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

Feng Cao[1], **Zishuo Feng**[2], **Wei Shi**[1]
[1]Beihang University
[2]Beijing University of Posts and Telecommunications
kohaku@buaa.edu.cn, akatukifzs@bupt.edu.cn, shiweilab@buaa.edu.cn

Abstract

Extracellular recordings are brief voltage fluctuations recorded near neurons, widely used in neuroscience as the basis for decoding brain activity at single-neuron resolution. Spike sorting, which assigns each spike to its source neuron, is a critical 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 spike waveforms. By combining contrastive learning with a denoising autoencoder, HuiduRep learns latent representations that are robust to noise and drift. Built on HuiduRep, we develop a spike sorting pipeline that clusters spike representations without supervision. Experiments on hybrid and real-world datasets demonstrate that HuiduRep achieves strong robustness and the pipeline matches or outperforms state-of-the-art tools such as KiloSort4 and MountainSort5. These findings demonstrate the potential of self-supervised spike representation learning as a foundational tool for robust and generalizable processing of extracellular recordings.

Introduction

Neuroscientists frequently record extracellular action potentials, known as spikes, to monitor brain activity at single-cell resolution. These spikes, the extracellular voltage deflections from individual neurons, are considered the "fingerprints" of single-cell activity. By analyzing spike trains, sequences of temporally ordered spike times, researchers can infer neuronal coding and dynamics with millisecond precision. (Bod et al. 2022)

However, each electrode often captures spikes from many nearby neurons, so it is crucial to sort or cluster spikes by their source. (Dallal et al. 2016; Banga et al. 2022) Spike sorting is the process of assigning each detected spike waveform to its originating neuron. (Guzman et al. 2021) In practice, spike sorting is treated as a clustering problem on waveform features, often following initial steps of filtering and spike detection. (Souza et al. 2019) It is a foundational step in electrophysiology that enables single-unit analysis and studies of neuronal function. (Rey, Pedreira, and Quian Quiroga 2015)

In classical spike sorting pipelines, data are first preprocessed, for example, filtered and normalized. Spikes are then detected, typically via threshold crossings or template matching. Next, features such as waveform principal components or wavelet coefficients are extracted. Finally, spikes are clustered (via k-means, Gaussian mixtures, density peaks, etc.) to yield putative single units. Early automated sorters such as KlustaKwik (Kadir, Goodman, and Harris 2013) often required substantial manual curation because of imperfect clustering. More recent frameworks like MountainSort (Chung et al. 2017) and KiloSort (Vishnubhotla et al. 2023) have dramatically improved throughput. For example, MountainSort introduced a fully automatic clustering approach with accuracy comparable to or exceeding manual sorting. Likewise, KiloSort4 (Pachitariu et al. 2024) uses template-matching and deconvolution to scale sorting to hundreds of channels with high accuracy. These tools represent the state-of-the-art in spike sorting, but they still rely on traditional clustering steps and assumptions of stable, high-quality signals.

Despite advances, challenges remain. In real recordings, the signal-to-noise ratio (SNR) can be low, making spikes hard to detect or distinguish. Nearby neurons often produce overlapping or similar waveforms, leading to "compound" spikes that violate the assumption of one spike per neuron. Electrode drift, slow movement of neurons relative to the probe, causes spike waveforms to change over time, breaking the stationarity assumption. Indeed, electrode drift has been identified as a major source of sorting errors, and correcting for drift substantially improves sorting performance. Spatial overlap of neurons also complicates sorting: dense, high–channel-count probes produce many overlapping electrical fields, worsening the "collision" problem.

In practice, even the best algorithms degrade under such conditions: for example, methods without explicit drift correction such as SpyKING CIRCUS (Yger et al. 2018), earlier versions of MountainSort, lose accuracy when drift is large. Conventional methods also struggle with diverse waveform shapes, and cross-session variability may result in inconsistent unit identities across different recording sessions (Brockhoff et al. 2025). 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 spike waveforms' representations 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 (Vishnubhotla et al. 2023), HuiduRep combines contrastive learning with a denoising autoencoder (DAE) (Vincent et al. 2008). As a result, HuiduRep can learn robust and informative spike representations without any manual labeling.

We then cluster the learned representations using the Gaussian Mixture Model (GMM) to perform spike sorting. Based on this, we further design a complete pipeline for spike sorting. As we will show, this pipeline achieves robustness to low SNR and drift, and matches or outperforms state-of-the-art sorters such as KiloSort4 and MountainSort5 across diverse benchmark scenarios.

In summary, our main contributions are as follows:

- We propose HuiduRep, a novel self-supervised framework that combines contrastive learning and denoising autoencoder for robust spike representation learning.
- We design a complete pipeline for spike sorting, requiring no ground truth labels.
- We evaluate our method on diverse datasets, demonstrating its robustness, and show that it outperforms or matches state-of-the-art sorters.

Related Work

Template-based Spike Sorters

Template-based spike sorting algorithms remain one of the most widely used methods for processing extracellular recordings. These approaches typically detect spikes and then cluster them by matching their waveforms to a set of learned templates.

Kilosort is one of the most widely adopted template-matching sorters. It performs spike detection and sorting in a unified framework using a template-matching approach combined with drift correction. Operating directly on raw data, Kilosort can handle large-scale recordings, such as those produced by high-density Neuropixels probes (Steinmetz et al. 2021). Its core idea is to model the recorded signals as a superposition of spatiotemporally localized templates and to iteratively infer spike times and unit identities.

Despite the success of Kilosort and other template-based methods, they often rely on handcrafted heuristics or static templates that may not generalize well to low SNRs or rare waveform variations. These limitations have motivated the development of recent deep learning-based methods, including our proposed HuiduRep, which aims to learn robust representations directly from data without relying on fixed templates.

Representation Learning Models

In spike sorting, effective representation of spike waveforms plays a crucial role in enabling accurate clustering, particularly in noisy and drifting recordings. Recent methods have therefore adopted representation learning frameworks to learn spike features. Among these, CEED (Vishnubhotla et al. 2023) and SimSort (Zhang et al. 2025) have emerged as two representative approaches that leverage contrastive learning to derive meaningful spike features without manual labeling.

CEED is a SimCLR-based (Chen et al. 2020) contrastive representation learning framework for extracellular recordings. It is trained and evaluated on the IBL dataset (Laboratory et al. 2021), achieving promising performance in embedding spike features. Nevertheless, CEED is limited in this scope: it functions solely as a feature extractor and does not design a complete spike sorting pipeline. Moreover, its performance degrades sharply in embedding multiple neuron types, indicating its difficulty in capturing fine-grained inter-class distinctions.

Compared to CEED, SimSort not only proposes a representation learning model but also introduces a complete spike sorting pipeline. However, SimSort also has several limitations. For instance, it only supports 4-channel inputs, which limits its applicability to high-density probes such as Neuropixels recordings. Moreover, due to its relatively small model size, the performance improvement over existing sorters remains limited, especially in noisy and drifting recording conditions.

Method

In this section, we introduce our HuiduRep's overall architecture as well as the pipeline based on HuiduRep.

HuiduRep's Architecture

The overall architecture of HuiduRep is illustrated in Figure 1. Inspired by BYOL (Grill et al. 2020), our framework also consists of two main branches: an online network and a target network. The target network is updated via a momentum update based on the online network's parameters, and its gradients are frozen during training.

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 (He et al. 2015) in BYOL with a Transformer encoder (Vaswani et al. 2023). 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 **view1** is fed into the DAE branch, while **view2** does not participate in the denoising task.

The contrastive learning branch adapts the MoCo v3 style (Chen, Xie, and He 2021), where representations from positive pairs (*query* and *key*) and in-batch negative samples are compared. For contrastive learning, we adopt the InfoNCE loss (van den Oord, Li, and Vinyals 2019), while for denoising, we employ the mean squared error (MSE) loss:

$$\mathcal{L}_{\text{Contrastive}} = -\log \frac{\exp(q \cdot k^+/\tau)}{\exp(q \cdot k^+/\tau) + \sum_{k^-} \exp(q \cdot k^-/\tau)}$$

$$\mathcal{L}_{\text{Denoising}} = \frac{1}{n} \sum_{i=1}^{n} (v_i - \hat{v}_i)^2$$

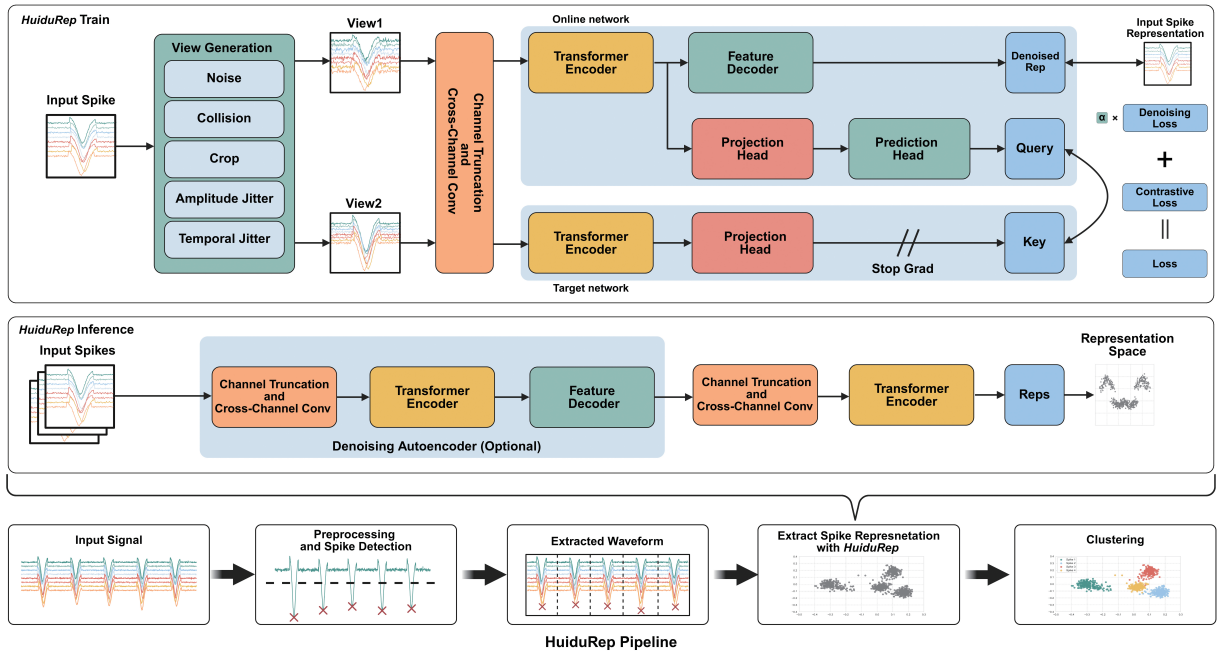


Figure 1: Overall architecture of our HuiduRep and the pipeline. During training, the contrastive learning branch adapts the MoCo v3 style framework, where the *query* is compared with that of *key* and other in-batch samples (not shown in the figure due to the limited space). Only the *view1* is passed to the DAE branch for reconstruction. During inference, only the encoder and projection head are used to extract representations.

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. 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.(Wu et al. 2018) for l_2-normalized q and k. 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 *view1*, enhancing the robustness and performance of the DAE.

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 placed before the encoder to further enhance the overall performance of the model.

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, spikeInterface's preprocessing and threshold-based detection modules were employed to process the recordings (Buccino et al. 2020). Following extraction, the spike representations were clustered using GMM from the scikit-learn library (Pedregosa et al. 2018) 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 matches or even outperforms the state-of-the-art and most widely used models such as Kilosort4.

Datasets

In this section, we present the datasets used for training and evaluating our model, as well as their characteristics.

Algorithm 1: HuiduRep's Pytorch Style Pseudocode

```
# conv: channel truncation + cross-channel convolution
# f_q: encoder + projection + prediction
# f_k: momentum encoder + momentum projection
# dae: encoder + feature decoder
# clf: contrastive loss function
# a: weight factor
# m: momentum coefficient

for x in loader:  # load data
    v1, v2 = aug(x), aug(x)  # augmentation
    v1, v2 = conv(v1), conv(v2)  # conv embeddings
    q1, q2 = f_q(v1), f_q(v2)  # queries
    k1, k2 = f_k(v1), f_k(v2)  # keys

    v = conv(x)  # conv embeddings
    v_hat = dae(v1)  # denoising batch

    loss1 = clf(q1, k2) + clf(q2, k1)  # symmetrized
    loss2 = MSELoss(v, v_hat)
    loss = loss1 + a * loss2  # weighted loss
    loss.backward()

    # optimizer update
    update(f_q), update(dae), update(conv)
    f_k = m*f_k + (1-m)*f_q  # momentum update
```

International Brain Laboratory (IBL) Dataset

The International Brain Laboratory (IBL) (Laboratory et al. 2021) is a global collaboration involving multiple research institutions, aiming to uncover the neural basis of decision-making in mice through standardized behavioral and electrophysiological experiments.

We selected DY016 and DY009 recordings from the datasets released by IBL to train and evaluate HuiduRep. Similar to the processing in CEED (Vishnubhotla et al. 2023), we used KiloSort2.5 (Pachitariu, Sridhar, and Stringer 2023) to preprocess the recordings and extracted a subset of spike units labeled as *good* according to IBL's quality metrics (Banga et al. 2022) 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.

All selected units from the DY016 and DY009 recordings were used for constructing the training set. 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 as the IBL train dataset and the IBL test dataset in the following sections.

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 (Pachitariu, Sridhar, and Stringer 2023). 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.

Paired MEA64C YGER Dataset

PAIRED_MEA64C_YGER is a real-world extracellular recording dataset (Yger et al. 2018) that includes ground-truth spike times, which were obtained using juxtacellular recording (Pinault 1996). The dataset 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.

Experiments

In this section, we will introduce the key experimental procedures, including hyperparameter search, performance evaluation, and ablation studies.

Implementation Details

For training HuiduRep, we used the AdamW optimizer (Loshchilov and Hutter 2019) 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}.

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.

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

We evaluated the model's performance 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 $\pm$ standard error (SEM), along with the maximum and minimum ARI values of each model across the 100 units. As shown in Table 1, the best overall performance was achieved when $\alpha = 0.2$, with the highest mean ARI score (71.9 ± 1.3) and the highest maximum value (92.7). 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 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 2, with the representation dimension increasing, the model's performance generally

ARI / α	0.0 (without Reconstruction)	0.2	0.4	0.6	0.8	1.0
Mean $\pm$ 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 1: HuiduRep's ARI scores (Mean $\pm$ SEM, Max, Min) across different DAE loss weight α, 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

Table 2: HuiduRep's ARI scores and time cost (Mean $\pm$ SEM) across different representation (Rep) dimensions, evaluated with IBL test dataset.

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 12.1. All experiments were performed on a server with a single NVIDIA RTX 4080 GPU and CUDA 12.8. A complete list of training hyperparameters is provided in the Appendix.

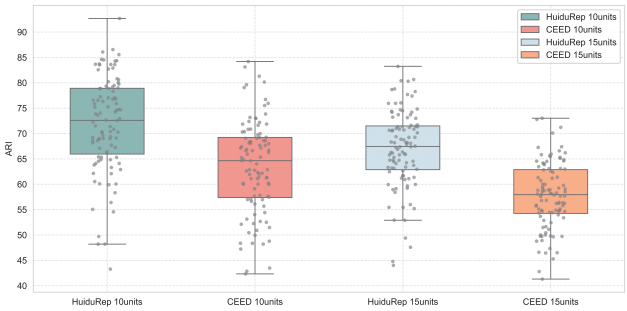
Performance Evaluation

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

As shown in Table 3, HuiduRep significantly outperforms CEED 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, we selected two publicly available datasets—Hybrid Janelia and Paired MEA64c YGER—as test sets. Multiple spike sorting tools, including Kilosort series (Pachitariu, Sridhar, and Stringer 2023) and MountainSort series (Chung et al. 2017), were evaluated. The performance of Kilosort4 and MountainSort5 was evaluated in our local evaluation environment. The results for SimSort were cited from its original publication (Zhang et al. 2025), while the performance data for the remaining methods were obtained from the results provided by SpikeForest (Magland et al. 2020).

We recorded three metrics: accuracy (Acc), precision and recall of different models across various test sets. It is worth



Model / ARI	Mean $\pm$ SEM	Max	Min
HuiduRep 10units	$\mathbf{71.9 \pm 1.3}$	**92.7**	**43.3**
CEED 10units	63.5 ± 1.3	84.2	42.3
HuiduRep 15units	$\mathbf{66.9 \pm 0.8}$	**83.2**	**44.0**
CEED 15units	57.7 ± 0.7	73.0	41.3

Table 3: ARI scores of HuiduRep and CEED (Vishnubhotla et al. 2023) across varying counts of selected units, evaluated with random seeds from 0 to 99.

noting that 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, but not identical 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:

$$Precision = \frac{n_2}{n_2 + n_3}$$

$$Recall = \frac{n_2}{n_1 + n_2}$$

$$Accuracy = \frac{n_2}{n_1 + n_2 + n_3}$$

As shown in Table 4 and Table 5, on the Hybrid Janelia dataset, the HuiduRep Pipeline consistently outperforms other models in terms of accuracy and precision, under both static and drift data. However, its recall is slightly lower than that of IronClust, potentially due to threshold-based spike detection missing low-amplitude true spikes.

On the high-density, multi-channel Paired MEA64c

Method	Hybrid_Janelia-Static (SNR > 3, 9 recordings)			Hybrid_Janelia-Drift (SNR > 3, 9 recordings)		
	Accuracy	**Recall**	**Precision**	**Accuracy**	**Recall**	**Precision**
HerdingSpikes2 (Hilgen et al. 2017)	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 (Jun and Magland 2020)	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 (Jun et al. 2017)	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 (Pachitariu et al. 2016)	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 (Pachitariu et al. 2020)	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 (Pachitariu et al. 2024)	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 (Magland 2022)	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 (Magland 2024)	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 (Yger et al. 2018)	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 (Pouzat and Garcia 2015)	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 (Zhang et al. 2025)	0.62 ± 0.04	0.68 ± 0.04	0.77 ± 0.03	0.56 ± 0.03	0.63 ± 0.03	0.69 ± 0.03
HuiduRep 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	$\mathbf{0.83 \pm 0.01}$
HuiduRep 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 4: 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	PAIRED_MEA64C_YGER (SNR > 3, 9 recordings)		
	Accuracy	**Recall**	**Precision**
HerdingSpikes2 (Hilgen et al. 2017)	0.77 ± 0.10	0.92 ± 0.04	0.80 ± 0.09
IronClust (Jun and Magland 2020)	0.73 ± 0.09	0.96 ± 0.02	0.74 ± 0.09
KiloSort (Pachitariu et al. 2016)	0.80 ± 0.09	0.96 ± 0.01	$\mathbf{0.82 \pm 0.09}$
KiloSort2 (Pachitariu et al. 2020)	0.69 ± 0.11	0.99 ± 0.01	0.70 ± 0.11
KiloSort4 (Pachitariu et al. 2024)	0.71 ± 0.10	$\mathbf{0.99 \pm 0.01}$	0.72 ± 0.11
MountainSort4 (Magland 2022)	0.80 ± 0.09	0.97 ± 0.02	0.81 ± 0.09
MountainSort5 (Magland 2024)	0.57 ± 0.10	0.85 ± 0.08	0.60 ± 0.10
SpykingCircus (Yger et al. 2018)	0.78 ± 0.10	0.98 ± 0.01	0.79 ± 0.10
Tridesclous (Pouzat and Garcia 2015)	0.79 ± 0.09	0.97 ± 0.02	0.80 ± 0.09
HuiduRep Pipeline with DAE	$\mathbf{0.80 \pm 0.08}$	0.94 ± 0.02	$\mathbf{0.82 \pm 0.09}$

Table 5: Spike sorting results (Mean ± SEM) on the PAIRED_MEA64C_YGER dataset. Results for other methods are obtained from SpikeForest. Best-performing values are highlighted in **bold**. Note: KiloSort2 was evaluated on 8 out of 9 recordings, as it is failed to run on one recording.

YGER dataset, the HuiduRep Pipeline achieves slightly higher accuracy and precision compared to other models, while the recall remains slightly lower. The performance on both datasets demonstrates the practical applicability of the HuiduRep pipeline for real-world spike sorting tasks.

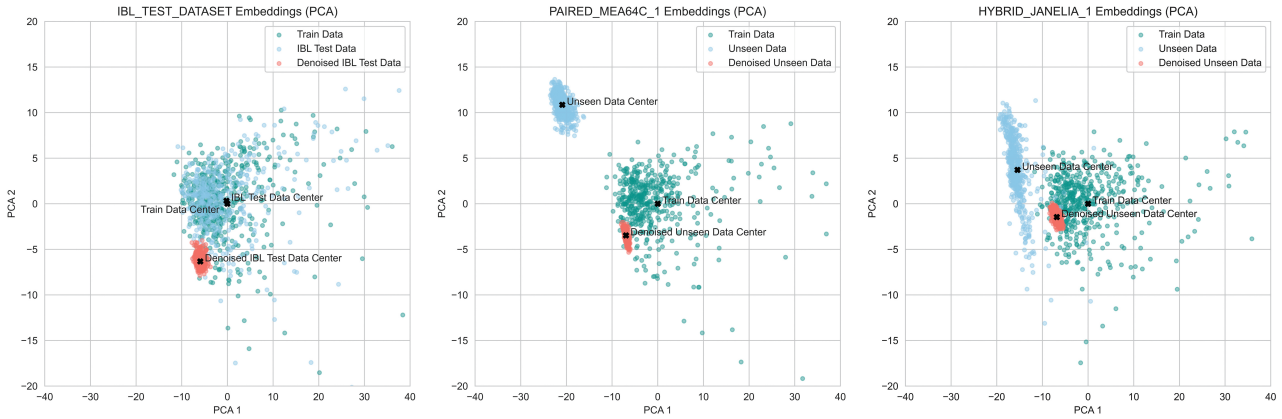
Notably, placing the DAE—originally an auxiliary module during training—before the contrastive learning encoder during inference leads to significant improvements in both accuracy and recall scores. We will provide an in-depth analysis of this effect in the next ablation study section.

Ablation Study

To find why the DAE can enhance model performance during inference and to understand its underlying mechanism, we randomly selected 500 spike samples from each test dataset and the IBL training dataset. For the test datasets, the samples were divided into two groups: one processed with the DAE and the other without DAE processing. Then, we applied Principal Component Analysis (PCA) to reduce the dimensionality of the spike data to 2 dimensions. For each test dataset, we repeated the experiment 20 times and computed the Euclidean distance between the centroid of the test samples and that of the IBL training set in the reduced representation space. We report the mean and standard deviation (STD) of the distances across the 20 experiments.

As shown in Table 6, applying the DAE to spike waveforms from out-of-distribution datasets (i.e., Paired MEA64C YGER and Hybrid Janelia) significantly reduces their Euclidean distance to the IBL training set in the reduced representation space. This indicates that the DAE has learned to capture the feature distribution of the original training data. By aligning out-of-distribution data closer to the training data, the DAE effectively performs domain alignment. Consequently, placing the DAE before the contrastive learning encoder enables 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:



IBL Train Dataset Vs.	IBL_TEST_DATASET	PAIRED_MEA64C_1	PAIRED_MEA64C_2
Distance without DAE	0.46 ± 0.23	23.43 ± 0.07	24.72 ± 0.08
Distance with DAE	8.64 ± 0.24	7.77 ± 0.01	7.53 ± 0.02
IBL Train Dataset Vs.	HYBRID_JANELIA_1	HYBRID_JANELIA_2	HYBRID_JANELIA_3
Distance without DAE	16.00 ± 0.14	14.53 ± 0.10	12.47 ± 0.09
Distance with DAE	7.02 ± 0.03	6.50 ± 0.02	6.41 ± 0.02

Table 6: Euclidean distances (Mean $\pm$ STD) between the centroid of the IBL training dataset and those of other datasets, with and without the DAE. The features of each dataset are reduced to 2 dimensions using PCA. In the figure, the centroid of each dataset is marked with a black X.

the DAE may compress spike waveforms into a more compact representation space, reducing inter-class variability and thereby making them less distinguishable and slightly reducing precision scores in the subsequent spike sorting task. Moreover, for in-distribution 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.

This suggests that while DAE effectively aligns out-of-distribution data, it may negatively impact performance when applied to data already well-aligned with the training distribution.

Conclusion

In HuiduRep, our view generation strategy not only produces different views that preserve semantic invariance but also maintains genuine physiological significance. This strategy simulates the jitters occurring during the firing process of real neural signals, as well as the overlapping and interference between signals from different neurons. In essence, it models the natural variability present in real neural recordings. Thus, the view generation strategy encourages the model to learn spike representations under more realistic conditions, enhancing its overall performance.

Furthermore, DAE learns to reconstruct augmented inputs back onto the original spike waveforms. This component is also remarkably biologically intuitive: many cortical circuits effectively perform noise suppression and normalization. For example, computational models show that topographic recurrent networks in the cortex can amplify signal-to-noise by adjusting the excitation–inhibition balance (Zajzon et al. 2023). In other words, cortex exhibits a denoising behavior that preserves stimulus features while suppressing irrelevant fluctuations. The DAE plays a similar role: it is trained to reconstruct a clean waveform from an augmented input. In our framework, this means that HuiduRep is encouraged to represent only the stable, informative representations, effectively filtering out the noise.

Overall, HuiduRep demonstrates strong robustness in spike representation learning, matching or surpassing state-of-the-art sorters across a wide range of datasets. By combining contrastive learning with a denoising autoencoder, it maintains high performance under low SNR, electrode drift, and overlapping conditions. Its architecture draws inspiration from neuroscience, offering greater resilience to real-world variability than conventional methods. Future work may focus on extending evaluations to more diverse probe types, incorporating richer biological priors, and integrating more advanced spike detection techniques to further enhance generalization and interpretability.

References

Banga, K.; Boussard, J.; Chapuis, G. A.; Faulkner, M.; Harris, K. D.; Huntenburg, J.; Hurwitz, C.; Lee, H. D.; Paninski, L.; Rossant, C.; et al. 2022. Spike sorting pipeline for the International Brain Laboratory.

Bod, R. B.; Rokai, J.; Meszéna, D.; Fiáth, R.; Ulbert, I.; and Márton, G. 2022. From End to End: Gaining, Sorting, and Employing High-Density Neural Single Unit Recordings. *Frontiers in Neuroinformatics*, Volume 16 - 2022.

Brockhoff, M.; Träuble, J.; Middya, S.; Fuchsberger, T.; Fernandez-Villegas, A.; Stephens, A.; Robbins, M.; Dai, W.; Haider, B.; Vora, S.; Läubli, N. F.; Kaminski, C. F.; Malliaras, G. G.; Paulsen, O.; and Schierle, G. S. K. 2025. PseudoSorter: A self-supervised spike sorting approach applied to reveal Tau-induced reductions in neuronal activity. *Science Advances*, 11(11): eadr4155.

Buccino, A. P.; Hurwitz, C. L.; Garcia, S.; Magland, J.; Siegle, J. H.; Hurwitz, R.; and Hennig, M. H. 2020. SpikeInterface, a unified framework for spike sorting. *Elife*, 9: e61834.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A Simple Framework for Contrastive Learning of Visual Representations. arXiv:2002.05709.

Chen, X.; Xie, S.; and He, K. 2021. An Empirical Study of Training Self-Supervised Vision Transformers. arXiv:2104.02057.

Chung, J. E.; Magland, J. F.; Barnett, A. H.; Tolosa, V. M.; Tooker, A. C.; Lee, K. Y.; Shah, K. G.; Felix, S. H.; Frank, L. M.; and Greengard, L. F. 2017. A fully automated approach to spike sorting. *Neuron*, 95(6): 1381–1394.

Dallal, A. H.; Chen, Y.; Weber, D.; and Mao, Z.-H. 2016. Dictionary learning for sparse representation and classification of neural spikes. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 3486–3489. IEEE.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P. H.; Buchatskaya, E.; Doersch, C.; Pires, B. A.; Guo, Z. D.; Azar, M. G.; Piot, B.; Kavukcuoglu, K.; Munos, R.; and Valko, M. 2020. Bootstrap your own latent: A new approach to self-supervised Learning. arXiv:2006.07733.

Guzman, E.; Cheng, Z.; Hansma, P. K.; Tovar, K. R.; Petzold, L. R.; and Kosik, K. S. 2021. Extracellular detection of neuronal coupling. *Scientific Reports*, 11(1): 14733.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Hilgen, G.; Sorbaro, M.; Pirmoradian, S.; Muthmann, J.-O.; Kepiro, I. E.; Ullo, S.; Ramirez, C. J.; Puente Encinas, A.; Maccione, A.; Berdondini, L.; Murino, V.; Sona, D.; Cella Zanacchi, F.; Sernagor, E.; and Hennig, M. H. 2017. Unsupervised Spike Sorting for Large-Scale, High-Density Multielectrode Arrays. *Cell Reports*, 18(10): 2521–2532.

Jun, J.; and Magland, J. 2020. IronClust: Terabyte-scale, drift-resistant spike sorter. https://github.com/flatironinstitute/ironclust. Accessed: 2025-07-19.

Jun, J. J.; Mitelut, C.; Lai, C.; Gratiy, S. L.; Anastassiou, C. A.; and Harris, T. D. 2017. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *bioRxiv*.

Kadir, S. N.; Goodman, D. F. M.; and Harris, K. D. 2013. High-dimensional cluster analysis with the Masked EM Algorithm. arXiv:1309.2848.

Laboratory, T. I. B.; Aguillon-Rodriguez, V.; Angelaki, D.; Bayer, H.; Bonacchi, N.; Carandini, M.; Cazettes, F.; Chapuis, G.; Churchland, A. K.; Dan, Y.; Dewitt, E.; Faulkner, M.; Forrest, H.; Haetzel, L.; Häusser, M.; Hofer, S. B.; Hu, F.; Khanal, A.; Krasniak, C.; Laranjeira, I.; Mainen, Z. F.; Meijer, G.; Miska, N. J.; Mrsic-Flogel, T. D.; Murakami, M.; Noel, J.-P.; Pan-Vazquez, A.; Rossant, C.; Sanders, J.; Socha, K.; Terry, R.; Urai, A. E.; Vergara, H.; Wells, M.; Wilson, C. J.; Witten, I. B.; Wool, L. E.; and Zador, A. M. 2021. Standardized and reproducible measurement of decision-making in mice. *eLife*, 10: e63711.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101.

Magland, J. 2022. MountainSort 4: Spike sorting software. https://github.com/magland/mountainsort4. Accessed: 2025-07-19.

Magland, J. 2024. MountainSort 5: Spike sorting software. https://github.com/flatironinstitute/mountainsort5. Accessed: 2025-07-19.

Magland, J.; Jun, J. J.; Lovero, E.; Morley, A. J.; Hurwitz, C. L.; Buccino, A. P.; Garcia, S.; and Barnett, A. H. 2020. SpikeForest, reproducible web-facing ground-truth validation of automated neural spike sorters. *eLife*, 9: e55167.

Pachitariu, M.; Rossant, C.; Steinmetz, N.; Colonell, J.; Winter, O.; Bondy, A. G.; Banga, K.; Bhagat, J.; Sosa, M.; O'Shea, D.; Nakamura, K. C.; Contributors, G. L.; Saxena, R.; Liddell, A.; Guzman, J.; Botros, P.; Denman, D.; Karamanlis, D.; and Beau, M. 2020. MouseLand/Kilosort2: 2.0 final. https://github.com/MouseLand/Kilosort/releases/tag/v2.0. Zenodo DOI: 10.5281/zenodo.4147288; Accessed: 2025-07-19.

Pachitariu, M.; Sridhar, S.; Pennington, J.; and Stringer, C. 2024. Spike sorting with Kilosort4. *Nature methods*, 21(5): 914–921.

Pachitariu, M.; Sridhar, S.; and Stringer, C. 2023. Solving the spike sorting problem with Kilosort. *bioRxiv*.

Pachitariu, M.; Steinmetz, N.; Kadir, S.; Carandini, M.; and Kenneth D., H. 2016. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *bioRxiv*.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Müller, A.; Nothman, J.; Louppe, G.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Édouard Duchesnay. 2018. Scikit-learn: Machine Learning in Python. arXiv:1201.0490.

Pinault, D. 1996. 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.

Pouzat, C.; and Garcia, S. 2015. Tridesclous: Offline/online spike sorting toolkit. https://github.com/tridesclous/tridesclous. Accessed: 2025-07-19.

Rey, H. G.; Pedreira, C.; and Quian Quiroga, R. 2015. Past, present and future of spike sorting techniques. *Brain Research Bulletin*, 119: 106–117. Advances in electrophysiological data analysis.

Souza, B. C.; Lopes-dos Santos, V.; Bacelo, J.; and Tort, A. B. 2019. Spike sorting with Gaussian mixture models. *Scientific reports*, 9(1): 3627.

Steinmetz, N. A.; Aydin, C.; Lebedeva, A.; Okun, M.; Pachitariu, M.; Bauza, M.; Beau, M.; Bhagat, J.; Böhm, C.; Broux, M.; Chen, S.; Colonell, J.; Gardner, R. J.; Karsh, B.; Kloosterman, F.; Kostadinov, D.; Mora-Lopez, C.; O'Callaghan, J.; Park, J.; Putzeys, J.; Sauerbrei, B.; van Daal, R. J. J.; Vollan, A. Z.; Wang, S.; Welkenhuysen, M.; Ye, Z.; Dudman, J. T.; Dutta, B.; Hantman, A. W.; Harris, K. D.; Lee, A. K.; Moser, E. I.; O'Keefe, J.; Renart, A.; Svoboda, K.; Häusser, M.; Haesler, S.; Carandini, M.; and Harris, T. D. 2021. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539): eabf4588.

van den Oord, A.; Li, Y.; and Vinyals, O. 2019. Representation Learning with Contrastive Predictive Coding. arXiv:1807.03748.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2023. Attention Is All You Need. arXiv:1706.03762.

Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 1096–1103.

Vishnubhotla, A.; Loh, C.; Srivastava, A.; Paninski, L.; and Hurwitz, C. 2023. Towards robust and generalizable representations of extracellular data using contrastive learning. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 42271–42284. Curran Associates, Inc.

Wu, Z.; Xiong, Y.; Yu, S.; and Lin, D. 2018. Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination. arXiv:1805.01978.

Yger, P.; Spampinato, G. L.; Esposito, E.; Lefebvre, B.; Deny, S.; Gardella, C.; Stimberg, M.; Jetter, F.; Zeck, G.; Picaud, S.; Duebel, J.; and Marre, O. 2018. A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo. *eLife*, 7: e34518.

Zajzon, B.; Dahmen, D.; Morrison, A.; and Duarte, R. 2023. Signal denoising through topographic modularity of neural circuits. *eLife*, 12: e77009.

Zhang, Y.; Han, D.; Wang, Y.; Lv, Z.; Gu, Y.; and Li, D. 2025. SimSort: A Data-Driven Framework for Spike Sorting by Large-Scale Electrophysiology Simulation. arXiv:2502.03198.

Reproducibility Checklist

This paper:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes)
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes)
- Provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper (yes)

Does this paper make theoretical contributions? (yes)

- All assumptions and restrictions are stated clearly and formally. (yes)
- All novel claims are stated formally (e.g., in theorem statements). (partial)
- Proofs of all novel claims are included. (yes) Proof sketches or intuitions are given for complex and/or novel results. (yes)
- Appropriate citations to theoretical tools used are given. (yes)
- All theoretical claims are demonstrated empirically to hold. (yes)
- All experimental code used to eliminate or disprove claims is included. (yes)

Does this paper rely on one or more datasets? (yes)

- A motivation is given for why the experiments are conducted on the selected datasets (yes)
- All novel datasets introduced in this paper are included in a data appendix. (yes)
- All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (yes)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (yes)
- All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing. (NA)

Does this paper include computational experiments? (yes)

- This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (yes)
- Any code required for pre-processing data is included in the appendix. (yes).
- All source code required for conducting and analyzing the experiments is included in a code appendix. (yes)
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes)
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (partial)
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes)
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (partial)

- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes)
- This paper states the number of algorithm runs used to compute each reported result. (yes)
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (no)
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (no)
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (yes)