of the reliability of the representations.

As shown in Table 5b, SimSort substantially outperformed these conventional methods under both static and drift conditions, achieving the highest average ARI values of **0.91** and **0.89**, respectively. These results revealed the robustness and effectiveness of our approach in deriving meaningful and reliable representations from extracellular data.

Additionally, we compared SimSort with CEED [21], a recent framework for representation learning on extracellular data, using the author-provided model. Note that CEED is trained on datasets derived from IBL DanLab DY016 and DY009 Neuropixels recordings, which are more closely related to our test sets. We evaluated both SimSort and CEED on fifty test sets from the **IBL Neuropixels dataset** (specifically derived from the IBL CortexLab KS046 recording). As presented in Table S4, Fig. S2, S3, our method, without training on the dataset, achieved performance comparable to CEED.

Case Studies: In Fig. 6, we visualized the results of spike identification on an example recording in the Hybrid static subset. SimSort's embeddings (first column) demonstrated superior clustering performance, with a GMM clustering ARI score of 0.90, closely aligning with true labels. In contrast, UMAP, t-SNE, and PCA embeddings (second to fourth columns) yielded lower clustering scores of 0.80, 0.78, and 0.67, respectively. SimSort embeddings showed clear separation between neuron clusters, while other methods exhibited significant overlap, highlighting the advantages of SimSort in learning robust and discriminative features.

Impact of Dimensionality Reduction and Clustering Methods: To evaluate the adaptability of SimSort, we conducted a study to assess the impact of different dimensionality reduction techniques (PCA, t-SNE, UMAP, and None) and clustering algorithms (GMM, KMeans [47], Mean Shift [42], and Spectral Clustering [48]). Dimensionality reduction was applied to latent features extracted from waveforms, with an additional setup where clustering was performed directly on the raw latent features without dimensionality reduction (None). As illustrated in Fig. S4, the combination of UMAP and Mean Shift achieved the highest accuracy on both hybrid static and drift subsets. Across many other combinations, SimSort also maintained competitive performance, demonstrating its flexibility and robustness to various configurations.

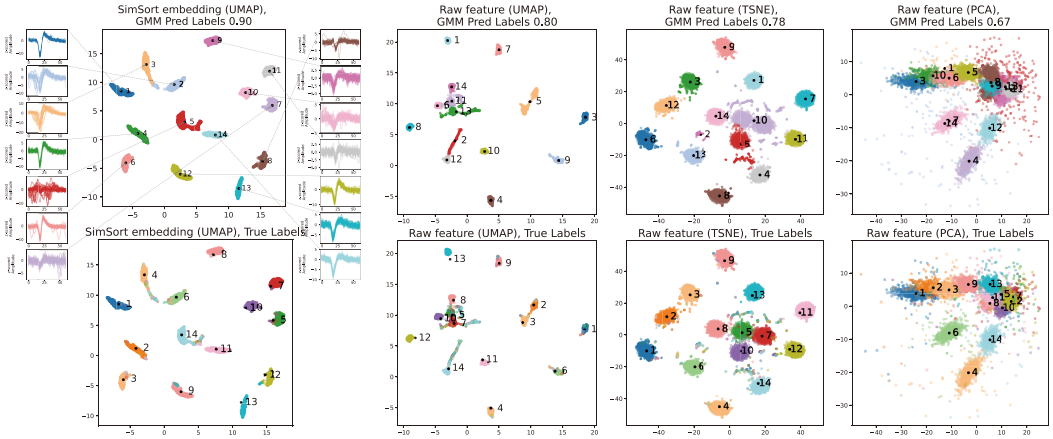

Figure 6: Visualized results of spike identification on an example recording in the Hybrid static subset. See Appendix C for visualization results for additional samples.

5.6 Scaling Law of Data Size

As we emphasized the importance of a large-scale dataset throughout this work, it was interesting to evaluate how SimSort's performance scaled with the size of the training dataset (RQ3). Specifically, we evaluated performance across sorting accuracy, detection accuracy, and identification ARI score, capturing how each metric varied with the growing size of the training dataset, which ranged from 2^8 to 2^{13} trials. As Fig. 7 shows, larger datasets led to almost consistent performance improvements on both hybrid static and drift subsets, reflecting that the large-scale dataset holds the key of SimSort's effectiveness.

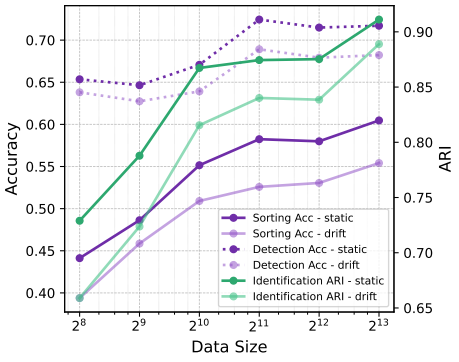

Figure 7: Performance scaling with increasing training data size on the Hybrid-drift and -static.

6 Conclusion

In this paper, we presented SimSort, a data-driven framework for automated spike sorting, built upon a biologically realistic large-scale extracellular dataset. SimSort achieves competitive or superior performance compared to commonly used spike sorting algorithms on both simulated and real-world benchmarks, demonstrating its robustness and scalability. Evaluations further highlighted its zero-shot transfer capability.

SimSort has limitations. The models were trained on 4-channel tetrode data. While SimSort shows promising performance on tasks with high-density probes (IBL Neuropixels), future work should extend the training with diverse electrode geometries.

Acknowledgments and Disclosure of Funding

This work was supported by AI for Science Foundation of Fudan University (FudanX24AI046 to Y.G.) and Microsoft Research.

References

[1] Michael S. Lewicki. A review of methods for spike sorting: The detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53, 1998.

[2] György Buzsáki. Large-scale recording of neuronal ensembles. *Nature Neuroscience*, 7(5):446–451, 2004.

[3] Rodrigo Quian Quiroga and Stefano Panzeri. Extracting information from neuronal populations: Information theory and decoding approaches. *Nature Reviews Neuroscience*, 10(3):173–185, 2009.

[4] Abhilasha Joshi, Eric L. Denovellis, Abhijith Mankili, Yagiz Meneksedag, Thomas J. Davidson, Anna K. Gillespie, Jennifer A. Guidera, Demetris Roumis, and Loren M. Frank. Dynamic synchronization between hippocampal representations and stepping. *Nature*, 617(7959):125–131, 2023.

[5] Soujatya Sarkar. Advanced Spike Sorting Approaches in Implantable VLSI Wireless Brain Computer Interfaces: A Survey. In *2024 IEEE Region 10 Symposium (TENSYMP)*, pages 1–7, 2024.

[6] 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.e6, 2017.

[7] Nicholas A. Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, Susu Chen, Jennifer Colonell, Richard J. Gardner, Bill Karsh, Fabian Kloosterman, Dimitar Kostadinov, Carolina Mora-Lopez, John O'Callaghan, Junchol Park, Jan Putzeys, Britton Sauerbrei, Rik J. J. van Daal, Abraham Z. Vollan, Shiwei Wang, Marleen Welkenhuysen, Zhiwen Ye, Joshua T. Dudman, Barundeb Dutta, Adam W. Hantman, Kenneth D. Harris, Albert K. Lee, Edvard I. Moser, John O'Keefe, Alfonso Renart, Karel Svoboda, Michael Häusser, Sebastian Haesler, Matteo Carandini, and Timothy D. Harris. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539):eabf4588, 2021.

[8] Jason E. Chung, Hannah R. Joo, Jiang Lan Fan, Daniel F. Liu, Alex H. Barnett, Supin Chen, Charlotte Geaghan-Breiner, Mattias P. Karlsson, Magnus Karlsson, Kye Y. Lee, Hexin Liang, Jeremy F. Magland, Jeanine A. Pebbles, Angela C. Tooker, Leslie F. Greengard, Vanessa M. Tolosa, and Loren M. Frank. High-Density, Long-Lasting, and Multi-region Electrophysiological Recordings Using Polymer Electrode Arrays. *Neuron*, 101(1):21–31.e5, 2019.

[9] Guosong Hong and Charles M. Lieber. Novel electrode technologies for neural recordings. *Nature Reviews Neuroscience*, 20(6):330–345, 2019.

[10] Baptiste Lefebvre, Pierre Yger, and Olivier Marre. Recent progress in multi-electrode spike sorting methods. *Journal of Physiology-Paris*, 110(4, Part A):327–335, 2016.

[11] Jérémie Sibille, Carolin Gehr, Jonathan I. Benichov, Hymavathy Balasubramanian, Kai Lun Teh, Tatiana Lupashina, Daniela Vallentin, and Jens Kremkow. High-density electrode recordings reveal strong and specific connections between retinal ganglion cells and midbrain neurons. *Nature Communications*, 13(1):5218, 2022.

[12] Yunzhe Liu, Matthew M. Nour, Nicolas W. Schuck, Timothy E. J. Behrens, and Raymond J. Dolan. Decoding cognition from spontaneous neural activity. *Nature Reviews Neuroscience*, 23(4):204–214, 2022.

[13] Jongkil Park, Gookhwa Kim, and Sang-Don Jung. A 128-Channel FPGA-Based Real-Time Spike-Sorting Bidirectional Closed-Loop Neural Interface System. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(12):2227–2238, 2017.

[14] Yuntao Han, Shiwei Wang, and Alister Hamilton. Live Demonstration: An Efficient Hardware for Real-time Multi-channel Spike Sorting with Localization. In *2024 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–1, 2024.

[15] R.Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Computation*, 16(8):1661–1687, 2004.

[16] Cyrille Rossant, Shabnam N. Kadir, Dan F. M. Goodman, John Schulman, Maximilian L. D. Hunter, Aman B. Saleem, Andres Grosmark, Mariano Belluscio, George H. Denfield, Alexander S. Ecker, Andreas S. Tolias, Samuel Solomon, György Buzsáki, Matteo Carandini, and Kenneth D. Harris. Spike sorting for large, dense electrode arrays. *Nature Neuroscience*, 19(4):634–641, 2016.

[17] Marius Pachitariu, Nicholas A Steinmetz, Shabnam N Kadir, Matteo Carandini, and Kenneth D Harris. Fast and accurate spike sorting of high-channel count probes with KiloSort. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[18] Gerrit Hilgen, Martino Sorbaro, Sahar Pirmoradian, Jens-Oliver Muthmann, Ibolya Edit Kepiro, Simona Ullo, Cesar Juarez Ramirez, Albert Puente Encinas, Alessandro Maccione, Luca Berdondini, Vittorio Murino, Diego Sona, Francesca Cella Zanacchi, Evelyne Sernagor, and Matthias Helge Hennig. Unsupervised Spike Sorting for Large-Scale, High-Density Multielectrode Arrays. *Cell Reports*, 18(10):2521–2532, 2017.

[19] Marius Pachitariu, Shashwat Sridhar, Jacob Pennington, and Carsen Stringer. Spike sorting with Kilosort4. *Nature Methods*, 21(5):914–921, 2024.

[20] Jin Hyung Lee, David E Carlson, Hooshmand Shokri Razaghi, Weichi Yao, Georges A Goetz, Espen Hagen, Eleanor Batty, E.J. Chichilnisky, Gaute T. Einevoll, and Liam Paninski. YASS: Yet Another Spike Sorter. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[21] Ankit Vishnubhotla, Charlotte Loh, Akash Srivastava, Liam Paninski, and Cole Lincoln Hurwitz. Towards robust and generalizable representations of extracellular data using contrastive learning. In *Thirty-Seventh Conference on Neural Information Processing Systems*, 2023.

[22] Alessio P. Buccino, Samuel Garcia, and Pierre Yger. Spike sorting: New trends and challenges of the era of high-density probes. *Progress in Biomedical Engineering*, 4(2):022005, 2022.

[23] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series*, 2:559–572, 1901.

[24] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[25] Rakesh Veerabhadrappa, Masood Ul Hassan, James Zhang, and Asim Bhatti. Compatibility Evaluation of Clustering Algorithms for Contemporary Extracellular Neural Spike Sorting. *Frontiers in Systems Neuroscience*, 14, 2020.

[26] Pierre Baldi. Autoencoders, Unsupervised Learning, and Deep Architectures. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[27] Tong Wu, Anikó Rátkai, Katalin Schlett, László Grand, and Zhi Yang. Learning to Sort: Few-shot Spike Sorting with Adversarial Representation Learning. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 713–716, 2019.

[28] 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, 2021.

[29] Etay Hay, Sean Hill, Felix Schürmann, Henry Markram, and Idan Segev. Models of Neocortical Layer 5b Pyramidal Cells Capturing a Wide Range of Dendritic and Perisomatic Active Properties. *PLOS Computational Biology*, 7(7):e1002107, 2011.

[30] Henry Markram, Eilif Muller, Srikanth Ramaswamy, Michael W. Reimann, Marwan Abdellah, Carlos Aguado Sanchez, Anastasia Ailamaki, Lidia Alonso-Nanclares, Nicolas Antille, Selim Arsever, Guy Antoine Atenekeng Kahou, Thomas K. Berger, Ahmet Bilgili, Nenad Buncic, Athanassia Chalimourda, Giuseppe Chindemi, Jean-Denis Courcol, Fabien Delalondre, Vincent Delattre, Shaul Druckmann, Raphael Dumusc, James Dynes, Stefan Eilemann, Eyal Gal, Michael Emiel Gevaert, Jean-Pierre Ghobril, Albert Gidon, Joe W. Graham, Anirudh Gupta, Valentin Haenel, Etay Hay, Thomas Heinis, Juan B. Hernando, Michael Hines, Lida Kanari, Daniel Keller, John Kenyon, Georges Khazen, Yihwa Kim, James G. King, Zoltan Kisvarday, Pramod Kumbhar, Sébastien Lasserre, Jean-Vincent Le Bé, Bruno R. C. Magalhães, Angel Merchán-Pérez, Julie Meystre, Benjamin Roy Morrice, Jeffrey Muller, Alberto Muñoz-Céspedes, Shruti Muralidhar, Keerthan Muthurasa, Daniel Nachbaur, Taylor H. Newton, Max Nolte, Aleksandr Ovcharenko, Juan Palacios, Luis Pastor, Rodrigo Perin, Rajnish Ranjan, Imad Riachi, José-Rodrigo Rodríguez, Juan Luis Riquelme, Christian Rössert, Konstantinos Sfyrakis, Ying Shi, Julian C. Shillcock, Gilad Silberberg, Ricardo Silva, Farhan Tauheed, Martin Telefont, Maria Toledo-Rodriguez, Thomas Tränkler, Werner Van Geit, Jafet Villafranca Díaz, Richard Walker, Yun Wang, Stefano M. Zaninetta, Javier DeFelipe, Sean L. Hill, Idan Segev, and Felix Schürmann. Reconstruction and Simulation of Neocortical Microcircuitry. *Cell*, 163(2):456–492, 2015.

[31] Steeve Laquitaine, Milo Imbeni, Joseph Tharayil, James B. Isbister, and Michael W. Reimann. Spike sorting biases and information loss in a detailed cortical model, 2025.

[32] M. L. Hines and N. T. Carnevale. The NEURON Simulation Environment. *Neural Computation*, 9(6):1179–1209, 1997.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[34] Sven Dorkenwald, Peter H. Li, Michał Januszewski, Daniel R. Berger, Jeremy Maitin-Shepard, Agnes L. Bodor, Forrest Collman, Casey M. Schneider-Mizell, Nuno Maçarico da Costa, Jeff W. Lichtman, and Viren Jain. Multi-layered maps of neuropil with segmentation-guided contrastive learning. *Nature Methods*, 20(12):2011–2020, 2023.

[35] Han Yu, Hanrui Lyu, Ethan Yixun Xu, Charlie Windolf, Eric Kenji Lee, Fan Yang, Andrew M. Shelton, Shawn Olsen, Sahar Minavi, Olivier Winter, International Brain Laboratory, Eva L. Dyer, Chandramouli Chandrasekaran, Nicholas A. Steinmetz, Liam Paninski, and Cole Hurwitz. In vivo cell-type and brain region classification via multimodal contrastive learning. *bioRxiv Preprint*, page 2024.11.05.622159, 2024.

[36] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[37] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[38] John P. Cunningham and Byron M. Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 17(11):1500–1509, 2014.

[39] Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, 2014. Association for Computational Linguistics.

[40] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, 2020.

[41] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[42] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[43] Jeremy Magland, James J Jun, Elizabeth Lovero, Alexander J Morley, Cole Lincoln Hurwitz, Alessio Paolo Buccino, Samuel Garcia, and Alex H Barnett. SpikeForest, reproducible web-facing ground-truth validation of automated neural spike sorters. *eLife*, 9:e55167, 2020.

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

[45] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.

[46] Alessio P Buccino, Cole L Hurwitz, Samuel Garcia, Jeremy Magland, Joshua H Siegle, Roger Hurwitz, and Matthias H Hennig. SpikeInterface, a unified framework for spike sorting. *eLife*, 9:e61834, 2020.

[47] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 5.1, pages 281–298. University of California Press, 1967.

[48] Andrew Ng, Michael Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[49] M. L. Hines and N. T. Carnevale. Neuron: A Tool for Neuroscientists. *The Neuroscientist*, 7(2):123–135, 2001.

[50] Nathan W. Gouwens, Jim Berg, David Feng, Staci A. Sorensen, Hongkui Zeng, Michael J. Hawrylycz, Christof Koch, and Anton Arkhipov. Systematic generation of biophysically detailed models for diverse cortical neuron types. *Nature Communications*, 9(1):710, 2018.

[51] Yazan N. Billeh, Binghuang Cai, Sergey L. Gratiy, Kael Dai, Ramakrishnan Iyer, Nathan W. Gouwens, Reza Abbasi-Asl, Xiaoxuan Jia, Joshua H. Siegle, Shawn R. Olsen, Christof Koch, Stefan Mihalas, and Anton Arkhipov. Systematic Integration of Structural and Functional Data into Multi-scale Models of Mouse Primary Visual Cortex. *Neuron*, 106(3):388–403.e18, 2020.

[52] Henrik Lindén, Espen Hagen, Szymon Leski, Eivind S. Norheim, Klas H. Pettersen, and Gaute T. Einevoll. LFPy: A tool for biophysical simulation of extracellular potentials generated by detailed model neurons. *Frontiers in Neuroinformatics*, 7, 2014.

[53] Espen Hagen, Torbjørn V. Ness, Amir Khosrowshahi, Christina Sørensen, Marianne Fyhn, Torkel Hafting, Felix Franke, and Gaute T. Einevoll. ViSAPy: A Python tool for biophysics-based generation of virtual spiking activity for evaluation of spike-sorting algorithms. *Journal of Neuroscience Methods*, 245:182–204, 2015.

[54] Espen Hagen, Solveig Næss, Torbjørn V. Ness, and Gaute T. Einevoll. Multimodal Modeling of Neural Network Activity: Computing LFP, ECoG, EEG, and MEG Signals With LFPy 2.0. *Frontiers in Neuroinformatics*, 12, 2018.

[55] Alessio Paolo Buccino and Gaute Tomas Einevoll. MEArec: A Fast and Customizable Testbench Simulator for Ground-truth Extracellular Spiking Activity. *Neuroinformatics*, 19(1):185–204, 2021.

[56] William R. Holmes. Passive Cable Modeling. In *Computational Modeling Methods for Neuroscientists*. 2009.

[57] Romain Brette and Alain Destexhe. Handbook of Neural Activity Measurement. In *Handbook of Neural Activity Measurement*, pages 92–135. Cambridge University Press, 2012.

[58] György Buzsáki, Costas A. Anastassiou, and Christof Koch. The origin of extracellular fields and currents — EEG, ECoG, LFP and spikes. *Nature Reviews Neuroscience*, 13(6):407–420, 2012.

[59] Gaute T. Einevoll, Christoph Kayser, Nikos K. Logothetis, and Stefano Panzeri. Modelling and analysis of local field potentials for studying the function of cortical circuits. *Nature Reviews Neuroscience*, 14(11):770–785, 2013.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We tested our framework and models on simulated and real-world data.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed limitations in the Conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have submitted the code and will also release the dataset upon acceptance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We release the source code. For dataset, we will release upon acceptance since the size of our simulation data are too large for an anonymous net disk.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided the full details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix A.7

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We are not aware of any violence.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See the Broader Impacts section in the beginning of Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our model is for spike sorting in neuroscience data analysis. We are not aware of any risk for misusage.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We obey the licenses.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have provided detailed information about our dataset, and will release it together with documentations upon acceptance.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Human subjects not involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Human subjects not involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Broader Impacts

Our work aims to advance the methodology of data processing in neuroscience experiments. It does not directly lead to any negative societal impact.

A Implementation Details of SimSort

A.1 Data Preparation

A.1.1 Train datasets

In Sec. 4.1, we introduced the preparation of our simulated training dataset, divided into the continuous signal dataset and the spike waveform dataset.

The **continuous signal dataset** was constructed by concatenating multiple simulated trials (e.g., 8192 trials), each containing 6,000,000 timesteps (dt = 0.1 ms) of extracellular data. Spike labels were derived by identifying peak times of intracellular action potentials and assigning binary labels to the corresponding region. This dataset, therefore, provided labeled, continuous extracellular signals for training the detection model and evaluating the overall spike sorting performance of SimSort.

The **spike waveform dataset** was constructed by extracting waveform segments centered on labeled spike events. For each spike, a 60-timestep waveform was segmented from the extracellular signals, and up to 400 such waveforms were collected per unit. Each waveform was then paired with its corresponding unit label, creating a dataset dedicated to training and evaluating the identification model.

A.1.2 Test datasets

BBP L6 dataset (simulation). This dataset comprised 20 simulated trials, generated as stated in Sec. 4.1, using BBP layer 6 neuron models. It provided 5 ground-truth units per recording for evaluating spike sorting performance.

Hybrid dataset. This dataset was composed of recordings provided by Spikeforest [43]. These recordings were generated by Kilosort, with waveform templates recorded at a 5 μm electrode spacing and 1/f background noise. It consisted of two subsets: 9 static tetrode recordings and 9 drift tetrode recordings that simulated sinusoidal probe movements. For each recording, we selected units with an SNR greater than 3 (unit IDs meeting this criterion were provided by SpikeForest to ensure a fair comparison), comprising 10–15 ground-truth units per recording. The dataset we prepared supported two modes of use. To evaluate overall spike sorting and detection performance, we used the raw data as input. For assessing identification performance, we constructed a waveform dataset by extracting 500 waveforms (60-timesteps length) per unit.

WaveClus dataset. This dataset consisted of four subsets (8 recordings in Easy1, 4 recordings in Easy2, 4 recordings in Difficult1, and 4 recordings in Difficult2). Each recording was generated by combining spike waveform templates derived from experimental recordings with background noise, with noise levels ranging from 0.05 to 0.4, and contained 3 ground-truth units. To increase the overall difficulty and adapt the original single-channel recording for tetrode analysis, we preprocessed the signals by duplicating them and modifying them with small random noise (standard deviation 0.01), amplitude scaling (ranging from 1 to 0.5), and random temporal shifts (-5 to +5 samples).

IBL Neuropixels dataset. This dataset was derived from a real-world extracellular recording, CortexLab KS046, published by the International Brain Laboratory (IBL). This recording was captured with a Neuropixels 1.0 probe across multiple brain regions from a mouse performing a decision-making behavior task. To compare spike identification performance with CEED, we prepared this dataset following the method described in the CEED paper (CEED trained and evaluated datasets derived from DanLab DY016 and DY009 Neuropixels recordings in the IBL database). We extracted waveforms from 400 units detected by Kilosort 2.5 and created fifty test sets for comparison between SimSort and CEED. Each test set contained 10 random neurons and 100 spikes per neuron. Note that CEED inputs 11-channel data into the model, while SimSort processes data from the center 4 channels.

Real-world Neural Recording. These recordings were collected by our wet lab (approved by research ethics committee). A 16-channel tetrode was implanted unilaterally in the primary visual cortex (V1) of mice. During the recording sessions, the mice were head-fixed, and the contralateral eye was stimulated with drifting grating stimuli of 8 different orientations. The stimuli were presented in a randomized sequence over 8 repeated trials. Each grating had a spatial frequency of 0.04 cycles per degree (cpd) and a temporal frequency of 2 Hz, lasting for 1.5 seconds per trial.

To quantify the orientation tuning of individual neurons, we computed the global orientation selectivity index (gOSI), as defined in the equation below:

$$gOSI = \frac{\left\|\sum R(\theta)e^{2i\theta}\right\|}{\sum R(\theta)}$$

where θ is the direction of the moving grating and $R(\theta)$ is the neuronal response to direction θ. The imaginary unit is denoted as i whose square is -1. The gOSI ranges from 0 to 1, with 0 indicating no orientation selectivity (equal response to all directions), and 1 indicating maximal selectivity (response only to a single orientation).

Table S1: Details information of test datasets used for evaluation.

Datasets	Num. Recordings	Sample Rate (Hz)	Num. Channels	Duration (sec per rec.)	Num. True Units (per rec.)	Ground-Truth Determination
BBP L6 dataset	20	10000	4	600	5	Simulation
Hybrid dataset	18	30000	4	600 / 1200	10-15	Real waveforms + background noise
WaveClus dataset	20	24000	4	60	3	Real waveforms + background noise
IBL Neuropixels dataset	/	30000	11	/	10	Sorted with KiloSort 2.5
Real-world Neural recording	10	40000	4	≈ 250	/	No Ground-truth

A.2 Technical Details of Simulation

Intracellular Simulation. Intracellular electrophysiological activity was simulated using the NEURON package [32, 49]. Pink noise, scaled to the rheobase (the minimum current needed to elicit an action potential in a neuron), was injected into each neuron's soma to evoke stochastic action potentials, simulating biologically realistic background noise. The temporal resolution was set to 0.1 ms, and simulations spanned a total duration of 600 seconds. Neurons were initialized with a resting membrane potential of -70 mV and maintained at a physiological temperature of 34°C. Synaptic mechanisms and biophysical properties were dynamically compiled and loaded to ensure compatibility with the NEURON simulation environment.

For each neuron, a multi-compartmental model [29, 30, 50, 51] was used to calculate transmembrane currents [52, 53, 54, 55]. The neuron was divided into compartments, each described as an equivalent electrical circuit. The dynamics of the membrane potential in each compartment n were governed by Kirchhoff's current law [56], where the sum of all currents entering or leaving a node must equal zero. Considering that extracellular potential changed much slower than ion channel dynamics, we assumed it constant. The temporal evolution of the membrane potential V_n in compartment n was given by $g_{n,n+1}(V_{n+1} - V_n) - g_{n-1,n}(V_n - V_{n-1}) = C_n \frac{dV_n}{dt} + \sum_j I_n^{(j)}$, where $g_{n,n+1}$ and $g_{n-1,n}$ represented the conductances between neighboring compartments, C_n was the membrane capacitance, and $\sum_j I_n^{(j)}$ accounted for ionic currents and any externally applied currents.

These transmembrane currents served as the source for extracellular potential modeling in the next simulation stage.

Extracellular Simulation. Extracellular potentials were modeled using the volume conductor theory, which links transmembrane currents to extracellular potentials recorded at electrode sites [57, 58, 59]. In each simulation trial, five neurons were randomly selected from the available models and positioned within a $100 \times 100 \times 100$ μm^3 cubic space. These neurons consisted of different types to represent the diversity of cortical microcircuits. A virtual tetrode electrode with four recording sites was placed randomly within the same space to capture extracellular potentials.

The extracellular potential at a given electrode site $\mathbf{r}_e$ was computed as the sum of contributions from all neuronal segments within the simulation space: $\phi(\mathbf{r}_e, t) = \frac{1}{4\pi\sigma} \sum_{n=1}^{N} \sum_{m=1}^{M_n} \frac{I_{n,m}(t)}{|\mathbf{r}_e - \mathbf{r}_{n,m}|}$, where

$I_{n,m}(t)$ was the transmembrane current of the m-th compartment of the n-th neuron, $\mathbf{r}_{n,m}$ was its position, σ is the extracellular medium's conductivity, and $|\mathbf{r}_e - \mathbf{r}_{n,m}|$ was the Euclidean distance between the segment and the electrode. Here, N was the total number of neurons, and M_n was the number of compartments in the n-th neuron. A reference electrode far from the source was assumed, setting $\phi = 0$ at infinity.

A.3 Spike Detection

Data Augmentation For each input segment $X_i \in \mathbb{R}^{T\times C}$, where T represents the number of time points and C denotes the number of electrode channels, a subset of channels $\mathcal{C} \subseteq \{1, 2, \ldots, C\}$ was randomly selected. A probability $p \in [0, 1]$ determined whether the following augmentations were independently applied to each channel $k \in \mathcal{C}$:

$$X'_{i,k} = \begin{cases} X_{i,k} + \epsilon, & \text{AddWithNoise,} \\ \beta \cdot X_{i,k}, & \text{RandomAmplitudeScaling,} \\ X_{i,k,\lfloor t\cdot n+\epsilon_t \rfloor}, & \text{RandomTimeJitter.} \end{cases} \tag{3}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ represents Gaussian noise, $\beta \sim \mathcal{U}(1 - s, 1 + s)$ is a random amplitude scaling factor, and $\epsilon_t \sim \mathcal{U}(-\delta, \delta)$ introduces small temporal shifts. The parameter n is an up-sampling factor enabling finer time resolution.

Model-Based Detector In Sec. 4.2, we introduced a model-based detector to predict spike events. These predictions were processed through the following steps:

1. Detection with Model: The model took as input a multi-channel signal $V \in \mathbb{R}^{t\times c}$, where t was the number of time samples and c was the number of channels. This input was processed simultaneously across all channels by the detection model, which predicted a sequence of binary labels $\hat{y}_i(t)$ for each time point:

$$\hat{y}_i(t) = f_{\text{model}}(V), \tag{4}$$

where f_{model} represented the detection model. The output $\hat{y}_i(t)$ indicated the presence or absence of spike events at each time point.

2. Adjacent Peak Merging: The predicted labels $\hat{y}_i(t)$ were refined by merging adjacent peaks to reduce false positives and consolidate overlapping detections. To identify the most relevant peaks, the channel with the maximum summed absolute signal across all time points was selected:

$$c^* = \arg\max_{c\in\{1,\ldots,C\}} \sum_{t=1}^{T} |V(t, c)|, \tag{5}$$

where $V(t, c)$ denoted the signal value at time t for channel c. Using this peak channel, adjacent peaks were grouped within a window size Δt, and only the time point with the maximum absolute signal was retained within each group:

$$t_{\max} = \arg\max_{t\in[t_s,t_e]} |V(t, c^*)|, \tag{6}$$

where $[t_s, t_e]$ denoted the boundaries of a group of adjacent peaks.

3. Spike Peak Extraction: After merging, the final spike peak positions $\{t_k\}$ were determined as the set of time points where $\hat{y}'_i(t) = 1$:

$$t_k = \{t \,|\, \hat{y}'_i(t) = 1\}. \tag{7}$$

Threshold-Based Detector In Sec. 5.5, we compared our model-based detector with a threshold-based detector. To ensure a fair comparison, the processing steps for the threshold-based detector were designed to align closely with those of the model-based detector. The threshold-based method processed the signal $V \in \mathbb{R}^{t\times c}$, where t was the number of time samples and c was the number of channels. Each channel was processed independently, as follows:

1. Local Maxima Detection: For each channel c, the signal $V(t, c)$ was scanned to identify local maxima that satisfied the following conditions:

$$V(t, c) > Th_{\text{single\_ch}}, \quad V(t, c) > V(t - r, c), \quad V(t, c) > V(t + r, c), \tag{8}$$

where $Th_{\text{single\_ch}}$ was the amplitude threshold, and $r \in \{1, \ldots, \text{loc\_range}\}$. These conditions ensured that the detected peaks were above the threshold and were local maxima within the range r. Peaks identified by this criterion were further validated within a broader range long_range to ensure they represented significant spike events.

2. Adjacent Peak Merging: For each channel c, adjacent peaks were grouped within a window size Δt, and only the time point with the maximum absolute signal was retained within each group:

$$t_{\text{max}} = \arg \max_{t \in [t_s, t_e]} |V(t, c)|, \tag{9}$$

where $[t_s, t_e]$ defined the boundaries of a cluster of adjacent peaks. This step ensured that closely spaced peaks were consolidated into a single spike.

3. Spike Peak Extraction: After processing all channels independently, the final detected spike positions for channel c were determined as:

$$t_k = \{t \mid V(t, c) = t_{\text{max}}\}. \tag{10}$$

A.4 Spike Identification

Waveform Snippet Extraction. For each detected spike at position t_k, a waveform snippet $W_k \in \mathbb{R}^{L \times c}$ was extracted from the multi-channel continuous signal $V(t, c)$, where L was the snippet length (number of time samples) and c was the number of channels. The snippet was centered around t_k and spanned a time window determined by the pre- and post-padding values:

$$W_k = V(t_k - p_{\text{pre}} : t_k + p_{\text{post}}, :), \tag{11}$$

where $V(t, c)$ represented the input signal value at time t for channel c, and p_{pre} and p_{post} were the padding values (in time samples) before and after the spike time t_k, respectively.

A.5 Definitions of Metrics

To evaluate the performance of spike sorting algorithms, we adopted standard metrics as defined in the context of spike sorting and ground-truth comparisons [43, 46]. These metrics included **accuracy**, **precision**, and **recall**, which were derived based on the matching between the sorted spike train and the ground-truth spike train.

1. Matching Spike Events: For a given sorted spike train k and ground-truth spike train l, the first step was to calculate the number of matching events ($n_{l,k}^{\text{match}}$). A sorted spike event $s_j^{(k)}$ was considered a match to a ground-truth event $t_i^{(l)}$ if the absolute time difference was less than or equal to the predefined matching window Δ, which was typically set to 1 millisecond:

$$n_{l,k}^{\text{match}} = \#\{i : |t_i^{(l)} - s_j^{(k)}| \leq \Delta \text{ for some } j\}. \tag{12}$$

2. Missed Events and False Positives: Using the number of matching events, we computed the number of missed events ($n_{l,k}^{\text{miss}}$) and false positives ($n_{l,k}^{\text{fp}}$) as follows:

$$n_{l,k}^{\text{miss}} = N_l - n_{l,k}^{\text{match}}, \quad n_{l,k}^{\text{fp}} = M_k - n_{l,k}^{\text{match}}, \tag{13}$$

where N_l was the total number of ground-truth events in l, and M_k was the total number of sorted events in k.

3. Accuracy: Accuracy ($a_{l,k}$) balanced the contributions of both missed events and false positives and was defined as:

$$a_{l,k} = \frac{n_{l,k}^{\text{match}}}{n_{l,k}^{\text{match}} + n_{l,k}^{\text{miss}} + n_{l,k}^{\text{fp}}}. \tag{14}$$

4. Precision and Recall: The precision (p_l) and recall (r_l) were calculated based on the best matching sorted unit $\hat{k}_l$ for each ground-truth unit l. The best match was determined as the sorted unit k with the highest accuracy:

$$\hat{k}_l = \arg \max_k a_{l,k}. \tag{15}$$

To identify the best matching sorted unit $\hat{k}_l$ for each ground-truth unit l, the Hungarian Algorithm was applied to establish the best one-to-one correspondence between sorted and ground-truth units. Using this best match, precision and recall were defined as:

$$p_l = \frac{n^{\text{match}}_{l,\hat{k}_l}}{n^{\text{match}}_{l,\hat{k}_l} + n^{\text{fp}}_{l,\hat{k}_l}}, \quad r_l = \frac{n^{\text{match}}_{l,\hat{k}_l}}{n^{\text{match}}_{l,\hat{k}_l} + n^{\text{miss}}_{l,\hat{k}_l}}. \tag{16}$$

A.6 Hyperparameter Search

We performed a grid search to optimize the hyperparameters for the SimSort detection and identification models. The hyperparameter ranges included the number of layers, hidden size, learning rate, number of attention heads, dropout rate, and augmentation settings. The final selected values are presented in Tables S2 and S3.

A.7 Infrastructures

Our simulation dataset are generated using CPU clusters for approximately 5000 CPU hours, which can be parallelized. Our models are trained on NVIDIA A100/V100 GPU, approximately 20 GPU hours for each time.

Table S2: Hyperparameters of SimSort detection model.

Hyperparameters	Description	Value
Num Layer	Number of Transformer encoder layers	4
Input Size	Dimensions of the input data	4
Batch Size	Number of items processed in a single operation	128
Learning Rate	Learning rate of the model	5e-4
Hidden Size	Dimension of the Transformer encoder	256
nhead	Number of Transformer attention heads	4
Dropout	Dropout probability in Transformer encoder	0.2
Weight Decay	Regularization parameter used to prevent overfitting	1e-5
Warmup Steps	Steps to warm up the learning rate	4000
Num Epochs	Number of epochs to run	20
Sigmoid Threshold	Probability threshold for classifying spike events	0.97
Noise Level	The level of noise applied to the input data	0.9
Maximum Time Jitter	Maximum variation in timing (in terms of upsampled sampling points)	5
Amplitude Scale Range	The range for scaling the amplitude of the input data	[0, 1]
Transform Probability	The probability to perform augmentations to input data	0.8
Max Channels	The maximum number of data channels augmentations are applied to	4

Table S3: Hyperparameters of SimSort identification model.

Hyperparameters	Description	Value
Num Layer	Number of GRU layers	1
Input Size	Dimensions of the input data	4
Batch Size	Number of items processed in a single operation	128
Learning Rate	Learning rate of the model	1e-4
Hidden Size	Latent dimention size of GRU	512
Weight Decay	Regularization parameter used to prevent overfitting	1e-5
Num Epochs	Number of epochs to run	10
k components	Number of components to reconstruct waveform using SVD	5
Noise Level	The level of noise applied to the input data	4
Maximum Time Jitter	Maximum variation in timing (in terms of upsampled sampling points)	50
Amplitude Scale Range	The range for scaling the amplitude of the input data	[0.5, 1]
Transform Probability	The probability to perform augmentations to input data	0.5
Max Channels	The maximum number of data channels augmentations are applied to	4

B Supplementary Figures

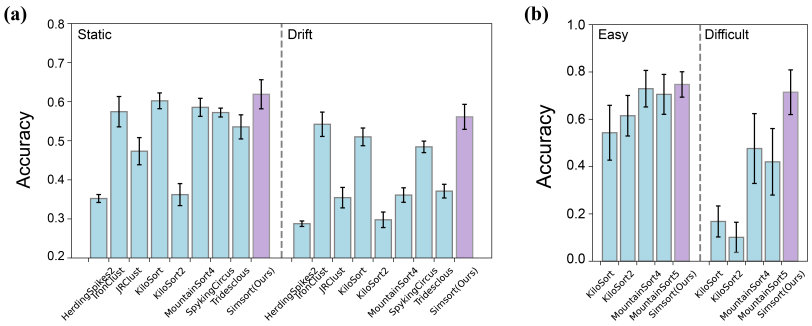

Figure S1: (a) Accuracy of spike sorting on the Hybrid dataset. SimSort significantly outperforms HerdingSpikes2 ($p = 0.0000^*$), JRClust ($p = 0.0000^*$), KiloSort2 ($p = 0.0000^*$), and Tridesclous ($p = 0.0001^*$) in static recordings, and also shows a significant advantage over IronClust ($p = 0.0296^*$). There is no significant difference compared to KiloSort ($p = 0.5023$), MountainSort4 ($p = 0.1799$), and SpykingCircus ($p = 0.1356$). In the drift condition, SimSort significantly outperforms HerdingSpikes2 ($p = 0.0000^*$), JRClust ($p = 0.0000^*$), KiloSort ($p = 0.0285^*$), KiloSort2 ($p = 0.0000^*$), MountainSort4 ($p = 0.0004^*$), SpykingCircus ($p = 0.0111^*$), and Tridesclous ($p = 0.0007^*$), with no significant difference compared to IronClust ($p = 0.3398$). (b) Accuracy of spike sorting on the WaveClus dataset. In the easy subset, SimSort shows no significant improvement compared to KiloSort ($p = 0.0770$), KiloSort2 ($p = 0.1569$), MountainSort4 ($p = 0.6137$), and MountainSort5 ($p = 0.4424$). In the difficult subset, SimSort significantly outperforms KiloSort ($p = 0.0007^*$), KiloSort2 ($p = 0.0004^*$), and MountainSort5 ($p = 0.0449^*$), with no significant difference compared to MountainSort4 ($p = 0.0935$). Statistical analysis was performed using a paired two-tailed t-test.

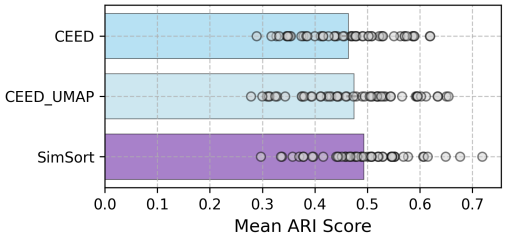

Figure S2: Performance comparison on Fifty test sets of IBL Neuropixels dataset.

Table S4: Performance comparison of spike identification on 50 test sets from IBL Neuropixels dataset. ARI values are presented as mean ± standard deviation. Statistical significance was assessed with paired t-tests: $p = 0.0012$ for SimSort vs. CEED, $p = 0.0554$ for SimSort vs. CEED+UMAP.

Methods	**Test sets (seed 0-49)**
CEED	0.46 ± 0.09
CEED+UMAP	0.47 ± 0.10
Ours	0.49 ± 0.09

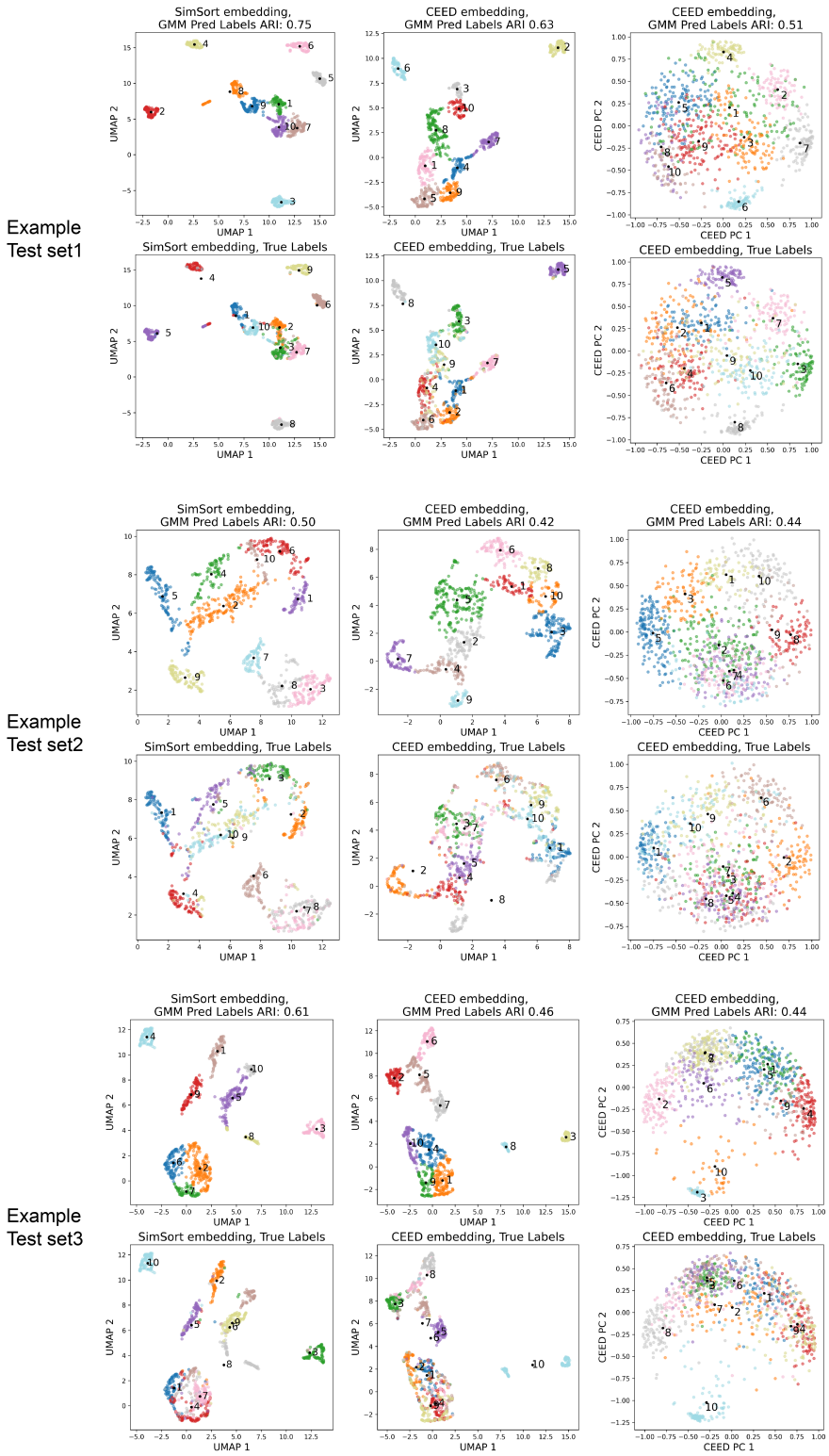

Figure S3: Visualized comparison results of three example test sets from IBL Neuropixels dataset. For each example, the first column presents GMM clustering results on UMAP-embedded SimSort representations, with predicted labels (top) and true labels (bottom). The second column displays GMM clustering on UMAP-embedded CEED representations. The third column illustrates GMM clustering directly on CEED representations (as in the CEED paper).

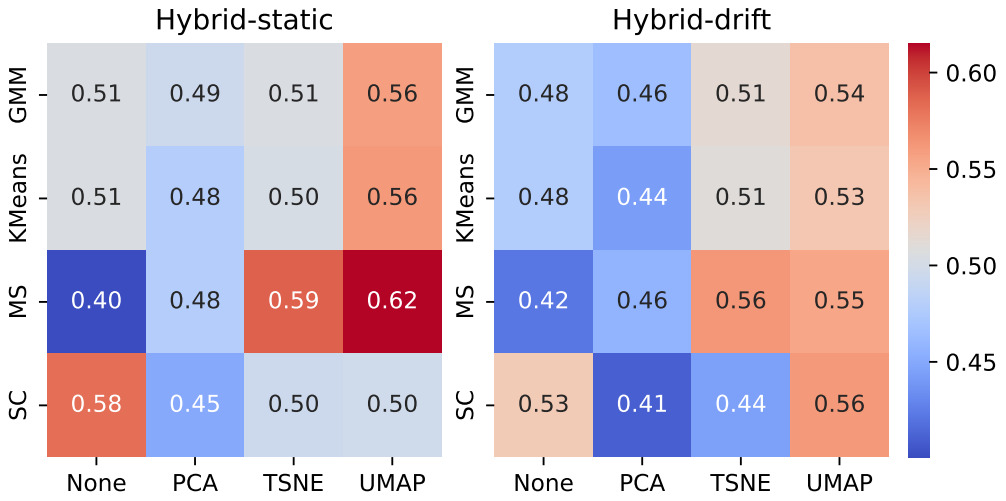

Figure S4: Impact of dimensionality reduction and clustering algorithms on the hybrid dataset. MS denotes Mean Shift, SC denotes Spectral Clustering. Comparing different combinations of dimensionality reduction methods (None, PCA, t-SNE, UMAP) and clustering algorithms (GMM, KMeans, Mean Shift, and Spectral Clustering) on static (left) and drift (right) subset. Here, "None" indicates that raw features learned by SimSort were used directly without dimensionality reduction.

C More Showcases

In this section, we present additional visualizations from various datasets to provide a more comprehensive evaluation of SimSort's performance. These visualizations include examples of spike detection, identification, and sorting tasks. The supplementary showcases illustrate key aspects such as the alignment between detected and ground-truth spikes and the clustering of spike waveforms.

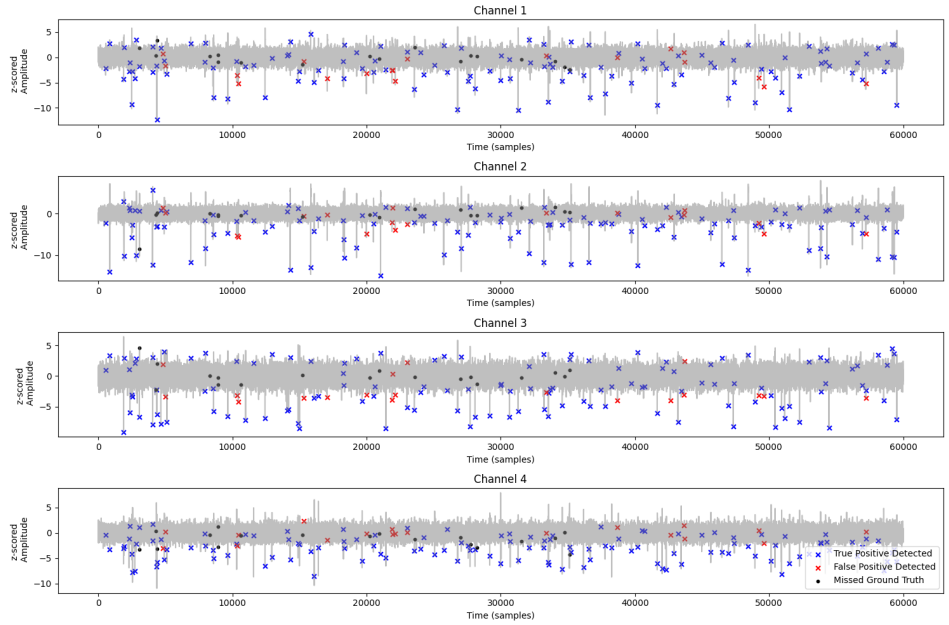

Figure S5: Visualization of spike detection performance on an example recording in Hybrid dataset.

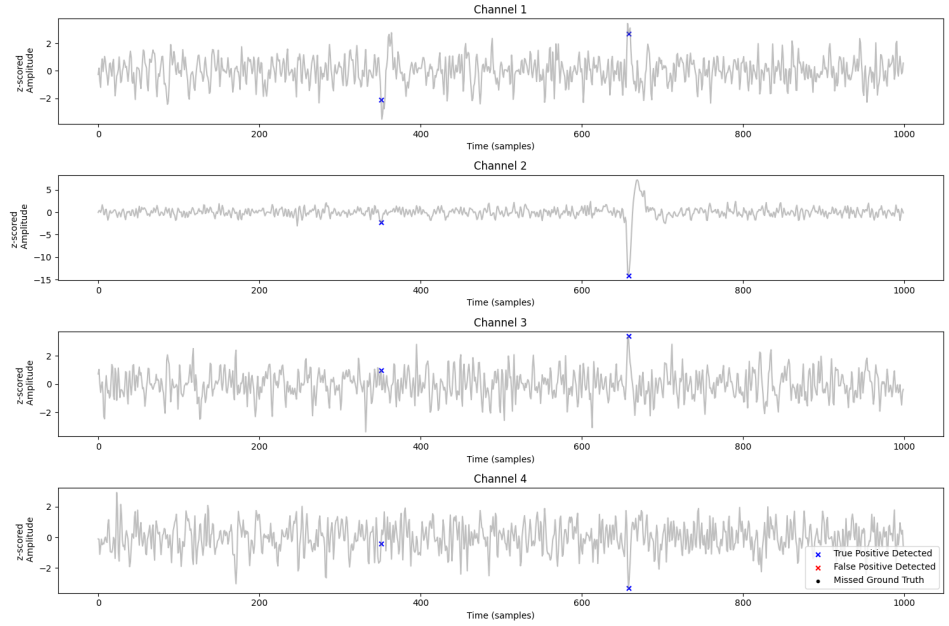

Figure S6: Zoomed-in visualization of spike detection performance on an example recording in Hybrid dataset.

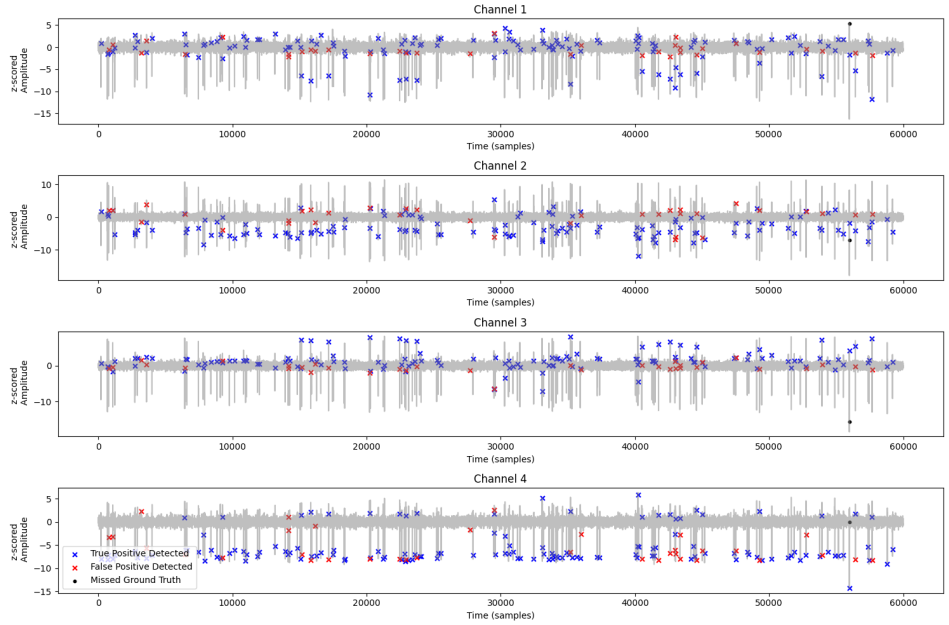

Figure S7: Visualization of spike detection performance on an example recording in WaveClus dataset.

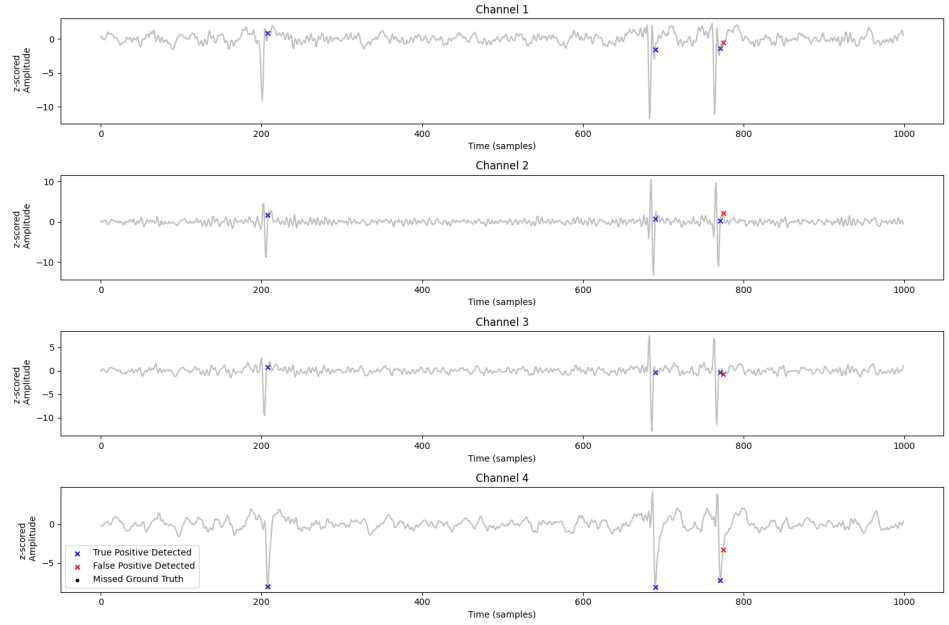

Figure S8: Zoomed-in visualization of spike detection performance on an example recording in WaveClus dataset.

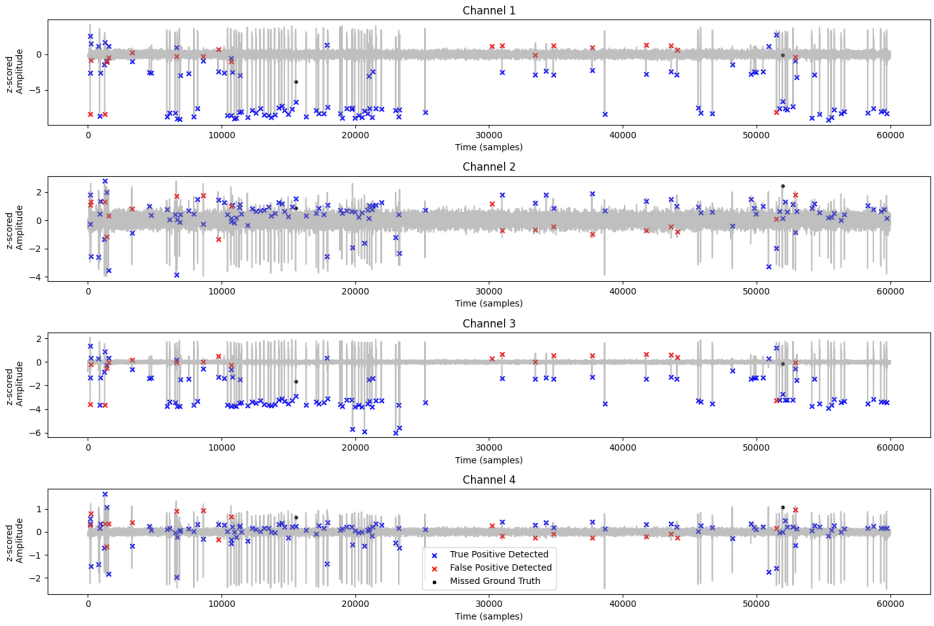

Figure S9: Visualization of spike detection performance on an example recording in BBP L6 dataset.

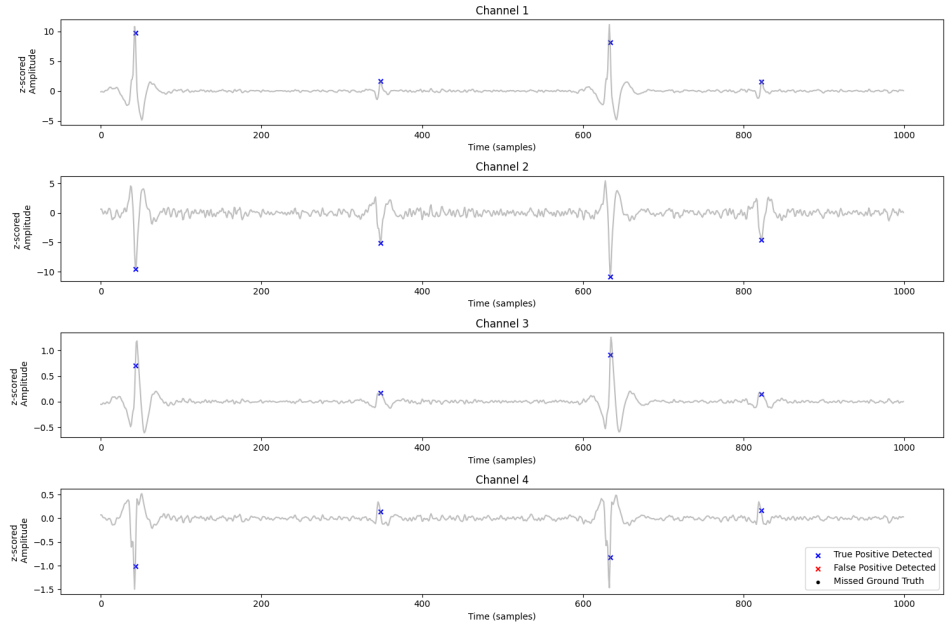

Figure S10: Zoomed-in visualization of spike detection performance on an example recording in BBP L6 dataset.

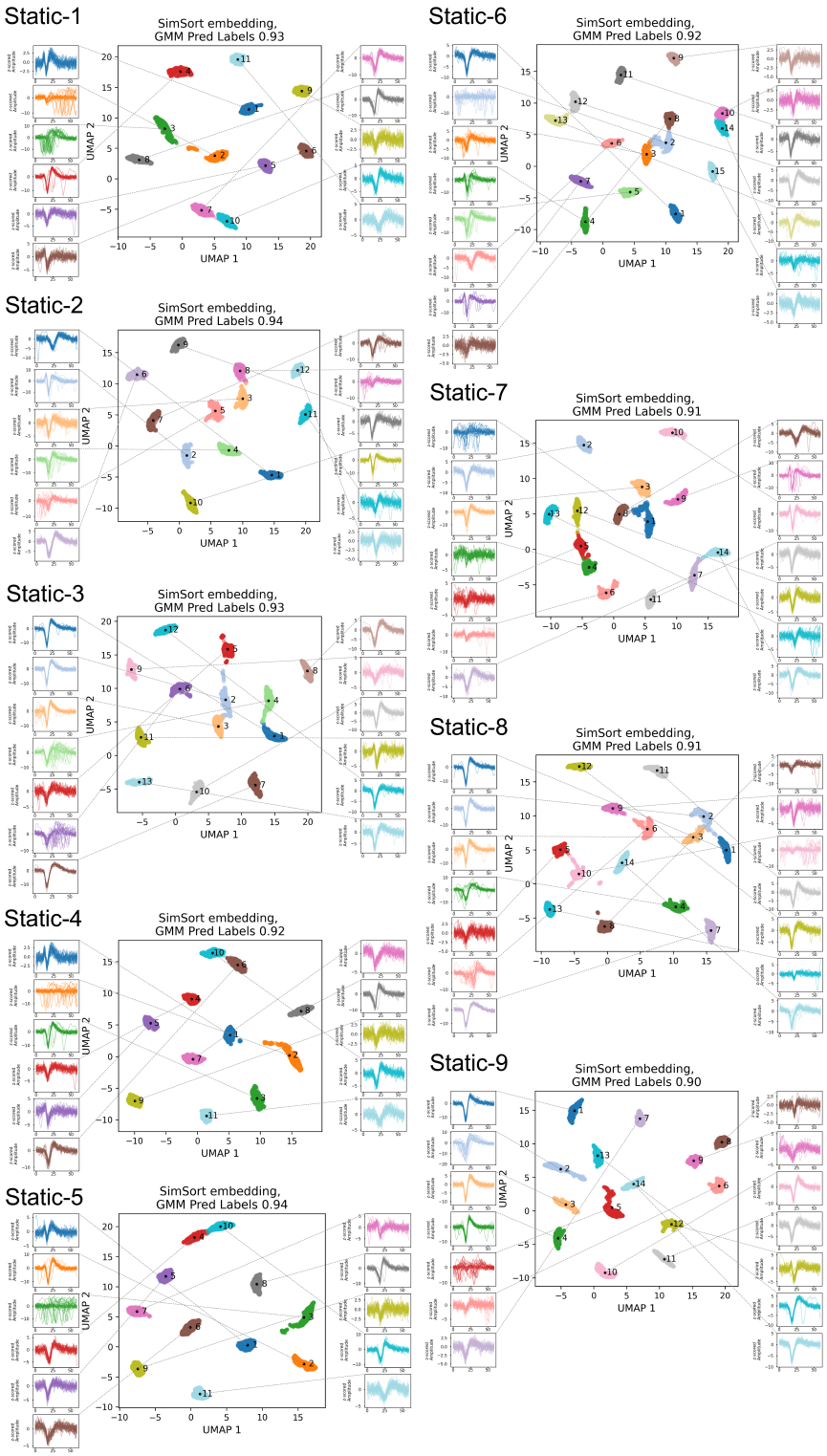

Figure S11: Visualization of spike identification performance on hybrid static subset.

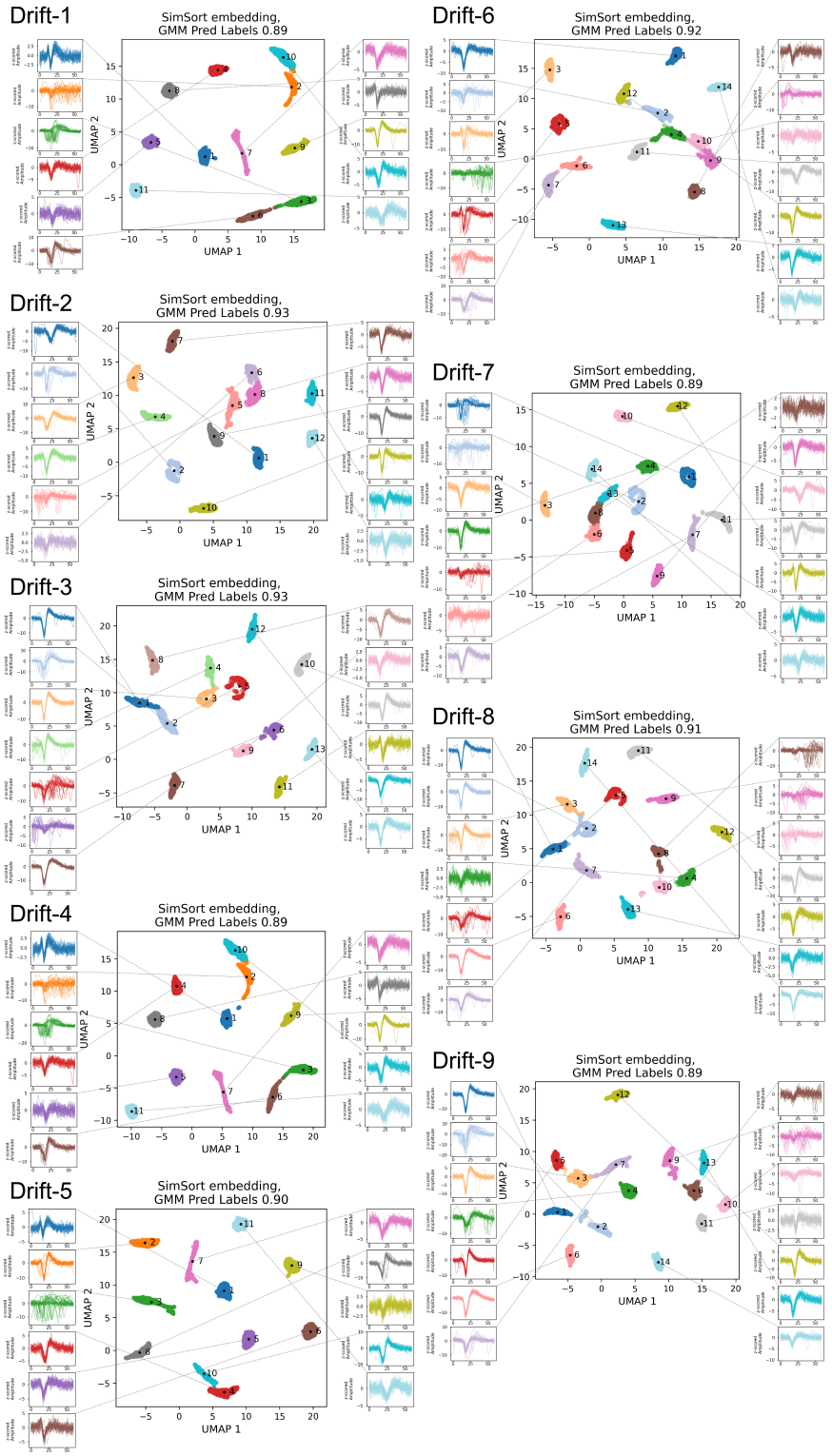

Figure S12: Visualization of spike identification performance on hybrid drift subset.

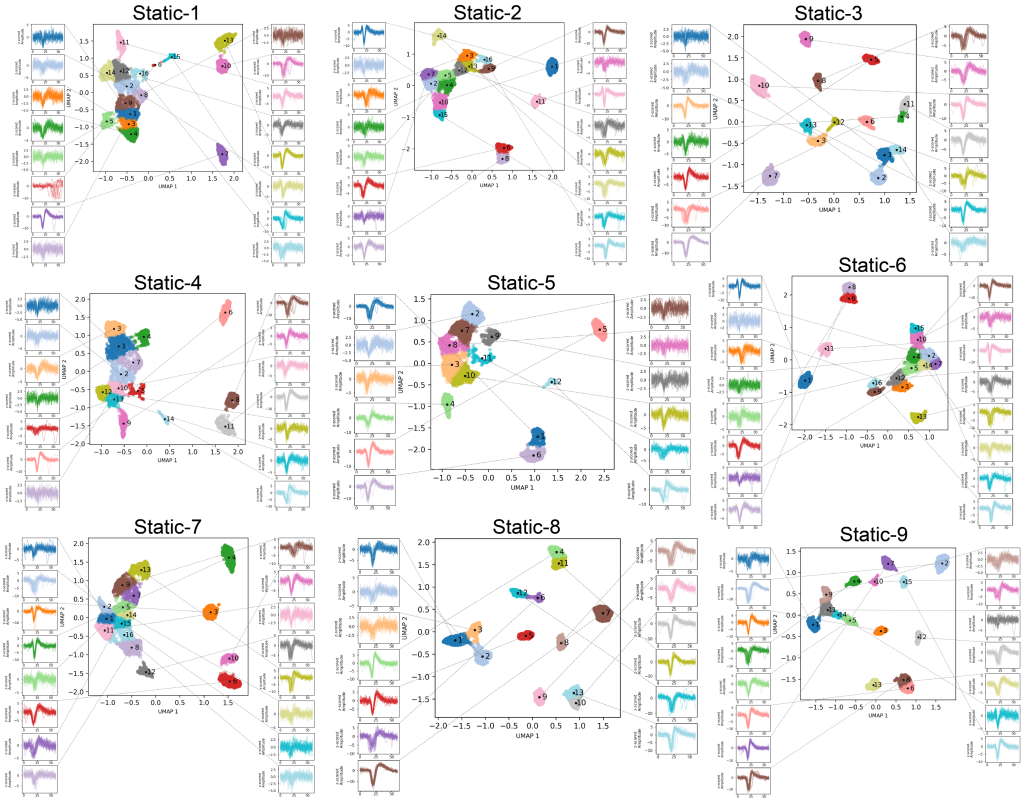

Figure S13: Visualization of spike sorting performance on hybrid static subset.

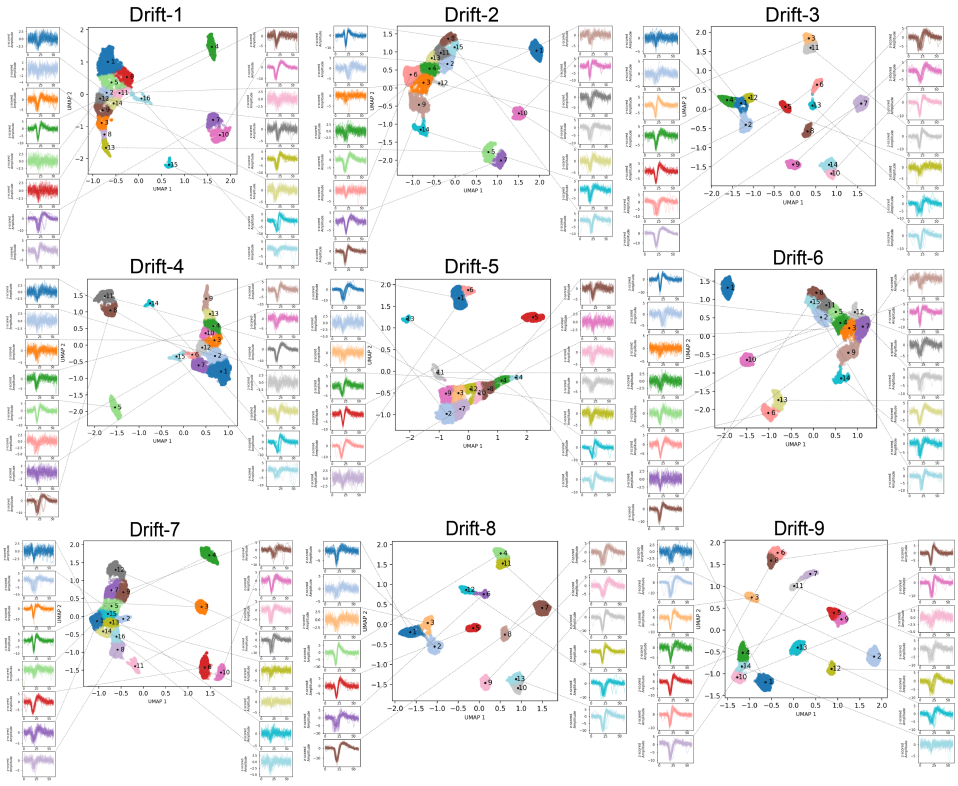

Figure S14: Visualization of spike sorting performance on hybrid drift subset.

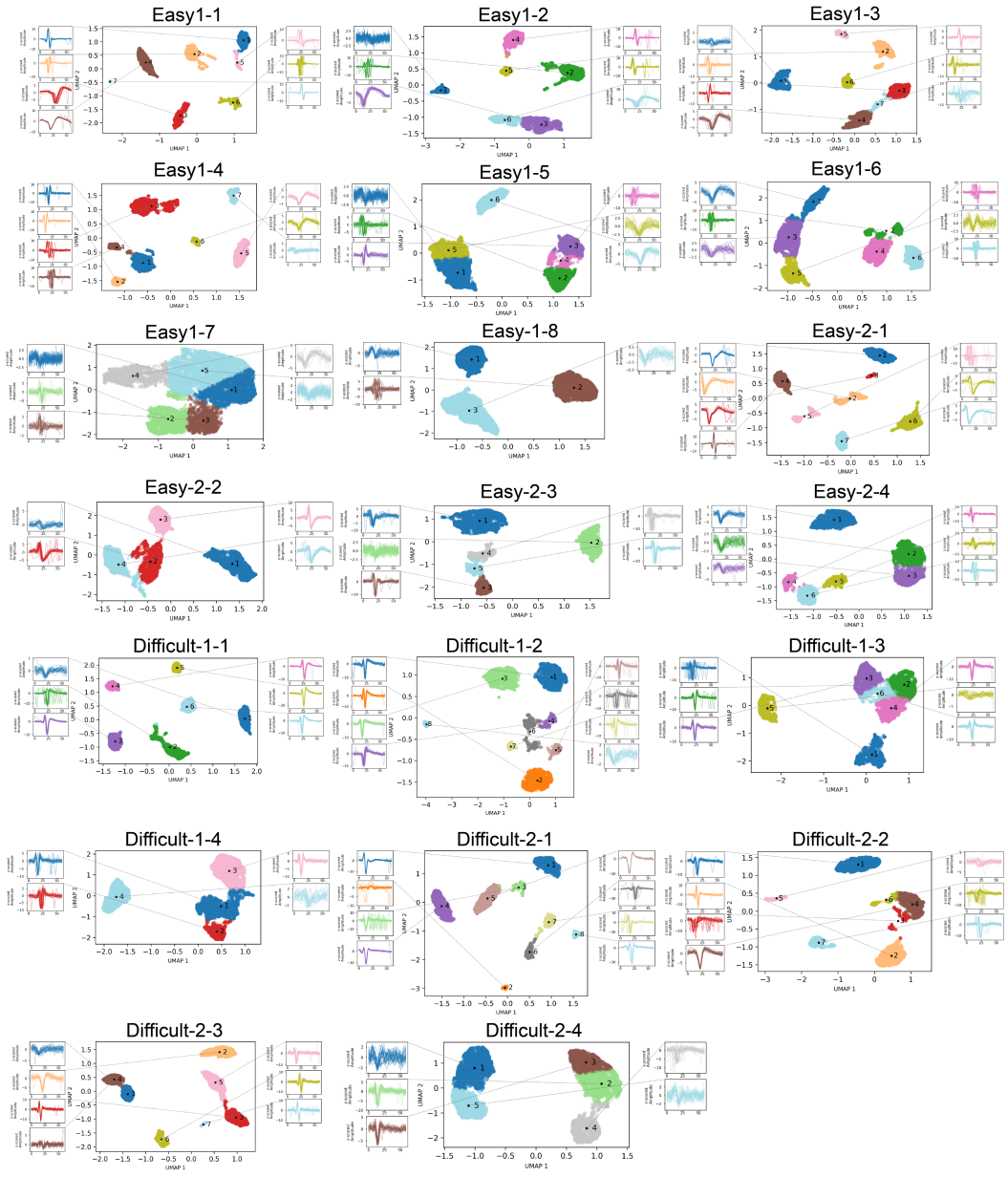

Figure S15: Visualization of spike sorting performance on WaveClus dataset.

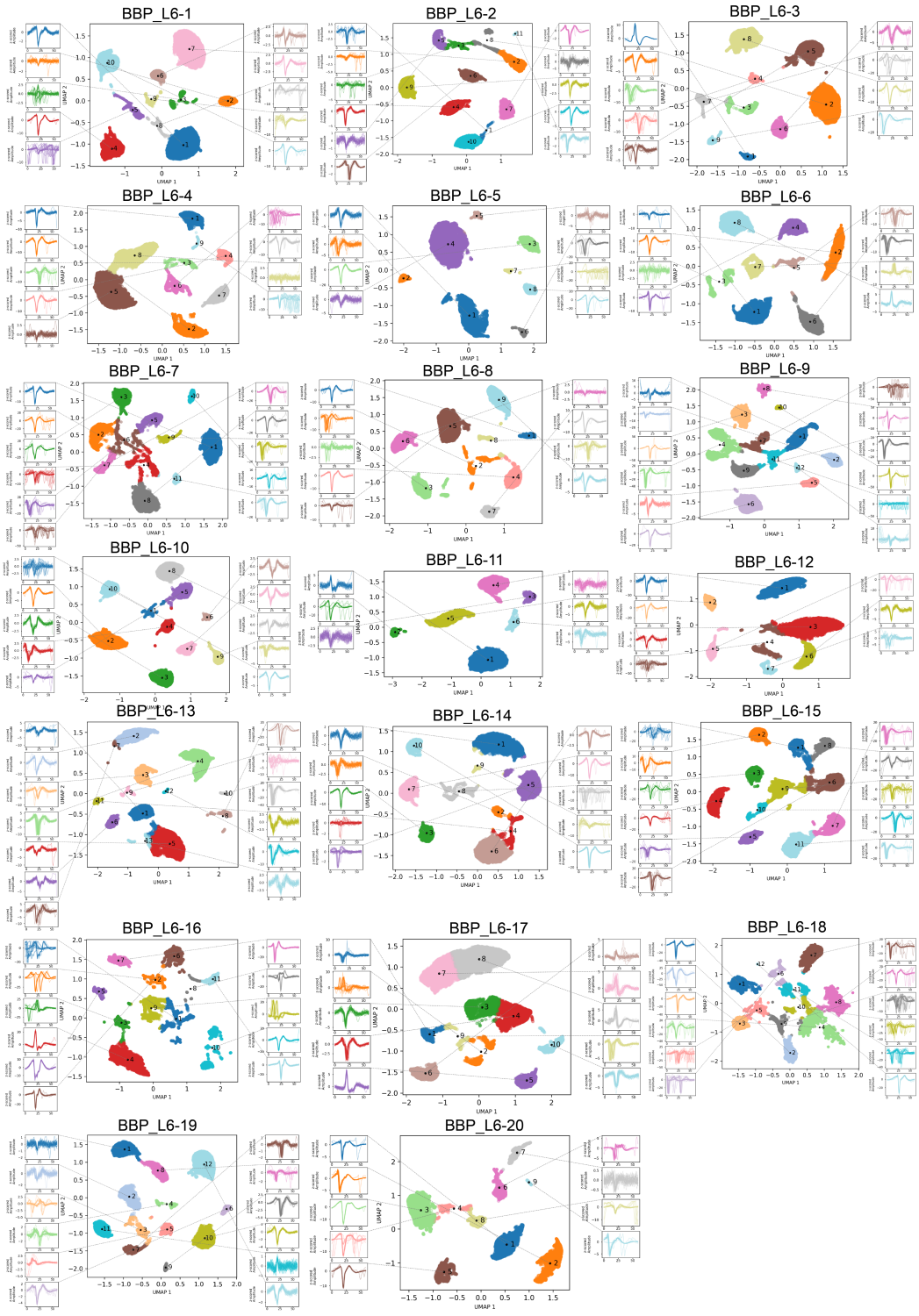

Figure S16: Visualization of spike sorting performance on BBP L6 dataset.

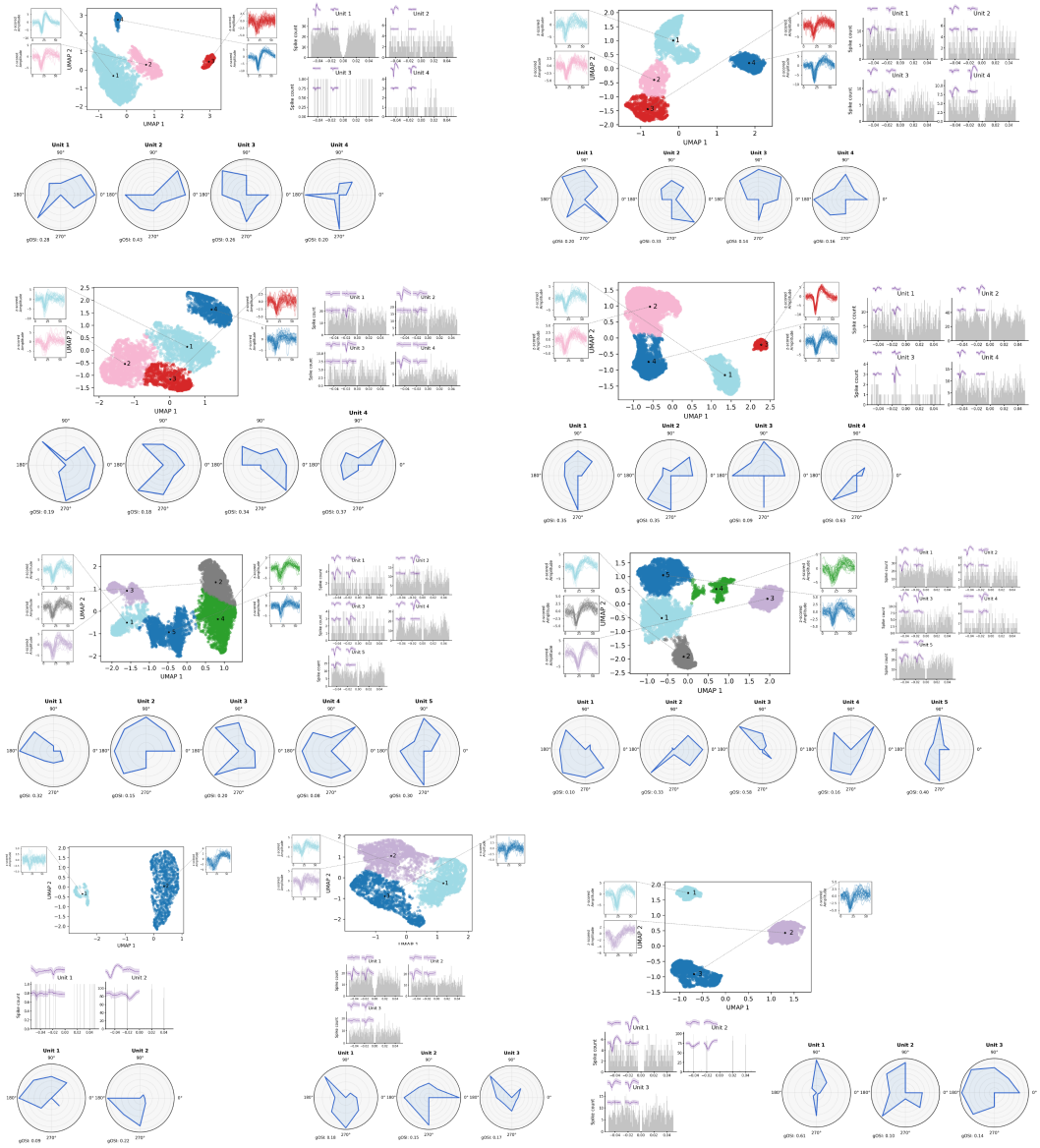

Figure S17: Visualization of spike sorting performance on real recording data.