UnitRefine: A Community Toolbox for Automated Spike Sorting Curation

Anoushka Jain[1,2], Robyn Greene[3], Chris Halcrow[4], Jake A. Swann[5], Alexander Kleinjohann[6], Federico Spurio[7], Severin Graff[1,2], Björn Kampa[1], Juergen Gall[7,8], Sonja Grün[6], Alessio Buccino[8], Matthias H. Hennig[3†], Simon Musall[1,2,9†]

[1]Institute for Zoology, RWTH Aachen

[2]Institute of Biological Information Processing (IBI-3), Forschungszentrum Jülich

[3]Institute for Adaptive and Neural Computation, School of Informatics, The University of Edinburgh, UK

[4]Centre for Brain Discovery Sciences, Hugh Robson Building, University of Edinburgh, UK

[5]Department of Neuroscience, Physiology and Pharmacology, University College London, London, UK

[6]Institute of Advanced Simulations (IAS-6), Forschungszentrum Jülich

[7]Institute of Computer Science, University of Bonn

[8]Lamarr Institute for Machine Learning and Artificial Intelligence, Technical University Dortmund

[9]Allen Institute for Neural Dynamics, Seattle, WA, USA

[10]Institute of Experimental Epileptology and Cognition Research, University of Bonn, Bonn, Germany

† Shared project supervision

Abstract

Electrophysiological recordings capture signals from hundreds of neurons simultaneously, but isolating single-unit activity often requires manual curation due to limitations in spike-sorting algorithms. As dataset sizes keep increasing, the time and expertise required for accurate and consistent human curation pose a major challenge for experimental labs. To address this issue, we developed UnitRefine, a classification toolbox that leverages diverse machine-learning algorithms to minimize manual curation efforts. Using acute recordings with Neuropixel probes, we collected a large neural dataset with reproducible experimental conditions and had multiple expert human curators label each recording for reliable cluster identification. This carefully labeled dataset served as the foundation for our automated curation system to learn from human annotations and replicate curator decisions. UnitRefine incorporates existing and newly developed quality metrics, including hyper-synchronous spiking events and drifts in firing rate, to automatically separate noise from neural clusters with high accuracy. To address inherent labeling imbalances between well-isolated single-unit clusters and mixed-population activity, we implemented a cascading classification system. UnitRefine uses a comprehensive hyperparameter optimization search across various classification algorithms to identify optimal model parameters. Across recordings, an optimized Random Forest decoder outperformed other approaches, with up to 83% balanced decoding accuracy for unseen recordings. The broad applicability of UnitRefine is demonstrated by its successful performance across diverse labs and recording conditions, including high-density probes in mice, rats, mole rats, primates, and clinical recordings in epilepsy patients. UnitRefine is developed for broad community adoption. It is fully integrated into the open-source SpikeInterface toolbox, allowing users to either apply our pre-trained models or generate new decoders based on their own curation. Our models and new models can be downloaded and shared via HuggingFaceHub.

1 Introduction

Extracellular electrophysiology is a cornerstone technique in neuroscience, enabling researchers to explore how individual neurons and neural populations encode and transmit information. Each recording electrode can capture action potentials from multiple individual neurons, necessitating the isolation and clustering of spiking activity, a process referred to as 'spike sorting'. The goal of spike sorting is to

first identify signals in the recording that represent putative action potentials, and then cluster them into groups, representing the activity of individual neurons (single-units)[1].

Traditional spike sorting techniques rely on manual sorting and curation, and yield reliable results for individual electrodes or low-density probes such as tetrodes[2]. However, with the increasing number of recording sites on modern devices[3], automated spike sorting solutions have been developed to improve the scalability and reliability of spike sorting while reducing the need for manual curation. These developments are critical for high-density multi-electrode arrays (HDMEAs) with hundreds or thousands of electrodes, where manual approaches are infeasible[4]. Large-scale datasets from acute recordings of brain-wide neural activity during different behaviors are now also openly available, establishing HDMEA recordings as a quasi-standard for many efforts to study neural circuit function[5].

The significant volume of data generated by HDMEAs is a substantial challenge for scalable data processing and analysis[6], [7]. Although automated spike sorters continuously improve and leverage HDMEAs' features, such as recording the same action potential across multiple electrodes, current algorithms still struggle to reliably isolate single-unit activity[8]. Different experimental settings also introduce various sources of contaminating noise and artifacts that can strongly affect spike sorting performance. For example, combining electrophysiology with optogenetic manipulation or functional imaging can introduce light-induced artifacts that can occlude spiking activity[9]. Similarly, recordings in awake animals or humans introduce movement-related artifacts from muscle activity. These factors can limit the quality of single-unit isolation, and, in turn, affect the subsequent interpretation of neural recordings.

A potential solution lies in using metrics that quantify the quality of single-unit identification[10], [11]. These metrics focus on different features of action potential waveforms, cluster separation quality, or biophysical properties such as refractory periods[12]. However, the computed values of these metrics may vary depending on experimental conditions, such as recording probes, data acquisition systems, preprocessing steps, and spike sorting algorithms.

How different quality metrics should be used to reject noise and identify single-unit activity also differs across brain regions and species. This variability makes it challenging to standardize metric computations across laboratories and experimental conditions to derive robust rules for automated curation[13].

One promising approach is Bombcell[14], which aims to address these issues by providing an analysis workflow to compute quality metrics for Neuropixels data, subsequent application of adjustable metric thresholding rules, and a user interface to inspect the results and further refine classifications. However, this approach still requires manual fine-tuning to achieve optimal results and lacks the flexibility for seamless application across different brain areas without substantial manual intervention.

To overcome these limitations and enable an automated spike sorting curation, we have developed UnitRefine, a flexible open-source framework for the classification of spike clusters. We use manually labeled data to train a classifier based on a large set of quality metrics computed from the clustered units. We show that this method works, giving accuracy scores comparable to expert human curators. Moreover, our approach generalizes reliably across different experimental settings, with high classification accuracy across recording devices, including Neuropixels probes, Utah arrays, or wire bundles, and different species, including mice, rats, mole rats, monkeys, and humans. To make the approach widely accessible, UnitRefine is fully integrated into the Python-based open source toolbox SpikeInterface[8]. This allows processing of electrophysiological data from many different recording systems, the use of various spike sorters, and a standardized computation of quality metrics. To apply our pre-trained curation model, a SpikeInterface user only needs to add a single new function call to their analysis pipeline. Users can also train their own classifier models using the quality metrics integrated into SpikeInterface or incorporate custom-made metrics tailored to their specific experiments. Our approach represents a highly versatile and powerful tool to efficiently curate large neural datasets and accelerate neuroscientific discovery.

2 Results

2.1 Evaluation of manual curation of sorted clusters

To assess the quality of automated spike sorting, we analysed a large set of acute recordings in mice with Neuropixels 1.0 probes[15], targeting cortical and subcortical structures (Figure 1A, left). We find

that such recordings can be of highly variable quality and differ in baseline noise levels and visibility of spiking activity (Figure 1A, right traces). Moreover, the introduction of optogenetic stimulation created transient deflections across multiple channels that temporally coincided with light delivery.

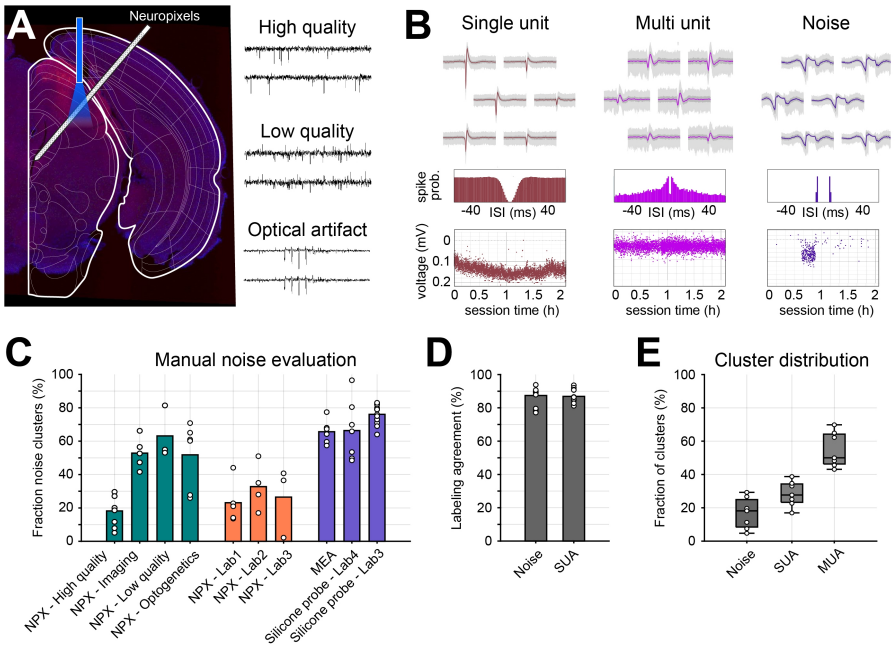

Figure 1: **Multi-curator evaluation of different electrophysiological datasets. A)** Left: Example brain slice with placement of a Neuropixels (NPX) probe in V1 and the deeper SC. The blue marker indicates placement of a light fiber for optogenetic stimulation. Right: Example traces for high- and low-quality recordings and optical artifacts. **B)** Example clusters for single-unit (SUA), multi unit (MUA), and noise clusters. Top row shows the spike waveforms of each cluster across 6 recording channels. Middle row shows cross-correlogram of inter-spike intervals (ISIs) and bottom row shows the amplitude of all detected spikes across the entire recording. **C)** Manual evaluation of spike sorted clusters shows the fraction of noise clusters (%) across different recording tpyes: 1) NPX of high or low quality and with additional functional imaging or optogenetic stimulation (green). 2) NPX recordings from other laboratories (orange), and 3) silicon probe and MEA recordings (purple). **D)** Labeling agreement across human curators for either separating noise from neural and MUA from SUA clusters (n = 11 recordings). **E)** Distribution of cluster labels, showing the fraction of noise, MUA, and SUA labels across all high quality recordings (n = 7 recordings). Error bars indicate median ± 95% confidence intervals; individual recordings are shown as circles.

For spike sorting, we first isolated working channels on the probe, removed common-mode noise using a time-shifted median subtraction (a technique that estimates and removes shared background fluctuations across channels), applied a highpass filter and then used the Kilosort 2.5 sorter[16] to extract clusters.

We then implemented a standardized manual labeling protocol with five well-trained human curators to classify the resulting clusters using the curation software Phy[1]. All clusters were categorized into three distinct groups as shown in Figure 1B: single-unit activity (SUA), multi-unit activity (MUA), or noise. SUA clusters were identified using the following three criteria: 1.) Consistent spike waveforms across the cluster, with minimal variability and negligible interference from other waveform shapes. 2.) A smooth correlogram of inter-spike-intervals (ISIs) with a clear dip in the center. 3.) Stable spike amplitudes throughout the recording session without much dispersion along the Y-axis (Figure 1B, bottom row). MUA clusters contained neural but overlapping waveforms with lower amplitude and less variation across recording sites, and a larger number of refractory period violations due to contributions from multiple neurons. Noise clusters were characterized by irregular waveforms, ISI correlograms lacking clear refractory periods, and inconsistent spike amplitude distributions throughout the recordings.

Manual curation revealed a large fraction of noise clusters in all recordings (Figure 1C, green). The

[1] https://github.com/cortex-lab/phy/

proportion of noise clusters was particularly high in low-quality recordings, or those combined with optogenetic stimulation or functional imaging (52.6-60.6% median noise occurrence). However, even in high-quality recordings, approximately 20% of all clusters were due to non-neural noise, demonstrating that this issue is not limited to recordings with compromised experimental settings. Such high noise levels significantly impact downstream analyses by potentially introducing false correlations between neurons, skewing population activity patterns, and leading to inaccurate estimates of neural response properties. To show this issue is common across labs, we also analyzed Neuropixels data from other research groups and found similar results (20.9-36.5% median noise occurrence; Figure 1C, orange). Moreover, the proportion of noise clusters was even larger for other recording methods such as multi-channel silicone probes or in vitro multi-electrode array (MEA) recordings (65.7-77.5% median noise occurrence; Figure 1C, purple).

To assess the consistency of curator labels, we computed the cross-curator labeling agreement across 11 recordings (7 high quality and 4 optogenetic recordings). Each dataset was annotated by two to five trained humans. Across all recordings, we found high inter-curator reliability for noise and SUA ($\text{Agreement}_{\text{Noise}} = 87.9\%\ [77.9 - 89.9], \text{Agreement}_{\text{SUA}} = 86.4\%\ [84.5 - 91.0]$; median [95% CIs]) (Figure 1D).

We next focused on the 7 high-quality recordings without additional experimental noise sources as representative acute Neuropixels recordings, to develop a curation model. All recordings were performed in the primary visual cortex (V1) and the superior colliculus (SC, as shown in Figure 1A) and we only used cluster labels where the majority of curators agreed on the label. This approach resulted in a dataset with 2,945 well-curated clusters from 5 animals from recordings which were all obtained under similar recording conditions. This comprehensively labeled dataset with annotations from all human curators is publicly available at the UnitRefine Github repository[2].

2.2 Novel cluster metrics for automated cluster classification

After manually labeling the spike-sorted clusters, we next quantified how effectively different cluster metrics could predict the human-assigned labels. SpikeInterface implements the computation of many quality metrics in a consistent framework [3]. We provide a description of all metrics used in Appendix 4. Following the SpikeInterface convention, we splitted the metrics into two broad subcategories:

- **Quality metrics** are widely used in the field to assess the reliability of spike sorting results[11], [17]. These can be further subcategorised. Contamination metrics, such as the number and ratio of refractory period violations, identify false positives by highlighting physiologically implausible spike trains. Completeness metrics, such as presence ratio and amplitude cutoff, quantify the fraction of potentially missing spikes in each cluster due to low signal amplitude or tissue drift. Isolation metrics, such as isolation distance or L-ratio, evaluate how well a given cluster is separated from other clusters in feature space, providing a measure of cluster separability. A detailed description of different quality metrics can also be found in the SpikeInterface documentation[4] and the Allen SDK[5]).

- **Template Metrics** capture the biophysical properties of neural waveforms that distinguish genuine neural activity from artifacts. These include temporal characteristics of action potentials, such as the peak-to-valley duration or full-width half maximum, the peak-to-trough ratio of spike amplitudes, and dynamic waveform features, such as the recovery and repolarization slopes. Template metrics are particularly valuable because they leverage the distinctive shape properties of neural action potentials, which often differ substantially between neural activity and noise[6].

In addition to existing metrics, we devised several new quality metrics to address specific features of noise and SUA clusters that were not fully captured by established approaches. We developed three novel metric types:

[2] https://github.com/anoushkajain/UnitRefine
[3] https://spikeinterface.readthedocs.io/en/latest/tutorials/qualitymetrics/plot_3_quality_metrics.html
[4] https://spikeinterface.readthedocs.io/en/latest/modules/qualitymetrics.html
[5] https://allensdk.readthedocs.io/en/latest/_static/examples/nb/visual_behavior_neuropixels_quality_metrics.html
[6] https://spikeinterface.readthedocs.io/en/latest/modules/postprocessing.html

1. Synchrony metrics to quantify the simultaneous occurrence of spikes across multiple clusters. This metric was based on earlier work that highlighted the potential of synchronicity measures to identify noise clusters due to electrical interference or crosstalk between channels[18]. To leverage this feature, we established 3 separate measures to quantify the number of simultaneous spikes across either 2, 4 or 8 different clusters.

2. Amplitude variation metrics to quantify the stability of spike amplitudes over time, which should be high for well-isolated neurons. The amplitude variation is based on the ratio between the standard deviation and the mean of spike amplitude across time and then quantified by computing the range between the 5th and 95th percentile across the recording.

3. The firing range quantifies the dynamic range of the firing rate of a given cluster over the recording, which should also be largely stable for neural clusters. In contrast, noise clusters tend to exhibit aberrant fluctuations in firing rate with moments of large spread at specific recording times (see Fig. 1B, bottom right panel).

We have implemented these new quality metrics in SpikeInterface, including detailed documentation on their use and implementation[7].

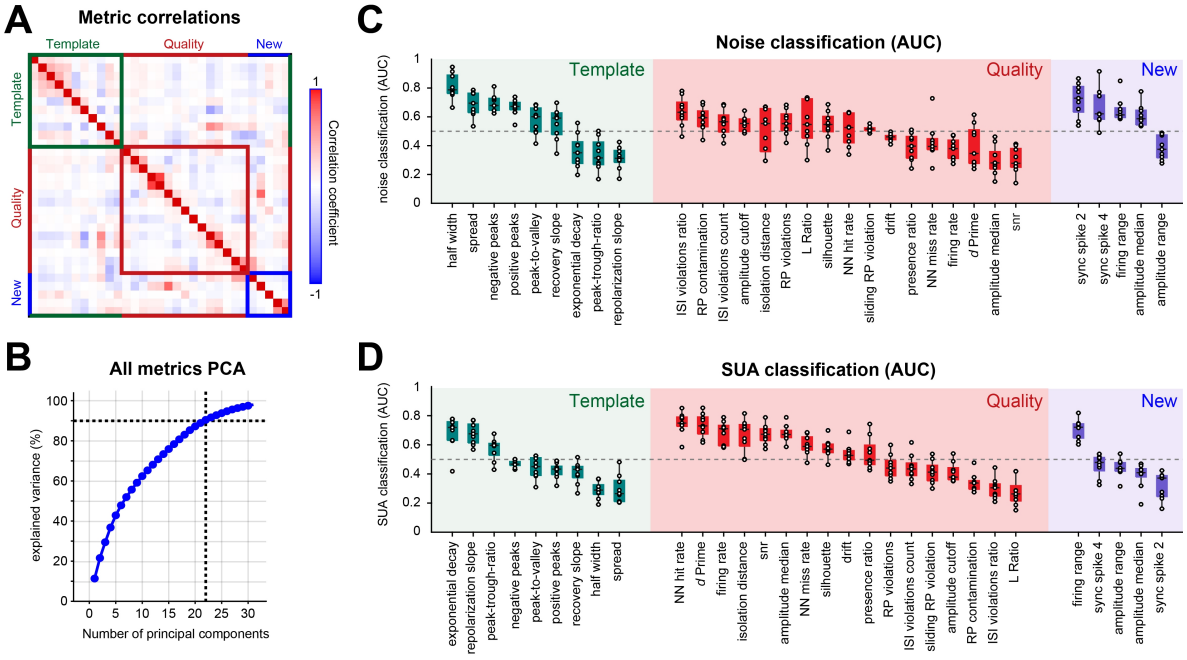

Figure 2: **Discriminative power of neural quality metrics. A)** Correlation matrix showing relationships between all metrics, with hot and cold colors indicating positive and negative correlations, respectively. **B)** Cumulative explained variance from principal component analysis, demonstrating that 22 components are required to explain 90% of variance (dashed lines). **C)** AUC values for distinguishing noise from non-noise clusters, using established template metrics (green), quality metrics (red), and new quality metrics (purple). **D)** Same as in C) but with AUCs for SUA labels. Box plots show median (central line), interquartile range (box), and 95% confidence intervals (whiskers). Metrics are arranged in descending order of the median AUC.

2.3 Discriminative power of individual cluster metrics

Cluster metrics are designed to capture various aspects of noise, MUA, and SUA clusters[19], making them suitable candidates for an automatized cluster classification approach. Before designing a metric-based classifier, we evaluated how informative individual metrics are to identify noise, MUA and SUA clusters. To test for redundancy, we first computed the correlation matrix of all metrics across all curated clusters (Figure 2A). The metrics were grouped as template metrics (dark green), quality

[7] https://spikeinterface.readthedocs.io/en/0.100.1/modules/qualitymetrics.html

metrics (red), and new metrics (blue) and we used hierarchical clustering for each group to identify groups of correlated metrics (visible as clusters in the correlation matrix). Overall, cluster metrics were only weakly correlated to each other with few exceptions, such as high correlation between the geometric median spike amplitude and the signal-to-noise ratio.

To further evaluate redundancy among different cluster metrics, we performed principal component analysis (PCA) on all metrics and plotted the cumulative explained variance of the principal components (Figure 2B). The cumulative explained variance increased gradually, requiring 22 principal components to account for 90% of the total variance across metrics. This gradual increase, along with the low correlation among metrics, indicates that information is distributed across multiple metrics rather than being concentrated in a few key components. The high-dimensional structure of the metrics space suggests that combining all metrics as inputs to a classifier should enhance classification performance compared to using a reduced set of metrics or applying dimensionality reduction as a pre-processing step.

To assess the relationship between individual metrics and curator labels, we quantified their ability to differentiate between various cluster labels using area under the receiver operating characteristic (AUC) analysis[20]. High AUC values approaching 1 indicate that the metric is predictive of a specific cluster label, while low values nearing 0 suggest an inverse predictive relationship. Values around 0.5 imply no clear association between the metric and cluster labels. First, we computed the AUC for each metric to distinguish noise from neural labels (Figure 2C). As expected, several metrics were informative of noise clusters, especially the half-width of the cluster waveform and our new synchrony metrics (sync spike 2 and 4). Individual quality metrics tended to be less predictive but almost all metrics were significantly different from 0.5 (Wilcoxon signed-rank test, $p < 0.05$), indicating their potential utility for noise identification. Next, we calculated the AUCs for each metric to distinguish MUA from SUA labels (Figure 2D). Again, most metrics showed significant differences from 0.5, suggesting they can be leveraged to identify SUA clusters. Quality metrics were particularly informative, with low inter-spike-interval violations and L ratios being especially predictive of SUA clusters. Moreover, the new firing range metric showed high predictive potential, while synchronicity was inversely related to SUA clusters.

These results reveal three important insights for automated curation. First, no single metric adequately captures all aspects of neural quality; a combination of metrics from different categories is necessary for robust classification. Second, the most effective metrics represent all three categories (template metrics, quality metrics and new metrics). Finally, the high dimensionality (22 PCs for 90% variance) suggests that it is challenging to manually develop heuristic curation approaches with high specificity. Instead, a machine learning model trained on a comprehensively annotated and sufficiently large dataset of labeled clusters can potentially be more effective for automated curation.

2.4 Development and evaluation of automated curation approach

To assess the effectiveness of a machine learning-based curation approach, we systematically evaluated the performance of different classifiers for decoding cluster labels. We split the classification problem into two parts, first we trained a model to distinguish between noise and neural clusters, and subsequently, for the neural clusters we distinguish between MUA and SUA clusters. We trained models using multiple classification paradigms: ensemble-based methods (Random Forest, Gradient Boosting, XGBoost, and LGBM), linear models (Logistic Regression), neural networks (MLP implemented in PyTorch), and transformer-based approaches (TabPFN)[21]. The balanced decoder accuracy of each classifier was evaluated across individual recordings under a leave-one-out scheme. Here, the training set for each classifier consisted of all available recordings, except for one held-out recording that was used for testing decoder performance (Figure 3A). Each data point therefore represents the performance of a given classifier on a single unseen recording.

Developing a robust decoder for neural activity classification involves several challenges, including imbalanced labels and effective hyperparameter tuning, because noise and SUA clusters are often very unevenly distributed and vary across different recordings, which can bias model predictions. To address this, we employed stratified sampling to ensure that all cluster labels were equally represented in the training sets. Moreover, we used BayesSearchCV for systematic hyperparameter optimization of each classifier.

For both noise versus neural and MUA versus SUA classification, a Random Forest classifier achieved the highest median balanced accuracy ($\text{Accuracy}_{\text{Noise}} = 84.2\,[76.6 - 89.8]\%, \text{Accuracy}_{\text{SUA}} =$

79.6 [73.2 − 84.5]%; median [95% CIs], (Figure 3A, D)), closely followed by LGBM, XGBoost, and Gradient Boosting. These ensemble-based models consistently performed well, demonstrating their suitability for decoding cluster labels from tabular data. Additionally, the Random Forest decoder exhibited consistent performance characterized by high precision and recall, along with low false positive and false negative rates (Figure 3B, E). In contrast, neural networks such as MLP and the more specialized TabPFN did not achieve the highest classification accuracy in our experiments. However, these results may vary for other experimental datasets. Due to the wide range of models that are implemented in our approach, different classifiers can therefore be flexibily leveraged to achieve the best decoding accuracy for a given application.

To evaluate how classifier performance scales with increasing size of training data set, we examined learning curves for all classifiers (Figure 3C, F). The balanced decoder accuracy improved rapidly as the number of training examples was increased, with the fastest learning classifiers reaching a plateau, roughly within the first 500 labeled clusters. This sugests that while additional data enhances classifier performance, a reasonably small training data set is already sufficient to achieve high classifier performance. Increasing the training data size further still increased the decoder performance further for most classifiers but these gains were modest, suggesting that our dataset with a total of 2945 clusters was sufficiently large.

Notably, Random Forest and boosting-based classifiers (LGBM, XGBoost, Gradient Boosting) maintained a performance advantage across all dataset sizes, suggesting strong generalization capabilities. These findings underscore the importance of dataset size in training effective classifiers, with at least 500 labeled clusters being recommended for stable performance. This comparative evaluation provides key insights relevant to automated neural data curation by highlighting the necessity for models that generalize well across different recording sessions and experimental conditions while emphasizing the need for effective hyperparameter optimization.

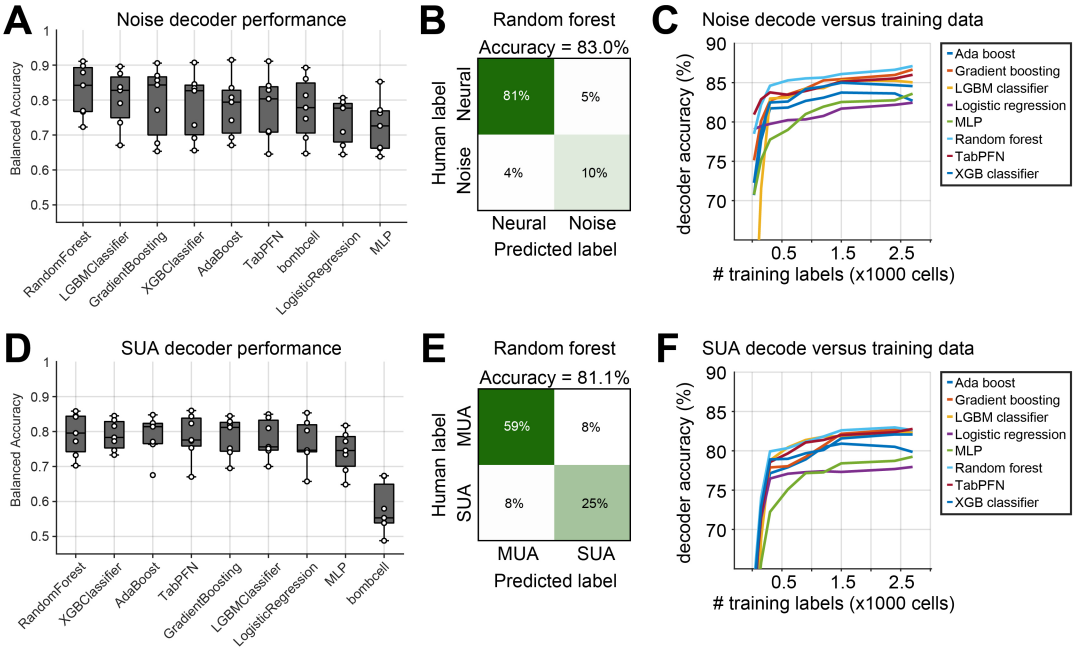

Figure 3: **Comparison of machine learning classifiers for noise and SUA separation. A)** Comparison of balanced accuracy across different classifiers for noise decoding, with Random Forest achieving the highest median performance among all tested models. **B)** Confusion matrix for the Random Forest model, showing an overall balanced accuracy of 83%, with low false positive and false negative rates. **C)** Learning curves show how the balanced accuracy of each classifier improves with increasing training data for noise classification. Random Forest reaches high accuracy the fastest, stabilizing around 500 labeled clusters. **D-F)** Same analyses as (A-C), but for SUA classification. Again, Random Forest outperforms other models, achieving the highest median accuracy with fewer training samples.

Overall, our findings demonstrate that machine learning classifiers are well suited to predict human curator labels from quality metrics. In addition to classifier models, we also compared the

human labels to curation results from bombcell (v1.7.2), a threshold-based method for cluster classification of Neuropixels data[14]. Bombcell also computes relevant quality metrics for each cluster and then performs a series of thresholding steps to categorize cells into noise, MUA, SUA, and non-somatic clusters. Since our curated dataset did not include a non-somatic cluster label, these were excluded before computing balanced decoder accuracy. Note that we only used bombcell with standard settings without further manual optimization of the thresholds to achieve potentially higher classification performance. This approach also achieved high classification accuracy for noise clusters ($\text{BalancedAccuracy}_{\text{Noise}} = 77.8\,[69.2 - 86.1]\%; \text{median}[95\%\ \text{CIs}]$). However, balanced decoding accuracy was lower for SUA ($\text{BalancedAccuracy}_{\text{SUA}} = 55.3\,[53.8 - 67.3]\%$) compared to other classification approaches. Aside from a potential lack of manual optimization, this difference was due to a more conservative identification of SUA clusters, resulting in very high precision (very few false positives) but lower recall (more false negatives) of SUA clusters (Supplementary Figure 1). We also observed similar outcomes when applying a combination of recommended thresholds from the International Brain Laboratory. These findings highlight an inherent trade-off between employing conservative thresholding rules that clearly identify SUA clusters while potentially rejecting other useful clusters and training classifiers that emulate human curator behavior to be more sensitive but might be less principled. Here, one advantage of the Random Forest decoder over neural networks is that the decision tree of a trained model can be further investigated to gain a better understanding of how metrics are used for classification.

2.5 UnitRefine robustly generalizes across experimental settings and species

We next tested whether UnitRefine can robustly generalize across different recording systems, species, and experimental contexts. To assess this, we evaluated the Random Forest model on diverse electrophysiological datasets recorded using different technologies and species (Figure 4).

In 4-shank Neuropixels 2.0 recordings from the CA1 region of the rat hippocampus, we achieved a balanced decoding accuracy of 87.4% (n = 1062 clusters). Similarly, we achieved high balanced decoding accuracy of 81.0% in Neuropixels 2.0 recordings in mole rats (n = 506 clusters). Going beyond Neuropixels, we tested UnitRefine in Utah array recordings in the visual cortex (V1 and V4, n = 80 clusters) of non-human primates during resting-state conditions[22]. Here, the model was trained on data from one Utah array and tested on another within the same brain region, achieving a very high balanced decoding accuracy of 96.0%. Lastly, we tested the performance of UnitRefine on human intracranial electrophysiological recordings from epileptic patients. Unlike previous datasets, which were spike-sorted using Kilosort4, these recordings were processed with the spike sorter Combinato[23]. Despite these methodological differences and a relatively small dataset size (n = 516 clusters), the classifier maintained a reasonable balanced decoding accuracy of 77.8%. This suggests that our approach may also have potential applications in clinical neuroscience to facilitate robust classification of neural units.

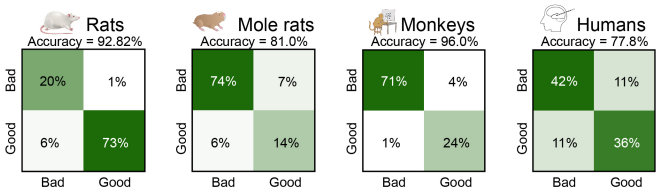

Figure 4: **Label classification performance across different species and recording systems.** Confusion matrices for classification results across species and recording devices. From left to right: Rats (92.82% accuracy), mole rats (81.0% accuracy), monkeys (96.0% accuracy), and humans (77.8% accuracy). To predict labels from different species and recording devices, the Random Forest model was re-trained on labels and cluster metrics for each respective dataset.

Overall, these results demonstrate that UnitRefine effectively generalizes across different electrophysiological recording technologies, species, brain regions, and spike-sorting pipelines while maintaining high overall classification accuracy. The confusion matrices from different application examples further illustrate the balanced performance with equally high precision and recall across all cases.

2.6 Classifier performance with reduced cluster metric combinations

Beyond individual feature importance, we evaluated how model performance depends on different subsets of cluster features. We therefore grouped cluster features into 5 predefined metric lists (Table 2), either based on their physiological relevance or isolating features that required the least computational time:

- The full set of metrics, detailed in Section 2.2. This set has been used to train the classifier studied in all previous sections.
- Light decoder. A reduced set of metrics that are efficient to compute; those which only require the spike times and templates of each cluster.
- Quality metrics. All quality metrics as described in Section 2.2, including the newly added quality metrics.
- Spike time metrics: Temporal properties of spike events, including inter-spike interval violations and synchrony. These are the fastest to compute because they are only based on the spike sorter outputs.
- Template metrics: All template metrics as described in Section 2.2.

We then computed the 3-way balanced decoder accuracy for each classifier, combining classifier performance for noise, MUA, and SUA labeling (Figure 5A, Table 1). As expected, using all metrics achieved the highest overall decoder accuracy, but the light decoder also achieved very high accuracy that was not significantly lower than the classifier with all metrics (Wilcoxon rank sum test, $p = 0.45$). Reducing the set of metrics further decreased the decoder accuracy, with template metrics alone achieving the lowest overall performance.

Metric set	Median balanced accuracy (%)	95% Confidence interval
All metrics	76.2	71.2 - 83.7
Light decoder	76.1	73.4 - 78.7
Quality metrics	72.8	68.1 - 78.4
Spike times metrics	73.0	68.9 - 77.0
Template metrics	66.2	64.8 - 66.0

Table 1: The median balanced accurate for decoders trained using different metric subsets. The list of metrics included in each subset can be found in Appendix 5.

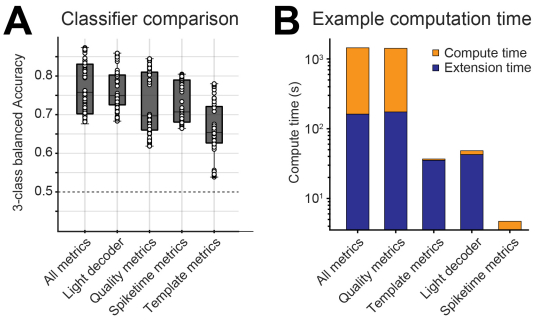

Figure 5: **Label Classifier evaluation with reduced metric sets. A)** Performance of 3-way classifier with different metric sets, combining balanced decoder accuracy for noise and SUA labels. Datapoints show results from individual folds across high quality recordings (n = 70 folds from 7 recordings). Classifier accuracy was highest when using all metrics and lowest when only using template metrics. **B)** Computation time for different metric sets for an example 25-minute long Neuropixels recording. Time in blue is needed for feature extraction, such as the waveforms from individual spikes. Time in orange is required for computing cluster metrics after extraction.

Discussion

In this work, we demonstrate that fully automated curation of spike sortings is feasible and achieves performance comparable to skilled human curators. UnitRefine employs a supervised machine learning approach using established spike sorting quality metrics, trained on manually curated sortings, and generalizes well across similar recordings. We further show that this method achieves high curation performance across recordings from different species, brain areas, and recording systems. UnitRefine is fully integrated into the SpikeInterface framework, enabling standardized computation of quality metrics and facilitating easy reuse and sharing of trained models. This integration provides an intuitive interface for training new models within diverse spike sorting pipelines. Moreover, we offer a standalone implementation for expert users to incorporate into custom analysis workflows—allowing flexible integration of additional cluster metrics for specialized applications such as the automated identification of neurons with specific functional tuning properties.

Designed as a flexible analysis pipeline, UnitRefine accepts diverse raw data preprocessing algorithms and spike sorting methods. It significantly reduces expert manual curation labor, while retaining fidelity in detecting SUA clusters, which are fundamental to numerous scientific studies. Such automated curation of cluster labels has the potential to save hundreds of hours in expert time by either replacing manual curation or strongly reducing the number of clusters to be re-checked to ensure best results. Importantly, as the number of curated datasets increases, the model can be easily retrained to further improve curation performance. To evaluate its generalizability, we also tested the curation performance across different spike sorters, species, recording devices, and brain regions, demonstrating robust applicability across a large range of experimental contexts.

The reported balanced accuracy scores for our models are generally around 80% across different contexts. Since the models were trained on manually curated labels, their overall accuracy can be inherently limited by variability in human curator performance. Indeed, different studies found significant variability among expert curators. The curators in Buccino et al.[8] agreed on $400/525 = 76\%$ of curated clusters. A more extensive study[24] asked experts to manually curate a sorted dataset, allowing them to split and merge clusters as well as label them. After this more invovled process the five curators found 18, 27, 28, 32 and 35 "good" clusters respectivelty, demonstrating the wide variability of results based on the individual curator. For our study, we have shown that curator training increased the curator reliability with higher cross-curator accuracy. We further improved the quality of cluster labels of our core dataset by using the majority vote across curators. Overall, the $\sim 80\%$ accuracy level of our models therefore demonstrates strong classification performance, achieving as comparable or even higher performance as human curator.

The models were trained using thousands of curated clusters utilizing 37 distinct metrics per cluster. Many of these metrics are computationally expensive due to integration of specific features, such as individual spike amplitudes and principal components, which require a scan of the entire raw recording. Despite this complexity, we identified fast-to-compute metric sets yielding acceptable classification results, further increasing the applicability of UnitRefine to applications where long compute times are prohibitife, such as particularly long recording sessions. Additioanlly, only around 500 curated clusters were needed to train a successful decoder with gains in clasification accuracy being much lower after this number. We therefore believe it is viabale for a small lab to curate a reasonably small number of clusters, and implement UnitRefine using a small amount of computation time. By further refinement of initial automated curation results, the amount of available clusters for re-training can also be quickly increased to achieve higher classification performance.

Importantly, UnitRefine is not limited to the specific cluster labels that we used in these study and different users are likely to find new applications, such as the automated classification of specific cluster types, such as interneurons, or flagging cluster merges, Moreover, auto-curated labels could be leveraged to evaluate different analysis pipelines or spike sorters by quickly testing which analysis approach offers a high yield of high-quality clusters. We therefore believe that UnitRefine can be a highly valuable tool across a wide range of experimental settings that include electrophysiological recordings, promoting standardization in the field while reducing manual labor.

One significant obstacle in developing novel SUA classification algorithms is the absence of ground-truth datasets. A notable exception is SpikeForest[25] which combines real ground-truth datasets (of which there are few) and simulated data, which can then be used to compare sorters. Such standardized datasets are essential for reproducibility and comparison between different steps in the spike sorting process. To contribute to these efforts, the spiking data and human labels that were used in the study

have been made openly available in an online repository[26]. With the emergence of large standardized datasets from organizations like the International Brain Laboratory[27] and the Allen Institute for Brain Science, complementing these with diverse datasets from individual laboratories would also enable researchers to evaluate how well methods generalize across different experimental conditions and confirm their resilience to variations in recording setups and preprocessing approaches.

To ensure accessibility and reproducibility, UnitRefine has been developed as an open-source Python package. This facilitates systematic versioning and efficient bug reporting while streamlining troubleshooting processes. Our publicly available pipeline offers researchers a user-friendly tool for more scalable and reliable electrophysiological analysis, serving as a valuable open-source resource to accelerate advances in automated spike sorting and unit classification across the neuroscience community.

3 Methods

3.1 Neural Recording Acquisition and Processing

Neural recordings were acquired using multiple electrode systems including Neuropixels 1.0, Neuropixels 2.0, and other silicon probe and microelectrode array (MEA) configurations. Recordings were obtained from diverse experimental preparations spanning multiple species including rats, mole rats, monkeys, and humans. For Neuropixels recordings, probes were stereotaxically implanted following standard procedures[15]. For details, see also Balla et al, 2023[28]. In experiments with optogenetic stimulation, the light delivery was also coordinated with electrophysiological recording to examine artifactual responses.

Raw electrophysiological data was analyzed with common preprocessing steps, including removing of bad channels, bandpass filtering (300-6000 Hz) and common mode referencing. Spike detection and sorting was performed using Kilosort2.5[29] with default parameters. The initial automated sorting results were then manually curated using Phy 2.0b1 (https://github.com/kwikteam/phy). Units were then categorized into three main classes: (1) single-unit activity (SUA), representing well-isolated individual neurons; (2) multi-unit activity (MUA), representing activity from multiple neurons; and (3) noise, representing non-neural electrical artifacts, optical artifacts, or other non-biological signals. Each dataset was curated by at least 2 human experts that were trained to consistently perform the curation with the criteria, described in the results section.

3.2 Quality metric calculation

We calculated a comprehensive set of quality metrics for each sorted unit to enable automated classification. These metrics included established template-based measures alongside newly developed quality indicators. All metrics were calculated using version 102.0 of SpikeInterface. Descriptions of all metrics used can be found in Appendix 4.

To compute the linear relation between pairs of different cluster metrics, we computed the Pearson correlation coefficients between all pairs of metrics. The resulting correlation matrix was then visualized as a heatmap, with color intensity indicating their respective correlation strength. To ensure that correlated metrics would be clearly visible, we performed hierarchical clustering with Ward's method for each group of cluster metrics. This analysis helped identify redundancies among metrics and informed the selection of the most informative, non-redundant subset of metrics for unit classification.

Principal component analysis was performed on the standardized quality metrics to assess their combined discriminative power and dimensionality. All metrics were normalized to zero mean and unit variance before PCA. The cumulative explained variance was then plotted as a function of the number of principal components to determine the intrinsic dimensionality of the feature space. Based on this analysis, we established that 22 components were required to explain 90% of the variance in the quality metrics, indicating substantial non-redundant information across the metrics.

3.3 AUC analysis for metric performance evaluation

The performance of individual quality metrics and machine learning classifiers was evaluated using the area under the receiver operating characteristic curve (AUC). For each metric and classifier, we calculated the AUC for two classification tasks: (1) distinguishing noise from non-noise (neural signals), and (2) identifying single-unit activity (SUA) from other signal types.

For individual metrics, AUC values were computed by varying the decision threshold across the full range of the metric and calculating the sensitivity and specificity at each threshold. Metrics were then ranked by their median AUC values to identify the most discriminative individual features for each classification task. The AUC ranges from 0 to 1, with 1 being a perfect classifier and 0.5 indicating random guessing.

3.4 Machine Learning Classification Framework

We implemented and compared multiple machine learning classifiers to automate the identification of noise and single-unit activity. We fine-tuned our Sklearn-based[8] classifiers using BayesSearchCV[9], which begins by testing a few random settings and builds a model (usually a Gaussian Process) to predict which settings might yield the best classifier results. The most promising configurations are then tested, with the model updated after each iteration. To ensure reliable performance, we employed 5-fold cross-validation (cv=5), splitting the data into five parts: training on four while testing on the fifth, and repeating this process for each hyperparameter setting. This approach balances exploration and accuracy while minimizing the risk of overfitting.

We evaluated the following classifier implementations: Random Forest, Gradient Boosting, LGBM Classifier, XGB Classifier, AdaBoost, TabPFN, Logistic Regression, Multi-layer Perceptron (MLP) and Bombcell (as a comparison baseline). All classifiers were trained and evaluated using a nested cross-validation approach. For each classifier, we assessed the classifier performance by computing a 2x2 confusion matrix for the held-out test data (as shown in figure 3B,E) and the corresponding balanced accuracy. The training data size requirements were systematically evaluated by incrementally increasing the number of labeled clusters that were used in the training set.

To obtain an estimate of the computation time for each classifier with different metric sets, we analyzed a 25-minute long NeuroPixels 1.0 recording with 384 channels and 242 units. The preprocessed recording was then saved locally before computing all the required extensions. The computation was done on an 8-core workstation with an Intel i-7 CPU and 32 GB of RAM.

In the confusion matrix, given a positive and a negative class, true positive (TP) and true negative (TN) samples are correctly labeled samples, while false positives (FP) are samples which were predicted positive although their true label is negative and false negatives (FN) have been predicted negative although their true label is positive. Based on the confusion matrix, several metrics can be derived. The overall accuracy is the fraction of all correctly classified samples and is calculated as the total number of true positives and true negatives divided by the total number of samples. It measures how well a classifier performs overall but ignores potential imbalances in the data,

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{1}$$

Precision, also known as sensitivity, is the fraction of true positives over the number of positive classified samples (true positive and false positives). It measures the ability of a classifier not to label as positive a sample that is negative,

$$\text{precision} = \frac{TP}{TP + FP}. \tag{2}$$

Recall is the fraction of true positives over the number of all samples that should have been identified as positive (true positives and false negatives). It measures the ability of a classifier to find all the positive samples,

$$\text{recall} = \frac{TP}{TP + TN}. \tag{3}$$

Balanced accuracy, as reported in the classifier results, is the average of recall obtained in each class. It is used to measure the classifier performance when the data set is imbalanced. It gives equal weight to each class, unlike accuracy which is biased towards the class with more observations. Therefore, we used preferred balanced accuracy to quantify decoder performance,

$$\text{balanced_accuracy} = \frac{\text{sensitivity} + \text{specificity}}{2}, \tag{4}$$

where sensitivity = precision = $\frac{TP}{TP+FP}$ and specificity = $\frac{TN}{TN+FN}$.

[8] https://scikit-learn.org/stable/supervised_learning.html
[9] https://scikit-optimize.github.io/stable/modules/generated/skopt.BayesSearchCV.html

3.5 Inter-curator agreement and classifier generalization

To establish ground truth labels and assess variability in human classification, a subset of recordings ($n = 11$) were independently annotated by multiple expert curators. The agreement rates between curators were calculated separately for noise and SUA classifications to establish an upper bound on the expected classifier performance of subsequent analyses.

We evaluated the generalizability of our classification framework across different recording systems (Neuropixels 1.0, Neuropixels 2.0, other silicon probes, and MEAs) and across species (rats, mole rats, monkeys, and humans). For each system and species combination, confusion matrices were generated to assess classification accuracy, true positive rates, and true negative rates.

Acknowledgements

CH is supported by the UKRI Biotechnology and Biological Sciences Research Council (BBSRC) grant number BB/X01861X/1. RG is supported by the UKRI Biotechnology and Biological Sciences Research Council (BBSRC) grant number BB/T00875X/1. AJ and SM are supported by the Helmholtz Association VH-NG-1611 and the state of North-Rhine-Westhphalia through the iBehave initative.

We would like to express our gratitude to our curators—Nilufar Lahiji, Sacha Abou Rachid, Severin Graff, Luca Koenig, and Natalia Babushkina—for their invaluable help in curating the data.

We also thank Alana Darcher and Dr. Florian Mormann for providing the human patient data, Aitor Morales-Gregorio for contributing macaque data, Jake Swann for sharing rat data, and Sylvia Schroeder for Neuropixels data.

Additionally, we thank Alireza Saeedi, Runita Shirshankar, and Pascal Malkemper for providing data from Fukomys anselli. We also thank Joe Ziminski for giving valuable feedback during the SpikeInterface integration process.

4 Appendix: Quality Metric Descriptions

When training our model, we use the implementations of quality and template metrics contained in the SpikeInterface package[8]. These are re-implementations of metrics from a diverse range of papers and from GitHub repositories. We have tried to cite the first appearance of each metric in the literature, though this is difficult for very commonly used metrics.

The `firing_range`, `amplitude_cv_median`, `amplitude_cv_range` and `sync_n` metrics are new, implemented by the authors of this paper. In this appendix, we have collated the metric definitions using unified notation defined below. All metrics listed here are computed for each cluster.

Terminology, conventions and notation

Cluster: Set of spikes, determined by a spike sorting algorithm to be clustered together.
Waveform: The preprocessed voltage trace of a spike. Can exist on many probe channels.
Template: The representative waveform of a single cluster, usually computed as the mean of the cluster's spike waveforms.

We index spikes with lower case letters, clusters with upper case letters.

T : total time of the recording (s)
f : sampling frequeny (Hz)
N : number of spikes
N_I : number of spikes in cluster I.

We define further notation, if needed, in the metric subsections.

Spike train metrics

`num_spikes`
Number of spikes in the cluster.

`firing_rate`
Average number of spikes in the cluster, per second.

`firing_range`
For each cluster, the spike train is binned in time and a firing rate is computed for each time bin. This produces a distribution of firing rates. The firing range is the difference between the 5th and 95th percentile firing rates.

`presence_ratio`
For each cluster, the spike train is binned in time. The presence ratio is equal to the fraction of bins containing at least one spike.
`sync_n`
Take all spike times from cluster I. Count all spikes which occur at the same timepoint as at least $n-1$ spikes from any other cluster. `sync_n` is equal to this count.

Refractory period violation metrics

Additional notation

T_r : refractory period (s).
Once a neuron has fired an action potential, it cannot do so again for a duration known as the refractory period. If two spiking events occur within this time interval, a refractory period violation has occurred. The length of the refractory period is different in different neuron types, but is usually a few milliseconds.

`isi_violations_count`
The number of inter-spike-interval (isi) violations that occur for each cluster. Here, the number of violated refractory periods are counted. Hence if two spikes violate one period, one violation is counted (contrast with `rp_violations`).

`isi_violations_ratio` [11]
Given the `isi_violations_count`, the `isi_violations_ratio` approximates the contamination rate of the cluster, assuming random contamination. For a cluster I, the approximation is given by

$$\texttt{isi_violations_ratio} = \frac{\texttt{isi_violations_count}}{N_I^2} \times \frac{T}{2(T_r - 1/f)} \tag{5}$$

`rp_violations`
The number of refractory period (rp) violations that occur for each cluster. Here the total number of violations is counted. Hence if two spikes violate another spikes' refractory period, two violations are counted (contrast with `isi_violations`).

`rp_contamination` [30]
Given the `rp_violations` count, the contamination rate of the cluster is approximated, assuming random contamination. The approximation is a refinement of `isi_violations_ratio` and works better for highly contaminated clusters. For a given cluster I, the ratio is given by

$$\texttt{rp_contamination} = 1 - \sqrt{1 - \frac{\texttt{rp_violations}}{N_I^2}\frac{T}{T_r}} \tag{6}$$

`sliding_rp_violation` [31]
Assume that a percentage of the cluster's spikes come from contamination. Compute the likelihood of observing the computed `isi_violations` for a given refractory period, assuming Poisson spiking. Do this for several refractory periods. If the confidence is above a 90% confidence threshold, keep this datapoint. The `sliding_rp_violation` is equal to the minimum contamination rate of these clusters which have pass the threshold. Allows a metric to be computed for neurons with an unknown refractory period.

Amplitude metrics

Additional notation

spike amplitudes: Using either `max`, `min` or `max_abs` as the extremal function, find the channel whose cluster template has the maximal extremal value. Call this the extremal channel. For each spike, the spike amplitude is equal to the extremal amplitude on the extremal channel.
amplitude cv: Consider the distribution of spike amplitudes for some subset of spikes. The ampltiude coefficient of variation (cv) is equal to the standard deviation of this distribution, divided by the mean.

`snr`
The snr is equal to the extremal amplitude of the template on the extremal channel, divided by the standard deviation of the noise on this channel.

`sd_ratio` [32]
Compute the spike amplitudes for a simulated cluster composed of many snippets of noise, giving a noise-amplitude distribution. The `sd_ratio` is equal to the standard deviation of the cluster's amplitude distribution divided by the standard deviation of the noise-amplitude distribution.

`amplitude_cv_median`
For each cluster, time bin the spike amplitudes and compute the `amplitude_cv` for each bin. The `amplitude_cv_median` is the median of he values across all bins.

`amplitude_cv_range`
For each cluster, time bin the spike amplitudes and compute the `amplitude_cv` for each bin. Find the 5th and 95th percentile values. The `amplitude_cv_range` is the difference between these values.

`amplitude_median`
The median of the distribution of the spike amplitudes for each cluster.

`amplitude_cutoff`[5]
For each cluster, make a histogram of spike amplitude values. Assume the distribution is normally distributed and fit the distribution using the high-value tail. Under this assumption, estimate how many spikes are missing from the low-value tail. The `amplitude_cutoff` is equal to the estimated fraction of spikes missing from the tail.

Drift metrics

Additional notation

Spike location difference distribution: for each cluster, compute the location of each spike, giving a distribution of spike locations relative to the probe. Then compute the difference between the locations and the median location. This gives a distribution of spike location differences for each cluster.

`drift_ptp`
The difference between the maximum and minimum values in the spike location difference distribution.

`drift_std`
The standard deviation of the spike location difference distribution.

`drift_mad`[5]
The median absolute deviation of the spike location difference distribution. More precisely,

$$\texttt{drift_mad} = \texttt{median}\left(\texttt{abs}\left(\texttt{spike_location_difference} - \texttt{mean}\left(\texttt{spike_location_difference}\right)\right)\right) .$$

Principal Components metrics

For these metrics, the principal components of each spike waveform are first computed. When distance is measured, this means distance measured by a metric, most commonly the L_2 metric, on principal

component space. The Mahalanobis distance accounts for cluster means and variations.

`isolation_distance` [33]
Compute the the Mahalanobis distance between cluster I and all other clusters. The `isolation_distance` for cluster I is equal to the minimum of all these distances.

`silhouette`[34]
Take a cluster I. For every spike $i \in I$, compute the average distance to every other spike from its own cluster and call this $a(i)$. Call the other clusters $B_1, B_2, \ldots$. Now compute the average distance from i to every spike in each of these clusters and call these distances $b_1(i), b_2(i), \ldots$. Take the shortest of these distances, $b(i) = \min_I b_I(i)$. The sillhoutte score of this spike is equal to

$$s(i) = \frac{\max(b(i) - a(i))}{\max\left((a(i), b(i)\right)} \,. \tag{7}$$

The cluster sillhoutte score of the cluster, `silhouette`, is the average of s(i) across all spikes.

`nn_hit_rate`[5]
For each spike in a cluster A, find `n` nearest neighbour spikes. The spike hit rate is the fraction of these spikes which are also in A. The cluster spike rate is the mean of all spike hit rates for spikes in A.

`nn_miss_rate`[5]
For each spike not in a cluster A, find the `n` nearest neighbour spikes. The spike miss rate is the fraction of these spikes which are in A. The cluster spike rate is the mean of all spike hit rates for spikes not in A.

`d_prime`[11]
Consider the spikes in one cluster A, and the set of all other spikes $\neg A$. Construct a one-component Linear Discrimiation Model with respect to these sets, which contains a mean and standard deviation for each set of spikes $\mu_A, \mu_{\neg A}, \sigma_A, \sigma_{\neg A}$. The `d_prime` is equal to the normalised difference in these means:

$$\texttt{d_prime} = \frac{\mu_A - \mu_{\neg A}}{\sqrt{2\left(\sigma_A^2 + \sigma_{\neg A}^2\right)}} \tag{8}$$

`l_ratio`[35]
Assume all principal components for each cluster come from indepdendent, normal distributions. This gives a probability χ^2 distribution on pca space. Consider all spikes that do not belong to cluster I, s_j. Using the Mahalanobis distance in PCA space between the spike s_j and the cluster centroid C_I, we can compute the probability that the spike s_j should have been sorted into I. It is given by

$$P(s_j, C_I) = 1 - \mathrm{CDF}_\chi\left(D^2_{s_j, C_I}\right) \tag{9}$$

The `l_ratio` for a cluster I is then equal to

$$\texttt{l_ratio} = \sum_j P(s_j, C_I)\,. \tag{10}$$

Template metrics

This set of template metrics mostly derive from [5].

Additional notation

A template is defined on a number of channels. The template on each channel has a `peak` (the maximum value) and a `valley` (the minimum value). We call the channel with the extremal value (either with respect to `max`, `min` or `abs_max`) the extremal channel. Many metrics (`peak_to_valley`, `get_peak_trough_ratio`, `half_width`, `repolarization_slope`, `recovery_slope`, `num_positive_peaks`, `num_negative_peaks`) only use the template on the extremal channel. We plot visual representations of several of these metrics in Figure 6.

`peak_to_valley` (s)

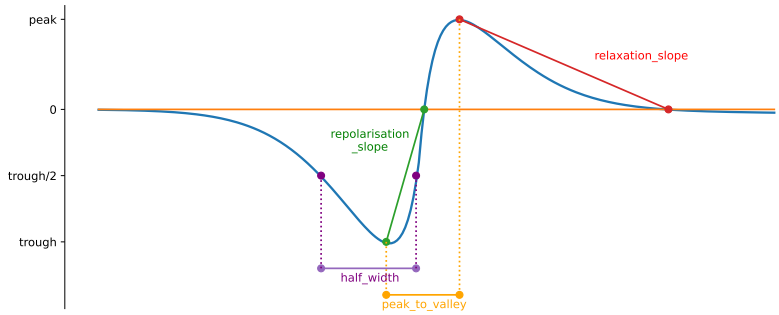

Figure 6: A visual representation of some of the template metrics.

The time between the peak and valley of the template on the extremal channel.

`peak_trough_ratio`
The ratio ‘peak ÷ trough‘.

`num_positive_peaks`
The number of positive peaks of the template on the extrumum channel.

`num_negative_peaks`
The number of negative peaks of the template on the extrumum channel.

`half_width` (s)
Consider the `trough`. The half_width is equal to the maximum interval around the valley index where all template values lie between the `trough` value and half of the `trough` value.

`repolarization_slope` ($\mu V/s$)
Consider the line from the `trough` to the first zero-value following the `trough`. The `repolarization_slope` is the gradient of this line.

`recovery_slope` ($\mu V/s$)
Consider the line from the `peak` to the first zero-value following the `peak`. The `recovery_slope` is the gradient of this line.

`exp_decay` ($1/\mu m$)
On every channel, find both the maximum amplitude and the distance from the extremum channel. Use this to produce amplitude as a function of distance $A(d)$. Fit this function to an exponential decay $A(d) \approx A_0 \exp(-ad)$. The metric `exp_decay` is equal to the decay rate a.

`velocity_above`
On every channel above the extremum channel, find the time index of the extremal value of the template and the distance from the extremum channel. Using this, produce distance as a function of extremum-amplitude-temporal-location, $d(t)$. Fit this using a linear regression. The gradient represents the velocity of the spike, and we set `velocity_above` equal to this.

`velocity_below`
On every channel below the extremum channel, find the time index of the extremal value of the template and the distance from the extremum channel. Using this, produce distance as a function of extremum-amplitude-temporal-location, $d(t)$. Fit this using a linear regression. The gradient represents the velocity of the spike, and we set `velocity_below` equal to this.

`spread`
Find all channels containing a channel-template whose amplitude is somewhere above a threshold. The `spread` is equal to the maximum distance between the channels which pass this threshold test.

5 Appendix: Metrics used in each decoder

Metric List	Features
All Metrics	num_spikes, firing_rate, presence_ratio, snr, isi_violations_ratio, isi_violations_count, rp_contamination, rp_violations, sliding_rp_violation, amplitude_cutoff, amplitude_median, amplitude_cv_median, amplitude_cv_range, sync_spike_2, sync_spike_4, sync_spike_8, firing_range, drift_ptp, drift_std, drift_mad, isolation_distance, l_ratio, d_prime, silhouette, nn_hit_rate, nn_miss_rate, peak_to_valley, peak_trough_ratio, half_width, repolarization_slope, recovery_slope, num_positive_peaks, num_negative_peaks, velocity_above, velocity_below, exp_decay, spread
Quality Metrics	num_spikes, firing_rate, presence_ratio, snr, isi_violations_ratio, isi_violations_count, rp_contamination, rp_violations, sliding_rp_violation, amplitude_cutoff, amplitude_median, amplitude_cv_median, amplitude_cv_range, sync_spike_2, sync_spike_4, sync_spike_8, firing_range, drift_ptp, drift_std, drift_mad, isolation_distance, l_ratio, d_prime, silhouette, nn_hit_rate, nn_miss_rate
Template Metrics	peak_to_valley, peak_trough_ratio, half_width, repolarization_slope, recovery_slope, num_positive_peaks, num_negative_peaks, velocity_above, velocity_below, exp_decay, spread
Spike Times Metrics	num_spikes, firing_rate, presence_ratio, snr, isi_violations_ratio, isi_violations_count, rp_contamination, rp_violations, sliding_rp_violation, sync_spike_2, sync_spike_4, sync_spike_8, firing_range
Light Decoder Metrics	firing_rate, presence_ratio, snr, isi_violations_ratio, isi_violations_count, rp_contamination, rp_violations, sliding_rp_violation, sync_spike_2, sync_spike_4, sync_spike_8, firing_range, sd_ratio, peak_to_valley, peak_trough_ratio, half_width, repolarization_slope, recovery_slope, num_positive_peaks, num_negative_peaks, velocity_above, velocity_below, exp_decay, spread

Table 2: Overview of the different metric lists that were included in the reduced classifiers.

6 Appendix: Supplementary Figures

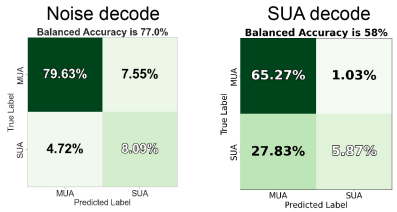

Figure S1: **Label classification performance for Bombcell curation.** Confusion matrices for classification of either noise or SUA clusters. While noise classification shows balanced accuracy and recall, SUA clustering is conservative with high precision but lower recall.

References

[1] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. "Past, present and future of spike sorting techniques". In: *Brain Research Bulletin* 119 (2015), pp. 106–117.

[2] Lynn Hazan, Michaël Zugaro, and György Buzsáki. "Klusters, NeuroScope, NDManager: A free software suite for neurophysiological data processing and visualization". In: *Journal of Neuroscience Methods* 155.2 (2006), pp. 207–216. DOI: 10.1016/j.jneumeth.2006.01.017. URL: https://doi.org/10.1016/j.jneumeth.2006.01.017.

[3] Nicholas A. Steinmetz et al. "Challenges and Opportunities for Large-Scale Electrophysiology With Neuropixels Probes". In: *Current Opinion in Neurobiology* (2018). DOI: 10.1016/j.conb.2018.01.009.

[4] Devyn Lee Bauer et al. "3D Printed Guide Tube System for Acute Neuropixels Probe Recordings in Non-Human Primates". In: *Journal of Neural Engineering* (2023). DOI: 10.1088/1741-2552/acd0d7.

[5] Joshua H. Siegle et al. "Survey of Spiking in the Mouse Visual System Reveals Functional Hierarchy". In: *Nature* (2021). DOI: 10.1038/s41586-020-03171-x.

[6] Jelena Dragas et al. "Complexity Optimization and High-Throughput Low-Latency Hardware Implementation of a Multi-Electrode Spike-Sorting Algorithm". In: *Ieee Transactions on Neural Systems and Rehabilitation Engineering* (2015). DOI: 10.1109/tnsre.2014.2370510.

[7] Alessio Paolo Buccino, Samuel Garcia, and Pierre Yger. "Spike sorting: new trends and challenges of the era of high-density probes". In: *Progress in Biomedical Engineering* 4.2 (2022), p. 022005. DOI: 10.1088/2516-1091/ac6b96. URL: https://doi.org/10.1088/2516-1091/ac6b96.

[8] Alessio Paolo Buccino et al. "SpikeInterface, a unified framework for spike sorting". In: *Elife* 9 (2020), e61834.

[9] Thomas Zhihao Luo et al. "An Approach for Long-Term, Multi-Probe Neuropixels Recordings in Unrestrained Rats". In: *Elife* (2020). DOI: 10.7554/elife.59716.

[10] Felix Jung. "DMC-BrainMap - An Open-Source, End-to-End Tool for Multi-Feature Brain Mapping Across Species". In: (2025). DOI: 10.1101/2025.02.19.639009.

[11] Daniel N. Hill, Samar B. Mehta, and David Kleinfeld. "Quality metrics to accompany spike sorting of extracellular signals". eng. In: *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 31.24 (June 2011), pp. 8699–8705. ISSN: 1529-2401. DOI: 10.1523/JNEUROSCI.0971-11.2011.

[12] Xiaoxuan Jia et al. "High-Density Extracellular Probes Reveal Dendritic Backpropagation and Facilitate Neuron Classification". In: *Journal of Neurophysiology* (2019). DOI: 10.1152/jn.00680.2018.

[13] Alessio Paolo Buccino et al. "A Deep Learning Approach for the Classification of Neuronal Cell Types". In: (2018). DOI: 10.1109/embc.2018.8512498.

[14] Julie M.J. Fabre et al. *Bombcell: automated curation and cell classification of spike-sorted electrophysiology data*. Version 1.0.0. July 2023. DOI: 10.5281/zenodo.8172822. URL: https://doi.org/10.5281/zenodo.8172822.

[15] James J. Jun et al. "Fully integrated silicon probes for high-density recording of neural activity". en. In: *Nature* 551.7679 (Nov. 2017). Number: 7679 Publisher: Nature Publishing Group, pp. 232–236. ISSN: 1476-4687. DOI: 10.1038/nature24636. URL: https://www.nature.com/articles/nature24636 (visited on 12/21/2022).

[16] Marius Pachitariu, Nicholas A. Steinmetz, and Jon Colonell. *Kilosort 2.5*. https://github.com/MouseLand/Kilosort. 2021.

[17] International Brain Laboratory et al. *Reproducibility of in vivo electrophysiological measurements in mice*. en. Pages: 2022.05.09.491042 Section: New Results. Dec. 2024. DOI: 10.1101/2022.05.09.491042. URL: https://www.biorxiv.org/content/10.1101/2022.05.09.491042v7 (visited on 02/19/2025).

[18] Eduardo Torre et al. "Synchronous Spike Patterns in Macaque Motor Cortex during an Instructed-Delay Reach-to-Grasp Task". In: *Journal of Neuroscience* 36.32 (Aug. 2016), pp. 8329–8340. DOI: 10.1523/JNEUROSCI.4375-15.2016.

[19] International Brain Laboratory. "Spike sorting pipeline for the International Brain Laboratory". In: (May 2022). DOI: 10.6084/m9.figshare.19705522.v4. URL: https://figshare.com/articles/online_resource/Spike_sorting_pipeline_for_the_International_Brain_Laboratory/19705522.

[20] Andrew P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern Recognition* 30.7 (1997), pp. 1145–1159. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(96)00142-2. URL: https://www.sciencedirect.com/science/article/pii/S0031320396001422.

[21] N. Hollmann, S. Müller, L. Purucker, et al. "Accurate predictions on small data with a tabular foundation model". In: *Nature* 637 (2025), pp. 319–326. DOI: 10.1038/s41586-024-08328-6. URL: https://doi.org/10.1038/s41586-024-08328-6.

[22] Xing Chen et al. "1024-channel electrophysiological recordings in macaque V1 and V4 during resting state". en. In: *Scientific Data* 9.1 (Mar. 2022). Publisher: Nature Publishing Group, p. 77. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01180-1. URL: https://www.nature.com/articles/s41597-022-01180-1 (visited on 03/25/2025).

[23] Johannes Niediek et al. "Reliable Analysis of Single-Unit Recordings from the Human Brain under Noisy Conditions: Tracking Neurons over Hours". en. In: *PLOS ONE* 11.12 (Dec. 2016). Publisher: Public Library of Science, e0166598. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0166598. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0166598 (visited on 03/24/2025).

[24] F. Wood et al. "On the variability of manual spike sorting". In: *IEEE Transactions on Biomedical Engineering* 51.6 (2004), pp. 912–918. DOI: 10.1109/TBME.2004.826677.

[25] Jeremy Magland et al. "SpikeForest, reproducible web-facing ground-truth validation of automated neural spike sorters". In: *eLife* 9 (2020), e55167. DOI: 10.7554/eLife.55167. URL: https://doi.org/10.7554/eLife.55167.

[26] Anoushka Jain. *Curated dataset*. en. Jan. 2025. DOI: 10.6084/m9.figshare.28282799.v1. URL: https://figshare.com/articles/dataset/Curated_dataset/28282799/1 (visited on 03/25/2025).

[27] International Brain Laboratory et al. *A Brain-Wide Map of Neural Activity during Complex Behaviour*. en. Pages: 2023.07.04.547681 Section: New Results. Dec. 2024. DOI: 10.1101/2023.07.04.547681. URL: https://www.biorxiv.org/content/10.1101/2023.07.04.547681v4 (visited on 03/25/2025).

[28] Elisabeta Balla et al. *Broadband visual stimuli improve neuronal representation and sensory perception*. en. Pages: 2023.05.03.539238 Section: New Results. May 2023. DOI: 10.1101/2023.05.03.539238. URL: https://www.biorxiv.org/content/10.1101/2023.05.03.539238v1 (visited on 05/04/2023).

[29] Marius Pachitariu et al. "Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels". In: *bioRxiv* (2016). DOI: 10.1101/061481. URL: https://doi.org/10.1101/061481.

[30] Victor Llobet, Aurélien Wyngaard, and Boris Barbour. "Automatic post-processing and merging of multiple spike-sorting analyses with Lussac". In: *bioRxiv* (2022), pp. 2022–02. DOI: https://doi.org/10.1101/2022.02.08.479192.

[31] Kush Banga et al. "Spike sorting pipeline for the International Brain Laboratory". In: (2022).

[32] Christophe Pouzat, Ofer Mazor, and Gilles Laurent. "Using noise signature to optimize spike-sorting and to assess neuronal classification quality". In: *Journal of neuroscience methods* 122.1 (2002), pp. 43–57. DOI: https://doi.org/10.1016/S0165-0270(02)00276-5.

[33] Jason E Chung et al. "A fully automated approach to spike sorting". In: *Neuron* 95.6 (2017), pp. 1381–1394. DOI: https://doi.org/10.1016/j.neuron.2017.08.030.

[34] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65. DOI: https://doi.org/10.1016/0377-0427(87)90125-7.

[35] Neil Schmitzer-Torbert and A David Redish. "Neuronal activity in the rodent dorsal striatum in sequential navigation: separation of spatial and reward responses on the multiple T task". In: *Journal of neurophysiology* 91.5 (2004), pp. 2259–2272. DOI: 10.1152/jn.00687.2003.