**A study of deep clustering in spike sorting**

Eugen-Richard Ardelean1,\*, Raluca Laura Portase1

1Department of Computer Science, Technical University of Cluj-Napoca, Cluj-Napoca, Romania

**\* Corresponding authors:** [ardeleaneugenrichard@gmail.com](mailto:ardeleaneugenrichard@gmail.com)

**ORCID Author IDs:**

Eugen-Richard Ardelean: 0000-0002-0098-4228

Raluca Laura Portase: 0000-0002-8985-4728

**Abstract: [**TBWritten]

**Keywords:** clustering, deep clustering, spike sorting, neuroscience

# Introduction

## Spike Sorting

Spikes sorting (1) is the process of organizing the instances of activity of neurons, also known as spikes, into groups depending on the putative neurons. This process is applied to extracellular data where the activity of multiple neurons is captured by a recording electrode (2). Therefore, the provenience of each spike is unknown during recording. Spike sorting is often referred to as analogous (3) to the cocktail party problem (4). The latter requires the isolation of an individual’s speech in complex environment, similarly spike sorting attempts to extract the words (as in spikes) of a single person (as in neuron) in a complex environment that is riddled with noise and where individuals may speak at once (spike superposition) and at different cadences (different firing rates of neurons). Spike sorting operates on the assumption that each neuron produces spikes of a similar shape, while simultaneously different from the spikes of any other neuron.

In its most traditional form, the spiking sorting process is separated into four consecutive steps (1): filtering of the recording signal, spike detection of the filtered signal, feature extraction of the detected spikes (to reduce dimensionality) and clustering for the assignment of spikes to a specific neuron. The filtering of the recording signals employs a band-pass filter between 300 and 3000Hz (5) to capture the frequency components of spikes. The spike detection step is traditionally a simple amplitude thresholding which imposes a comprise between the precise identification of spikes and the number of instances identified. In other words, the more spikes are identified the higher is the chance to include noise; however, as many spike as possible must be extracted. This study focuses on the last two steps of the pipeline, the feature extraction and the clustering. The feature extraction step attempts to identify the most informative feature for the generation of lower-dimensional working space, while the clustering step attempts to separate clusters in the space obtained by the feature extraction. In this case, the most informative features represent the features that bestow the most separability between clusters. Finally, each cluster should represent all instances of activity of a single neuron.

The spike sorting pipeline has seen a number of iterations during the years, starting from a manual approach (43) where spikes were separated and assigned by a researcher based on simple charateristics such as amplitude and width (39). One such discriminatory feature found was the peak-to-trough ratio (44) which allowed researchers to distinguish between the spikes of inhibitory (narrow spikes) and excitatory (wide spikes) neurons. More complex probabilistic models which could process a low number of electrodes were created that were able to make use of the entire spike waveform (40). Later, more complex algorithms were employed principal components (41), the wavelet transform (42) and various combinations of them to project the high-dimensional space of spikes into a low-dimensional space.

Due to the recent advances in recording hardware such methods are rapidly turning intractable. The number of neurons captured in recordings has been increasing exponentially since the 1950s (6) and now with the development of multi-array silicon probes (7,8), thousands of neurons can be captured in a single recording. Depending on the approach, online or offline, different variants can be used. In offline spike sorting the use of more sophisticated algorithms is allowed by the lack of a time constraint, while in online spike sorting it must be done during the recording and a faster approach is required to abide by the time constraints.

Lately, template matching has been increasingly utilized as an alternative for the spike detection and feature extraction steps due to its performance and computational efficiency (9), as it is usually applied to only a subset of the dataset. One such method that focuses on the Wavelet Transform for detection and template matching is M-Sorter (10). It detects spikes from the band-pass filtered spike waveforms by computing the correlation of wavelet coefficients, templates are generated through the use of K-Means, and are spikes are matched to the closest templates. Another pipeline that employs template matching and K-Means is KiloSort (9,11). Kilosort creates spike templates through mathematical models which are then used to initialize a modified K-Means. Computational efficiency is the main advantage of KiloSort, however it also allows the possibility of human intervention as a post-processing step.

In this work, we endevour in the pursuit of identifying the suitability of deep clustering algorithms for spike sorting. Although, many feature extraction and clustering algorithms have been employed in the task of spike sorting, no golden standard (1,5,12,13) has been yet found as the performance of each algorithm is dependent upon the specific characteristics of the data. Here, we propose the use of deep clustering algorithms in the pursuit of identifying a more performant option for the spike sorting task.

## Deep Clustering Algorithms

Deep clustering algorithms (14) are neural network approaches to clustering based on autoencoders (15,16). Traditionally, autoencoders are composed of two inter-linked parts: an encoder and a decoder. Their task is to compress the input data into a latent representation, usually lower-dimensional, and reconstruct the input data at the ouput. Autoencoders have been applied for many different applications such as feature extraction, dimensionality reducion, generative modelling and anomality detection. Autoencoders have been also been demonstrated to be an adequate approach in the feature extraction of image datasets (17–19), such as MNIST. Due to inherent non-linearity of autoencoders from the activation functions, autoencoders are a suitable approach for the task of spike sorting (20).

Traditional clustering methods have been shown to struggle with high-dimensional complex data. Deep clustering algorithms have been proposed a solution for this issue and have been demonstrated to have a high performance on image datasets (21–28). Most of these methods (21,22,25,27,29,30) have been designed with a modified loss function to include both the reconstruction and the clustering as well. A subset of deep clustering methods (23,24) have also been designed based on pretraining followed by iterative refinement based on the statistical dip-test (31) for modality in iterative loops for updating labels (23) or as a postprocessing step of cluster merging (24). Even a tree approach (27) has been designed that uses a joint optimization strategy for clustering. Simpler approaches have also been taken, where a 2-stage approach (32) is taken, the autoencoder beings by creating a low-dimensional representation of the input which is then further reduced through the t-SNE algorithm to a 2-dimensional space that is clustered by a density-based approach. Thus, deep learning approaches are strong candidates for spike sorting.

## The challenges of spike sorting [TBRewritten]

As it was alluded to in the cocktail party problem, spike sorting suffers from an assortment of challenges. Realistically, even if the idea of neural coding would be invalidated, background noise induces variability into the shapes of spikes which would still generate clusters. Consequently, feature extraction techniques are an important step in improving the robustness of clustering by removing redundant information. As pointed to above, neurons can have different firing rates (33) (34). Within the finite frame of a recording, different firing rates results in a different number of spikes which leads to imbalanced clusters. This happens due to neuronal activity being modulated by entire brain circuits rather than a single neuron deciding. Besides noise, the shape of spikes can be disrupted by phenomena such as electrode drift (8). These can lead to more similar spike shapes which result in overlapping clusters. Finally, the time scale of neuronal activity is of milliseconds, implying that even a brief recording will generate a high volume of data (35). From a terminological perspective, single unit activity refers to a cluster that is composed from the spikes of a single neuron, while multiunit activity refers to a “cluster” that is composed of the spikes of multiple neurons (usually more distant from the recording electrode) (5).

The aim of feature extraction is to generate a new feature space that is resistant to small changes in spike shape, thus offering separability in clusters. The purpose of clustering is to group the different groups of activity to identify the activity of different neurons. As the spikes of neurons are muddled by the inherent background noise of brain recordings, autoencoder which have been demonstrated a robust ability for denoising may be able to offer a latent representatino that is invariant to noise (15). Autoencoders have seen previous use in spike sorting (20,36) with promising results. Many variants have been developed for the introduction of deep learning into spike sorting (37). Yet, the suitability of deep clustering methods in spike sorting which employ autoencoders has not yet been determined.

The paper is organized in the following manner: Section 2 presents of traditional feature extraction methods, clustering methods and performance metrics metrics used in the analysis. Moreover, it provides a description of the proposed methods for spike sorting, and of the datasets. In Section 3, a thorough evaluation is made on each method on multiple datasets from the perspective of various performance metrics, simultaneously offering a critical interpretation of the results. Finally, Section 4 explores the limitations of our proposal and the conclusions reached.

# Materials and Methods

## Feature Extraction

Feature extraction is a key step in the spike sorting pipeline in which spike waveforms are represented through a smaller informative feature space. In spike sorting, for computational reasons, feature extraction attempts to reduce the dimensionality of the original feature space while retaining the information that allows for the discrimination of spikes from different neurons. This implies creating features that are invariant to the noise that differentiates the spikes produced by the same neuron. Techniques for feature extraction methods can be categorized by multiple attributes such as linearity, thus PCA is a linear convex algorithm while ICA is a linear non-convex approach.

### Linear feature extraction methods

One of the most widely used techniques for feature extraction, in general and in spike sorting (45), is the Principal Component Analysis (PCA) (46). Despite its limitations, PCA has been extensively used in spike sorting over the years (5) and it is still is used in modern spike sorting pipelines (47). PCA transforms the input data into a new feature space of orthogonal axes – called principal components – which are derived through eigenvalue decomposition. It can reduce dimensionality by discarding components with low variance. Often by retaining only the first few principal components (48) (49), more than 70% of the data variance is captured. However, by retaining variance it is not guaranteed that an optimal space for clustering is created (1) (5).

Independent Component Analysis (ICA) (50) is another linear method, generally employed in source separation that has been shown to have applications in spike sorting (51) (52). ICA focuses on maximizing independence among the components it can find, rather than variance as PCA does. This unsupervised approach identifies independent sources in the data allowing it to isolate individual instances of neuronal activity and it has been demonstrated to have a high performance in spike sorting (51) (52).

### Non-linear feature extraction methods

In the category of unsupervised non-linear dimensionally reduction techniques Isomap (53) uses Isometric Mapping to learn the low-dimensional projection in a manifold space while retaining the distances of the original space. It uses the geodesic distance, which can be thought of as the shortest path along the curved surface of the manifold space.

T-distributed Stochastic Neighbor Embedding (t-SNE) (54) is a non-linear dimensionality reduction method that minimizes the divergence between input features and the reduced feature space by using pairwise probability similarities. The divergence of two distributions is calculated using KL divergence, which is minimized by applying gradient descent. Due to its high time complexity, several orders of magnitude higher than PCA, and its main function being visualization, t-SNE was not considered a suitable candidate. A computation of a few seconds for PCA can become tens of minutes for t-SNE. Furthermore, from empirical observations, the separation offered by t-SNE for the datasets used here was small to non-existent.

## Traditional clustering algorithms

K-Means (55) is a partition-based clustering algorithm. It partitions the space into k partitions, where each sample is appointed to the closest centroid based on the Euclidean distance. K-Means has several disadvantages. First, it requires the number of clusters as an input which is hard to provide for real data. Second, in its most basic form it is not deterministic, such that each execution may result in a different clustering. Through recent optimizations it has been improved and has increased stability. Third, K-Means has difficulties in separating overlapping clusters. In our case, this is an advantage: If the performance of K-Means is higher for a certain feature extraction method it denotes that the method provides better separation.

DBSCAN?

## Deep clustering algorithms

Traditional clustering algorithms struggle with complex data structures. Deep clustering techniques combine representation learning with clustering objectives to enhance performance, often using autoencoders. Most of these methods have been tested on the MNIST dataset (21–28), showing a satisfactory performance in clustering high dimensional datasets; thus, proving their potential for complex tasks such as spike sorting. The deep clustering algorithms analyzed here have their code provided by the authors. For consistency of the results, we have used the implementations of these algorithms from clustpy (56), with some modifications to improve performance.

ACeDeC (25), introduced in 2021, is a deep clustering approach that separates the latent representation into distinct spaces: a clustering space for cluster-specific information and a shared space for general data variation. ACeDeC measures the importance of each dimension within these spaces. Additionally, the loss function used accounts for the cluster information by minimizing distances to centroids, the shared information by modelling the distance to the mean of the embedded data and for the reconstruction of the autoencoder. By separating the embedded space and using a reformulated loss function, ACeDeC enables the learning of detailed reconstructions and cluster-specific abstractions and it improves clustering performance. Experiments on various datasets demonstrate ACeDeC's superior performance compared to existing methods, even DCN (29) another deep clustering approach.

AEC (26), introduced in 2013, is a deep clustering approach that proposes using autoencoders for mapping data to a more suitable space. This method incorporates both data reconstruction and cluster compactness through its proposed loss function, leading to more stable and effective clustering. The model iteratively refines data representation and cluster centres, achieving superior performance compared to conventional approaches like K-means. Experiments on benchmark datasets demonstrate the improved accuracy and normalised mutual information of this auto-encoder-based clustering technique.

DCN (29), introduced in 2017, proposes the use of deep neural networks (DNNs) for dimensionality reduction and K-means for the clustering of high-dimensional data. This method learns a 'clustering-friendly' latent space by simultaneously optimising data reconstruction, dimensionality reduction, and cluster structure. DCN uses an autoencoder network structure (with a step of greedy layer-wise pre-training (57)) with a K-means clustering objective at the bottleneck layer to avoid trivial solutions, and an alternating stochastic gradient algorithm for optimisation. Experiments on synthetic and real-world datasets demonstrate the effectiveness of DCN in improving clustering performance compared to state-of-the-art methods, particularly in cases with unbalanced clusters. It was shown to outperform other deep clustering approaches, such as DEC (58) and simpler approaches that used an autoencoder to reduce dimensionality and a clustering algorithm such as K-Means.

DDC (32), introduced in 2020, employs a two-stage approach: first, it uses a deep convolutional autoencoder to learn low-dimensional feature representations, and then applies a new density-based clustering technique. DDC uses a deep autoencoder to learn deep feature representations of data. It adopts t-SNE to further reduce the learned features to a 2-dimensional space while preserving the pairwise similarity of data instances. It develops a novel density-based clustering method that considers both the local structures of clusters and the importance of instances to generate the final clustering results. This method addresses limitations in existing deep clustering algorithms, specifically the need for a pre-defined number of clusters and instability with non-spherical cluster shapes. Experiments demonstrate that DDC achieves state-of-the-art performance, even when the number of clusters is unknown, making it a robust solution for various image clustering tasks. Moreover, DDC was shown to outperform other deep clustering methods, specifically DEC (58), IDEC (22), DKM (21) and VaDE (59).

DEC (58), introduced in 2016, proposes the use of DNNs, specifically an autoencoder, to simultaneously learn feature representations and cluster assignments. It iteratively refines clusters by optimising a clustering objective in a lower-dimensional space. This process involves computing soft assignments and minimising Kullback-Leibler divergence using an auxiliary target distribution to map the autoencoder’s embeddings to cluster centroids. DEC applies a greedy layer-wise pre-training (57) on the autoencoder starting with weights initialized from a normal distribution. The authors demonstrate significant improvements over existing clustering methods on image and text datasets. Furthermore, DEC exhibits robustness to hyperparameter variations, making it practical for real-world applications. The algorithm's linear complexity enables it to scale effectively to large datasets.

DeepECT (27,28), introduced in 2019, is a deep hierarchical clustering approach that combines the strengths of deep learning and traditional clustering methods. It uses a generic feedforward autoencoder with a clustering layer that builds a cluster tree (without needing the number of clusters specified beforehand) in an embedded space, and both the embedding and the tree are trained simultaneously. DeepECT uses a projection-based optimisation strategy that enhances cluster boundaries and preserves orthogonal structural information through a compression loss that penalises the distance between data points and their assigned node centres. It also includes an extension that utilises augmentation methods to ignore known invariances within the data. Experimental results demonstrate that DeepECT excels in creating high-quality cluster trees and performs competitively with flat clustering methods. It was shown to outperform other deep clustering approaches, such as IDEC (22) and simpler approaches that used an autoencoder to reduce dimensionality and a clustering algorithm such as K-Means.

DipDECK (24), introduced in 2021, is a deep clustering approach that simultaneously learns data representations and estimates the number of clusters present. DipDECK integrates a cluster number estimation within the deep learning process, addressing limitations in scalability and reliance on pre-defined cluster numbers. The algorithm uses an autoencoder to embed data, overestimates the initial cluster count, and then applies Hartigan's Dip-test to merge structurally similar clusters. Experiments demonstrate that DipDECK achieves competitive clustering results, accurately estimates cluster numbers, and exhibits robustness across various datasets and parameter settings. Moreover, it was compared with other deep clustering methods and shown to outperform them, specifically, DEC (58), IDEC (22), DCN (29) and VaDE (59) on 7 out of 8 datasets.

DipEncoder (23), introduced in 2022, is a deep clustering algorithm that leverages Hartigan's Dip-test to enforce multimodality in autoencoders. This approach combines an autoencoder with the Dip-test, enabling the creation of embeddings that clearly separate clusters within a dataset. The DipEncoder uses gradients of the Dip-value with respect to both the projection axis and the data itself to improve cluster separation. It uses two loss terms, one to minimize the modality of within separate clusters and another to maximize modality between combinations of clusters. The algorithm updates cluster labels using the Dip-test and requires only the number of clusters as a parameter, offering a parameter-free method for deep clustering. By maximizing multimodality between clusters while ensuring unimodality within individual clusters, the DipEncoder achieves competitive performance compared to state-of-the-art deep clustering methods, specifically, DEC (58), IDEC (22), DCN (29) and DipDECK (24) on 6 out of 10 various datasets, including image, numerical, and text data.

DKM (21), introduced in 2020, is a deep clustering algorithm that jointly learns data representations and performs K-Means clustering. It uses joint optimization through stochastic gradient descent to learn autoencoder-based representations and it uses a differentiable parametrized softmax instead of argmin for K-Means. It uses a greedy layer-wise pre-training (57) for the autoencoder in one variant and an annealing approach for a second variant. DKM uses a continuous reparametrisation of the objective function. Experiments on image and text datasets demonstrate DKM's superior clustering performance compared to other deep clustering models such as DCN (29) and IDEC (22). The pretrained variant obtained a slightly higher and more stable performance when compared with the annealing variant.

IDEC (22), introduced in 2017, is a deep clustering approach that seeks to simultaneously cluster data and learn meaningful feature representations by integrating an autoencoder with a clustering loss function. This combination allows the algorithm to scatter data points while preserving the local structure of the data. It is stated that preserving this structure is vital for effective deep clustering as clustering losses can sometimes corrupt the feature space, leading to non-representative and meaningless feature. IDEC uses an under-complete autoencoder. IDEC uses a stacked denoising autoencoder (with a step of greedy layer-wise pre-training (57)), followed by an under-complete (the latent code is of lower size than the input) autoencoder after initialization to preserve the local structure of the data generating distribution. This constrains the manipulation of the feature space while using a clustering loss to scatter data points. Moreover, IDEC has been shown to outperform its precursor DEC (58) and simpler approaches that used an autoencoder to reduce dimensionality and a clustering algorithm such as K-Means

N2D (60), introduced in 2021, is a deep clustering approach that simplifies existing methods by replacing a deep clustering network with manifold learning. N2D uses an autoencoder to create an initial data representation, then employs manifold learning techniques, especially UMAP, to uncover a more cluster-friendly structure. This manifold learning step focuses on preserving local distances while retaining global structure, improving cluster quality. The resulting embedding is then clustered using a shallow algorithm, achieving competitive, and sometimes superior, performance on image and time-series datasets. Experiments demonstrate N2D's efficiency and effectiveness compared to traditional and state-of-the-art deep clustering methods.

VaDE (59) or Variational Deep Embedding, introduced in 2017, is an unsupervised, generative clustering approach that uses variational autoencoders (VAE). It models data generation by combining a Gaussian Mixture Model (GMM) with a deep neural network (DNN), where the GMM selects a cluster to produce a latent embedding and the DNN decodes this into an observable output. An encoder network is used to infer latent embeddings from observables to maximise the evidence lower bound (ELBO). The method aims to learn suitable representations for clustering tasks and generate realistic samples without supervised training. The experiments presented demonstrate VaDE's ability to outperform state-of-the-art methods on benchmark datasets.

AutoClustering (30), introduced in 2018, a clustering algorithm based on feed-forward neural networks (FFNN), offering an alternative to methods like Self-Organising Maps (SOM). This approach employs an encoder-decoder structure and a loss function to map data records to clusters and their exemplars through distance. The proposed approach of exemplars is conceptually similar to K-means’ cluster centroids. This work introduces an improved activation function, facilitating a smooth transition from soft-max to max functions. Experimental results, assessed via homogeneity and completeness metrics, demonstrate the algorithm's effectiveness, especially with blob-shaped datasets, although stability issues related to local minima are noted. Comparisons with Gaussian mixture models, k-means models, and affinity propagation show AutoClustering's performance.

## Performance metrics

Rather than relying on a single metric, we opted to use several metrics to capture the performance of these methods from a variety of angles helping us avoid evaluation bias. A method demonstrating high scores across all 6 performance metrics indicates a robust ability for clustering.

Six metrics were employed for the evaluation of performance: Adjusted Rand Index (ARI), Adjusted Mutual Information (AMI), V-Measure (VM), Calinski-Harabasz Score (CHS), Davies-Bouldin Score (DBS), and Silhouette Score (SS). Each of these methods is shortly described in Table 1 along with an interpretation of its internal workings, range and type. As clustering follows feature extraction in spike sorting, internal metrics (61) also reflect the separability imparted by the feature extraction algorithm as they evaluate the compactness and separation of the clusters in a given space without requiring ground truth labels. Conversely, external metrics evaluate the correspondence between the true labels and the predicted labels.

**Table 1** – A short description of each performance evaluation metric, specifying its type and range.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Range [worst, best] |
| ARI | External | Agreement between true and predicted labels using mutual information using a pairwise comparison, with an added normalization to account for random assignments. | [-1, 1] |
| AMI | External | Agreement between true and predicted labels using mutual information, with an added normalization to account for random assignments. | [0, 1] |
| Purity | External | Percentage of data points assigned to the correct class, assuming each cluster is labeled by majority vote, obtains the proportion of correctly assigned points. | [0, 1] |
| DBS | Internal | Ratio of within-cluster scatter to between-cluster separation, assessing cluster compactness and separation. | (Inf, 0] |
| CHS | Internal | The fraction between the dispersion between clusters and the dispersion within clusters, evaluating how distinct and compact the clusters are. | [0, Inf) |
| SS | Internal | Average of individual scores per data point that compare intra-cluster closeness with nearest-cluster separation, indicating overall clustering quality. | [-1, 1] |

[TBRewritten]

External metrics require the labels of the clustering algorithm, and the ground truth labels. Therefore, a clustering algorithm has to be applied after the feature extraction and we have chosen K-Means (55). K-means has a long history of use as a clustering algorithm and many variations have been developed. It was introduced in spike sorting in 1988 and remained the de facto clustering algorithm for a long time (62) (63). Furthermore, newly developed spike sorting techniques and pipelines are based on it or use it (9) (64) and in recent evaluations K-Means has been shown to still be a highly performant option, as it placed third in the evaluation of 25 clustering algorithms (62).

ARI (65) (66) (67) (3) is an adjustment of the Rand Index (RI) metric in order to handle chances. ARI is an external clustering metric; therefore, it requires a ground truth for the dataset. RI (68) (2) makes comparisons between pairs of points to determine if it is an agreement, when the two points are in the same cluster for both the predicted and the true labels, or a disagreement, when they belong to different clusters. The formulas used to calculate the metric are the following:

where *MaxRI* is the upper bound and *ExpectedRI* is theexpected placement of pairs in the same class using the permutation model and calculated based on the contingency table (65).

AMI (66) (69) (5) is an adjustment of the Mutual Information (MI) metric through the use of entropy, denoted as H. Moreover, AMI also contains the normalization (66) (70) (71) of Normalized Mutual Information. MI (4) is calculated between two clusters U and V, where N is the size of the dataset and |X| is the number of points in subset X.

V-Measure (72) (6) is the harmonic mean of Homogeneity and Completeness. A cluster is considered to be homogeneous (7) when all the points of that cluster are part of the same class. By switching the predicted and true labels, completeness is obtained. Completeness (8) is achieved when all the points of a class are part of the same cluster. We have chosen beta equal to 1 as given by the original formula (72).

where H(C|K) is the conditional entropy of the true cluster given the predicted cluster, H(C) is the entropy of the true cluster, while H(K|C) is the conditional entropy of the predicted cluster given the true cluster and H(K) is the entropy of the predicted cluster.

All the metrics presented until this point are external metrics and require a ground truth to compare with the predicted labels. Furthermore, all these metrics have bounded scores in the [0, 1] interval with higher values being more desirable.

The following three metrics are internal and therefore do not require a ground truth to be used. The internal metrics were used with the ground truth labels for the evaluation of the synthetic datasets. These metrics evaluate the intra-cluster and inter-cluster distances and the morphology of the clusters producing an adequate evaluation of the feature extraction capabilities.

DBS (73) (74) (75) (10) finds the mean similarity between clusters, where similarity, denoted as *R* (9), is defined by the distance between clusters and their sizes. The minimum value of this index is 0. The closer the result is to 0, the better separation exists between clusters. This may come as counterintuitive as it is the only metric where lower values represent a higher performance. The DBS metric is given by the following equations:

where *si* is the mean of all distances between the points of cluster *i* and its centroid, *di,j* is the distance between clusters *i* and *j* given by their centroids, and *max(Ri,j)* is the maximum similarity of clusters *i* and *j*.

CHS (61) (72) (11), also known as Variance Ratio Criterion, calculates the ratio between the intra-cluster and inter-cluster dispersion. Where *tr(X)* denotes the trace of between cluster *Bk* or within-cluster *Wk* dispersion matrix, *n* denotes the size of the dataset and *k* the number of clusters. The dispersion is defined as the sum of squared distances. For this metric, a higher value indicates a better result.

SS (72) (76) (12) is calculated by measuring the mean distance between a point and the rest of the points of that cluster and the mean distance between the point and all the points of the nearest cluster. The score is bound between [-1, 1] where -1 represents an incorrect clustering, 0 overlapping clusters, and 1 a dense clustering. SS aims for the standard concept of a cluster, dense and well separated, therefore such cases will give a higher score. The equation of SS is the following:

where *b* denotes the average of all distances between a point in cluster *i* and all points of the nearest cluster *j*, and *a* the average of all distances between a point in cluster *i* and all other points in the same cluster.

It is important to mention that although used in evaluation of spike sorting techniques (36) (20), accuracy is not a suitable performance metric. First, because spike sorting is unsupervised and accuracy requires labels. Second, neuronal data is imbalanced because of the various firing rates of individual neurons, and it is has been extensively shown that accuracy is not appropriate for evaluating tasks on imbalanced data (77) (78) (79) (80). Nevertheless, through the use of the chosen metrics, we are able to evaluate the separation and shapes of cluster using the internal metrics and the correctness of clustering using the created features using the external metrics.

## Data [TBRewritten]

### Synthetic datasets

The validation of deep clustering methods was made by comparing the different methods with [\*\*\*PCA, ICA and Isomap\*\*\*]. The chosen datasets, 95 in number and denominated as simulations, originate from the Department of Engineering, University of Leicester UK and are publicly available. Each simulation is a dataset. The creation of these simulations was based on recordings from the neocortex of a monkey. They were generated using 594 different spike shapes (12). The original study that introduces the simulations (12) also reviews different clustering algorithms and their results. Out of 20 different units, these algorithms were able to detect 10 in the best case.

The datasets were generated based on a real dataset recorded “in vivo”. The waveform contains 316 points originally sampled at 96 KHz; afterwards this frequency was reduced to 24KHz, therefore 79 samples describe a spike. Being synthetic datasets, each of these spikes has a label, which allows for the use of external metrics to evaluate performance. Each simulation contains a multi-unit cluster, which is the noise, and a number of clusters that varies between 2 and 20. Each unique number of clusters has 5 simulations. Thus, there are 5 simulations with 2 clusters, 5 simulations with 3 clusters, and so on.

All but one of the clusters are single-units between 0 and 50μm away from the electrode. The firing rate follows a Poisson distribution with a mean between 0.1 and 2Hz. The amplitudes follow a normal distribution and have been scaled to values between 0.9 and 2 to simulate real data. No spikes with temporal overlapping are present in the data, such that spikes have at least 0.3ms between them.

The generated multi-unit cluster was added in order to increase the complexity of clustering for the tested algorithms. The simulated multi-unit contains 20 spike shapes, each of the 20 neurons firing being between 50-140μm away from the electrode. The amplitude of the spikes was fixed to 0.5, with an overall composite firing rate of 5Hz, with each of the 20 individual composing neurons having a firing rate mean of 0.25Hz following an independent Poisson distribution. Here, in order to increase clarity, the multi-unit cluster is always color-coded in white in all figures.

To evaluate the proposed approach in comparison with other state-of-the-art methods we have chosen the following 4 simulations out of the 95 available as they are representative of the issues that are present in feature extraction methods and allow for the evaluation of the methods on varying numbers of clusters covering a wide range and enabling a comprehensive evaluation of performance:

* Simulation 1 (Sim1 - Fig 3a), containing 16 single-unit clusters and a multi-unit cluster (in total 17) with 12012 samples.
* Simulation 4 (Sim4 - Fig 3b), containing 4 single-unit clusters and a multi-unit cluster (in total 5) with 5127 samples.
* Simulation 16 (Sim16 - Fig 3c), containing 8 single-unit clusters and a multi-unit cluster (in total 9) with 7556 samples.
* Simulation 35 (Sim35 - Fig 3d), containing 12 single-unit clusters and a multi-unit cluster (in total 13) with 9481 samples.
* Simulation 14, containing 3 single-unit clusters and a multi-unit cluster (in total 4) with 4507 samples. This dataset was used for the visualization of the impact of alignment on feature extraction in section 3.1.

These simulations can also be viewed in Fig 3 through the use of PCA to reduce the dimensionality from 79 to 2. The overlapping clusters produced by PCA can be clearly seen in Fig 3, in none of the datasets is it able to perfectly separate all clusters.

Scatter chart

Description automatically generated

**Fig 3. PCA projection of the synthetic datasets.** PCA projection of 4 different simulations with distinct numbers of clusters; the colors represent the cluster assignment given in the ground truth.

## Data preprocessing

Besides the traditional scaling and shuffling of data, a first step of preprocessing that might improve spike sorting performance is the alignment of spikes (the necessity of this step is dependent upon the spike detection. The following expression was employed for to align all spikes to a given index:

The starting index of a spike waveform in the recorded signal is given by the *old\_start* term, which must be shifted to *new\_start* to align all spikes to the same *index*. The reference point for alignment can be any index of the spike. However, the superior choice regarding performance is the maximum peak of the spike, also called the amplitude which is represented by the *peak* term. This formula offers flexibility as any reference point could be chosen for alignment (81).

Chart, scatter chart

Description automatically generated

**Fig 4. Impact of alignment.** PCA applied on Sim29 with and without alignment. The white cluster is kept together but the overlap with the blue cluster remains.

# Results

## Performance evaluation

## Performance evaluation of all datasets

A thorough analysis of performance requires varying levels of complexity in the data used. Our analysis based on 95 datasets (12) containing a range of cluster counts and spike shapes allows for an extensive evaluation of performance.

Fig X shows the results obtained by each method across all 95 datasets by each performance metric.

In Fig X, a statistical analysis can be reviewed carried out via Bonferroni corrected t-tests.

In Table X, a Borda aggregation-based ranking of the methods according to their performance on each metric is presented.

# Discussion

# Bibliography

1. Quiroga RQ. Spike sorting. Scholarpedia. 2007 Dec 21;2(12):3583.

2. Carter M, Shieh J. Chapter 4 - Electrophysiology. In: Carter M, Shieh J, editors. Guide to Research Techniques in Neuroscience (Second Edition) [Internet]. San Diego: Academic Press; 2015 [cited 2022 Aug 2]. p. 89–115. Available from: https://www.sciencedirect.com/science/article/pii/B9780128005118000046

3. Brown GD, Yamada S, Sejnowski TJ. Independent component analysis at the neural cocktail party. Trends Neurosci. 2001 Jan;24(1):54–63.

4. Haykin S, Chen Z. The cocktail party problem. Neural Comput. 2005 Sep;17(9):1875–902.

5. Rey HG, Pedreira C, Quian Quiroga R. Past, present and future of spike sorting techniques. Brain Res Bull. 2015 Oct 1;119:106–17.

6. Stevenson IH, Kording KP. How advances in neural recording affect data analysis. Nat Neurosci. 2011 Feb;14(2):139–42.

7. Jun JJ, Steinmetz NA, Siegle JH, Denman DJ, Bauza M, Barbarits B, et al. Fully integrated silicon probes for high-density recording of neural activity. Nature. 2017 Nov;551(7679):232–6.

8. Steinmetz NA, Aydin C, Lebedeva A, Okun M, Pachitariu M, Bauza M, et al. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. Science. 2021 Apr 16;372(6539):eabf4588.

9. Pachitariu M, Steinmetz N, Kadir S, Carandini M, D HK. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels [Internet]. bioRxiv; 2016 [cited 2022 Jul 18]. p. 061481. Available from: https://www.biorxiv.org/content/10.1101/061481v1

10. Yuan Y, Yang C, Si J. The M-Sorter: An automatic and robust spike detection and classification system. J Neurosci Methods. 2012 Sep 30;210(2):281–90.

11. Pachitariu M, Sridhar S, Pennington J, Stringer C. Spike sorting with Kilosort4. Nat Methods. 2024 May;21(5):914–21.

12. Pedreira C, Martinez J, Ison MJ, Quian Quiroga R. How many neurons can we see with current spike sorting algorithms? J Neurosci Methods. 2012 Oct 15;211(1):58–65.

13. Estivill-Castro V. Why so many clustering algorithms: a position paper. SIGKDD Explor Newsl. 2002 Jun 1;4(1):65–75.

14. Min E, Guo X, Liu Q, Zhang G, Cui J, Long J. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. IEEE Access. 2018;6:39501–14.

15. Pinaya W, Vieira S, Garcia-Dias R, Mechelli A. Autoencoders. In 2019. p. 193–208.

16. Baldi P. Autoencoders, Unsupervised Learning, and Deep Architectures. In: Proceedings of ICML Workshop on Unsupervised and Transfer Learning [Internet]. JMLR Workshop and Conference Proceedings; 2012 [cited 2021 Dec 22]. p. 37–49. Available from: https://proceedings.mlr.press/v27/baldi12a.html

17. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006 Jul 28;313(5786):504–7.

18. Wang W, Huang Y, Wang Y, Wang L. Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2014. p. 496–503.

19. Wang Y, Yao H, Zhao S. Auto-encoder based dimensionality reduction. Neurocomputing. 2016 Apr 5;184:232–42.

20. Radmanesh M, Rezaei AA, Jalili M, Hashemi A, Goudarzi MM. Online spike sorting via deep contractive autoencoder. Neural Netw [Internet]. 2022 Aug 5 [cited 2022 Aug 11]; Available from: https://www.sciencedirect.com/science/article/pii/S089360802200301X

21. Moradi Fard M, Thonet T, Gaussier E. Deep *k*-Means: Jointly clustering with *k*-Means and learning representations. Pattern Recognit Lett. 2020 Oct 1;138:185–92.

22. Guo X, Gao L, Liu X, Yin J. Improved Deep Embedded Clustering with Local Structure Preservation. 2017;1753–9.

23. Leiber C, Bauer LGM, Neumayr M, Plant C, Böhm C. The DipEncoder: Enforcing Multimodality in Autoencoders. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining [Internet]. New York, NY, USA: Association for Computing Machinery; 2022 [cited 2025 Feb 9]. p. 846–56. (KDD ’22). Available from: https://dl.acm.org/doi/10.1145/3534678.3539407

24. Leiber C, Bauer LGM, Schelling B, Böhm C, Plant C. Dip-based Deep Embedded Clustering with k-Estimation. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining [Internet]. New York, NY, USA: Association for Computing Machinery; 2021 [cited 2025 Feb 9]. p. 903–13. (KDD ’21). Available from: https://dl.acm.org/doi/10.1145/3447548.3467316

25. Miklautz L, Bauer LGM, Mautz D, Tschiatschek S, Böhm C, Plant C. Details (Don’t) Matter: Isolating Cluster Information in Deep Embedded Spaces. In 2021 [cited 2025 Feb 10]. p. 2826–32. Available from: https://www.ijcai.org/proceedings/2021/389

26. Song C, Liu F, Huang Y, Wang L, Tan T. Auto-encoder Based Data Clustering. In: Ruiz-Shulcloper J, Sanniti di Baja G, editors. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Berlin, Heidelberg: Springer; 2013. p. 117–24.

27. Mautz D, Plant C, Böhm C. Deep Embedded Cluster Tree. In: 2019 IEEE International Conference on Data Mining (ICDM) [Internet]. 2019 [cited 2025 Feb 10]. p. 1258–63. Available from: https://ieeexplore.ieee.org/abstract/document/8970987

28. Mautz D, Plant C, Böhm C. DeepECT: The Deep Embedded Cluster Tree. Data Sci Eng. 2020 Dec 1;5(4):419–32.

29. Yang B, Fu X, Sidiropoulos ND, Hong M. Towards K-means-friendly spaces: simultaneous deep learning and clustering. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. Sydney, NSW, Australia: JMLR.org; 2017. p. 3861–70. (ICML’17).

30. Kimura M. AutoClustering: A Feed-Forward Neural Network Based Clustering Algorithm. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW) [Internet]. 2018 [cited 2025 May 2]. p. 659–66. Available from: https://ieeexplore.ieee.org/document/8637379

31. Hartigan JA, Hartigan PM. The Dip Test of Unimodality. Ann Stat. 1985 Mar;13(1):70–84.

32. Ren Y, Wang N, Li M, Xu Z. Deep density-based image clustering. Knowl-Based Syst. 2020 Jun 7;197:105841.

33. Buzsáki G. Rhythms of the Brain [Internet]. New York: Oxford University Press; 2006 [cited 2021 Dec 8]. 464 p. Available from: https://oxford.universitypressscholarship.com/10.1093/acprof:oso/9780195301069.001.0001/acprof-9780195301069

34. Lewicki MS. A review of methods for spike sorting: the detection and classification of neural action potentials. Netw Bristol Engl. 1998 Nov;9(4):R53-78.

35. Bear M, Connors B, Paradiso M. Neuroscience: Exploring the brain: Fourth edition. 2015. 1 p.

36. Eom J, Park IY, Kim S, Jang H, Park S, Huh Y, et al. Deep-learned spike representations and sorting via an ensemble of auto-encoders. Neural Netw. 2021 Feb 1;134:131–42.

37. Meyer LM, Zamani M, Rokai J, Demosthenous A. Deep learning-based spike sorting: a survey. J Neural Eng. 2024 Nov;21(6):061003.

38. Dimensionality reduction: a comparative review | BibSonomy [Internet]. [cited 2022 Aug 11]. Available from: https://www.bibsonomy.org/bibtex/2ed03568f0e9bca9cdaf6b25304e55940/peter.ralph

39. Meister M, Pine J, Baylor DA. Multi-neuronal signals from the retina: acquisition and analysis. J Neurosci Methods. 1994 Jan 1;51(1):95–106.

40. Pouzat C, Mazor O, Laurent G. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. J Neurosci Methods. 2002 Dec 31;122(1):43–57.

41. Litke AM, Bezayiff N, Chichilnisky EJ, Cunningham W, Dabrowski W, Grillo AA, et al. What does the eye tell the brain?: Development of a system for the large-scale recording of retinal output activity. IEEE Trans Nucl Sci. 2004 Aug;51(4):1434–40.

42. Hulata E, Segev R, Ben-Jacob E. A method for spike sorting and detection based on wavelet packets and Shannon’s mutual information. J Neurosci Methods. 2002 May 30;117(1):1–12.

43. Chung JE, Magland JF, Barnett AH, Tolosa VM, Tooker AC, Lee KY, et al. A fully automated approach to spike sorting. Neuron. 2017 Sep 13;95(6):1381-1394.e6.

44. Ebbesen CL, Reifenstein ET, Tang Q, Burgalossi A, Ray S, Schreiber S, et al. Cell Type-Specific Differences in Spike Timing and Spike Shape in the Rat Parasubiculum and Superficial Medial Entorhinal Cortex. Cell Rep. 2016 Jul 14;16(4):1005–15.

45. Adamos DA, Kosmidis EK, Theophilidis G. Performance evaluation of PCA-based spike sorting algorithms. Comput Methods Programs Biomed. 2008 Sep 1;91(3):232–44.

46. Mishra S, Sarkar U, Taraphder S, Datta S, Swain D, Saikhom R, et al. Principal Component Analysis. Int J Livest Res. 2017 Jan 1;1.

47. Toosi R, Akhaee MA, Dehaqani MRA. An automatic spike sorting algorithm based on adaptive spike detection and a mixture of skew-t distributions. Sci Rep. 2021 Jul 6;11(1):13925.

48. Glaser EM, Marks WB. ON-LINE SEPARATION OF INTERLEAVED NEURONAL PULSE SEQUENCES. In: Enslein K, editor. Data Acquisition and Processing in Biology and Medicine [Internet]. Pergamon; 1968 [cited 2022 Aug 11]. p. 137–56. Available from: https://www.sciencedirect.com/science/article/pii/B9780080035437500124

49. Abeles M, Goldstein MH. Multispike train analysis. Proc IEEE. 1977 May;65(5):762–73.

50. Hyvärinen A. Independent component analysis: recent advances. Philos Transact A Math Phys Eng Sci. 2013 Feb 13;371(1984):20110534.

51. Tiganj Z, Mboup M. Neural spike sorting using iterative ICA and a deflation-based approach. J Neural Eng. 2012 Dec;9(6):066002.

52. Lopes MV, Aguiar E, Santana E, Santana E, Barros AK. ICA feature extraction for spike sorting of single-channel records. In: 2013 ISSNIP Biosignals and Biorobotics Conference: Biosignals and Robotics for Better and Safer Living (BRC). 2013. p. 1–5.

53. Tenenbaum JB, de Silva V, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science. 2000 Dec 22;290(5500):2319–23.

54. Zhou H, Wang F, Tao P. t-Distributed Stochastic Neighbor Embedding Method with the Least Information Loss for Macromolecular Simulations. J Chem Theory Comput. 2018 Nov 13;14(11):5499–510.

55. MacQueen J. Some methods for classification and analysis of multivariate observations. Proc Fifth Berkeley Symp Math Stat Probab Vol 1 Stat. 1967 Jan 1;5.1:281–98.

56. Leiber C, Miklautz L, Plant C, Böhm C. Benchmarking Deep Clustering Algorithms With ClustPy. In: 2023 IEEE International Conference on Data Mining Workshops (ICDMW) [Internet]. 2023 [cited 2025 Feb 10]. p. 625–32. Available from: https://ieeexplore.ieee.org/document/10411702

57. Bengio Y, Lamblin P, Popovici D, Larochelle H, Montreal U. Greedy layer-wise training of deep networks. Vol. 19, Advances in Neural Information Processing Systems. 2007.

58. Xie J, Girshick R, Farhadi A. Unsupervised deep embedding for clustering analysis. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48. New York, NY, USA: JMLR.org; 2016. p. 478–87. (ICML’16).

59. Jiang Z, Zheng Y, Tan H, Tang B, Zhou H. Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. 2017;1965–72.

60. McConville R, Santos-Rodríguez R, Piechocki RJ, Craddock I. N2D: (Not Too) Deep Clustering via Clustering the Local Manifold of an Autoencoded Embedding. In: 2020 25th International Conference on Pattern Recognition (ICPR) [Internet]. 2021 [cited 2025 Feb 10]. p. 5145–52. Available from: https://ieeexplore.ieee.org/document/9413131

61. Rendón E, Abundez I, Arizmendi A, Quiroz EM. Internal versus External cluster validation indexes. 2011;5(1):8.

62. Veerabhadrappa R, Ul Hassan M, Zhang J, Bhatti A. Compatibility Evaluation of Clustering Algorithms for Contemporary Extracellular Neural Spike Sorting. Front Syst Neurosci [Internet]. 2020 [cited 2022 Jul 18];14. Available from: https://www.frontiersin.org/articles/10.3389/fnsys.2020.00034

63. Salganicoff M, Sarna M, Sax L, Gerstein GL. Unsupervised waveform classification for multi-neuron recordings: a real-time, software-based system. I. Algorithms and implementation. J Neurosci Methods. 1988 Oct;25(3):181–7.

64. Caro-Martín CR, Delgado-García JM, Gruart A, Sánchez-Campusano R. Spike sorting based on shape, phase, and distribution features, and K-TOPS clustering with validity and error indices. Sci Rep. 2018 Dec 12;8(1):17796.

65. Hubert L, Arabie P. Comparing partitions. J Classif. 1985 Dec 1;2(1):193–218.

66. Vinh NX, Epps J, Bailey J. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. :18.

67. Steinley D. Properties of the Hubert-Arable Adjusted Rand Index. Psychol Methods. 2004;9(3):386–96.

68. Fowlkes EB, Mallows CL. A Method for Comparing Two Hierarchical Clusterings. J Am Stat Assoc. 1983;78(383):553–69.

69. Strehl A, Ghosh J. Cluster Ensembles --- A Knowledge Reuse Framework for Combining Multiple Partitions. J Mach Learn Res. 2002;3(Dec):583–617.

70. Lazarenko D, Bonald T. Pairwise Adjusted Mutual Information. 2021.

71. Vinh N, Epps J, Bailey J. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? ICML. 2009. 135 p.

72. Rosenberg A, Hirschberg J. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In 2007. p. 410–20.

73. Caliński T, JA H. A Dendrite Method for Cluster Analysis. Commun Stat - Theory Methods. 1974 Jan 1;3:1–27.

74. Davies DL, Bouldin DW. A Cluster Separation Measure. IEEE Trans Pattern Anal Mach Intell. 1979 Apr;PAMI-1(2):224–7.

75. Halkidi M, Batistakis Y, Vazirgiannis M. On Clustering Validation Techniques. J Intell Inf Syst. 2001 Dec 1;17(2):107–45.

76. Rousseeuw PJ. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math. 1987 Nov 1;20:53–65.

77. Wegier W, Ksieniewicz P. Application of Imbalanced Data Classification Quality Metrics as Weighting Methods of the Ensemble Data Stream Classification Algorithms. Entropy Basel Switz. 2020 Jul 31;22(8):E849.

78. Sun Y, Wong AKC, Kamel MS. Classification of imbalanced data: a review. Int J Pattern Recognit Artif Intell. 2009 Jun;23(04):687–719.

79. Joshi MV, Kumar V, Agarwal RC. Evaluating boosting algorithms to classify rare classes: comparison and improvements. In: Proceedings 2001 IEEE International Conference on Data Mining. 2001. p. 257–64.

80. Weiss GM. Mining with rarity: a unifying framework. ACM SIGKDD Explor Newsl. 2004 Jun;6(1):7–19.

81. Dipalo M, Amin H, Lovato L, Moia F, Caprettini V, Messina G, et al. Intracellular and Extracellular Recording of Spontaneous Action Potentials in Mammalian Neurons and Cardiac Cells with 3D Plasmonic Nanoelectrodes. Nano Lett. 2017 May 23;17.

82. Dwork C, Kumar R, Naor M, Sivakumar D. Rank aggregation methods for the Web. In: Proceedings of the 10th international conference on World Wide Web [Internet]. New York, NY, USA: Association for Computing Machinery; 2001 [cited 2022 Dec 6]. p. 613–22. (WWW ’01). Available from: https://doi.org/10.1145/371920.372165