HuiduRep: A Self-Supervised Learning Framework for More Robust Neural Representations from Extracellular Recordings

Feng Cao
Beihang University
kohaku@buaa.edu.cn

Zishuo Feng
Beijing University of Posts and Telecommunications
AkatukiFZS@bupt.edu.cn

Wei Shi
Beihang University
shiweilab@buaa.edu.cn

Jicong Zhang
Beihang University
jicongzhang@buaa.edu.cn

Abstract

Extracellular recordings are transient voltage fluctuations in the vicinity of neurons, serving as a fundamental modality in neuroscience for decoding brain activity at single-neuron resolution. Spike sorting, the process of attributing each detected spike to its corresponding neuron, is a pivotal step in brain sensing pipelines. However, it remains challenging under low signal-to-noise ratio (SNR), electrode drift, and cross-session variability. In this paper, we propose **HuiduRep**, a robust self-supervised representation learning framework that extracts discriminative and generalizable features from extracellular recordings. By integrating contrastive learning with a denoising autoencoder, HuiduRep learns latent representations robust to noise and drift. With HuiduRep, we develop a spike sorting pipeline that clusters spike representations without ground truth labels. Experiments on hybrid and real-world datasets demonstrate that HuiduRep achieves strong robustness. Furthermore, the pipeline outperforms state-of-the-art tools such as KiloSort4 and MountainSort5.

1 Introduction

Extracellular electrophysiology enables single-neuron resolution via spike sorting, a critical clustering step that assigns detected spikes to their source neurons for downstream analysis of neuronal coding and dynamics [1]. Classical spike sorting pipelines involve preprocessing, feature extraction, and clustering using algorithms such as Gaussian Mixture Model (GMM) or density-based methods.

Recent frameworks like MountainSort [2] and KiloSort [3] have improved throughput. MountainSort and KiloSort4 [4] improve throughput via automated clustering and scalable template-based inference. These tools represent the state-of-the-art in spike sorting, but they still rely on conventional clustering paradigms and presuppose stable, high-quality signals.

Despite recent advances, spike sorting remains challenging under realistic conditions. Low SNR hinders spike detection and separation. Overlapping or similar waveforms from nearby neurons can produce compound spikes, violating the assumption that each spike originates from a single neuron. Electrode drift causes gradual waveform changes, undermining stationarity assumptions and contributing significantly to sorting errors; correcting for drift substantially improves performance. Dense, high–channel-count probes also exacerbate waveform collisions due to overlapping electrical fields.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Foundation Models for the Brain and Body.

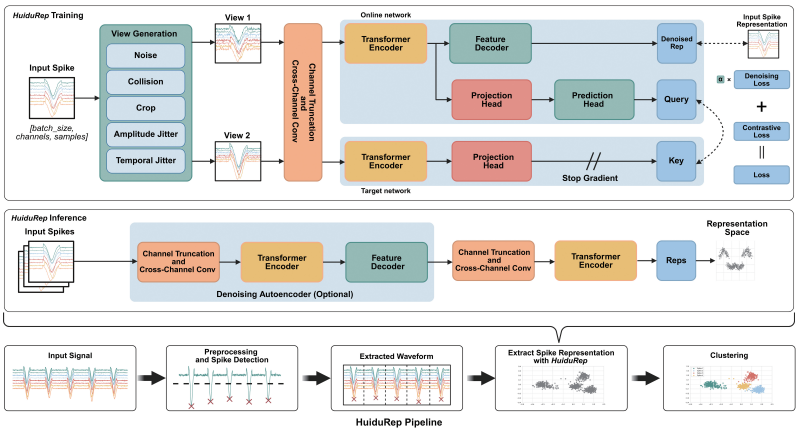

Figure 1: Overall architecture of HuiduRep and the pipeline. During training, the contrastive learning branch adapts the MoCo v3 style framework. During inference, only the transformer encoder and DAE module are used to extract representations.

In practice, even leading algorithms degrade under such conditions. Methods without explicit drift correction, such as SpyKING CIRCUS [5] and earlier versions of MountainSort, lose accuracy with substantial drift. Conventional approaches also struggle with waveform diversity and cross-session variability, often yielding inconsistent unit identities across recordings [6]. Thus, robustly clustering spikes in noisy, drifting data remains a key open problem.

To address these issues, we propose HuiduRep, a self-supervised representation learning framework for extracting representations of spike waveforms for spike sorting. HuiduRep learns features that are discriminative of neuron identity while being less affected by noise and drift. Inspired by recent trends in extracellular recordings representation learning [3], HuiduRep combines contrastive learning with a denoising autoencoder (DAE) [7]. As a result, HuiduRep can learn robust and informative spike representations without any manual labeling. We further design a complete pipeline for spike sorting with GMM clustering. The pipeline achieves robustness to low SNR and drift, and outperforms state-of-the-art sorters such as KiloSort4 and MountainSort5 on accuracy and precision across diverse datasets.

2 Method

2.1 Architecture of HuiduRep

The overall architecture of HuiduRep is illustrated in Figure 1. Inspired by BYOL [8], our framework also consists of two main branches: an online network and a target network. The target network, which is frozen during training, is updated via a momentum update based on the online network's parameters.

The key difference lies in the introduction of a DAE within the online network, which is designed to reconstruct the original signals from the augmented views generated by the view generation module. This DAE serves as an auxiliary module to guide representation learning. Moreover, we replace the original ResNet encoder [9] in BYOL with a Transformer encoder [10]. Before feeding the input views into the encoder, we also apply cross-channel convolution to better capture the characteristics of spike waveforms. During training, only *View 1* is fed into the DAE branch, while *View 2* does not participate in the denoising task.

Furthermore, the contrastive learning branch adapts the MoCo v3 style [11], where representations from positive pairs (*query* and *key*) and in-batch negative samples are compared. For contrastive learning, we adopt the InfoNCE loss [12], while for denoising, we employ the mean squared error

(MSE) loss:

$$\mathcal{L}_{\text{Contrastive}} = -\log \frac{\exp(q \cdot k^+/\tau)}{\exp(q \cdot k^+/\tau) + \sum_{k^-} \exp(q \cdot k^-/\tau)}, \quad \mathcal{L}_{\text{Denoising}} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{v_i} - \hat{\mathrm{v}}_\mathrm{i})^2$$

Here q denotes the query vector output by the prediction head of the online network, k^+ represents the positive key generated by the target network for the same sample, and k^- refers to the negative keys, which are the outputs of other samples in the same batch passed through the target network. τ is a temperature hyper-parameter [13] for l_2 -normalized q and k. For MSE loss, v is the embedded feature obtained from the original input, while $\hat{v}$ is the reconstruction produced by the DAE. We apply a standard MSE loss to measure the reconstruction quality. The overall loss function of the model is a weighted sum of the denoising loss and the contrastive loss.

To generate input views, several augmentation strategies are employed to the original spike waveforms. These include: (1) Voltage and temporal jittering, which introduces small perturbations in both voltage amplitude and timing; (2) Channel cropping, where a random subset of channels is selected to create partial views of the original waveforms; (3) Collision, where noisy spikes are overlapped onto the original waveforms to simulate spike collisions; and (4) Noise, where temporally correlated noise is added to the waveforms to generate noised views. This Noise method is employed only for generating *View 1*, enhancing the robustness and performance of the DAE. The detailed view augmentation strategy is provided in the appendix.

During inference, HuiduRep uses the encoder from the contrastive learning branch to extract representations of input spikes for downstream tasks. In certain cases, the DAE can be optionally applied before the encoder to further enhance the overall performance of the model.

2.2 Spike Sorting Pipeline

Based on HuiduRep, we propose a complete pipeline for spike sorting. As illustrated in Figure 1, our pipeline consists of the following steps: (1) Preprocessing the raw recordings by removing bad channels and applying filtering; (2) Detecting spike events from the preprocessed recordings; (3) Extracting waveforms around the detected spike events; (4) Using HuiduRep to extract representations of individual spike waveforms; and (5) Clustering the spike representations to obtain their unit assignments.

In the pipeline, the preprocessing and threshold-based detection modules of SpikeInterface were employed to process the recordings [14]. Following extraction, the spike representations were clustered using GMM from the scikit-learn library [15] to produce the final sorting results.

Our pipeline is modular, meaning that each component can be replaced by alternative methods. For example, the threshold-based detection module can be substituted with more accurate detection algorithms. In the following experiments, we demonstrate that even when using a threshold-based detection module with relatively low accuracy, our pipeline still outperforms the state-of-the-art and most widely used models such as KiloSort4.

3 Datasets

3.1 International Brain Laboratory (IBL) Dataset

DY016 and DY009 recordings are selected from the datasets released by IBL [16] to train and evaluate HuiduRep. Both recordings were recorded from the hippocampal CA1 region and anatomically adjacent areas. Similar to the processing in CEED [3], we used KiloSort2.5 [17] to preprocess the recordings and extracted a subset of spike units labeled as *good* according to IBL's quality metrics [18] to construct our dataset. For every unit, we randomly selected 1,200 spikes for training and 200 spikes for evaluation. For each spike, we extracted a waveform with 121 samples across 21 channels, centered on the channel with the highest peak amplitude.

For evaluation, we randomly sampled 10 units from the IBL evaluation dataset for each random seed ranging from 0 to 99, resulting in a total of 100 data points. These two subsets are referred to as the IBL train dataset and the IBL test dataset in the following sections.

Table 1: ARI scores and time cost per data point (Mean ± SEM) of HuiduRep and other models across varying counts of selected units, evaluated with random seeds from 0 to 99.

Model	ARI	Time (seconds)
HuiduRep 10units	$\mathbf{71.9 \pm 1.3}$	$\mathbf{6.78 \pm 0.18}$
MoCo-v3 10units	66.9 ± 1.4	7.51 ± 0.20
CEED 10units	63.5 ± 1.3	21.25 ± 0.04
HuiduRep 15units	$\mathbf{66.9 \pm 0.8}$	$\mathbf{12.22 \pm 0.26}$
MoCo-v3 15units	61.3 ± 1.1	13.24 ± 0.28
CEED 15units	57.7 ± 0.7	24.93 ± 0.09

3.2 Hybrid Janelia Dataset

HYBRID_JANELIA is a synthetic extracellular recording dataset with ground truth spike labels, designed to evaluate spike sorting algorithms. It was generated by using the KiloSort2 eMouse [17]. The simulation includes a sinusoidal drift pattern with $20\mu m$ amplitude and 2 cycles over 1,200 seconds, as well as waveform templates from high-resolution electrode recordings.

We evaluated model performance on both the static and drift recordings of this dataset. To ensure a fair comparison, we reported results only on spike units with SNR greater than 3 for all models.

3.3 Paired MEA64C Yger Dataset

Paired_MEA64C_Yger is a real-world extracellular recording dataset [5] that includes ground-truth spike times, which were obtained using juxtacellular recording [19]. The dataset recorded from isolated retinal tissues primarily targets retinal ganglion cells. It was collected using a 16×16 microelectrode array (MEA) and an 8×8 sub-array was extracted for spike sorting evaluation. For each recording, there is one ground-truth unit.

We randomly selected 9 recordings in which the ground-truth unit has SNR greater than 3, and used them to evaluate our method with other baseline models.

4 Experiments

To evaluate the performance of HuiduRep and other models, we created datasets where each data point includes 15 units, using the same construction method as the IBL test dataset.

As shown in Table 1, HuiduRep significantly outperforms CEED and MoCo-v3 on both the 10-unit and 15-unit test datasets, indicating superior representation learning capability. Furthermore, during testing, HuiduRep has a lower number of active parameters (0.6M) compared to CEED (1.8M). These results demonstrate that HuiduRep not only achieves better performance with reduced model complexity, but also adapts more effectively to downstream tasks such as spike sorting, which require strong representational ability.

To evaluate the performance of the HuiduRep Pipeline in real-world spike sorting tasks, two publicly available datasets, Hybrid Janelia and Paired MEA64c Yger, are selected as test sets. Multiple spike sorting tools, including KiloSort series [17] and MountainSort series [2], were evaluated. The performance of KiloSort4 and MountainSort5 was evaluated on our local evaluation server. The results for SimSort were cited from its original publication [20], while the performance data for the remaining methods were obtained from the results provided by SpikeForest [21].

We recorded three metrics: accuracy (Acc), precision, and recall of different models across various test sets. Moreover, we adopted the SpikeForest definitions for computing these metrics, which slightly differ from the conventional calculation methods. The accuracy balances precision and recall, and it is similar to the F1-score. These metrics are computed based on the following quantities: n_1: The number of ground-truth events that were missed by the sorter; n_2: The number of ground-truth events that were correctly matched by the sorter; n_3: The number of events detected by the sorter that do not correspond to any ground-truth event. Based on these definitions, the metrics are calculated as:

$$\text{Precision} = \frac{n_2}{n_2 + n_3}, \quad \text{Recall} = \frac{n_2}{n_1 + n_2}, \quad \text{Accuracy} = \frac{n_2}{n_1 + n_2 + n_3}$$

Table 2: Spike sorting results (Mean ± SEM) on the HYBRID_JANELIA dataset. Results for other methods are obtained from SpikeForest. Best-performing values are highlighted in *bold*.

Method	Hybrid_Janelia-Static (SNR > 3)			Hybrid_Janelia-Drift (SNR > 3)		
	Accuracy	**Recall**	**Precision**	**Accuracy**	**Recall**	**Precision**
HerdingSpikes2 [22]	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 [23]	0.57 ± 0.04	$\mathbf{0.81 \pm 0.01}$	0.60 ± 0.04	0.54 ± 0.03	$\mathbf{0.71 \pm 0.02}$	0.65 ± 0.03
JRClust [24]	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 [25]	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 [26]	0.39 ± 0.03	0.37 ± 0.03	0.51 ± 0.03	0.30 ± 0.02	0.31 ± 0.02	0.57 ± 0.04
KiloSort4 [4]	0.40 ± 0.03	0.45 ± 0.03	0.52 ± 0.05	0.34 ± 0.02	0.35 ± 0.02	0.61 ± 0.03
MountainSort4 [27]	0.59 ± 0.02	0.73 ± 0.01	0.74 ± 0.03	0.36 ± 0.02	0.57 ± 0.02	0.61 ± 0.03
MountainSort5 [28]	0.40 ± 0.06	0.50 ± 0.05	0.52 ± 0.08	0.33 ± 0.04	0.40 ± 0.03	0.64 ± 0.05
SpykingCircus [5]	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 [29]	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 [20]	0.62 ± 0.04	0.68 ± 0.04	0.77 ± 0.03	0.56 ± 0.03	0.63 ± 0.03	0.69 ± 0.03
Pipeline without DAE	0.69 ± 0.02	0.72 ± 0.02	$\mathbf{0.87 \pm 0.01}$	0.56 ± 0.02	0.61 ± 0.02	$\mathbf{0.83 \pm 0.01}$
Pipeline with DAE	$\mathbf{0.70 \pm 0.02}$	0.75 ± 0.02	0.85 ± 0.01	$\mathbf{0.60 \pm 0.02}$	0.65 ± 0.02	$\mathbf{0.83 \pm 0.01}$

Table 3: Spike sorting results (Mean ± SEM) on the Paired_MEA64C_Yger dataset. Results for other methods are obtained from SpikeForest.Note: KiloSort2 was evaluated on 8 out of 9 recordings, as it is failed to run on one recording.

Method	Paired_MEA64C_Yger (SNR > 3, 9 recordings)		
	Accuracy	**Recall**	**Precision**
HerdingSpikes2 [22]	0.77 ± 0.10	0.92 ± 0.04	0.80 ± 0.09
IronClust [23]	0.73 ± 0.09	0.96 ± 0.02	0.74 ± 0.09
KiloSort [25]	0.80 ± 0.09	0.96 ± 0.01	$\mathbf{0.82 \pm 0.09}$
KiloSort2 [26]	0.69 ± 0.11	0.99 ± 0.01	0.70 ± 0.11
KiloSort4 [4]	0.71 ± 0.10	$\mathbf{0.99 \pm 0.01}$	0.72 ± 0.11
MountainSort4 [27]	0.80 ± 0.09	0.97 ± 0.02	0.81 ± 0.09
MountainSort5 [28]	0.57 ± 0.10	0.85 ± 0.08	0.60 ± 0.10
SpykingCircus [5]	0.78 ± 0.10	0.98 ± 0.01	0.79 ± 0.10
Tridesclous [29]	0.79 ± 0.09	0.97 ± 0.02	0.80 ± 0.09
Pipeline with DAE	$\mathbf{0.80 \pm 0.08}$	0.94 ± 0.02	$\mathbf{0.82 \pm 0.09}$

As shown in Tables 2 and 3, HuiduRep Pipeline consistently outperforms other models on the Hybrid Janelia dataset in terms of accuracy and precision, under both static and drift conditions. However, its recall is slightly lower than that of IronClust but significantly higher than that of the other models. This phenomenon is potentially due to threshold-based spike detection missing low-amplitude true spikes or IronClust detecting an excessive number of spikes, which leads to a high recall and lower precision. On the high-density, multi-channel Paired MEA64C Yger dataset, the HuiduRep Pipeline also achieves slightly higher accuracy and precision compared to other models. However, the recall remains slightly lower. The performance on both datasets demonstrates the practical applicability of the HuiduRep pipeline for real-world spike sorting tasks.

Notably, applying the DAE, originally an auxiliary module during training, before the contrastive learning encoder during inference leads to significant improvements in both accuracy and recall scores. We will provide an in-depth analysis of this effect in the appendix.

5 Conclusion

HuiduRep employs a view generation strategy that preserves semantic invariance while maintaining physiological plausibility, simulating natural variability such as spike jitter, overlap, and interference. This encourages learning robust spike representations under realistic conditions.

Across diverse datasets from distinct neural structures, HuiduRep outperforms current approaches by integrating contrastive learning with a DAE, showing resilience to low SNR, drift, and spike collisions. Its architecture, inspired by neuroscience, offers improved robustness to real-world variability.

While developed for extracellular recordings, the core methodology, self-supervised learning with physiologically grounded augmentations, can extend to other bioelectrical signals like EMG, ECoG, and EEG, which face similar challenges. Future work may explore such extensions and the integration of richer biological priors or advanced detection modules to further enhance generalization and interpretability.

References

[1] Réka Barbara Bod, János Rokai, Domokos Meszéna, Richárd Fiáth, István Ulbert, and Gergely Márton. From end to end: Gaining, sorting, and employing high-density neural single unit recordings. *Frontiers in Neuroinformatics*, Volume 16 - 2022, 2022.

[2] Jason E Chung, Jeremy F Magland, Alex H Barnett, Vanessa M Tolosa, Angela C Tooker, Kye Y Lee, Kedar G Shah, Sarah H Felix, Loren M Frank, and Leslie F Greengard. A fully automated approach to spike sorting. *Neuron*, 95(6):1381–1394, 2017.

[3] Ankit Vishnubhotla, Charlotte Loh, Akash Srivastava, Liam Paninski, and Cole Hurwitz. Towards robust and generalizable representations of extracellular data using contrastive learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 42271–42284. Curran Associates, Inc., 2023.

[4] Marius Pachitariu, Shashwat Sridhar, Jacob Pennington, and Carsen Stringer. Spike sorting with kilosort4. *Nature methods*, 21(5):914–921, 2024.

[5] Pierre Yger, Giulia LB Spampinato, Elric Esposito, Baptiste Lefebvre, Stéphane Deny, Christophe Gardella, Marcel Stimberg, Florian Jetter, Guenther Zeck, Serge Picaud, Jens Duebel, and Olivier Marre. A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo. *eLife*, 7:e34518, mar 2018.

[6] Marius Brockhoff, Jakob Träuble, Sagnik Middya, Tanja Fuchsberger, Ana Fernandez-Villegas, Amberley Stephens, Miranda Robbins, Wenyue Dai, Belquis Haider, Sulay Vora, Nino F. Läubli, Clemens F. Kaminski, George G. Malliaras, Ole Paulsen, and Gabriele S. Kaminski Schierle. Pseudosorter: A self-supervised spike sorting approach applied to reveal tau-induced reductions in neuronal activity. *Science Advances*, 11(11):eadr4155, 2025.

[7] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[8] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[11] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers, 2021.

[12] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[13] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination, 2018.

[14] Alessio Paolo Buccino, Cole Lincoln 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.

[15] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2018.

[16] The International Brain Laboratory, Valeria Aguillon-Rodriguez, Dora Angelaki, Hannah Bayer, Niccolo Bonacchi, Matteo Carandini, Fanny Cazettes, Gaelle Chapuis, Anne K Churchland, Yang Dan, Eric Dewitt, Mayo Faulkner, Hamish Forrest, Laura Haetzel, Michael Häusser, Sonja B Hofer, Fei Hu, Anup Khanal, Christopher Krasniak, Ines Laranjeira, Zachary F Mainen, Guido Meijer, Nathaniel J Miska, Thomas D Mrsic-Flogel, Masayoshi Murakami, Jean-Paul Noel, Alejandro Pan-Vazquez, Cyrille Rossant, Joshua Sanders, Karolina Socha, Rebecca Terry, Anne E Urai, Hernando Vergara, Miles Wells, Christian J Wilson, Ilana B Witten, Lauren E Wool, and Anthony M Zador. Standardized and reproducible measurement of decision-making in mice. *eLife*, 10:e63711, may 2021.

[17] Marius Pachitariu, Shashwat Sridhar, and Carsen Stringer. Solving the spike sorting problem with kilosort. *bioRxiv*, 2023.

[18] Kush Banga, Julien Boussard, Gaëlle A Chapuis, Mayo Faulkner, Kenneth D Harris, JM Huntenburg, Cole Hurwitz, Hyun Dong Lee, Liam Paninski, Cyrille Rossant, et al. Spike sorting pipeline for the international brain laboratory. 2022.

[19] Didier Pinault. A novel single-cell staining procedure performed in vivo under electrophysiological control: morpho-functional features of juxtacellularly labeled thalamic cells and other central neurons with biocytin or neurobiotin. *Journal of Neuroscience Methods*, 65(2):113–136, 1996.

[20] Yimu Zhang, Dongqi Han, Yansen Wang, Zhenning Lv, Yu Gu, and Dongsheng Li. Simsort: A data-driven framework for spike sorting by large-scale electrophysiology simulation, 2025.

[21] 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, may 2020.

[22] 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.

[23] James Jun and Jeremy Magland. Ironclust: Terabyte-scale, drift-resistant spike sorter. `https://github.com/flatironinstitute/ironclust`, 2020. Accessed: 2025-07-19.

[24] James J. Jun, Catalin Mitelut, Chongxi Lai, Sergey L. Gratiy, Costas A. Anastassiou, and Timothy D. Harris. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *bioRxiv*, 2017.

[25] Marius Pachitariu, Nicholas Steinmetz, Shabnam Kadir, Matteo Carandini, and Harris Kenneth D. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *bioRxiv*, 2016.

[26] Marius Pachitariu, Cyrille Rossant, Nick Steinmetz, Jennifer Colonell, Olivier Winter, Adrian Gopnik Bondy, Kush Banga, Jai Bhagat, Mari Sosa, Dan O'Shea, Kouichi C. Nakamura, Geffen Lab Contributors, Rajat Saxena, Alan Liddell, Jose Guzman, Paul Botros, Dan Denman, Dimokratis Karamanlis, and Maxime Beau. Mouseland/kilosort2: 2.0 final. `https://github.com/MouseLand/Kilosort/releases/tag/v2.0`, 2020. Zenodo DOI: 10.5281/zenodo.4147288; Accessed: 2025-07-19.

[27] Jeremy Magland. Mountainsort 4: Spike sorting software. `https://github.com/magland/mountainsort4`, 2022. Accessed: 2025-07-19.

[28] Jeremy Magland. Mountainsort 5: Spike sorting software. `https://github.com/flatironinstitute/mountainsort5`, 2024. Accessed: 2025-07-19.

[29] Christophe Pouzat and Samuel Garcia. Tridesclous: Offline/online spike sorting toolkit. `https://github.com/tridesclous/tridesclous`, 2015. Accessed: 2025-07-19.

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

A Technical Appendices and Supplementary Material

A.1 View Generation Methods

During training, our model adopts five types of view augmentation strategies as described below.

1. **Collision**: To simulate the overlap of spikes from different neurons, which frequently occurs during real extracellular recordings, we randomly sample a waveform from the training dataset and superimpose it onto the original waveform to create a collision-like scenario. We scale the selected waveform uniformly by a value $x \sim Uniform(0.2, 1)$, randomly shifts it forwards or backwards by a number of samples dictated by $s \sim Uniform(5, 60)$, and finally adds this random scaled, shifted waveform to the original waveform.
2. **Amplitude Jitter**: To simulate the natural voltage fluctuations of spikes observed in real extracellular recordings, we apply an amplitude scaling transformation to the original waveform. Specifically, each sample of waveform is multiplied by a randomly sampled factor $x \sim Uniform(0.8, 1.2)$.
3. **Temporal Jitter**: To simulate the subtle timing variations in extracellular recordings, we apply a three-step jittering strategy: (1) upsample the waveform $8\times$ via linear interpolation to increase temporal resolution; (2) downsample with a random offset ($1 \sim 8$) to introduce sub-sample timing jitter; and (3) randomly shift the waveform by ± 2 time steps to mimic small phase variations.
4. **Noise**: To simulate the structured noise characteristics observed in extracellular recordings, we inject spatio-temporally correlated Gaussian noise into the original waveform. The noise is generated using the square roots of precomputed spatial and temporal covariance matrices, which are estimated from the training data.

   Let $\mathbf{Z} \in \mathbb{R}^{C \times T}$ be a standard Gaussian noise matrix. We extend z to shape $C_{\text{total}} \times T$ by zero-padding along the spatial dimension, where C_{total} is the total number of channels used for estimating the spatial covariance.

   Given the spatial and temporal covariance matrices $\Sigma_s \in \mathbb{R}^{C_{\text{total}} \times C_{\text{total}}}$ and $\Sigma_t \in \mathbb{R}^{T \times T}$, we compute their matrix square roots $\sqrt{\Sigma_s}$ and $\sqrt{\Sigma_t}$ via singular value decomposition (SVD). The final noise is then constructed as:

   $$\mathbf{N} = \sqrt{\Sigma_s} \cdot z \cdot \sqrt{\Sigma_t}$$

   Here, $\mathbf{N} \in \mathbb{R}^{C_{\text{total}} \times T}$ is the correlated noise. We take the first C rows of n and add it to the original waveform $\mathbf{X} \in \mathbb{R}^{C \times T}$:

   $$\widetilde{\mathbf{X}} = \mathbf{X} + \mathbf{N}$$
5. **Crop**: To prevent the model from always relying on the central channel as the dominant feature, which might lead to biased representations, we adopt a cropping strategy over the channel dimension. Specifically, for each input waveform $\mathbf{X} \in \mathbb{R}^{C \times T}$, we define a cropping window of size $C' = 11$ and extract a subset $\mathbf{X}_{\text{crop}} \in \mathbb{R}^{C' \times T}$ centered around a selected channel index.

   To diversify the location of the dominant channel (typically the one with the highest amplitude), we randomly select the cropping center with two options:

   With probability $p = 0.5$, we center the crop at the middle channel;

   With probability $1 - p$, we select a nearby off-center channel such that the true dominant channel still lies within the cropped region.

Each view augmentation strategy is applied with a specific probability to generate views of the input spike waveform. The specific probabilities used for each augmentation method are summarized in Table 4.

While our view augmentation strategies are inspired by CEED [3], they are not identical; notably, we also introduce the Noise augmentation method. Importantly, *Noise* augmentation is applied exclusively when constructing **View 1**. This design ensures that only View 1 contains localized perturbations that simulate realistic noise conditions, which are essential for training the denoising autoencoder (DAE) branch. In contrast, **View 2** is kept relatively clean to preserve the integrity of the contrastive learning objective and is not used for DAE training.

Table 4: Probability of each augmentation strategy

Augmentation	View1	View2
Collision	0.4	0.4
Amplitude Jitter	0.7	0.7
Temporal Jitter	0.6	0.6
Noise	**0.6**	**0.0**
Crop	1.0	1.0

A.2 Preprocessing Steps

Prior to spike detection, we preprocess the extracellular recordings using the standard bandpass and notch filters adopted by SpikeForest [21]. Specifically, we apply a bandpass filter (typically 300–5000 Hz) to isolate spike-related activity and a notch filter (60 Hz) to suppress interference. These steps help to reduce baseline drift and remove irrelevant frequency components, facilitating more accurate spike detection.

The input to our HuiduRep is a waveform of shape $\mathbb{R}^{11\times121}$, where 11 denotes the number of channels and 121 denotes the number of sampling points. This means that for each detected spike waveform, HuiduRep can take into account the information from up to 11 adjacent channels, including its dominant channel. To handle recordings that do not conform to this shape (e.g., due to varying channel counts), we design a preprocessing function that automatically normalizes the input to the expected format. Specifically, our preprocessing function consists of the following steps to standardize input waveforms to the required shape:

1. **Normalization**: We perform z-score normalization independently along both the temporal and channel dimensions of the input waveform. Specifically, for the input tensor $\mathbf{X} \in \mathbb{R}^{C\times T}$, we compute:

$$\mathbf{X}'_{c,t} = \frac{\mathbf{X}_{c,t} - \mu_c}{\sigma_c}$$

   Here, μ_c and σ_c are the mean and standard deviation calculated along the temporal dimension for each channel c. This normalization standardizes the amplitude distribution across time within each channel.
2. **Interpolation**: To address spike waveforms with a low number of sampling points, we employ linear interpolation to increase the temporal resolution of the signals. Specifically, each waveform is upsampled by a factor of 1.5-2.5 via linear interpolation. Empirically, this upsampling factor was found to best preserve waveform morphology while minimizing computational overhead.
3. **Channel Repeating**: For spike waveforms with fewer than 11 channels, we replicate channels to match the required input dimensionality of the model. Specifically, each original channel is repeated multiple times until the total number of channels reaches or exceeds 11. For example, a 4-channel waveform with channels [1,2,3,4] is expanded to [1,1,1,2,2,2,3,3,3,4,4,4] by repeating each channel three times. For a more detailed description of the channel repetition strategy, please refer to our source code repository.
4. **Channel Truncation**: For spike waveforms containing more than 11 channels, we crop the input along the channel dimension to match the required input size. Similar to our cropping augmentation method, we select a fixed window of size 11 channels centered at the middle channel of the input. This ensures that the most relevant central channels are retained consistently across samples.

A.3 Implementation Details

For training HuiduRep, we used the AdamW optimizer [30] with a weight decay of 1×10^{-2} to regularize the model and reduce overfitting. Additionally, we employed a cosine annealing learning rate scheduler with a linear warm-up phase during the first 10 epochs, where the learning rate increased to a maximum of 1×10^{-4}.

Table 5: ARI scores (Mean ± SEM, Max, Min) across different weight factor α of HuiduRep, evaluated with IBL test dataset.

ARI / α	0.0	0.2	0.4	0.6	0.8	1.0
Mean ± SEM	70.5 ± 1.3	$\mathbf{71.9 \pm 1.3}$	67.0 ± 1.6	71.1 ± 1.4	65.3 ± 1.5	69.6 ± 1.4
Max	91.5	**92.7**	91.0	91.2	88.8	90.5
Min	43.9	43.3	37.0	**45.7**	37.2	39.8

Table 6: ARI scores and time cost per data point (Mean ± SEM) across different representation (Rep) dimensions of HuiduRep, evaluated with IBL test dataset.

Rep Dimensions	16	32	48
ARI	69.7 ± 1.4	71.9 ± 1.3	$\mathbf{72.9 \pm 1.3}$
Time (seconds)	$\mathbf{5.39 \pm 0.12}$	6.78 ± 0.18	7.47 ± 0.23

To balance the contrastive learning branch and the DAE branch, we assigned a weight factor α to the denoising loss to control its contribution during training:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{denoising}} + \mathcal{L}_{\text{contrastive}}$$

The model's performance is evaluated across different values of α to determine the optimal trade-off on the IBL test dataset. For each α setting, the learned representations were clustered using GMM, and the Adjusted Rand Index (ARI) was computed against the ground truth labels.

We report the mean ± standard error (SEM), along with the max and min ARI values of each model across the 100 data points. The result of each data point is averaged over 50 independent GMM runs. As shown in Table 5, the best overall performance was achieved when $\alpha = 0.2$, with the highest ARI score and the highest max value. Notably, both very low ($\alpha = 0.0$) and high values ($\alpha \geq 0.8$) led to decreased performance, indicating that a moderate contribution of the denoising branch is essential for improving robustness and the overall performance of HuiduRep.

In addition, using the same IBL test dataset and evaluation method, we also evaluated the effect of different representation dimensions on the model's performance with $\alpha = 0.2$. As shown in Table 6, with the representation dimension increasing, the model's performance generally improves, suggesting enhanced representational capacity. However, higher-dimensional representation also leads to greater computational costs. To balance efficiency and performance, we set the representation dimension to 32 and fixed α at 0.2 in all subsequent experiments.

All models under different settings were trained for 300 epochs with a batch size of 4096 and a fixed random seed on a server with a single NVIDIA L40s GPU and CUDA 12.4. A local evaluation server with a single NVIDIA RTX 5080 GPU and CUDA 12.8 is used to perform all experiments.

A.4 Ablation Study

To investigate why the DAE enhances model performance during inference and to gain insights into its underlying mechanism, we randomly selected 500 spike samples per unit from each test dataset and the IBL training dataset. For each test dataset, the same set of samples was processed using two different methods: one with the DAE and one without. Principal Component Analysis (PCA) was then applied to reduce the dimensionality of the spike data to two dimensions. We computed the Euclidean distance between the centroid of the test samples and that of the IBL training samples in the reduced feature space. Furthermore, we applied HuiduRep followed by GMM to both groups and calculated the silhouette scores along with ARI of the resulting clusters. Since each recording in the Paired MEA64C yger dataset contains only one ground truth unit, the ARI becomes inapplicable. Each experiment was repeated 20 times, and the mean and standard deviation (STD) were reported.

As shown in Table 6 and Figure 2, applying the DAE to spike waveforms from out-of-distribution (OOD) datasets (Paired MEA64C Yger and Hybrid Janelia) significantly reduces their Euclidean distance to the IBL training set in the reduced feature space. This indicates that the DAE has learned to capture the feature distribution of the original training data. By aligning OOD data closer to the training data, the DAE effectively performs domain alignment, improving the overall ARI. Consequently, as shown in Table 7, applying the DAE before the contrastive learning encoder enables

Table 7: Euclidean distances between the IBL training dataset and other datasets, along with silhouette score and ARI of each dataset with and without the DAE.

Dataset	without DAE			with DAE		
	Distance	**Silhouette Score**	**ARI**	**Distance**	**Silhouette Score**	**ARI**
IBL Test Dataset	0.46 ± 0.23	0.240 ± 0.010	0.72 ± 0.03	8.64 ± 0.24	0.087 ± 0.008	0.44 ± 0.03
Paired MEA64C 1	23.43 ± 0.07	0.176 ± 0.009	N/A	7.77 ± 0.01	0.133 ± 0.011	N/A
Paired MEA64C 2	24.72 ± 0.08	0.157 ± 0.006	N/A	7.53 ± 0.02	0.120 ± 0.008	N/A
Hybrid Janelia 1	16.00 ± 0.14	0.195 ± 0.011	0.60 ± 0.03	7.02 ± 0.03	0.150 ± 0.005	0.64 ± 0.03
Hybrid Janelia 2	14.53 ± 0.10	0.127 ± 0.005	0.57 ± 0.02	6.50 ± 0.02	0.103 ± 0.005	0.58 ± 0.01
Hybrid Janelia 3	12.47 ± 0.09	0.159 ± 0.005	0.55 ± 0.02	6.41 ± 0.02	0.131 ± 0.009	0.56 ± 0.03

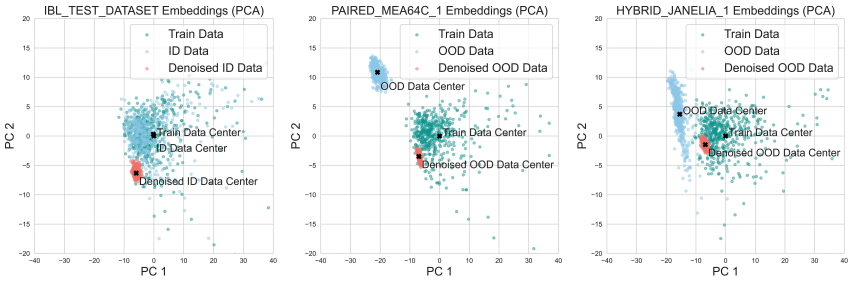

Figure 2: Reduced feature space of IBL training dataset and other datasets. The centroid of each dataset is marked with a black X. The features of each dataset are reduced to 2 dimensions using PCA.

HuiduRep to better handle distribution shifts, resulting in improved accuracy and recall scores, especially on noisy and drifting recordings.

However, this benefit comes with a potential trade-off: the DAE may compress spike waveforms into a more compact space, reducing inter-class variability and thereby making them less distinguishable and slightly reducing precision scores in the subsequent spike sorting task. This effect is reflected in the decreased silhouette scores observed after applying DAE. Moreover, for in-distribution (ID) test datasets such as the IBL test dataset, the use of DAE may distort the original data distribution, resulting in increased distance to the IBL training dataset along with lower ARI.

This suggests that while DAE effectively aligns OOD data, it may negatively impact performance when applied to data already well-aligned with the training distribution. Therefore, when processing a new dataset, one may first examine the data distribution with and without the DAE to assess its impact on the alignment of the data.