DeepSpike: Foundation Model-based Pipeline for Large-Scale Spike Sorting of Neural Activity

Anonymous authors
Paper under double-blind review

Abstract

Spike sorting of high-resolution neural recordings is essential for understanding brain activity, but it remains challenging when multiple units are recorded due to their overlapping spike timing, low signal-to-noise ratios and overlapping clusters. Here, we introduce DeepSpike, a self-supervised deep learning model that automates spike sorting and overcomes key limitations of conventional spike sorting methods. Pretrained on large-scale unlabeled spiking events as a reusable self-supervised encoder, it generalizes to new recordings without retraining. DeepSpike uses a self-supervised autoencoder to learn robust low-dimensional spike embeddings that facilitate accurate clustering and effective noise filtering. The model is trained on a new, large-scale dataset consisting of $255M$ spiking events (SpikeVault-255M) derived from real *in vivo* recordings of about 4560 minutes duration. The dataset consists of $15M$ ground truth spikes that are manually verified by an expert user. DeepSpike outperformed state-of-the-art spike sorting algorithms in both accuracy and robustness in our experiments on SpikeVault-255M, and two public benchmark datasets. Our results demonstrate that large-scale, self-supervised pre-training yields a powerful and generalizable encoder for automated spike sorting. The Spike Vault-255M dataset and the pre-trained DeepSpike model are made publicly available to facilitate further research and development. [1]

1 Introduction

Single-unit extracellular recordings are indispensable for linking the activity of individual neurons to behavior and for deciphering neural circuit dynamics (Gerstein & Clark, 1964). The process of detecting action potentials from collected raw data and assigning them to individual neurons – a process known as spike sorting – is the first critical step in neuronal data analysis, but it remains a significant challenge (Abeles & Goldstein, 1977; Atiya, 1992). Modern high-density multi-electrode probes can record neuronal signals at high sampling rates (e.g. $> 25kHz$) across hundreds of channels, rapidly producing massive data streams that demand automated and precise analysis (Wei et al., 2020). Despite the emergence of numerous semi- and fully-automated spike sorting algorithms in recent years (Buccino et al., 2022), classification accuracy of detected spikes is still a mounting challenge (Eom et al., 2021).

Spike sorting remains challenging due to several factors: (1) spike waveforms from different neurons are highly diverse with different space and time signatures; (2) the number of neurons present in the recording is unknown a priori; and (3) low signal-to-noise ratios between spikes and background signal (Wang et al., 2019). Additionally, signal drift during long-term recordings can alter waveform amplitude and shape, rendering static threshold-based detection ineffective.

In light of these challenges, we propose DeepSpike, a self-supervised autoencoder based spike sorting framework. DeepSpike follows a multi-stage pipeline to perform spike sorting, as shown in Figure 1, comprising four key steps. The pipeline consists of: (1) Data preprocessing, where the raw signals are filtered and normalized to extract candidate events; (2) Multi-threshold spike detection, an iterative strategy that captures spikes of varying amplitudes at progressively low signal-to-noise ratio (SNR); (3) Feature extraction via a

[1] Source code, data, models are available at: https://anonymous.4open.science/r/DeepSpike-DC19

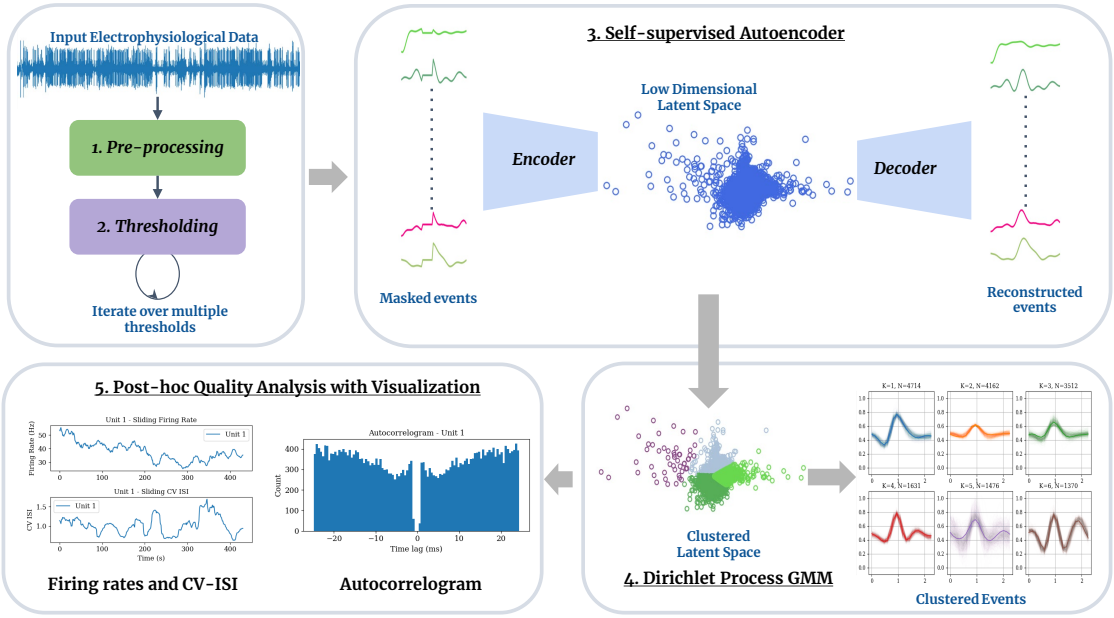

Figure 1: Overview of the DeepSpike pipeline. Raw multi-unit signals are preprocessed and run through an iterative multi-threshold detector to extract spike waveforms. An auto-encoder based foundation model then embeds each event into a low-dimensional latent vector, and finally Dirichlet Process Gaussian Mixture Model (DPGMM) based clustering is performed on the latent vectors such that each cluster corresponds to one putative neuron.

self-supervised autoencoder, which embeds each event waveform into a low-dimensional latent vector; and (4) Clustering, where the low-dimensional spike embeddings are grouped such that each cluster corresponds to a single putative neuron.[2] Each component builds on the previous one — for example, the enhanced spike detection feeds cleaner inputs to the autoencoder, and the learned embeddings enable more accurate clustering.

The key contributions of this work are:

1. A comprehensive spike sorting pipeline with a novel self-supervised foundation model at its core.
2. Use of non-parametric Bayesian clustering method that does not need number of clusters a prioi.
3. New large-scale dataset (SpikeVault-255M) comprising $255M$ spiking events with $15M$ manually verified ground truth spikes, derived from 95 *in vivo* recordings of 4560 minute duration.
4. Thorough evaluation of multiple baselines on SpikeVault-255M and two public datasets, with both quantitative and qualitative performance reporting.

2 Related Work

Traditional Spike Sorting Methods. Traditional spike sorting typically follows a fixed pipeline of filtering, threshold-based detection, feature extraction, and clustering (Hennig et al., 2019; Rey et al., 2015b; Vesanto & Alhoniemi, 2000; Adamos et al., 2008; Nguyen et al., 2015; Keshtkaran & Yang, 2017; Zhang et al., 2022). These methods rely on hand-crafted features (e.g., principal component analysis (PCA), wavelets) and require manual tuning of parameters such as detection thresholds and expected unit counts.

[2] A putative neuron is a cell presumed to be a neuron based on indirect or preliminary evidence, but has not yet been definitively identified as one through rigorous methods.

Recent high-performance open-source systems such as *Kilosort* (Pachitariu et al.), *MountainSort* (Chung et al., 2017), and *HerdingSpikes2* (Hilgen et al., 2017) introduce graphics processing unit (GPU) acceleration and more sophisticated clustering schemes, greatly improving throughput. However, these tools still struggle with high-noise conditions, overlapping spikes, and electrode drift (Buccino et al., 2022; Guenther et al., 2009; Lewicki, 1998; Harris et al., 2000). As a result, they often require manual curation and fail to generalize across datasets. Benchmark studies such as *SpikeForest* (Magland et al., 2020) highlight that no single method performs consistently well across varied recording conditions, further emphasizing the need for more general and robust approaches in spike sorting.

Deep Learning Approaches. Deep learning has emerged as a promising direction for spike sorting, aiming to reduce dependence on manual heuristics and improve robustness.

Some works target the detection stage: YASS (Lee et al., 2017) uses convolutional neural networks (CNNs) to directly identify spikes from raw signals, reducing noise and data volume. Boussard et al. (2021) integrate neural networks with physical models to estimate 3D spike locations, enabling long-term unit tracking and drift correction on Neuropixels probes (Jun et al., 2017).

Another major direction is unsupervised feature learning. Wu et al. (2018) employed autoencoders to compress spike waveforms into low-dimensional embeddings, replacing hand-engineered features. Subsequent work explored deep compressive autoencoders for denoising and reconstruction. Eom et al. (2021) combined multiple autoencoders for multi-channel input, improving robustness, while Rokai et al. (2021) proposed a variational autoencoder to learn probabilistic spike embeddings for clustering under uncertainty. Methods like Speiser et al. (2017) have used similar autoencoder based methods for spike sorting of calcium imaging data.

Other efforts focus on real-time spike sorting. Radmanesh et al. (2022) designed an online framework combining denoising autoencoders with adaptive clustering using the Gap Statistic, achieving near-offline accuracy with low latency.

Self-Supervised Representation Learning for Spike Sorting. More recently, self-supervised learning (SSL) has emerged as a powerful paradigm for learning from unlabeled electrophysiological data. Several approaches have utilized contrastive learning frameworks, where models learn to distinguish between similar and dissimilar spike waveforms. For example, CEED (Vishnubhotla et al., 2023) and HuiduRep (Cao & Feng, 2025) learn discriminative features by pulling representations of augmented spike waveforms closer while pushing others apart. PseudoSorter (Brockhoff et al., 2025) also leverages SSL to reveal activity changes in pathological models. While these contrastive methods are effective for learning discriminative features, our work explores a generative, reconstruction-based objective. We hypothesize that a masked autoencoder (MAE) approach is theoretically better suited for learning to denoise and reconstruct canonical spike waveforms—a key requirement for robust sorting in low-SNR conditions. DeepSpike is the first work to integrate this generative SSL strategy into a complete, end-to-end pipeline with automated non-parametric clustering and validate it at an unprecedented scale on the new Spike Vault-255M dataset.

We build on this body of work and address the challenges with generalization, label scarcity, and robustness under noise and data drift with our work. We bring insights from self-supervision and foundation models into spike sorting, and show promising results.

3 Methods

Consider a neural recording of duration T (s) sampled at frequency f (kHz) which is then divided into a collection of N events: $= \{_i\}_{i=1}^{N}$, each of duration t (ms). Each event consists of F samples i.e. $_i \in^F$ such that $F = f \cdot T/N$. The number of samples per event is chosen to correspond to feasible spike duration, which can range between $t = 1 - 4$ ms. Given these individual events, we are interested in performing spike sorting that is able to assign cluster identities to spikes that belong to the same putative neuron. This can be formulated as the task of learning the function $f_\theta :\in^F \rightarrow y \in \{0, 1, \cdots, K\}$, where the cluster $K = 0$ always corresponds to the noise cluster, and θ are the trainable model parameters.

3.1 Data Preprocessing

The raw electrophysiological signals undergo a series of preprocessing steps to enhance signal quality and to standardize the event representations. The key steps are:
1. **Bandpass Filtering:** : A 300–5000 Hz bandpass filter removes slow drifts and high-frequency noise, retaining the spectral content of action potentials, while subtracting a local moving average further centers the signal by addressing any residual baseline variations not fully eliminated by the bandpass filter.
2. Nonlinear Denoising: We estimate background noise via the median of extreme values from randomly sampled segments. Transients exceeding a threshold (e.g., large-amplitude artefacts) are clipped, suppressing spurious fluctuations unlikely to be genuine spikes.
3. Normalization: The filtered and denoised signal is normalized (e.g., scaled to [0,1]) to enable consistent thresholding across recordings, independent of electrode gain or impedance. A sample recording before and after preprocessing is shown in Fig. 2.
4. Event Detection (Initial Spike Extraction): We then perform an initial spike detection by thresholding the normalized signal. Any timepoint where the signal crosses a chosen threshold (e.g. 0.5 in normalized units) is flagged as a spike event . For each such event, we extract a fixed-length snippet of the waveform around the threshold-crossing point -– for example, 8 samples before the threshold crossing and 12 samples after, yielding a snippet of length $F = 20$ samples which would correspond to a spiking event of about $1ms$ if the signal is sampled at $25kHz$. By aligning these snippets on the spike peak, we obtain a set of candidate spike waveforms of equal length. Formally, the collection of initially detected spike waveforms $= \{_1, _2, \ldots, _N\}$ with each $_i \in \mathbb{R}^F$, may include false positives (noise waveforms) if the threshold was low, or it may have missed some small spikes if the threshold was high. We address this next with an iterative multi-threshold approach. To standardize inputs across datasets with varying sampling frequencies, all extracted waveforms are interpolated to a fixed length of $F = 20$ samples, enabling the pretrained model to process diverse recordings without modification.

Fixed-length windows are allowed to overlap when spikes occur in close succession; we do not perform explicit deconvolution of overlaps. Such cases enter the pipeline as separate windows if they pass thresholding and can later be separated by the embedding and clustering stages when their morphologies differ.

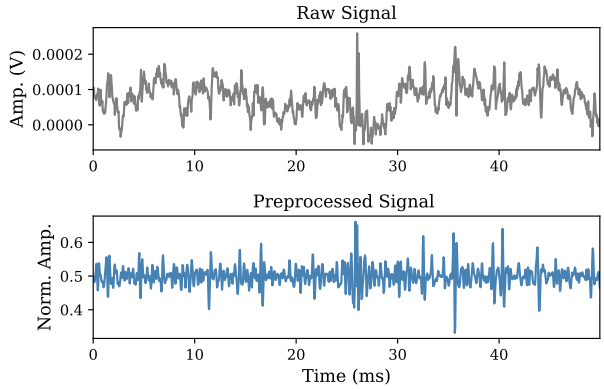

Figure 2: Raw signal (top row) compared to the preprocessed signal (bottom row) which shows the outlier removal and non-linear denoising.

3.2 Multi-Threshold Spike Detection

Relying on a single threshold for spike detection entails a trade-off between sensitivity and specificity: high thresholds reliably detect large-amplitude spikes but miss smaller ones, while low thresholds increase sensitivity at the cost of more false positives. Existing spike sorting methods strive to obtain a single, global threshold. This can have negative effects towards the specificity and/or sensitivity of the spike sorting methods.

To address this, in *DeepSpike* pipeline, we adopt an iterative multi-threshold strategy, where spike detection is performed multiple times using thresholds of decreasing magnitude. The results are then merged, allowing robust recovery of spikes across a wide amplitude range while suppressing spurious events.

Rather than using arbitrary values, thresholds are derived from the statistical properties of the signal. For each recording $_k$, a threshold τ_k is computed as: $\tau_k = \text{median}(_k) + \psi \cdot \text{std}(_k)$, where $_k$ denotes the preprocessed signal and ψ is a tunable scalar controlling the trade-off between false positives and false negatives. Alternatively, following Quiroga et al. (2004), the threshold can be defined as $\tau_k = 4\sigma_n$, where $\sigma_n = \frac{\text{median}(|_k|)}{0.6746}$, with the constant 0.6746 corresponding to the inverse cumulative distribution value of a

standard normal variable. This approach enables consistent and data-driven thresholding, improving the reliability of downstream feature extraction (Quiroga, 2007).

We perform spike detection iteratively across multiple threshold levels and refine the set of detected events at each iteration. The pseudocode for the multi-threshold spike detection algorithm is outlined in Algorithm **??** (in the Appendix).

3.3 Spike Representation Learning with Self-Supervised Autoencoder

Spike waveforms differ in their spatial and temporal signatures. Conventional spike sorters try to model the templates of unique spikes and perform template matching to then identify spikes fired from the same putative neuron. This can work in conditions with unique neurons, and when the SNR is high. In large-scale recordings with drifts, modelling all possible spike templates becomes infeasible, and as a result conventional sorters can fail.

With autoencoder-based spike sorters, the model attempts to learn *all* possible spike waveforms from the data. Learning these template representations instead of designing templates offers a powerful way to perform spike sorting (Eom et al., 2021; Rokai et al., 2021; Radmanesh et al., 2022).

Instead of using unsupervised autoencoders, in DeepSpike, we use a masked autoencoder to learn the latent representations of spike waveforms from a large-scale unlabelled dataset (He et al., 2022). The input to the encoder are masked spiking event data obtained using a random masking function $m(\cdot)$ which zeros out the event data in certain continuous regions. The encoder embeds this masked input into a low-dimensional latent space $f_\theta : m() \in^F \rightarrow \in^L$ where $L < F$ is the size of the latent dimension . The decoder g_ϕ is then tasked with reconstructing the complete spike event from the low-dimensional latent codes i.e. $g_\phi : \in^L \rightarrow \hat{} \in^F$. The encoder-decoder networks are trained jointly by minimizing the reconstruction loss $_{rec}(,\hat{})$.

This is a widely used strategy in self-supervised training of foundation models (Bommasani et al., 2021; Balestriero et al., 2023). The hypothesis with masked autoencoding is that by attempting to reconstruct complete spike events from masked inputs, the latent space learns to represent spikes and is able to distinguish noise events from true spikes, as spikes have more structure to them than noise events. We train DeepSpike in this masked autoencoder fashion using a large-scale dataset ((10^6) in our case).

In addition to the large-scale self-supervised training, we also regularize the latent space using a regularization term based on variational autoencoders (VAE) (Kingma et al., 2013). This forces the latent encoder distribution predicted by the encoder model $q(|;\theta)$ to align with a spherical Gaussian $p() = (0, 1)$ using Kullback-Leibler divergence (KLD). The regularization term is $_{reg} = KLD(q(|;\theta), p())$. The KL regularization enforces a smooth Gaussian latent space, expressing uncertainty and supporting generalization to unseen waveforms, consistent with robust performance across noise levels.

After training, during the inference time, we use the posterior mean obtained from the encoder neural network as the spike embedding i.e. $= f_\theta()$. Given a set of preprocessed events, , DeepSpike is used to obtain the low-dimensional representations, which are further used to perform clustering and neural unit identification i.e., $= \{f_\theta(_i), \forall_i \in\}$.

3.4 Bayesian Non-Parametric Clustering and Putative Neuron Identification

Given the latent embeddings from DeepSpike, , we are now interested in detecting clusters of events that belong to the same firing unit or the putative neuron. Under standard conditions, the firing pattern of each unit is unique and all the spike events that have a similar waveform should be marked as belonging to the same firing unit. This is the essential step in spike sorting – to attribute detected spikes to the right firing units. Formally, the objective of the clustering step is to assign a discrete neuronal firing unit label $y \in \{0, 1, \cdots, K\}$ to every latent code while remaining agnostic about the total number of neuronal firing units (K). This is performed in our pipeline using two levels of clustering: 1) Using simple binary clustering when iterating at a specific threshold to remove noise events, 2) Using DPGMM clustering to detect unique neuronal firing units.

Iterative Noise Filtering via Binary Clustering When we perform the multi-threshold detection, we first employ binary clustering. At each threshold level, we embed candidate events and apply k-means with $K = 2$ as a coarse split. We then compute template waveforms per cluster and re-assign events by template correlation: the cluster with higher within-cluster correlation is labeled "spike," and the other "noise." This two-step gate reduces false merges, because noise events lack a consistent template and exhibit low correlation even if a centroid-based split groups them together. The cluster with higher correlation are labelled as spikes, and the remaining are labelled as noise. In the next iteration when the threshold is lowered, all the non-spike events go through the same process: embed; cluster; label, until we reach the lowest SNR. At the end of this step, each event is either labelled has a binary label corresponding to if it is a spike or not.

Algorithm 1 Iterative Multi-Threshold Spike Detection and Filtering

Input: Preprocessed signal $\mathcal{D}$, Threshold levels $\{\tau_1, \ldots, \tau_M\}$ where $\tau_1 > \cdots > \tau_M$, Pre-trained encoder f_θ
Initialize: Set of accepted spikes $X_{\text{spikes}} \leftarrow \emptyset$, Set of candidate events $X_{\text{cand}} \leftarrow \mathcal{D}$
for each threshold τ_i in $\{\tau_1, \ldots, \tau_M\}$ **do**
  Detect all events in X_{cand} where signal exceeds τ_i. Let this set be X_{new}.
  if X_{new} is not empty **then**
    Embed all events in X_{new} into latent space: $Z_{\text{new}} = \{f_\theta(x) \forall x \in X_{\text{new}}\}$
    Perform binary clustering on Z_{new} (e.g., K-Means with $K = 2$) to get clusters C_1, C_2.
    Compute template waveforms T_1, T_2 for each cluster.
    Calculate within-cluster correlation scores $\text{Corr}_1, \text{Corr}_2$.
    if $\text{Corr}_1 > \text{Corr}_2$ **then**
      $X_{\text{accepted_at_i}} \leftarrow C_1$, $X_{\text{rejected_at_i}} \leftarrow C_2$
    else
      $X_{\text{accepted_at_i}} \leftarrow C_2$, $X_{\text{rejected_at_i}} \leftarrow C_1$
    end if
    $X_{\text{spikes}} \leftarrow X_{\text{spikes}} \cup X_{\text{accepted_at_i}}$
    $X_{\text{cand}} \leftarrow X_{\text{rejected_at_i}}$ {Rejected events become candidates for the next, lower threshold}
  end if
end for
Post-process X_{spikes} to merge duplicates.
Output: Final consolidated spike set X_{spikes}

Dirichlet Process Gaussian Mixture Model The binary discrimination step above removes obvious noise events and false detections. We then assign unique cluster identities to the remaining putative spikes, where the number of clusters K corresponds to the number of putative single units and is unknown *a priori* in extracellular recordings (Gasthaus et al., 2008b; Rey et al., 2015a; Lewicki, 1998). Conventional partitional clustering methods such as K-means, as well as fixed-K Gaussian mixture models, require specifying the partition level (or selecting K) and can be sensitive to overlapping and non-Gaussian cluster structure, leading to under-clustering (merging distinct units) or over-clustering (splitting one unit into multiple clusters) (Veerabhadrappa et al., 2020; Rey et al., 2015a; Ardelean et al., 2023). Gaussian-mixture assumptions in particular may overcluster by fitting multiple Gaussians to a single non-Gaussian cluster (Rey et al., 2015a), motivating nonparametric alternatives that infer the effective number of units in an unsupervised manner (Wood & Black, 2008; Gasthaus et al., 2008b).

To address these challenges we employ a Dirichlet Process Gaussian Mixture Model (DPGMM) which determines the optimal number of clusters data-dependent which has shown to be useful in spike sorting settings in (Gasthaus et al., 2008a). DPGMM removes the need to guess K and can capture subtle differences between spike clusters by flexibly allocating new clusters if required. Moreover, the Bayesian nature that is inherent to DPGMM allows incorporating prior knowledge (e.g. expected range of spike waveform variability) and yields a principled way to deal with the uncertainty in cluster assignments. This is particularly advantageous for handling artifacts like overlapping spikes. Such events, which produce atypical waveforms, are assigned to new, low-weight singleton clusters rather than being forced into and corrupting existing unit clusters, thus

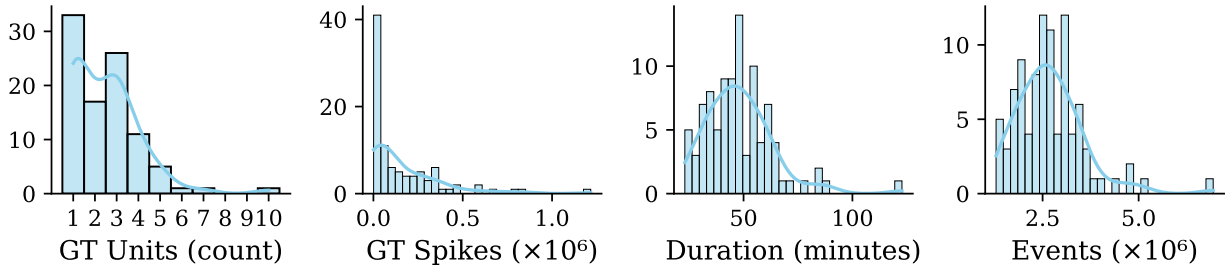

Figure 3: SpikeVault-255M Dataset Distributions

preserving the integrity of the final sort. We present a comprehensive study of different clustering methods in Sec.C, and fix DPGMM to be the default clustering scheme for DeepSpike.

4 Data and Experiments

Dataset with *in vivo* recordings: One of the contributions of our work is a new, large-scale neural recordings dataset that is made public. This dataset is obtained from *in vivo* recordings in the brainstem of `VGlut2-ChR2` transgenic mice (Hägglund et al., 2010). The experiments were approved by and performed in accordance with local regulatory authorities. The dataset is derived from single-channel recordings; the electrode consisted of a single channel coupled to an optical fiber. The construct was stereotaxically implanted into the pedunculopontine nucleus in the mouse brainstem. The recordings were performed in fully conscious mice in tethered conditions. The wide-band signal was recorded using Tucker-Davids RDZ5 preamplifier and a RD5 data-acquisitioner controlled by the Tucker-Davids synapse software at $25kHz$.

Using this set-up a total of 95 recordings with about 4560 minutes of data are used in this work. The recordings were manually spike sorted after the wide-band data was imported into Spike2[3] (v.7.20). The recordings were high pass filtered ($750Hz$) to remove the slow oscillatory activity. The spike events were isolated by setting a positive threshold. All events crossing the threshold were subjected to a principle component analysis using the inbuilt function in Spike2 and were visualized in a 3D space, using the 3 vectors with the highest weight. Spikes belonging to a single unit were grouped in the 3D space and noise was excluded. Finally, the spike time-stamp and cluster number was exported to a text file format and made available to be used as the ground truth. This ground truth data-set was then used for validating the performance of spike-sorting methods in this work which is also provided as part of the dataset.

Table 1: SpikeVault-255M Statistics

Sampling frequency	25 kHz
Total duration	4558.18 min.
Num. of events	255.15 M
Num. of GT spikes	15.19 M
Num. of GT units	237

The 95 recordings result in about $255M$ events after the preprocessing and event detection steps described in Sec. 3.1. These preprocessed events from the neural recordings will be referred to as the **SpikeVault-255M** dataset. The manual curation of this dataset yields about $15M$ spikes in the ground truth from 237 neural units. See the dataset statistics summarized in Table 1 and Figure 3.

Synthetic datasets: In addition to the SpikeVault-255M dataset we evaluate DeepSpike on two widely used synthetic datasets. The first is the WaveClus dataset (Quiroga et al., 2004), comprising 16 one-minute simulated recordings, each containing three spike classes and additive noise at four levels (0.05˘0.20), defined as the ratio of noise to spike amplitude. The data are labeled with ground-truth spike times and widely used for benchmarking spike sorting methods.

The second dataset, introduced by Pedreira et al. (2012), contains 95 simulated 10-minute recordings based on real primate neocortical data. Each recording includes between 2 and 20 neuron types, selected from a

[3] https://ced.co.uk/products/spike2

pool of 594 spike shapes, plus a noise cluster. Each spike is labeled, enabling external evaluation metrics. The dataset is designed to mimic realistic waveform variability and sorting complexity, with 5 simulations per unit count.

Experimental Set-up: We conduct multiple experiments to assess the performance of the proposed DeepSpike method. The key goal of the method is to perform reliable spike sorting; in order to evaluate this, we conduct experiments on the SpikeVault-255M dataset, and the two synthetic datasets. Both datasets have ground truth labels which are used for quantifying the performance. We evaluate DeepSpike in two modes: zero-shot setting, and with unsupervised fine-tuning. These experiments demonstrate the foundation model capabilities of DeepSpike.

Baseline Methods: We use SpikeInterface to run the baseline methods which has integrated several recent spike-sorting algorithms (Buccino et al., 2020) in a standardized manner. Among the methods supported in SpikeForest, we were able to reliably run four methods on SpikeVault-255M dataset: Kilosort4 (Pachitariu et al., 2024), MountainSort5 (Chung et al., 2017), SpykingCircus2 (Yger et al., 2018) and TridesClous (Pouzat & Garcia, 2016).

In addition to these methods, we were also able to compare with HerdingSpikes2 (Muthmann et al., 2015), Klusta (Rossant et al., 2016), and Waveclus (Chaure et al., 2018) for the two synthetic datasets. All baselines were run through SpikeInterface using recommended or default parameters. No baseline received per-recording calibration based on ground truth. Unless otherwise stated, evaluation follows unit-level matching with a fixed temporal tolerance (see Appendix C for the exact matching rule and tolerance), as implemented in SpikeInterface.

Metrics: We use the comparison module in SpikeInterface which provides standard metrics to compare to ground truth labels. These metrics are accuracy, precision, and recall. We refer to Sec. A for details on these metrics which are derived from the descriptions in SpikeInterface in (Buccino et al., 2020).

DeepSpike Model Training on SpikeVault: We want DeepSpike to learn intrinsic representations of diverse spike waveforms. As discussed in Sec. 3.3, we use a masked autoencoder approach to train DeepSpike. Instead of training on all the $255M$ events which might consist majority of noise events, we extract events at a higher SNR which should yield high quality spike events. We hypothesize that when the model is primarily trained on high quality spike waveforms, it can learn better representations of spike waveforms. As the model is only trained on possible spikes, when tested on other larger data with majority of noise events, it acts as an anomaly detection method (Sakurada & Yairi, 2014).

To achieve this training, we query all the events in SpikeVault-255M dataset that have a maximum intensity at $\tau \geq 0.7$.[4] This results in a subset of about $5M$ events which we denote as **SpikeVault-5M**. We train DeepSpike on SpikeVault-5M using input masking, and use the encoder f_θ from the converged model to obtain the low-dimensional representations used for clustering.

DeepSpike pretrained on SpikeVault-5M is also used in zero-shot setting on the synthetic datasets. Additional hyperparameters that describe the network architecture and train DeepSpike are reported in Sec. B.

5 Results and Discussions

Performance on SpikeVault-255M and SpikeVault-140M: The SpikeVault-255M dataset presented a particularly challenging test case due to its size, extended recording durations, and variable unit counts. Out of the 95 recordings in SpikeVault-255M, none of the four baseline methods were able to process all the recordings without failing or returning zero accuracy. To perform a fair comparison across the sorters, we curate the recordings that overlap between all the baseline sorters which yields 50 recordings (out of the 95), and the corresponding event dataset has about 140M events. We will refer to this as **SpikeVault-140M** dataset in the discussions. Statistics for this curation are provided in Table 2. DeepSpike was the only algorithm that successfully processed all 95 recordings without failures or zero-accuracy results.

[4]Recollect that the individual events $_i \in$ are amplitude normalized to be in $[0, 1]$.

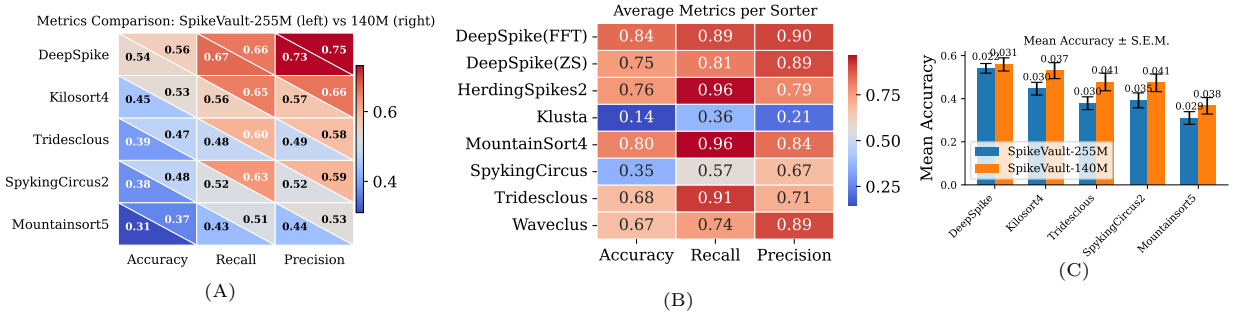

Figure 4: (A) Metrics (accuracy, recall, precision) on SpikeVault-255M (left) averaged over all recordings that each sorter processed successfully, including zero-accuracy cases; and on SpikeVault-140M (right), computed on the intersection of recordings processed by all sorters. (B) Metrics on the merged public datasets; FFT denotes fully fine-tuned and ZS denotes zero-shot DeepSpike. (C) Sorter comparison of mean accuracy, recall, and precision. blue: metrics averaged over all recordings that each sorter processed successfully on SpikeVault-255M (including zero-accuracy cases). yellow: the same metrics computed on the intersection of recordings successfully processed by all sorters (50 recordings; SpikeVault-140M). Error bars denote ± s.e.m.

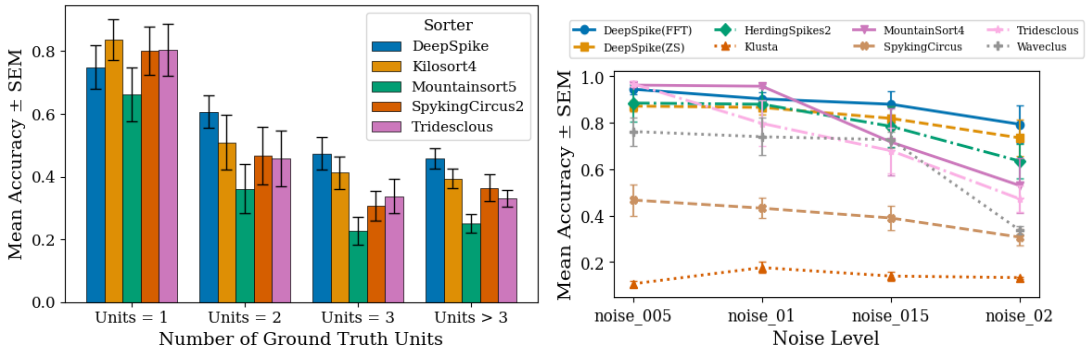

Figure 5: Robustness comparison across unit numbers in SpikeVault-255M and noise levels in the Quiroga dataset for each sorter; error bars denote ± s.e.m.

Table 2: Counts across 95 VGlut2 recordings where different sorters either failed to process or returned zero accuracy after processing. DeepSpike processed all 95 recordings successfully.

Sorters	DeepSpike	Kilosort4	Mountainsort5	SpykingCircus2	Tridesclous
Failed to process	0	6	0	1	10
Processed with zero acc.	0	11	18	11	17
Total	0	17	18	12	27

Figure 4-(A) quantifies this performance difference, showing that DeepSpike achieved the highest overall accuracy, recall, and precision across all recordings . Notably, its recall was 0.67 and precision was 0.73, compared to 0.56 and 0.57, respectively, for the next best method (Kilosort4). Similar large margins are also observed over other algorithms.

Qualitative results on one of the recordings from SpikeVault are shown in Fig. 6. Here again we observe that DeepSpike not only detects all units in the ground truth, but also matches the ground truth template waveforms with high fidelity. Baseline methods, however, miss some units and have average templates that do not match well with the ground truth, suggesting they may be mixing spikes from different neurons or including noise.

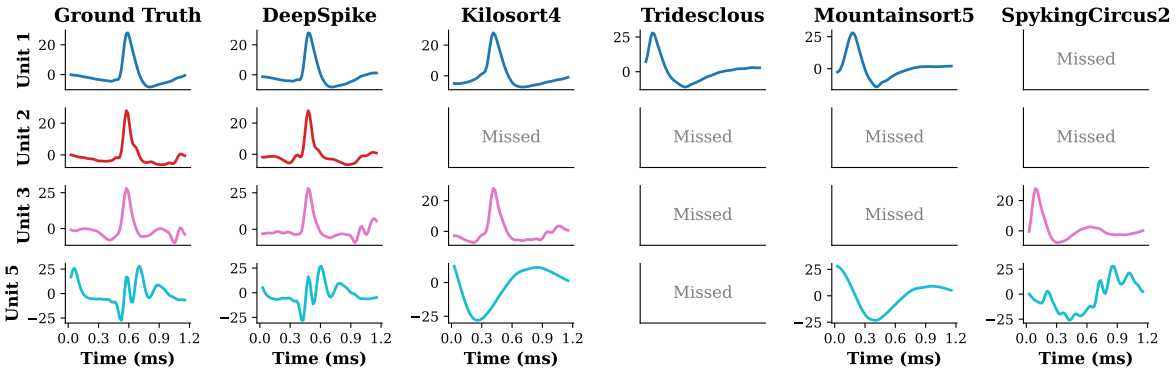

Figure 6: Average template waveforms of the ground truth units, compared with different sorters for one of the recordings in SpikeVault. We note that DeepSpike is able to detect all units and the average template is close to the ground truth template compared to other methods. These average templates were obtained from SpikeInterface based on the clustering results from all the methods.

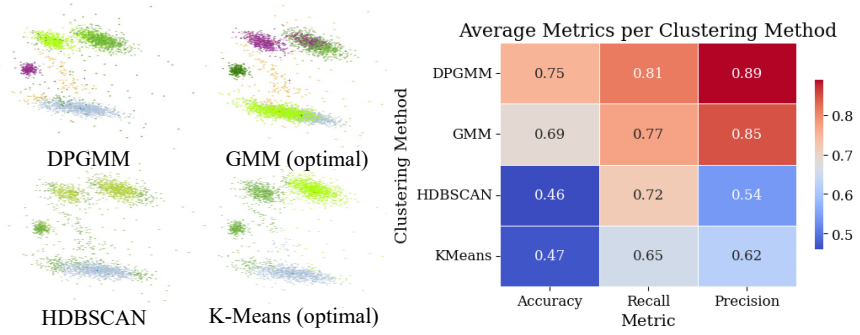

Figure 7: Influence of different clustering methods on DeepSpike.

Generalization Performance on Merged Public Benchmark Datasets: To directly test the generalization capabilities of our foundation model, we evaluated the DeepSpike model (pre-trained only on in vivo mouse brainstem data from SpikeVault-5M) on two public synthetic datasets. These evaluations included a zero-shot version ("DeepSpike (ZS)"), where the pretrained foundation model was applied directly, and a fully fine-tuned version ("DeepSpike (FFT)") which was further trained in an unsupervised manner on the synthetic datasets.

As shown in Figure 4-(B), both DeepSpike (ZS) and DeepSpike (FFT) demonstrated strong performance on the public datasets, outperforming many other established sorters. DeepSpike (FFT) generally achieved the highest accuracy, recall, and precision, indicating the benefit of fine-tuning for specific dataset characteristics. However, DeepSpike (ZS) also showed highly competitive results, underscoring the strong generalization achieved by the foundation model without any dataset-specific training on these public benchmarks. For instance, DeepSpike (FFT) achieved an accuracy of 0.84, while DeepSpike (ZS) achieved 0.75, both comparing favorably to other sorters like HerdingSpikes2 (0.76) and MountainSort4 (0.80). Notably, baselines like Waveclus rely on PCA/wavelet features; DeepSpike's superior accuracy highlights the benefits of learned, non-linear embeddings over hand-crafted ones.

Robustness to Noise: The Quiroga dataset's controlled noise levels allowed us to specifically test DeepSpike's robustness to noise. As shown in Figure 4-(C), DeepSpike maintained high performance across all noise levels: $[0.05, 0.10, 0.15, and 0.20]$, with only minimal degradation at the highest noise level. This contrasts with most other algorithms, which showed significant performance drops as noise increased. DeepSpike's resilience to noise can be attributed to its multi-threshold detection strategy and the denoising capabilities of its variational autoencoder.

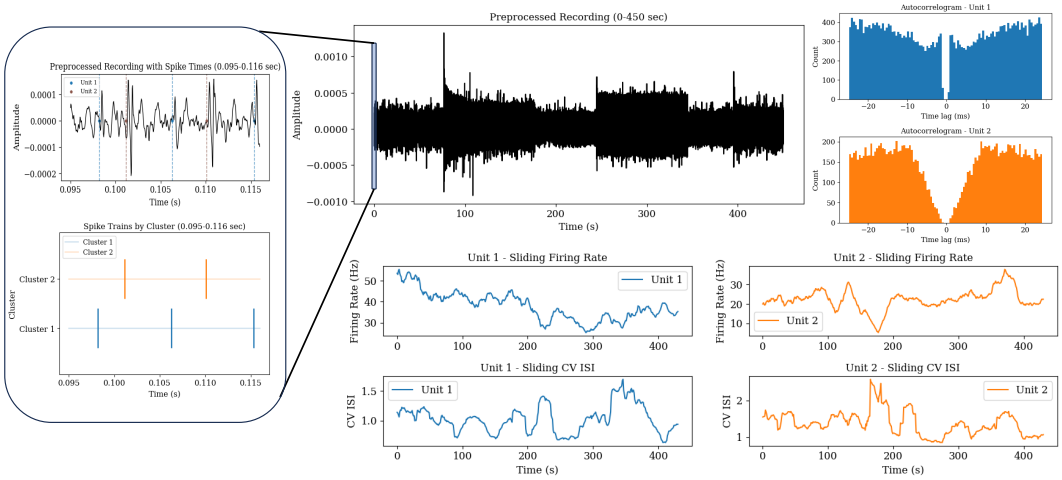

Figure 8: Illustration of activity windows used for trend analysis. Short windows: 0.02 seconds. Long windows: 450 seconds.

Robustness to Noise and Unit Count: Figure 5 summarizes robustness along two axes. We analyzed how the algorithms performed relative to the number of ground truth units in the SpikeVault-255M dataset (Figure 5, left panel). Interestingly, all algorithms showed their poorest performance with three units, with improved results for both fewer and more units. DeepSpike demonstrated the most consistent performance across different unit counts, with particularly strong results for single-unit recordings. Across the Quiroga noise levels $\{0.05, 0.10, 0.15, 0.20\}$, DeepSpike maintains strong performance with modest degradation at higher noise, whereas several baselines exhibit pronounced declines.

Clustering Methods Comparison: We also evaluated the effectiveness of different clustering methods within the DeepSpike framework on the synthetic datasets (Figure 7). DPGMM achieved the best performance across all the metrics, closely matching a standard GMM with an optimally predefined number of clusters. The key advantage of DPGMM is that it automatically determines the appropriate number of clusters without requiring manual specification or additional metrics. While HDBSCAN showed lower performance than expected and had longer runtime, it still offered valuable noise-handling capabilities in certain scenarios. Additional discussions on clustering methods, and their implications are presented in Sec. C.

Qualitative Analysis and Visualization: Spike sorting is a critical component in analyzing neural activity. After performing spike sorting with DeepSpike, we also demonstrate the further downstream analyses implemented as a comprehensive framework (Figure 8) that includes: 1) Short-time window visualization ($0.01 - 0.02$ seconds) showing preprocessed data with color-coded waveform overlays for each identified cluster and individual spike train plots for each cluster. Also, autocorrelograms calculated from shorter segments to ensure clarity, 2) Long-time window visualization ($15 - 20$ minutes) displaying temporal changes in firing rates and coefficients of variation of inter-spike intervals (CV ISIs) within a sliding window.

This framework in our pipeline allows neuroscientists to correlate spike firing patterns with animal behaviors. For instance, in the SpikeVault-255M dataset, the mice are running on treadmills at varying speeds, and the visualization enables analysis of variability in firing rates and ISIs relative to running speed, providing deeper insights into neuronal activity underlying different behavioral states.

Limitations and Future Work: Spike sorting pipelines are extremely complex requiring various steps that require several design choices. Some of this complexity is also reflected in our DeepSpike-based pipeline. We have tried to automate bulk of the analyses – starting from tuning the preprocessing thresholds, to event detection, and removing the need to specify the number of clusters. However, these settings might not always work for all recordings due to variations in acquisition conditions. This would require some tuning. We have, however, shown with our generalization performance that DeepSpike was able to yield the best results even in the zero shot setting, indicating our pipeline can be used for diverse recordings.

Modern Neuropixels probes (Jun et al., 2017) record from hundreds of channels simultaneously, and extending DeepSpike to multi-channel data is an important next step. In principle, our method can be scaled to multi-channel recordings by processing each channel's waveforms through the autoencoder and clustering independently, assuming each channel captures distinct units (an independence assumption). Alternatively, the model could be adapted to accept multi-channel waveforms as input, leveraging spatial information across electrodes (similar to Eom et al. (2021)'s approach for multi-channel autoencoders). We leave this extension to future work and hope that the core components of DeepSpike (unsupervised feature learning and DPGMM clustering) will remain applicable in the multi-channel setting, or even useful in spike sorting calcium imaging data (Speiser et al., 2017). Additionally, SpikeVault-255M is derived from mouse brainstem recordings; while zero-shot results on synthetic primate data suggest applicability to other regions and species, future extensions to cortical or pathological datasets would enhance universality.

6 Conclusion

Automated spike sorting has been an active area of research for years, driven by the need to make sense of increasingly large-scale neural recordings. While substantial progress has been made in developing algorithms for automatic analysis, the scale and complexity of modern datasets continue to pose challenges. Recent advances in foundation models offer a promising opportunity to revisit this problem, bringing new capabilities from large-scale machine learning.

To this end, we presented DeepSpike, a self-supervised autoencoder based foundation model at the core of a comprehensive spike sorting pipeline. Through comprehensive evaluations on a new large-scale dataset (SpikeVault-255M) derived from *in vivo* recordings and comparisons with multiple relevant baselines, we have shown that DeepSpike achieves state-of-the-art performance. We have also demonstrated the zero short capabilities of the model pretrained on SpikeVault on other public datasets. This is very promising as practitioners can use our public, pretrained model to perform competitive spike sorting on their in-house data. If needed, fine-tuning can also be performed to obtain better performance.

By combining deep representation learning with Bayesian non-parametric clustering, DeepSpike addresses key challenges in spike sorting (feature learning, unknown cluster count, noise robustness) without supervision. We believe this approach can scale to meet the demands of emerging large-scale neural recordings (Jun et al., 2017), paving the way for more automated and generalizable neural data analysis.

Broader Impact. The data was collected with approvals from relevant ethical boards. We do not foresee any harmful impact due to the algorithmic work itself.

Generative AI Usage Statement. ChatGPT version 5.2 and Google Gemini were used to support programming tasks, including the development of scripts for visualization and setting-up experiments, to edit and refine language and grammar in selected sections of the manuscript.

Acknowledgments. Authors thank several people.

References

Moshe Abeles and Moise H Goldstein. Multispike train analysis. *Proceedings of the IEEE*, 65(5):762–773, 1977.

Dimitrios A Adamos, Efstratios K Kosmidis, and George Theophilidis. Performance evaluation of pca-based spike sorting algorithms. *Computer methods and programs in biomedicine*, 91(3):232–244, 2008.

Eliza-Roxana Ardelean, Andrei-Marius Ichim, Mihai Dînşoreanu, and Raul C. Mureşan. Improved space breakdown method – a robust clustering technique for spike sorting. *Frontiers in Computational Neuroscience*, 17:1019637, 2023. doi: 10.3389/fncom.2023.1019637.

Amir F Atiya. Recognition of multiunit neural signals. *IEEE Transactions on Biomedical Engineering*, 39 (7):723–729, 1992.

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Julien Boussard, Erdem Varol, Hyun Dong Lee, Nishchal Dethe, and Liam Paninski. Three-dimensional spike localization and improved motion correction for neuropixels recordings. *Advances in Neural Information Processing Systems*, 34:22095–22105, 2021.

Marius Brockhoff, Jakob Träuble, Sagnik Middya, Tanja Fuchsberger, Ana Fernandez-Villegas, Amberley Stephens, Miranda Robbins, Wenyue Dai, Belquis Haider, Sulay Vora, et al. Pseudosorter: A self-supervised spike sorting approach applied to reveal tau-induced reductions in neuronal activity. *Science Advances*, 11(11):eadr4155, 2025.

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

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

Feng Cao and Zishuo Feng. Huidurep: A robust self-supervised framework for learning neural representations from extracellular spikes. *arXiv preprint arXiv:2507.17224*, 2025.

F. J. Chaure, H. G. Rey, and R. Quian Quiroga. A novel and fully automatic spike sorting implementation with variable number of features. *Journal of Neurophysiology*, 120(4):1859–1871, 2018. doi: 10.1152/jn.00339.2018.

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

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

Jan Gasthaus, Frank Wood, Dilan Gorur, and Yee Teh. Dependent dirichlet process spike sorting. *Advances in neural information processing systems*, 21, 2008a.

Jan Gasthaus, Frank Wood, Dilan Görür, and Yee Whye Teh. Dependent dirichlet process spike sorting. In *Advances in Neural Information Processing Systems*, 2008b.

GL Gerstein and WA Clark. Simultaneous studies of firing patterns in several neurons. *Science*, 143(3612): 1325–1327, 1964.

Frank H Guenther, Jonathan S Brumberg, E Joseph Wright, Alfonso Nieto-Castanon, Jason A Tourville, Mikhail Panko, Robert Law, Steven A Siebert, Jess L Bartels, Dinal S Andreasen, et al. A wireless brain-machine interface for real-time speech synthesis. *PloS one*, 4(12):e8218, 2009.

Martin Hägglund, Lotta Borgius, Kimberly J Dougherty, and Ole Kiehn. Activation of groups of excitatory neurons in the mammalian spinal cord or hindbrain evokes locomotion. *Nature neuroscience*, 13(2):246–252, 2010.

Kenneth D Harris, Darrell A Henze, Jozsef Csicsvari, Hajime Hirase, and Gyorgy Buzsaki. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *Journal of neurophysiology*, 84(1):401–414, 2000.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Matthias H Hennig, Cole Hurwitz, and Martino Sorbaro. Scaling spike detection and sorting for next-generation electrophysiology. *In Vitro Neuronal Networks: From Culturing Methods to Neuro-Technological Applications*, pp. 171–184, 2019.

Gerrit Hilgen, Martino Sorbaro, Sahar Pirmoradian, Jens-Oliver Muthmann, Ibolya Edit Kepiro, Simona Ullo, Cesar Juarez Ramirez, Albert Puente Encinas, Alessandro Maccione, Luca Berdondini, et al. Unsupervised spike sorting for large-scale, high-density multielectrode arrays. *Cell reports*, 18(10):2521–2532, 2017.

James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Çağatay Aydın, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, 2017.

Mohammad Reza Keshtkaran and Zhi Yang. Noise-robust unsupervised spike sorting based on discriminative subspace learning with outlier handling. *Journal of neural engineering*, 14(3):036003, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

JinHyung Lee, David Carlson, Hooshmand Shokri, Weichi Yao, Georges Goetz, Espen Hagen, Eleanor Batty, Ej Chichilnisky, Gaute Einevoll, and Liam Paninski. YASS: Yet Another Spike Sorter, June 2017.

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

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

Jens-Oliver Muthmann, Hayder Amin, Evelyne Sernagor, Alessandro Maccione, Dagmara Panas, Luca Berdondini, Upinder S Bhalla, and Matthias H Hennig. Spike detection for large neural populations using high density multielectrode arrays. *Frontiers in neuroinformatics*, 9:28, 2015.

Thanh Nguyen, Asim Bhatti, Abbas Khosravi, Sherif Haggag, Douglas Creighton, and Saeid Nahavandi. Automatic spike sorting by unsupervised clustering with diffusion maps and silhouettes. *Neurocomputing*, 153:199–210, 2015.

Marius Pachitariu, Nicholas A Steinmetz, Shabnam N Kadir, Matteo Carandini, and Kenneth D Harris. Fast and accurate spike sorting of high-channel count probes with KiloSort.

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

Carlos Pedreira, Juan Martinez, Matias J Ison, and Rodrigo Quian Quiroga. How many neurons can we see with current spike sorting algorithms? *Journal of neuroscience methods*, 211(1):58–65, 2012.

Christophe Pouzat and Samuel Garcia. trisdesclous: spike sorting with a french touch. Github, 2016. URL https://tridesclous.readthedocs.io/en/latest/#.

R Quian Quiroga, Zoltan Nadasdy, and Yoram Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004.

Rodrigo Quian Quiroga. Spike sorting. *Scholarpedia*, 2(12):3583, 2007.

Mohammadreza Radmanesh, Ahmad Asgharian Rezaei, Mahdi Jalili, Alireza Hashemi, and Morteza Moazami Goudarzi. Online spike sorting via deep contractive autoencoder. *Neural Networks*, 155:39–49, 2022.

Hernán G. Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. Past, present and future of spike sorting techniques. *Brain Research Bulletin*, 119:106–117, 2015a. doi: 10.1016/j.brainresbull.2015.04.007.

Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. Past, present and future of spike sorting techniques. *Brain research bulletin*, 119:106–117, 2015b.

János Rokai, Melinda Rácz, Richárd Fiáth, István Ulbert, and Gergely Márton. Elvisort: encoding latent variables for instant sorting, an artificial intelligence-based end-to-end solution. *Journal of Neural Engineering*, 18(4):046033, 2021.

Cyrille Rossant, Shabnam N Kadir, Dan FM Goodman, John Schulman, Maximilian LD Hunter, Aman B Saleem, Andres Grosmark, Mariano Belluscio, George H Denfield, Alexander S Ecker, et al. Spike sorting for large, dense electrode arrays. *Nature neuroscience*, 19(4):634–641, 2016.

Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pp. 4–11, 2014.

Artur Speiser, Jinyao Yan, Evan W Archer, Lars Buesing, Srinivas C Turaga, and Jakob H Macke. Fast amortized inference of neural activity from calcium imaging data with variational autoencoders. *Advances in neural information processing systems*, 30, 2017.

Rakesh Veerabhadrappa, Masood Ul Hassan, James Zhang, and Asim Bhatti. Compatibility evaluation of clustering algorithms for contemporary extracellular neural spike sorting. *Frontiers in Systems Neuroscience*, 14, 2020. doi: 10.3389/fnsys.2020.00034.

Juha Vesanto and Esa Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on neural networks*, 11(3):586–600, 2000.

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

Yueqi Wang, Ari Pakman, Catalin Mitelut, JinHyung Lee, and Liam Paninski. Spike sorting using the neural clustering process. In *Real Neurons & Hidden Units: Future directions at the intersection of neuroscience and artificial intelligence@ NeurIPS 2019*, 2019.

Ziqiang Wei, Bei-Jung Lin, Tsai-Wen Chen, Kayvon Daie, Karel Svoboda, and Shaul Druckmann. A comparison of neuronal population dynamics measured with calcium imaging and electrophysiology. *PLoS computational biology*, 16(9):e1008198, 2020.

Frank Wood and Michael J. Black. A nonparametric bayesian alternative to spike sorting. *Journal of Neuroscience Methods*, 173(1):1–12, 2008. doi: 10.1016/j.jneumeth.2008.04.030.

Tong Wu, Wenfeng Zhao, Edward Keefer, and Zhi Yang. Deep compressive autoencoder for action potential compression in large-scale neural recording. *Journal of neural engineering*, 15(6):066019, 2018.

Pierre Yger, Giulia LB Spampinato, Elric Esposito, Baptiste Lefebvre, Stéphane Deny, Christophe Gardella, Marcel Stimberg, Florian Jetter, Guenther Zeck, Serge Picaud, et al. A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo. *elife*, 7:e34518, 2018.

Yiwei Zhang, Jiawei Han, Tengjun Liu, Zelan Yang, Weidong Chen, and Shaomin Zhang. A robust spike sorting method based on the joint optimization of linear discrimination analysis and density peaks. *Scientific reports*, 12(1):15504, 2022.

A Details on Metrics for Comparing Spike Sorting Methods

Let $\{s_i\}_{i=1}^{n_{\mathrm{GT}}}$ be the ground truth firing times, and let $\{t_{k,j}\}_{j=1}^{n_k}$ denote the firing times of sorted unit k. We define an allowable temporal error Δt, assumed to be less than half the refractory period of any neuron.

A ground truth event s_i is considered matched if there exists a sorted spike $t_{k,j}$ such that $|t_{k,j} - s_i| < \Delta t$. At most one sorted event may match each s_i, and no $t_{k,j}$ may match multiple s_i, due to the refractory constraint.

The number of matched events is

$$n_k^{\mathrm{match}} := |\{i : \exists j \text{ such that } |t_{k,j} - s_i| < \Delta t\}|\,.$$

The number of missed ground truth events is

$$n_k^{\mathrm{miss}} := n_{\mathrm{GT}} - n_k^{\mathrm{match}},$$

and the number of false positives is

$$n_k^{\mathrm{fp}} := n_k - n_k^{\mathrm{match}}.$$

We now define the three evaluation metrics used on the main SpikeForest page: precision, recall, and accuracy. These are computed for each ground truth unit, in association with a sorted unit k, using the counts introduced above.

Precision measures the proportion of detected events that are correct:

$$p_k := \frac{n_k^{\mathrm{match}}}{n_k^{\mathrm{match}} + n_k^{\mathrm{fp}}} = \frac{n_k^{\mathrm{match}}}{n_k}.$$

Equivalently, precision equals one minus the false positive rate.

Recall measures the proportion of ground truth events that are successfully detected:

$$r_k := \frac{n_k^{\mathrm{match}}}{n_k^{\mathrm{match}} + n_k^{\mathrm{miss}}} = \frac{n_k^{\mathrm{match}}}{n_{\mathrm{GT}}}.$$

Equivalently, recall equals one minus the false negative rate.

Accuracy combines both errors, reflecting the proportion of correctly identified events among all relevant and retrieved events:

$$a_k := \frac{n_k^{\mathrm{match}}}{n_k^{\mathrm{match}} + n_k^{\mathrm{miss}} + n_k^{\mathrm{fp}}} = \frac{n_k^{\mathrm{match}}}{n_k + n_{\mathrm{GT}} - n_k^{\mathrm{match}}}.$$

B Model Hyperparameters

DeepSpike consists of a symmetric VAE architecture trained on spike waveform segments. The encoder is a 4-layer MLP with hidden dimensions {2048, 1024, 512, 512} followed by two heads predicting the latent mean and log-variance in an 8-dimensional latent space. Each layer is followed by batch normalization and GELU activations. The decoder mirrors the encoder, reconstructing the original waveform from the latent code. The model is trained for 100 epochs using the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 2e-4 and a batch size of 4096. A cosine annealing schedule with warm restarts (initial $T_0 = 10$, $T_{\mathrm{mult}} = 2$, $\eta_{\min} = 1e^{-5}$) is used for learning rate scheduling. The loss consists of an L1 reconstruction term and a KL divergence regularizer. The input dimension is 20, corresponding to the interpolated waveform length. The total number of trainable parameters is approximately 5.9 M trainable parameters. All models are implemented in PyTorch and trained with gradient accumulation disabled.

C Clustering Algorithms for Spike Embeddings

This appendix summarizes the four clustering approaches evaluated in *DeepSpike*: (1) K-Means, (2) finite Gaussian Mixture Models (GMM), (3) HDBSCAN, and (4) our main method—Dirichlet-Process Gaussian Mixture Models (DPGMM). Figure 9 and Table 3 provide visual and qualitative comparisons.

C.1 Fixed-K Clustering Methods

K-Means. A partition-based algorithm that minimizes the sum of squared distances to cluster centroids:

$$\arg\min_{S_1,\ldots,S_K} \sum_{i=1}^{K} \sum_{z \in S_i} \|z - \mu_i\|_2^2.$$

It assumes roughly spherical clusters of similar size and is fast on well-separated data.

Gaussian Mixture Model (GMM). A probabilistic model representing data as a weighted sum of K Gaussians:

$$p(z) = \sum_{i=1}^{K} \pi_i \, \mathcal{N}(z \mid \mu_i, \Sigma_i),$$

estimated via Expectation–Maximization. GMMs allow soft assignments and anisotropic covariances.

Selecting K. When K is unknown, we choose it by a majority vote over Silhouette score, BIC/AIC, and the GAP statistic (definitions in Sec. C.1.1). This mitigates manual tuning.

C.1.1 Cluster-Number Selection Criteria

- **Silhouette.** $s(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$, where $a(i)$ is the mean intra-cluster distance and $b(i)$ the nearest-cluster distance.
- **BIC / AIC.** $\mathrm{BIC} = \ln N\, k - 2 \ln \hat{L},\ \mathrm{AIC} = 2k - 2 \ln \hat{L}$, with N samples, k parameters, and likelihood $\hat{L}$.
- **GAP.** $\mathrm{GAP}(K) = \mathbb{E}[\log W_K] - \log W_K$, where W_K is within-cluster dispersion under the null reference.

C.2 HDBSCAN (Density-Based Clustering)

HDBSCAN is a non-parametric, density-based algorithm that *(i)* detects clusters of arbitrary shape and variable density, *(ii)* labels outliers as noise, and *(iii)* requires no pre-specified K. We employ it as a fallback on recordings with extremely high noise or unknown structure. Its flexibility comes at the cost of tuning a few hyperparameters (e.g. minimum cluster size) and higher runtime on very large datasets.

C.3 Dirichlet–Process Gaussian Mixture Model (DPGMM)

Generative process. For each latent spike embedding $\mathbf{z}_t \in \mathbb{R}^d$ we place a Dirichlet–Process (DP) prior over Gaussian components:

$$\begin{aligned} &G \sim \mathrm{DP}\big(\alpha,\, G_0\big), \quad (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \overset{\text{i.i.d.}}{\sim} G_0, \\ &\mathbf{z}_t \sim \sum_{k=1}^{\infty} \pi_k \, \mathcal{N}(\mathbf{z}_t \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad \pi_k = v_k \prod_{j<k} (1 - v_j),\ v_k \sim \mathrm{Beta}(1, \alpha). \end{aligned} \tag{1}$$

Explanation of variables.

- $\mathbf{z}_t$ – d-dimensional code of spike t (output of the VAE).
- $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ – mean and covariance of the k-th Gaussian component.
- G_0 – Normal–Inverse-Wishart (NIW) base measure over $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.
- α – DP concentration: small α favours few large clusters; large α yields many fine clusters.
- v_k – stick-breaking weights; π_k – resulting mixture proportion for component k.

The stick-breaking construction makes the *effective* number of clusters data-dependent.

Accommodation of non-Gaussian noise. Outliers or burst-overlap artefacts form elongated manifolds that fixed-K GMMs cannot model well. Because the DP can instantiate arbitrarily small new components, such atypical points are assigned to low-weight singleton clusters rather than distorting existing ones. In Chinese-Restaurant terminology, the probability of opening a new cluster is $\alpha/(N{-}1+\alpha)$, where N is the current sample size.

Conjugate inference. With an NIW base measure G_0 the model is conjugate: after observing data, each component's posterior remains NIW. We therefore integrate the parameters out and sample cluster labels $c_t \in \{1, \ldots, K{+}1\}$ directly.

Collapsed Gibbs step. Let $n_{-t,k}$ be the current size of cluster k (excluding $\mathbf{z}_t$), and let $(\mathbf{m}_k, \kappa_k, \nu_k, \mathbf{S}_k)$ be the NIW hyper-parameters after observing that cluster. Then

$$\Pr(c_t = k \mid \mathbf{z}_t, \mathbf{c}_{-t}) \propto \begin{cases} n_{-t,k}\; t_{\nu_k - d+1}\big(\mathbf{z}_t;\, \mathbf{m}_k,\, \frac{\kappa_k+1}{\kappa_k(\nu_k-d+1)}\mathbf{S}_k\big), & k \le K, \\ \alpha\; t_{\nu_0 - d+1}\big(\mathbf{z}_t;\, \mathbf{m}_0,\, \frac{\kappa_0+1}{\kappa_0(\nu_0-d+1)}\mathbf{S}_0\big), & k = K{+}1. \end{cases} \tag{2}$$

Here $t_\nu(\cdot; \mathbf{m}, \mathbf{\Lambda})$ denotes the multivariate Student–t density obtained after collapsing NIW parameters. Intuitively, embeddings join clusters that are both large ($n_{-t,k}$) and predictive (high t-density); otherwise they start a new cluster weighted by α.

Marginal likelihood. Integrating over all partitions and parameters yields

$$p(\mathbf{z}_{1:T} \mid \alpha) = \sum_{K=1}^{\infty} \frac{\alpha^K \Gamma(\alpha)}{\Gamma(\alpha+T)} \prod_{k=1}^{K} \Gamma(n_k) \int \mathcal{N}(\mathbf{z}_k \mid \boldsymbol{\mu}, \mathbf{\Sigma})\, G_0(d\boldsymbol{\mu}\, d\mathbf{\Sigma}), \tag{3}$$

automatically penalising superfluous clusters through the Gamma factors.

Scalable variational alternative. For million-spike datasets we adopt a mean-field variational approximation truncated at $K_{\max} = 20$ components; in practice far fewer are activated, preserving the non-parametric spirit while enabling fast, vectorised GPU updates.

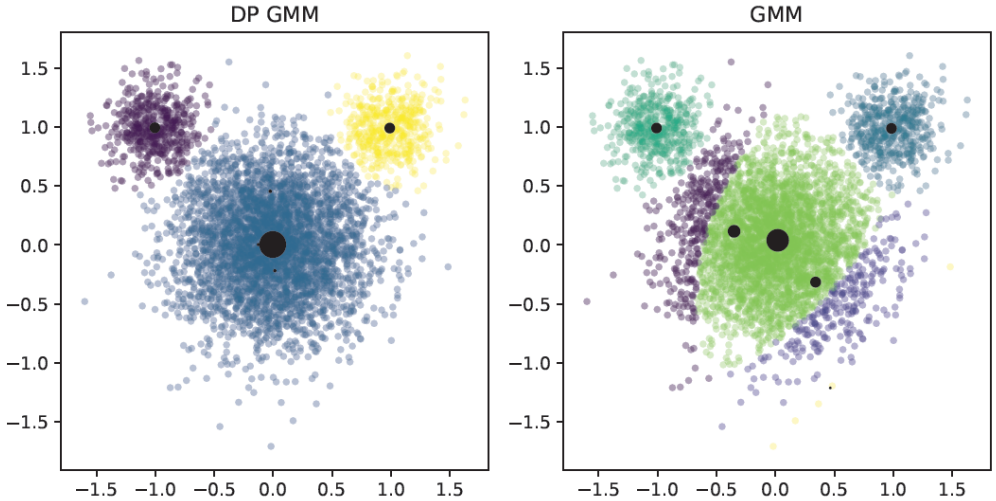

Figure 9: GMM vs. DPGMM with five active components. Marker size encodes mixture weight; colours denote clusters. DPGMM allocates extra low-weight atoms to outliers, whereas the finite GMM distorts existing clusters.

C.4 Overview and Qualitative Comparison

Table 3: Qualitative comparison of clustering strategies.

Property	**K-means**	**Finite GMM**	**DPGMM (ours)**
Unknown K	Predetermined	Predetermined	Inferred by posterior
Cluster covariances	Spherical	Fixed / tied	Full, learned
Heavy-tailed noise	Poor	Moderate	Handled via new atoms
Waveform drift	Not captured	Post-hoc heuristic	DDP random walk
Uncertainty quantification	None	Asymptotic	Explicit posterior