A data-driven spike sorting feature map for resolving spike overlap in the feature space

J. Wouters, F. Kloosterman, A. Bertrand

***Abstract*—Objective: Spike sorting is the process of extracting neuronal action potentials, or spikes, from an extracellular brain recording, and assigning each spike to its putative source neuron. Spike sorting is usually treated as a clustering problem. However, this clustering process is known to be affected by overlapping spikes. Existing methods for resolving spike overlap typically require an expensive post-processing of the clustering results. In this paper, we propose the design of a domain-specific feature map, which enables the resolution of spike overlap directly in the feature space.**

Approach: The proposed domain-specific feature map is based on a neural network architecture that is trained to simultaneously perform spike sorting and spike overlap resolution. Overlapping spikes clusters can be identified in the feature space through a linear relation with the single-neuron clusters for which the neurons contribute to the overlapping spikes. To aid the feature map training, a data augmentation procedure is presented that is based on biophysical simulations.

Main results: We demonstrate the potential of our method on independent and realistic test data. We show that our novel approach for resolving spike overlap generalizes to unseen and realistic test data. Furthermore, the sorting performance of our method is shown to be similar to the state-of-the-art, but our method does not assume the availability of spike templates for resolving spike overlap.

Significance: Resolving spike overlap directly in the feature space, results in an overall simplified spike sorting pipeline compared to the state-of-the-art. For the state-of-the-art, the overlapping spike snippets exhibit a large spread in the feature space and do not appear as concentrated clusters. This can lead to biased spike template estimates which affect the sorting performance of the state-of-the-art. In our proposed approach, overlapping spikes form concentrated clusters and spike overlap resolution does not depend on the availability of spike templates.

A conference precursor of this paper has been published in [1]. This work was carried out at the ESAT Laboratory of KU Leuven. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 802895) and the Research Foundation Flanders (FWO) project FWO G0D7516N. This research received funding from the Flemish Government (AI Research Program). A. Bertrand and J. Wouters are affiliated to KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics and Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium. F. Kloosterman is affiliated to Neuro-Electronics Research Flanders (NERF), Leuven, Belgium, KU Leuven, Brain & Cognition Research Unit, Belgium and VIB, Leuven, Belgium. The scientific responsibility is assumed by its authors.

I. Introduction

Neurons communicate with each other through electrochemical signalling mechanisms. Such neuronal signalling is believed to underlie cognition. In order to better understand the brain and how it gives rise to cognition, neuroscientists perform experiments that are designed to measure neural activity and relate this activity to behavioural data. When performing electrophysiological experiments, the electrical component in neural signalling is studied. Neuronal action potentials, or "spikes", are a frequently studied component of the electrical brain signals because they reflect the output signals of neurons.

When performing extracellular electrophysiological recordings, electrodes are lowered into the brain near the neuron population that is under study. Modern neural probes contain a dense grid of hundreds of recording electrodes [2] [3] and are becoming increasingly popular for performing large-scale neural recordings. Neural probes can record a mixture of spikes from hundreds of neurons simultaneously. The spiking activity at each electrode channel is referred to as multi-unit activity as each electrode typically capture spikes from multiple neurons. However, when an experimenter is interested in the neural activity of individual neurons, i.e., single-unit activity, a neural source separation problem has to be solved. The transformation of the activity of multiple neurons into the individual activity of the different recorded neurons is known as spike sorting [4]. While multi-unit activity is often sufficient for coarse brain-computer interfacing tasks, having access to the neural activity of individual neurons is essential to neuroscientists that try to understand how individual neurons contribute to brain function [5] [6] [7].

A common approach for solving the spike sorting problem is through the use of an unsupervised learning framework [8]. The framework is characterized by four consecutive processing steps: 1) spike detection, 2) spike alignment, 3) feature extraction, and, 4) clustering. The spike detection results in

the extraction of spike snippets from the neural recording. These spike snippets are then mutually aligned according to some alignment criterion to support the feature extraction. Next, feature values are computed for every spike snippet, which ideally results in the discriminability of spikes from different neurons, while promoting the grouping of spikes from the same neuron. Finally, the single-neuron clusters are identified through an automated clustering analysis. Several implementations and variations on this common framework have been proposed [9]–[15].

When two or more neurons fire near-simultaneously in time, a compound spike waveform, also referred to as an overlapping spike snippet, is generated. If the vanilla clustering framework from the previous paragraph is used for spike sorting, the overlapping spike snippets are likely going to be misinterpreted, leading to wrongly clustered spikes and an overall bias in the cluster identification itself. It has been shown that overlapping spikes are affecting spike sorting accuracy in practice [16]. Therefore, strategies have been designed to cope with overlapping spikes. To the best of our knowledge these strategies all depend on the use of spike templates [17]. The strategies consist of either applying matched filters that are derived from the spike templates following the biased clustering analysis [18]–[23], or rely on an iterative clustering/template fitting procedure to resolve spike overlap [16] [24] [25]. Spike template estimates will also be biased by overlapping spikes when they are based on biased clustering results. In case of the iterative template estimation, the template estimation bias can be overcome, but only at a considerable increase in computational cost.

Recently, modern supervised machine learning concepts such as (deep) neural networks have become increasingly popular for use in spike sorting [26]–[30]. These supervised learning building blocks are typically intended for replacing classical spike detection and feature extraction methods, resulting in domain-specific detectors and feature maps. However, these recent feature maps, as well as preceding classical techniques, do not take into account spike overlap and overlap resolution mechanisms. In this work we propose the use of a neural network spike sorting feature map that is capable of resolving spike overlap directly in the feature space. In this way, the clustering bias due to spike overlap is overcome. Through the use of this specialized feature map, there is no need for matched filter post-processors, nor for iterative fitting techniques to resolve spike overlap, such that a minimal processing pipeline is delivered that is capable of resolving spike overlap.

In Section II we formalize the classical spike sorting process and show how overlapping spikes hamper the sorting performance. Then, the domain-specific feature map that is designed for resolving spike overlap is presented in Section III. In Section IV we explore the design space of the proposed feature map and quantify the feature map spike sorting performance on realistic ground truth data. Finally, we conclude the presented work in Section V.

II. Spike sorting and overlapping spikes

Consider $\mathbf{x}[k] \in \mathbb{R}^N$ to be the high-pass filtered extracellular recording at time sample k as measured simultaneously on N recording electrodes. A single neuronal spike contributes to several consecutive time samples of the extracellular recording when using typical sampling frequencies (25-30 kHz). To facilitate the study of the neuronal spikes, we define the stacked vector $\bar{\mathbf{x}}[k] = \left[\mathbf{x}[k-L+1]^T \ \ldots \ \mathbf{x}[k-1]^T \ \mathbf{x}[k]^T\right]^T \in \mathbb{R}^{NL}$ to represent a spatio-temporal window on the extracellular recording of N channels by L samples. The bar notation $\bar{\blacksquare}$ is also used in combination with other symbols to indicate a similar delay-line extension on the data that is represented by that symbol.

A spike sorting algorithm takes such an extracellular recording $\mathbf{x}[k]$ at its input to output the sample times at which the individual neurons embedded in the recording generate spikes. The output thus consists of a set collection $\left\{\hat{\mathcal{S}}_n \ \forall n\right\}$, where an individual set $\hat{\mathcal{S}}_n$ contains the spike time estimates (i.e., the spike train) of the sortable neuron n. The different spike waveforms as generated by a specific neuron are very similar throughout time, which is why clustering techniques are commonly used in spike sorting. A typical clustering approach for transforming the input recording into the spike sorting output consists of the following sequential processing steps [8]:

1) **detection**: Extract a set of unsorted spike times $\mathcal{U}$ from the extracellular recording. Although several detection approaches are in use, usually they involve an amplitude thresholding operation that can be optionally preceded by a preemphasis filtering.
2) **alignment**: Given the set of unsorted spike times $\mathcal{U}$, every spike time is adjusted such that the corresponding spike waveforms are mutually aligned in their L samples wide temporal window according to some alignment criterion. A popular alignment criterion is to place the local spike minimum in the middle of the temporal window. This processing step results in a set of aligned, yet unsorted spike time $\mathcal{U}_{\|}$.

3) **feature extraction**: Given the aligned spike times, a set of features $\mathcal{F}$ is derived from the spike waveform as follows:

$$\mathcal{F} = \{\ f(\bar{\mathbf{x}}[u])\ \ \forall u \in \mathcal{U}_{\|}\ \},$$

with $f : \mathbb{R}^{NL} \rightarrow \mathbb{R}^{M}$ the feature map to the M-dimensional feature space. A popular choice of feature map for spike sorting is based on a principal component analysis (PCA), which results in a linear feature map [8].

4) **cluster analysis**: Finally, a cluster analysis is performed on $\mathcal{F}$, where the spike feature projections, which are also referred to as spike embeddings, of the same neuron are supposed to cluster together, and clusters from different neurons are separable in that feature space. Based on the clustering in $\mathcal{F}$, the original space $\mathcal{U}_{\|}$ can be partitioned into $\left\{\hat{\mathcal{S}}_n\ \forall n \in \hat{\mathcal{N}},\ \mathcal{S}_{\text{unsorted}}\right\}$, where $\hat{\mathcal{N}}$ is the set of sortable neurons which typically results from a manual curation of the clustering results and $\mathcal{S}_{\text{unsorted}}$ is a 'garbage' set which contains all multi-neuron and noise clusters that could not be assigned to a single neuron.

Ideally, $\mathcal{S}_{\text{unsorted}}$ is empty. However, in practice, $\mathcal{S}_{\text{unsorted}}$ usually contains a significant fraction of the detected spike times. A well-known problem that hampers the spike sorting process is the occurrence of spike overlap. To illustrate this problem consider the following model for the extracellular recording:

$$\mathbf{x}[k] = \sum_{n \in \mathcal{N}} \sum_{t \in \mathcal{S}_n} \alpha_{n,t} \mathbf{s}_n[k-t] + \mathbf{n}[k], \tag{1}$$

where $\mathbf{s}_n$ is the finite-support prototypical spike waveform of neuron n, also referred to as the spike template of neuron n. The spike template $\mathbf{s}_n$ is assumed to have W_n non-zero elements centered around $k = 0$. Furthermore, $\alpha_{n,t}$ is a template scaling factor to model amplitude variations and $\mathbf{n}$ is defined to be an additive noise component that accounts for both the neural and electrical noise. We say spike overlap occurs between a neuron i and j if $\exists\,(t_i, t_j) \in \mathcal{S}_i \times \mathcal{S}_j$: $|t_i - t_j| < \left\lfloor \frac{W_i + W_j}{2} \right\rfloor$, i.e., their respective spike templates overlap and are superimposed in $\mathbf{x}$.

Consider now a recording sample in which the spiking activity of two neurons i and j is present. For the sake of intelligibility, let us first consider a model of spike overlap where $|t_i - t_j| = 0$ for the spike times at which overlap occurs, i.e., all overlapping spikes consist of the sum of perfectly aligned single-neuron templates. For this scenario, a linear feature map f (e.g. the commonly used PCA feature map) has the following interesting linearity property:

$$f(\bar{\mathbf{s}}_i + \bar{\mathbf{s}}_j) = f(\bar{\mathbf{s}}_i) + f(\bar{\mathbf{s}}_j), \tag{2}$$

where $\bar{\mathbf{s}}_i$ and $\bar{\mathbf{s}}_j \in \mathbb{R}^{NL}$ are the mutually aligned spatio-temporal spike templates of neuron i and j, respectively. This property enables the resolution of these overlapping spikes in the feature space: if the sum of two cluster centroids maps onto the centroid of a third cluster, this third cluster is believed to contain the overlapping spikes that are composed of the sum of the aligned templates associated with the two former clusters. The overlap resolution property is illustrated in Fig. 1.

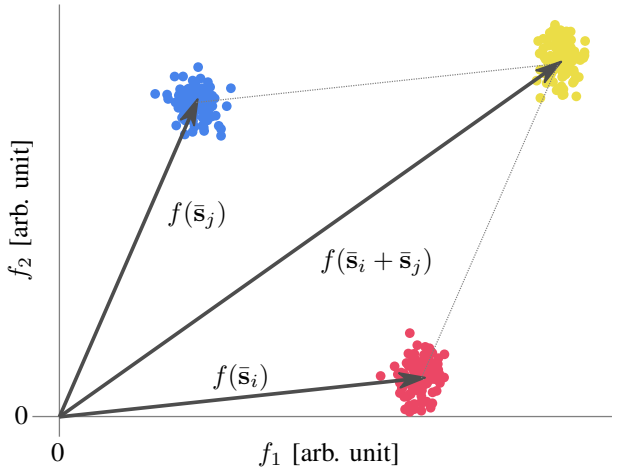

Fig. 1. Three clusters are distinguishable in the feature space. Both the red and blue clusters consist of single-neuron spike projections. The yellow cluster consists of projected overlapping spikes that are composed of aligned single-neuron spikes of both the red and blue cluster. For the case of perfectly aligned overlapping spikes and a linear feature map, spike overlap can be resolved by establishing relationships between clusters through the vector addition of cluster centroids that potentially map on overlapping spikes clusters.

Unfortunately, the assumption of perfect spike alignment is of little practical use. Although neurons might exist for which the alignment assumption holds, e.g., when such neurons are coupled through gap junctions, general spike overlap is believed to occur by chance because two or more neurons happen to fire near-simultaneously in time. When such random overlap happens for two neurons i and j, $t_i - t_j$ results from a uniform random distribution. To concisely represent such randomly aligned spike overlap we introduce the following notation:

$$\bar{\mathbf{s}}_i[k - t_i] + \bar{\mathbf{s}}_j[k - t_j] = \bar{\mathbf{s}}_i \oplus \bar{\mathbf{s}}_j, \tag{3}$$

where $\oplus$ denotes the random-shift-and-sum operator. The random time shift between spikes introduces a non-linearity that leads to the smearing of the overlapping spikes cluster when using a linear feature map, i.e., $f(\bar{\mathbf{s}}_i \oplus \bar{\mathbf{s}}_j) \neq f(\bar{\mathbf{s}}_i) + f(\bar{\mathbf{s}}_j)$. Such cluster smearing is shown in Fig. 2. This smearing

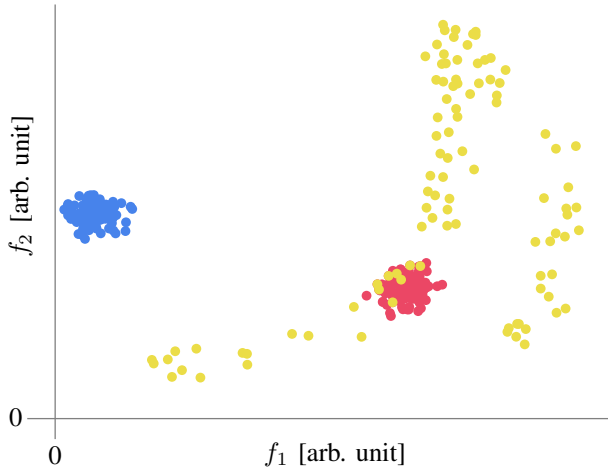

Fig. 2. The red and blue clusters consists of single-neuron spike projections. The projections of overlapping spikes with random alignment are shown in yellow and are smeared out in feature space, which prevents the overlapping spikes cluster from being recovered. Note that the colors in this figure are assigned based on ground-truth information and not based on an actual clustering analysis.

prevents a reliable recovery of the clusters when compared to Fig. 1. Without well-defined clusters, the resolution of overlapping spikes in the feature space becomes infeasible. Therefore, in the remainder of this work, we will investigate the use of a non-linear feature map g that is intended to behave as if it were a linear function applied to a linear combination:

$$g(\bar{\mathbf{s}}_i \oplus \bar{\mathbf{s}}_j) = g(\bar{\mathbf{s}}_i) + g(\bar{\mathbf{s}}_j), \tag{4}$$

such that the feature embedding has similar properties as the embedding presented in Fig. 1, but now also in case of overlapping spikes with random alignment.

III. Overlap resolving feature map design

The design of the feature map is approached as a supervised machine learning problem. Because spike sorting is inherently unsupervised, the feature map is envisioned to be applied in a train once, apply "anywhere" fashion. However, the meaning of anywhere is limited to the usage of a fixed probe geometry and spatio-temporal window, as will become apparent from the details in this section.

Given its supervised nature, the design of the feature map can be broken down into the following components:

- neural network architecture selection,
- overlap resolving cost function design,
- training data generation and augmentation process, and,
- training procedure.

In the following sections we will go into more detail on each of these design components.

A. Neural network architecture selection

In this work we focus on the use of a multi-layer perceptron (MLP) neural network architecture for the non-linear feature map $g : \mathbb{R}^{NL} \to \mathbb{R}^{M}$. Although other function families exist, the MLP with a single hidden layer is known to be a universal function approximator [31], meaning that it can fit any continuous function up to an error that can be made arbitrarily small by increasing the dimensionality of the hidden layer. By incorporating additional hidden layers in the network, the total number of artificial neurons that are needed to achieve a certain error bound, can be drastically reduced [32].

The deep neural network architecture that is used in this work is shown in Fig. 3. The neural network takes an NL-dimensional vector as input. The network contains three fully-connected hidden layers with $2NL$, NL and $\frac{1}{5}NL$ hidden artificial neurons, respectively. Because of the fixed network input dimensionality, we assume that the sampling frequency of the neural recordings is constant throughout both the training and actual spike sorting phases. The hidden neurons use rectified linear unit (ReLU) activation functions [33]. The output consists of M artificial neurons with linear activation, where M is the desired dimension of the feature space. For training regularization purposes we have included a dropout layer [34] before every neuron layer. To complement the dropout layer, the weights incident to each artificial neuron are constrained to not exceed a specified maximum norm as also proposed in [34]. The output of the hidden layers is subjected to an optional batch normalization [35]. The middle hidden layer is marked optional as well. Here, by optional we mean that whether or not the optional feature is used, is considered a hyperparameter. The hyperparameter exploration will be discussed in Section IV.

B. Cost function

The model selection, as presented in the previous section, only results in an architectural skeleton. In order to apply the neural network g to recording data, the appropriate function coefficients have to be chosen, leading to a concrete realization of g.

In a typical supervised learning task, the coefficients of the network are chosen to enable classification or regression. To obtain such behaviour, the class labels or dependent variables values $\mathbf{y}_k$ are required during training for all $k \in \mathcal{T}$, with $\mathcal{T}$ the set of training samples. A realization of $\hat{g}$ is then

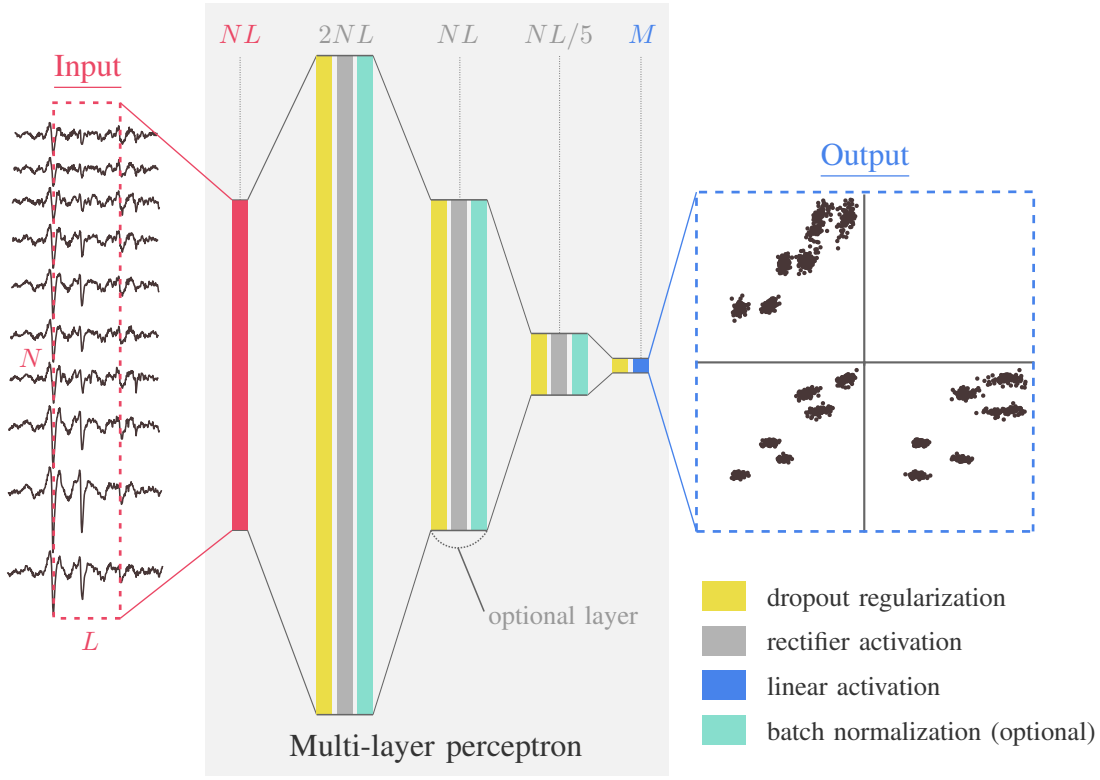

Fig. 3. Schematic representation of the fully connected MLP neural network architecture as used in this work. The network accepts an NL-dimensional vector at its input (red). This input vector is transformed to an M-dimensional output vector (blue) via a chained projection onto three consecutive hidden layers, prior to a projection onto the output neurons. The dimensionality of the hidden layers ordered from input to output is $2NL$, NL and $\frac{1}{5}NL$, respectively. Both the hidden layers and the output layer contain a dropout layer (yellow) for training regularization purposes. The hidden neurons make use of a non-linear rectifier activation function (gray). The use of batch normalization (green) is explored in this work. The second hidden layer is marked as optional because we will explore networks both with and without this layer.

obtained by tuning the coefficients until the (local) minimum of a certain cost function is obtained:

$$\hat{g} = \arg\min_{g} \sum_{k \in \mathcal{T}} \mathcal{L}\left(g\left(\bar{\mathbf{x}}_k\right), \mathbf{y}_k\right), \tag{5}$$

where $\mathcal{L}$ is a training sample loss function, e.g., a least squares criterion.

Irrespective of the choice of loss, the classical approach to supervised learning with explicit training labels $\mathbf{y}_k$ is not suitable for use in spike sorting because of its inherent unsupervised nature. Therefore, we propose the use of a cost function where the training labels are used implicitly, i.e., the network is not forced to produce a specific output, rather it is instructed to learn an output representation which behaves according to that cost function. The cost function that is proposed in this work consists of four distinct terms that each have a specific intended purpose. In what follows, the different terms will be introduced one-by-one.

The loss function that is associated with the first term of the cost acts on three spatio-temporal spike waveforms and is given by

$$\mathcal{L}_1\left(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j, \bar{\mathbf{x}}_{i\oplus j}\right) = \left\| g(\bar{\mathbf{x}}_i) + g(\bar{\mathbf{x}}_j) - g(\bar{\mathbf{x}}_{i\oplus j}) \right\|_2^2. \tag{6}$$

with a slight abuse of notation, let $\bar{\mathbf{x}}_i = \bar{\mathbf{s}}_i + \bar{\mathbf{n}}$ and $\bar{\mathbf{x}}_j = \bar{\mathbf{s}}_j + \bar{\mathbf{n}}$ be a training sample spike of neuron i and j, respectively, each with their own random noise component. $\bar{\mathbf{x}}_{i\oplus j} = \bar{\mathbf{s}}_i \oplus \bar{\mathbf{s}}_j + \bar{\mathbf{n}}$ is a training sample representing two overlapping spikes involving neuron i and j. Minimizing $\mathcal{L}_1$ over g has the potential to results in a feature map g that *behaves linearly for non-aligned overlapping spikes* as described by (4).

Minimizing only $\mathcal{L}_1$, will result in the trivial solution $g(\bar{\mathbf{x}}) = \mathbf{0}\ \forall \bar{\mathbf{x}}$. To obtain a non-trivial feature map g, which is suitable for clustering, $\mathcal{L}_1$ is combined with a second loss term $\mathcal{L}_2$:

$$\mathcal{L}_2\left(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j\right) = \exp\left(-\left\| g(\bar{\mathbf{x}}_i) - g(\bar{\mathbf{x}}_j) \right\|_2^2\right). \tag{7}$$

When minimizing $\mathcal{L}_2$ over g, it will favour realizations of g that result in a large distance in the feature space between a pair of spikes from different neurons. As such, this term is intended to maximize the *between-cluster distance*.

To promote the clustering of spikes from the same neuron, $\mathcal{L}_3$ is defined as

$$\mathcal{L}_3(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}'_i) = \|g(\bar{\mathbf{x}}_i) - g(\bar{\mathbf{x}}'_i)\|_2^2 , \tag{8}$$

where $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{x}}'_i$ are distinct spike samples from the same neuron. As such this term serves to *minimize the within-cluster distance* for the given neuron.

The compound cost function used in this work is composed from the above three loss functions as follows:

$$\begin{aligned}\sum_{\mathcal{T}} \mathcal{L}_1\left(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j, \bar{\mathbf{x}}_{i\oplus j}\right) + \; & \kappa_1 \mathcal{L}_2\left(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j\right) + \\ & \kappa_2 \left(\mathcal{L}_3(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_i^{\sigma}) + \mathcal{L}_3(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_j^{\sigma})\right) + \\ & \kappa_3 \left(\mathcal{L}_3(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_i^{\delta}) + \mathcal{L}_3(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_j^{\delta})\right),\end{aligned} \tag{9}$$

where the sum operator $\sum$ acts on the training set $\mathcal{T}$. A single entry of the training set is a collection of training example spikes $\left\{\bar{\mathbf{x}}_i,\ \bar{\mathbf{x}}_j,\ \bar{\mathbf{x}}_{i\oplus j},\ \bar{\mathbf{x}}_i^{\sigma},\ \bar{\mathbf{x}}_j^{\sigma},\ \bar{\mathbf{x}}_i^{\delta},\ \bar{\mathbf{x}}_j^{\delta}\right\}$ from a pair of training neurons (i, j) with $i \neq j$. Note that for a given pair of neurons, multiple distinct entries can be present in the training set. In this work the neuron identities, e.g., i and j, serve as implicit labels, i.e., the cost function describes how the feature map should behave for spikes of different neurons, but it does not push the output to a predefined value. $\bar{\mathbf{x}}^{\sigma}$ represents an amplitude scaled version of $\bar{\mathbf{x}}$. As such, the terms that involve $\bar{\mathbf{x}}^{\sigma}$ (i.e. the terms weighted with κ_2) serve to penalize spike amplitude scaling variance in the desired feature space. $\bar{\mathbf{x}}^{\delta}$ represent a spike snippet that is superimposed with a spike waveform of a different neuron that reaches its maximum energy on another channel. As discussed in the following section, neurons that reach their maximum spike energy on a different channel are to be treated in separate clustering problems (cf. divide-and-conquer clustering). Therefore, the superimposed spike waveform is to be treated as physiological noise and feature robustness against such noise is desirable. More details on the specifics for $\bar{\mathbf{x}}^{\sigma}$ and $\bar{\mathbf{x}}^{\delta}$ are given in the next section. κ_1, κ_2 and κ_3 are hyper parameters that allow for an explicit trade-off between (a) the linear behavior within the feature space[1] , (b) the between-cluster distance and (c) the within-cluster distance. Note that $\mathcal{L}_1$ does not have an explicit hyper parameter because its importance can be controlled relative to the other terms through κ_1, κ_2 and κ_3.

[1]Linearity *within* the feature space does not imply a linear behavior between the input space and the feature space. Indeed, an amplitude scaling of a spike will not result in an equal scaling of its feature vector, which is even discouraged due to the κ_3 term minimizing the within-cluster distance.

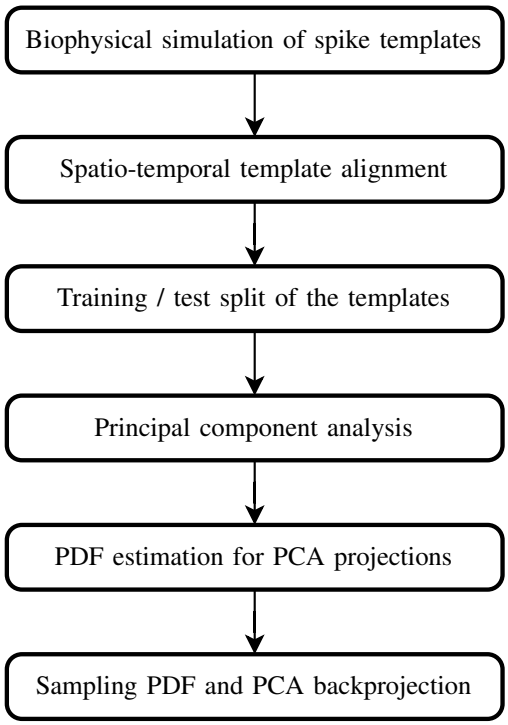

Fig. 4. Flowchart that contains the different steps that are involved for the generation of training data.

C. Training data

Now that we have defined the structure of g, as well as a criterion to optimally choose the coefficients of g, we need to construct the actual training set $\mathcal{T}$ to perform the training. Fig. 4 depicts the different steps that are involved in the training data generation process. These steps will be discussed in detail in the remainder of this section.

In this work we make use of a data augmentation approach for the generation of the training data. First, 634 spike templates $\bar{\mathbf{s}}_n \in \mathbb{R}^{N2L}$ from 634 distinct biophysically detailed neuron models [36] are simulated using MEArec [37]. Note that for these templates a temporal window of $2L$ is used rather than L. This is done to make sure that the minimum of an overlapping spikes segment with $|s_i - s_j| \neq 0$ can be temporally aligned within an L-window prior to computing the feature map projection. In this work, the probe geometry is equivalent to a single column of a neuropixel probe [2]. A single-column probe design is chosen here to aid the training data generation process in terms of its spatio-temporal aligment procedure (see below)[2]. The simulated sampling frequency is chosen to be 32 kHz. The temporal window length

[2]Although the proposed feature map design is deemed to be probe-dependent, the feature map design approach is not limited to the probe geometry used in this work, i.e., the generation of training data can be re-done when a different probe geometry is used.

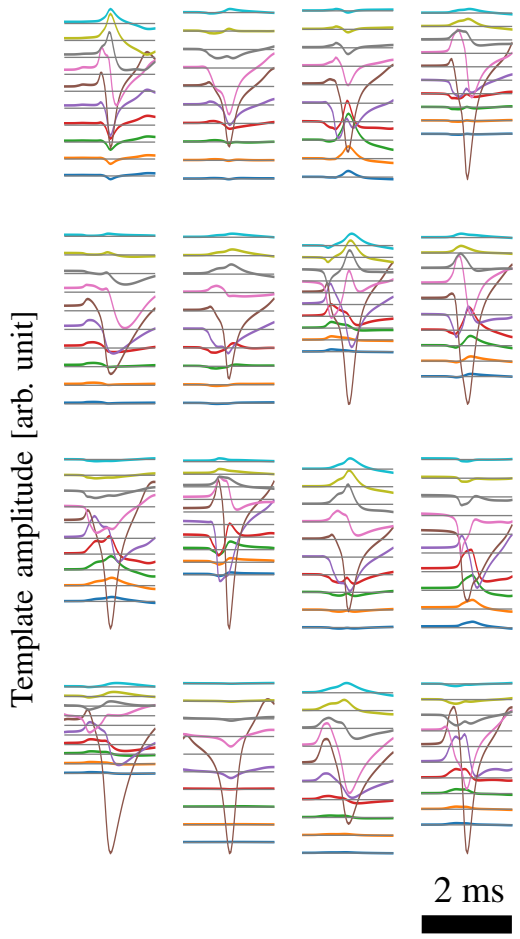

Fig. 5. Sixteen example simulated spatio-temporal spike templates are shown. The waveform as impinging on a specific channel is given a unique color for improved readability. The plotting distance between the channels is constant.

is chosen to be $L = 32$, which corresponds to 1 ms. The minimum of every $\bar{\mathbf{s}}_n$ is positioned at the same sample index through a spatio-temporal alignment procedure. The spatial alignment results in all templates reaching their maximum on the same channel. This might sound limiting, but we envision the feature map to be used in a divide-and-conquer type of spike sorting [38]. In a divide-and-conquer scheme, the clustering analysis is performed for every electrode separately. During this clustering, only spatio-temporal snippets are considered that reach their minimum on the same "dominant" electrode. In such a case N is often taken to be the number of electrodes that make up a local neighbourhood surrounding the dominant electrode. In this work $N = 10$ is used. A trained feature map can be re-used for all local neighbourhoods that have a similar spatial configuration. Such a divide-and-conquer approach has the advantage that it scales linearly with an increasing number of electrodes and is an easy subject for parallelism. One third of the simulated spike templates is randomly selected and excluded from the further steps in the augmentation process. This set of excluded templates are reserved for testing in Section IV-B. We will refer to this set as $\mathcal{S}_{\text{test}}$ The other two thirds, i.e., a total of 423 templates, are further processed to create the augmented training data. We will refer to this set of templates as $\mathcal{S}_{\text{training}}$. Example simulated templates are shown in Fig. 5. Note that these templates will not act as training samples themselves, but instead will form the basis for a synthetic model to generate training data. A principal component analysis is performed on the set of training templates. This requires the computation of the sample covariance matrix over the training templates:

$$\mathbf{R}_{\mathbf{ss}} = \frac{1}{|\mathcal{S}_{\text{training}}|} \sum_{\bar{\mathbf{s}}_n \in \mathcal{S}_{\text{training}}} \left(\bar{\mathbf{s}}_n - \langle \mathcal{S}_{\text{training}} \rangle\right) \left(\bar{\mathbf{s}}_n - \langle \mathcal{S}_{\text{training}} \rangle\right)^T, \tag{10}$$

where $|\mathcal{S}_{\text{training}}|$ indicates the cardinality of $\mathcal{S}_{\text{training}}$ and $\langle \mathcal{S}_{\text{training}} \rangle$ is the mean training template. The principal components are then computed through the eigendecomposition of $\mathbf{R}_{\mathbf{ss}} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$, where the columns of $\mathbf{U}$ contain the normalized eigenvectors and $\mathbf{\Sigma}$ is a diagonal matrix with the eigenvalues on the main diagonal. Without loss of generality, we assume that the elements of $\mathbf{\Sigma}$ are sorted in ascending order. We then select the first 30 columns of $\mathbf{U}$ to construct a principal components subspace $\mathbf{U}^{1:30} \in \mathbb{R}^{N2L \times 30}$ that accounts for approximately 99% of the training templates energy.

Consider $\mathbf{v}_n = \mathbf{U}^{1:30^T} \left(\bar{\mathbf{s}}_n - \langle \mathcal{S}_{\text{training}} \rangle\right)$ to be the principal component projections of a mean-corrected training template. Let $\mathbf{v}_n^k$ be the projection of $\bar{\mathbf{s}}_n$ onto the k^{th} principal component. For each $k = 1 : 30$, we estimate the probability density function (PDF) $P_k(\mathbf{v}^k)$ across all samples $\mathbf{v}_n^k$ for $n = 1, \ldots, |\mathcal{S}_{\text{training}}|$. We use a non-parametric kernel-density PDF estimation method with a Gaussian kernel [39] and a bandwidth factor of 0.1. The bandwidth factor was chosen based on visual inspection. Fig. 6 shows a visual representation of the estimated PDFs for the first four principal components, which are shown on top of the normalized projection histograms. A similar quality of fit was seen for the other principal components as well. Note that we model the PDF for each k separately instead of a joint PDF, to avoid the curse of dimensionality, which is supported by the uncorrelatedness of the entries in $\mathbf{v}$ due to the PCA procedure. Sampling each of the thirty distributions consecutively leads to an observed random mean-corrected training template in the principal component subspace. By projecting this observation back to the original space and re-adding the mean template, a new training template can be generated:

$$\hat{\bar{\mathbf{s}}} = \mathbf{U}^{1:30}\mathbf{v} + \langle \mathcal{S}_{\text{training}} \rangle, \tag{11}$$

where the k^{th} entry of $\mathbf{v}$ is sampled from P_k. Such example augmented spike templates are shown in Fig. 7. This figure also contains a non-spatially aligned example. Such non-aligned examples are removed from the set of augmented training templates. In this work a set of 25000 random augmented test templates is generated.

Given two distinct augmented training templates $\hat{\bar{\mathbf{s}}}_i$ and $\hat{\bar{\mathbf{s}}}_j$, a training set entry can be generated from following equations:

$$\bar{\mathbf{x}}_i = r_L\left(\hat{\bar{\mathbf{s}}}_i\right) + \bar{\mathbf{n}} \tag{12}$$

$$\bar{\mathbf{x}}_j = r_L\left(\hat{\bar{\mathbf{s}}}_j\right) + \bar{\mathbf{n}} \tag{13}$$

$$\bar{\mathbf{x}}_{i\oplus j} = r_L\left(\hat{\bar{\mathbf{s}}}_i \oplus \hat{\bar{\mathbf{s}}}_i\right) + \bar{\mathbf{n}} \tag{14}$$

$$\bar{\mathbf{x}}_i^{\sigma} = r_L\left(\alpha_i \hat{\bar{\mathbf{s}}}_i\right) + \bar{\mathbf{n}} \tag{15}$$

$$\bar{\mathbf{x}}_j^{\sigma} = r_L\left(\alpha_j \hat{\bar{\mathbf{s}}}_j\right) + \bar{\mathbf{n}} \tag{16}$$

$$\bar{\mathbf{x}}_i^{\delta} = r_L\left(\hat{\bar{\mathbf{s}}}_i \oplus \hat{\bar{\mathbf{s}}}_k^{\delta}\right) + \bar{\mathbf{n}} \tag{17}$$

$$\bar{\mathbf{x}}_j^{\delta} = r_L\left(\hat{\bar{\mathbf{s}}}_j \oplus \hat{\bar{\mathbf{s}}}_k^{\delta}\right) + \bar{\mathbf{n}}, \tag{18}$$

with r_L the temporal realignment-and-truncate operator that results in a temporally aligned spike waveform with a temporal window size of L samples. The alignment procedure used in this work consists of placing the minimum value of the (compound) spike waveform at the center of the temporal window. In this work the random relative shift between two spike waveforms is sampled from a uniform distribution that is bounded between -10 and 10. With a slight abuse of notation, let all $\bar{\mathbf{n}}$ be random Gaussian spatio-temporal white noise with an average SNR of 30 dB (see Section IV-B). α_i is a random scaling factor associated with the neuron i that is sampled from a uniform distribution that is bounded between 0.8 and 1.2. $\hat{\bar{\mathbf{s}}}_k^{\delta}$ is a spike waveform that reaches its maximum energy on a different channel than $\hat{\bar{\mathbf{s}}}_i$ and $\hat{\bar{\mathbf{s}}}_j$. As such, $\hat{\bar{\mathbf{s}}}_k^{\delta}$ is included to model overlap from neurons that reach their minimum spike trough on a different channel. Note that such non-spatially aligned overlapping spikes have to be treated as a source of noise in a divide-and-conquer clustering setting. By randomly sampling (with replacement) $\hat{\bar{\mathbf{s}}}_i$ and $\hat{\bar{\mathbf{s}}}_j$ with $i \neq j$ from the 25000 augmented templates, a total of 1000000 training set entries $\{$(12)-(18)$\}$ are generated to form the training set $\mathcal{T}$.

D. Training procedure

Given the model structure, cost function and training data, we can optimize the model to minimize the cost over the training data. To this end, we make use of Adam [40], which is a stochastic gradient descent algorithm that makes use of parameter-specific adaptive learning. Adam is operated with its default parameter settings as implemented in TensorFlow [41]. We perform the training in random batches of 1000 training set entries per iteration.

To prevent overfitting during training we apply dropout regularization, as already mentioned in Section III-A. The dropout fraction, i.e., the fraction of random connections that are removed from the network during every training step, is treated as a hyperparameter and will be discussed in the next section. To further improve the training immunity with respect to overfitting, we make use of early stopping [42] to terminate the training iteration. If the cost evaluated on a validation set does not decrease for 5 consecutive optimization passes over the entire training set, the training is terminated. The validation set $\mathcal{V}$ consists of a random 30% sample of the training data set $\mathcal{T}$. The final model that is retained is the model with the lowest validation cost.

IV. Experiments

A. Hyper parameter exploration

The model as presented in Section III-A depends on four continuous hyper parameters: κ_1, κ_2, κ_3, and the dropout fraction. For each of the four continuous hyper parameters we explore three predefined values: $\kappa_1 \in \{1, 10, 100\}$, $\kappa_2 \in \{0.1, 1, 10\}$, $\kappa_3 \in \{0.1, 1, 10\}$ and drop out fraction $\in \{0.2, 0.4, 0.6\}$. Note that the values of the different κ are relative to each other and to the first term of (9). Only three values are considered for every parameter to limit the dimensionality of the exploration space. Furthermore, we consider the usage of two optional features as additional binary hyper parameter: the use of batch normalization and the use of a third hidden layer. Training a feature map for all possible combinations of the described hyper parameter settings results in a total of 324 models.

Because the cost function is directly modulated by the choice of κ, the cost function value can not be used to assess the spike sorting performance of the different models. Furthermore, note that the training is example-based, i.e., the training is performed over the training set entries that each consist of only seven spike waveform examples at a time. In this section we use a higher number of spike waveforms for each pair of neurons (i, j) to enable a proper clustering analysis. For every pair of neurons, a set $\mathcal{U}_{(i,j)}$ of 300 spike snippet are generated:

- 100 samples of $\bar{\mathbf{x}}_i^{\alpha}$,
- 100 samples of $\bar{\mathbf{x}}_j^{\alpha}$, and,
- 100 samples of $\bar{\mathbf{x}}_{i\oplus j}^{\alpha}$ that is defined as an overlapping spikes snippet where each of the individual spike tem-

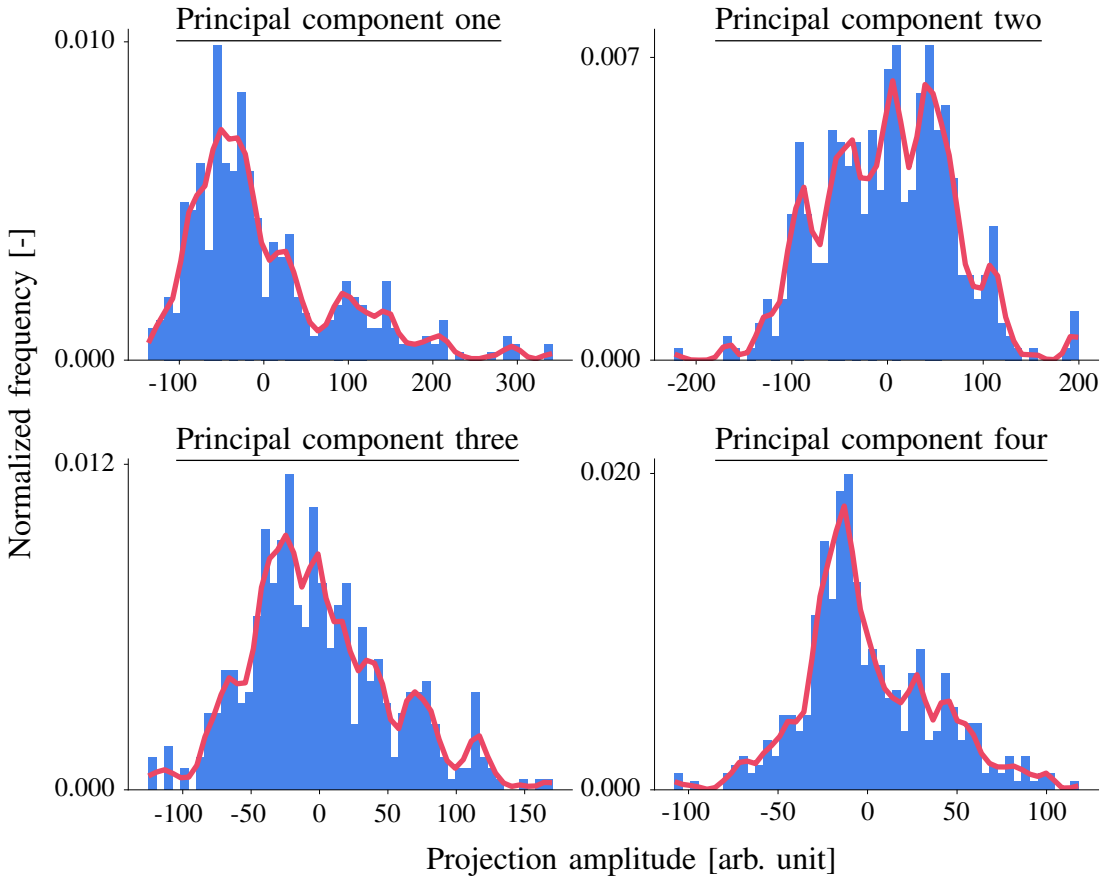

Fig. 6. Normalized projection histograms are shown in blue for the first four principal components of the set of "training templates" $\mathcal{S}_{\text{training}}$. The projection histograms are constructed by projecting all entries of $\mathcal{S}_{\text{training}}$ onto the respecitve principal component. For every histogram, the respective continuous probability density estimate is shown in red.

plates are randomly scaled prior to the application of the $\oplus$-operator.

For each spike snippet there is a 5% chance that a random non-spatially aligned spike is superimposed. The neuronal spike templates from which the snippets are derived are templates from the validation set $\mathcal{V}$ that is used for the early stopping regularization[3]. Given these 300 spikes, a K-means clustering is performed with $K = 3$.

To quantify the clustering performance, we compute two additional metrics: the adjusted Rand index [43] and the center prediction error (CPE).The adjusted Rand index is a widely used metric to assess the clustering performance. The center prediction error $e_{(i,j)}$ between neurons i and j is defined here to be a measure of the targeted 'linear behavior' of the feature map as shown in (4):

$$e_{(i,j)} = \frac{\|\mathbf{c}_{i\oplus j} - (\mathbf{c}_i + \mathbf{c}_j)\|}{\sqrt{\frac{1}{|\mathcal{U}_{(i,j)}|}\sum_{\bar{\mathbf{x}}_k \in \mathcal{U}_{(i,j)}} \|g\left(\bar{\mathbf{x}}_k\right) - \mathbf{c}_{\bar{\mathbf{x}}_k}\|^2}}, \tag{19}$$

[3]The fact that these templates have been used for training purposes is not an issue for the model selection, because the model is not directly optimized for the model selection metrics that are used here.

where the different $\mathbf{c}$ indicate the cluster centers. $\mathbf{c}_{\bar{\mathbf{x}}_k}$ denotes the cluster center from the cluster to which a certain spike waveform $\bar{\mathbf{x}}_k$ from $\mathcal{U}_{(i,j)}$ is assigned. The denominator of $e_{(i.j)}$ is a measure of the overall within-cluster spread that is related to the root mean square error.

Given the clustering results for a pair of neurons (i, j), both the Rand index and CPE can be computed for that pair. This procedure is repeated for a total of 100 validation pairs, resulting in an average rand index and CPE for every feature map. The average metrics for every feature map resulting from this hyper parameter exploration experiment are shown in Fig. 8.

As a first feature map selection criterion, we choose to minimize the center prediction error because we highly value the resolution of overlapping spikes in this work. Fortunately, from our experimental data (see Fig. 8) there seems to be a strong negative correlation between the center prediction error and Rand index, such that feature map selection based on the center prediction error also leads to excellent clustering behaviour.

From Fig. 8 we can also obtain an insight into the effects

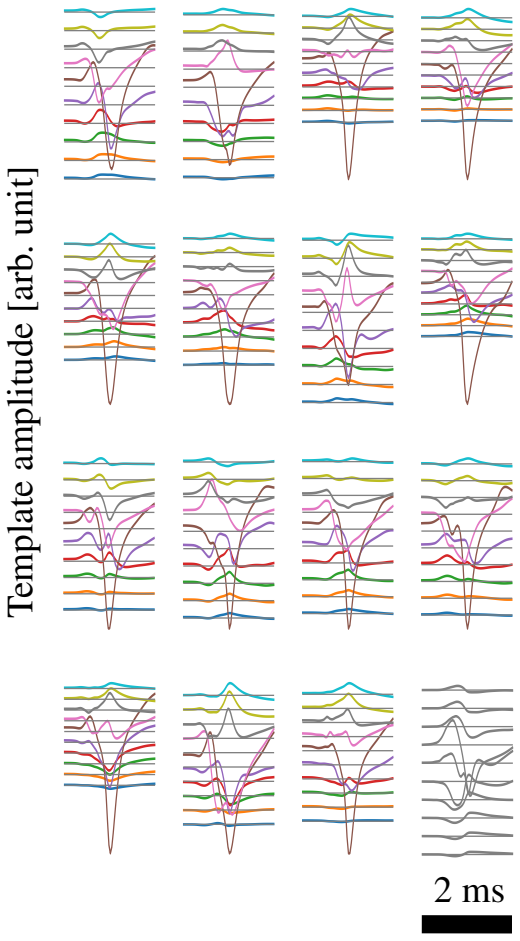

Fig. 7. Sixteen example augmented spike templates generated from (11). The waveform as impinging on a specific channel is given a unique color for improved readability. The plotting distance between the channels is constant. For illustrative purposes, an augmented spike template that reaches its maximum energy on a different channel than intended (and which is excluded from the further analysis) is shown in grey.

of the different parameters on the spike sorting performance. For κ_1 we see that its actual value does not have a major impact on the spike sorting performance, both in terms of Rand index and center prediction error. For κ_2 we see that this parameter affects the CPE. This dependency is likely due to the normalization with respect to the within-cluster spread in the computation of the CPE. The effect of κ_3 is most pronounced for the clustering performance, i.e., focusing too much on robustness against non-spatially aligned overlap will deteriorate the clustering performance. For the use of batch normalization, the optional layer and the value of the dropout fraction, we could not see any clear effect on the spike sorting performance. The minimum CPE model that we finally selected for further analysis and comparison has an average CPE of 0.300 and average Rand index of 0.971 for the validation data. The hyper parameters associated with this model are: $\kappa_1 = 1$, $\kappa_2 = 0.1$, $\kappa_3 = 0.1$, batch normalization is not used, the optional layer is used (resulting in a so-called deep neural network [32]) and a dropout fraction of 0.2 is used during training. It is noted that the model with maximum Rand index only reached a slightly higher Rand index of 0.975 but had a more than double center prediction error of 0.660.

B. Sorting overlapping spikes benchmark

Given the final feature map from the previous section, we can benchmark our method against the state-of-the-art for overlap resolution. The method that we have presented here is unique, in that it is the only method according to our knowledge that has the capability of resolving spike overlap directly in the feature space without the need for first extracting single-unit templates from the recording. Given that there is no feature space benchmark available that we could compare against, we resort to a state-of-the-art matched filtering technique for resolving spike overlap. The matched filter post-processors that we compare against are signal-to-peak-interference ratio (SPIR) optimal linear filters [23]. The desired SPIR was set to 20 dB and a subspace regularization [44] was applied, which accounts for 90% of the signal power. Finally, a fixed detection threshold of 25% of the template response output power (which is known by design) was used on the SPIR-optimal filter output to perform the final spike detection/classification. In this section, we also report the spike sorting accuracy for a clustering-based approach (also using K-means with K=3 for every pair of test neurons) which uses PCA features. The features are computed by projecting the data on three principal components. By comparing our method against a PCA clustering-based approach, we can assess the loss in accuracy when not accounting for overlapping spikes.

It is important to realize that the presented feature map approach and SPIR-optimal benchmark are very different, and this difference should be taken into consideration when comparing the sorting accuracy of the different methods. Use of the presented feature map approach is ultimately an unsupervised clustering approach, whereas the use of template matching post-processors is a binary PU-learning classification problem. This means that the feature map approach is uninformed about the spike waveforms of the neurons under study during the feature map training, whereas the template matching approach depends on the availability of the spike waveform for the matched filter design. Therefore, we do not expect the feature space approach to perform better than template matching approaches due to the fundamentally different prior knowledge assumptions. At best, our presented feature map approach

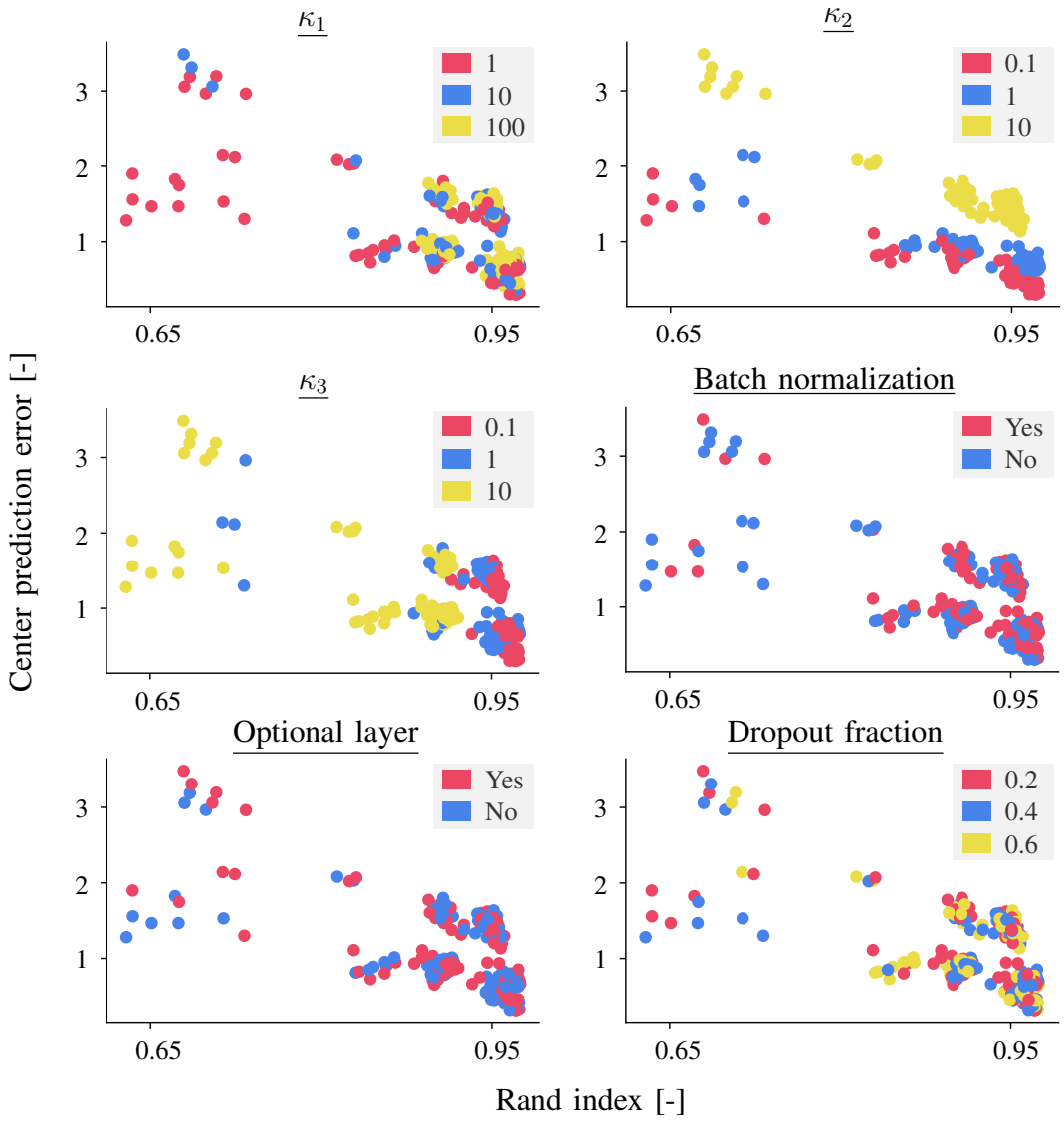

Fig. 8. The six plots in this figure show the average rand index (x-axis) and center prediction error (y-axis) over the the validation pairs for all feature maps in the exploration experiment. The lowest right-most point corresponds to the best performance. Each plot contains the same values, but has a unique color coding that is informed by one of the hyper parameters as indicated in the respective legends.

would result in a similar performance, but using a simplified processing pipeline. Furthermore, in this section, we will make use of other accuracy metrics, as compared to the previous section, that are applicable to both approaches and that are also more commonly used for spike sorting validation, i.e., precision and recall. Precision is equal to the fraction of true positive detections over the total number of detection, whereas recall is the fraction of true positive detections over the total number of ground truth positives.

The test data that is used in this section consists again of 100 pairs of neurons, but here we make use of the biophysically simulated test templates in $\mathcal{S}_{\text{test}}$. Note that these templates were obtained from neuronal models that are kept out of the data augmentation process for the generation of random training and validation templates. Furthermore, rather than using spatio-temporal white noise as during the training and hyper parameter exploration, here, we embed test spikes in real neural recordings [45] which are acquired from a recording device with the same probe geometry as was used for the simulations. The injection of the biophysically simulated spikes into this recording allows to quantify the spike sorting performance since the ground-truth spikes/clusters are known. For every pair we generate 300 spike snippets as in the previous section, but we don't add artificial non-spatially aligned spikes, because the recording already contains such

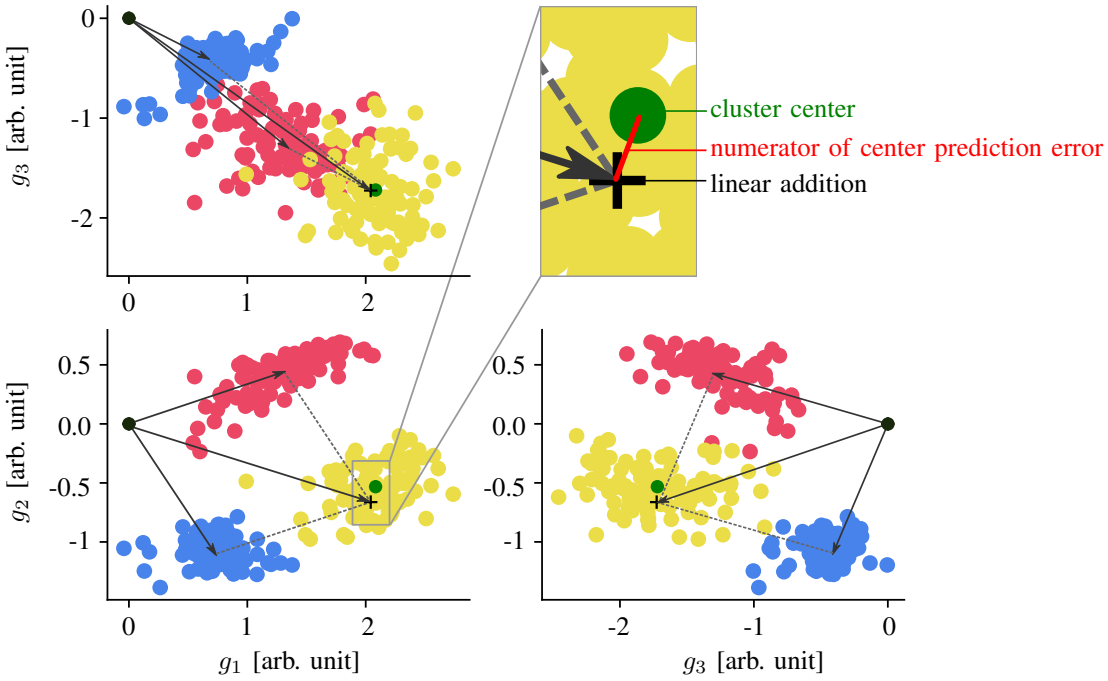

Fig. 9. Three orthogonal projections of the three dimensional feature space are shown. The spike projections are color coded based on the ground truth labels: red for spikes of neuron one, blue for spikes of neuron two and yellow for overlapping spikes of neuron one and two. The green dot represents the cluster center of the overlapping spikes cluster. The vector addition of the cluster centers of the red and blue clusters, which is indicated by a black cross, is shown to map closely to the cluster center of the overlapping spikes cluster.

interfering spikes.

In Fig. 9, the spike embeddings for a pair of test neurons injected in a real recording are shown. From this figure, it is clear that the different clusters are largely separable. Note that the color coding is based on ground-truth labels and not on the actual clustering results. Through the vector addition of the cluster centers from the single-neuron clusters, an estimate of the center of the overlapping spikes cluster can be obtained. The estimated center is indicated in Fig. 9 by a black cross and it is shown to be close to the actual cluster center of the overlapping spikes cluster. The cluster centers that are used here are obtained directly from the clustering analysis (i.e., they are not based on the ground truth labels). From this example, it is clear that the feature map generalizes in terms of spike sorting and overlap resolving capabilities for this test pair.

Fig. 10 summarizes the spike sorting performance for the proposed feature map approach, SPIR-optimal filtering approach and PCA-based approach for all test pairs. The feature map approach has a median precision of 99% and a median recall of 98.5%, whereas for the SPIR-optimal filter this is 100% for both. For the PCA-based clustering approach a median precision of 93.3% and a median recall of only 50% is noted. From this comparison it is clear that both the proposed approach and SPIR-optimal filter achieve a very good spike sorting performance, although the SPIR-optimal filter approach slightly outperforms the proposed feature map approach. However, our proposed feature map approach is superior to the PCA-based approach in the context of overlapping spikes. As already mentioned, the SPIR-optimal filter approach requires the spike templates to be known. In this work we assume that all spike templates are known, which is very unlikely to be the case in practice. Furthermore, because the feature map approach is unsupervised, it can not benefit from this assumption. Although the proposed feature map performs worse than the SPIR-optimal filter benchmark, its performance is still at an acceptable level. Note that we are only able to express the clustering performance of 94% and 90% of the test pairs in terms of precision and recall for the feature map approach and the PCA-based approach, respectively. This can be understood from the fact that the K-means clustering used here, sometimes groups together the single-neuron clusters into a single cluster such that this cluster can only be arbitrarily assigned to one of the two neurons.

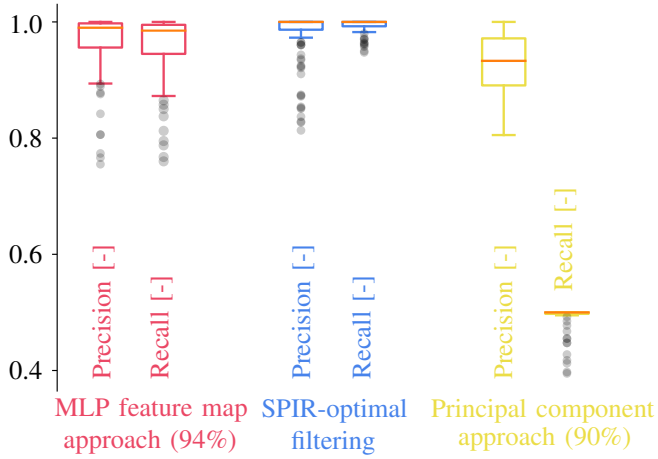

Fig. 10. The precision and recall boxplots are shown for both the proposed MLP feature map approach (red), the SPIR-optimal filter approach (blue) and a principal component feature map approach (yellow). The median values for the different metrics are depicted in orange. Outliers are shown as grey dots. For the MLP and principal component feature map approaches, we could only express the sorting performance in terms of precision and recall for 94% and 90% of the test pairs, respectively (see text).

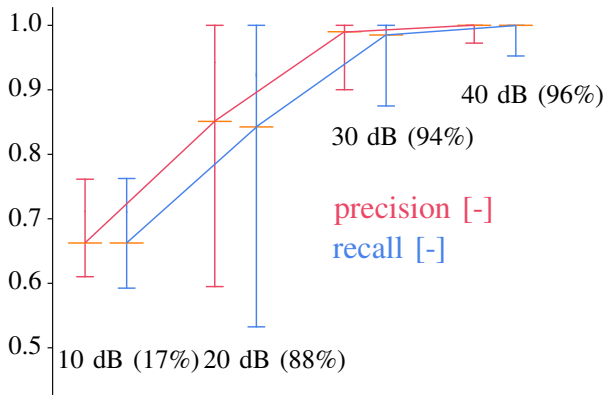

Fig. 11. The precision (red) and recall (blue) are shown for four different mean SNRs: 10 dB, 20 dB, 30 dB and 40 dB. The median metrics for each scenario are indicated in orange. The depicted measure of spread corresponds to the non-outlier boxplot interval, i.e., $Q_1 - 1.5\text{IQR}$ and $Q_3 + 1.5\text{IQR}$, where IQR is the inter-quartile range. For each mean SNR the proportion of test pairs for which the sorting accuracy can be expressed in terms of precision and recall is shown between brackets.

This is a common problem in spike sorting, which implies that no single-unit activity can be extracted for these neurons. This issue can be potentially resolved by using more advanced clustering techniques [46], which is beyond the scope of this paper. As the SPIR-optimal filtering is here computed from the ground-truth templates (instead of the cluster centroids), it does not suffer from this issue. However, in reality these templates should also be extracted from a prior unsupervised clustering [23].

The overall high spike sorting accuracy for both the proposed approach and SPIR-optimal filtering approach is driven by the high SNR spike trains that we have used for both training and testing. The signal power that is used here to compute the SNR is the peak power of the spike waveforms. The SNR of the test templates that are used in this section are situated between 25 and 38 dB, with an average of 30 dB. This SNR range is similar to the SNR range used during training. For such high SNR neurons our proposed feature space approach can thus be used to reliably sort and resolve overlapping spikes. Peak SNRs of 20 to 30 dB do occur in practice and these spike trains are usually easy to detect and can be reliably sorted, as was shown in the previous analysis.

In Fig. 11 we investigate whether the resulting feature map generalizes to other SNR scenarios. The SNR measures that are reported in the graph are the mean SNRs that correspond to a specific noise level. The 30 dB case corresponds to the case that was presented in Fig. 10. When further increasing the SNR to an average of 40 dB the spike sorting performance further increases with a median precision and precision of both 100%. We were able to express the sorting accuracy in terms of precision and recall for 96% of the test pairs. When decreasing the SNR to an average of 20 dB, the median precision and recall decrease to 85.1% and 84.2% respectively. For the 20 dB case we were able to express the spike sorting performance in terms of precision and recall for 88% of the test pairs. Finally, for the 10 dB case the median precision and recall further decrease to both 66.2%. For this final case, we were only able to express the sorting performance for 17% of the test pairs (which also explains the low spread compared to the 20 dB case). We can conclude from this experiment that the resulting feature map does not generalize for the lower SNR scenarios that were unseen during training. A potential retraining with lower SNR spike waveforms could improve the sorting performance for these scenarios.

V. Discussion and Conclusion

We have presented a feature map that is capable of resolving spike overlap in the feature space. Overlapping spikes clusters are identifiable by means of a simple vector addition of pairs of cluster centers. If the sum of the cluster centers from a pair of clusters is sufficiently close to the center of a third cluster, this third cluster is believed to contain spike snippets that are a superposition of spikes from the pair of neurons corresponding to the pair of clusters. This approach is particularly interesting,

because the feature map has to be trained only once. After this initial training, the feature map is intended to be used without any retraining on other recordings that have a similar recording setting. We have validated this usage scenario by showing that the proposed feature map and its overlap resolution capabilities generalize to data that was unseen during training.

In Section III we have broken down the design of the feature map into its different ingredients: model selection, cost function, training data and training procedure. Although we have shown that our feature map is capable of sorting overlapping spikes to a large extent, our method performed slightly worse compared to the state-of-the-art. Although we don't expect a feature map approach to outperform a state-of-the-art (SPIR-optimal) matched filtering approach due to the fundamental difference in learning paradigm and available prior knowledge, a further and deeper exploration of the design space could further close the gap between the two paradigms. Unfortunately, the design space is infinitely large, which forces researchers to narrow down the free parameters.

However, in case of future research in this field, the proposed multi-layer perceptron architecture could be further improved or other neural network families could be considered, e.g., convolutional neural networks. The proposed cost function can also be further tweaked, e.g., to account for overlapping spikes snippets from the activity of three or more neurons. However, support for multiple overlap (as well as other additional cost terms) is likely to trade off with other feature map characteristics, such as cluster separability, which could hamper the sorting performance. Therefore, it is important to assess the need for additional cost term modelling based on how these terms affect the sorting performance, i.e., both with and without the additional term involved. Other cost function related improvement could be integrated to account for neuronal bursting and probe drift [47]. A design component in this works that remains largely unexplored are the use of different optimization algorithms and regularization approaches, that might also lead to alternative feature map approaches with improved sorting capabilities.

We believe that the general success of domain-specific learning-based spike detection and feature map building blocks for spike sorting will depend on the availability of data augmentation approaches that generalize to real recordings. In this work we have proposed a simple approach based on simulated templates, that was capable of generalizing to realistic recordings that contained simulated test templates embedded in real recording noise. Note that the realistic recording noise was not available during training. Apparently, the feature map did not work well for low-SNR regimes. In future research, the training could be repeated, taking into account realistic noise conditions and a wider range of signal-to-noise ratio conditions. Such a training might even lead to a feature map with noise-reduction capabilities. Finally, we don't think learning-based building blocks will generalize between distinct recording devices, e.g., different probe geometries, and this is a limiting practical factor when compared to non-domain-specific approaches that are widely used today. We recognize that the proposed data augmentation method heavily depends on the availability of realistic simulation data, which could hamper its adoption when a wide variety of recording devices are of interest to a specific user. Therefore, we encourage future research on data augmentation procedures that take real recordings at their input. Nonetheless, the use of domain-specific building block has the potential to overcome long standing spike sorting challenges. For example, in our work, the template estimation bias due to overlapping spikes is circumvented through an integrated feature space approach.

REFERENCES

[1] J. Wouters, F. Kloosterman, and A. Bertrand, "A neural network-based spike sorting feature map that resolves spike overlap in the feature space," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1175–1179, 2020.

[2] J. J. Jun, N. A. Steinmetz, J. H. Siegle, D. J. Denman, M. Bauza, B. Barbarits, A. K. Lee, C. A. Anastassiou, A. Andrei, Ç. Aydın, *et al.*, "Fully integrated silicon probes for high-density recording of neural activity," *Nature*, vol. 551, no. 7679, pp. 232–236, 2017.

[3] J. E. Chung, H. R. Joo, J. L. Fan, D. F. Liu, A. H. Barnett, S. Chen, C. Geaghan-Breiner, M. P. Karlsson, M. Karlsson, K. Y. Lee, *et al.*, "High-density, long-lasting, and multi-region electrophysiological recordings using polymer electrode arrays," *Neuron*, vol. 101, no. 1, pp. 21–31, 2019.

[4] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network: Computation in Neural Systems*, vol. 9, no. 4, pp. R53–R78, 1998.

[5] E. I. Moser, E. Kropff, and M.-B. Moser, "Place cells, grid cells, and the brain's spatial representation system," *Annu. Rev. Neurosci.*, vol. 31, pp. 69–89, 2008.

[6] A. Khatoun, B. Asamoah, and M. Mc Laughlin, "Simultaneously excitatory and inhibitory effects of transcranial alternating current stimulation revealed using selective pulse-train stimulation in the rat motor cortex," *Journal of Neuroscience*, vol. 37, no. 39, pp. 9389–9402, 2017.

[7] Ç. Aydın, J. Couto, M. Giugliano, K. Farrow, and V. Bonin, "Locomotion modulates specific functional cell types in the mouse visual thalamus," *Nature communications*, vol. 9, no. 1, pp. 1–12, 2018.

[8] S. Gibson, J. W. Judy, and D. Marković, "Spike sorting: The first step in decoding the brain," *IEEE Signal processing magazine*, vol. 29, no. 1, pp. 124–143, 2011.

[9] S. Shoham, M. R. Fellows, and R. A. Normann, "Robust, automatic spike sorting using mixtures of multivariate t-distributions," *Journal of neuroscience methods*, vol. 127, no. 2, pp. 111–122, 2003.

[10] T. I. Aksenova, O. K. Chibirova, O. A. Dryga, I. V. Tetko, A.-L. Benabid, and A. E. Villa, "An unsupervised automatic method for sorting neuronal

spike waveforms in awake and freely moving animals," *Methods*, vol. 30, no. 2, pp. 178–187, 2003.

[11] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural computation*, vol. 16, no. 8, pp. 1661–1687, 2004.

[12] C. Rossant, S. N. Kadir, D. F. Goodman, J. Schulman, M. L. Hunter, A. B. Saleem, A. Grosmark, M. Belluscio, G. H. Denfield, A. S. Ecker, *et al.*, "Spike sorting for large, dense electrode arrays," *Nature neuroscience*, vol. 19, no. 4, pp. 634–641, 2016.

[13] M. Pachitariu, N. A. Steinmetz, S. N. Kadir, M. Carandini, and K. D. Harris, "Fast and accurate spike sorting of high-channel count probes with kilosort," *Advances in neural information processing systems*, vol. 29, pp. 4448–4456, 2016.

[14] J. E. Chung, J. F. Magland, A. H. Barnett, V. M. Tolosa, A. C. Tooker, K. Y. Lee, K. G. Shah, S. H. Felix, L. M. Frank, and L. F. Greengard, "A fully automated approach to spike sorting," *Neuron*, vol. 95, no. 6, pp. 1381–1394, 2017.

[15] P. Yger, G. L. Spampinato, E. Esposito, B. Lefebvre, S. Deny, C. Gardella, M. Stimberg, F. Jetter, G. Zeck, S. Picaud, *et al.*, "A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo," *Elife*, vol. 7, p. e34518, 2018.

[16] J. W. Pillow, J. Shlens, E. Chichilnisky, and E. P. Simoncelli, "A model-based spike sorting algorithm for removing correlation artifacts in multi-neuron recordings," *PloS one*, vol. 8, no. 5, p. e62123, 2013.

[17] M. Abeles and M. H. Goldstein, "Multispike train analysis," *Proceedings of the IEEE*, vol. 65, no. 5, pp. 762–773, 1977.

[18] D. A. Adamos, N. A. Laskaris, E. K. Kosmidis, and G. Theophilidis, "Nass: an empirical approach to spike sorting with overlap resolution based on a hybrid noise-assisted methodology," *Journal of neuroscience methods*, vol. 190, no. 1, pp. 129–142, 2010.

[19] O. Marre, D. Amodei, N. Deshmukh, K. Sadeghi, F. Soo, T. E. Holy, and M. J. Berry, "Mapping a complete neural population in the retina," *Journal of Neuroscience*, vol. 32, no. 43, pp. 14859–14873, 2012.

[20] F. Franke, R. Q. Quiroga, A. Hierlemann, and K. Obermayer, "Bayes optimal template matching for spike sorting–combining fisher discriminant analysis with optimal filtering," *Journal of computational neuroscience*, vol. 38, no. 3, pp. 439–459, 2015.

[21] Y. Mokri, R. F. Salazar, B. Goodell, J. Baker, C. M. Gray, and S.-C. Yen, "Sorting overlapping spike waveforms from electrode and tetrode recordings," *Frontiers in neuroinformatics*, vol. 11, p. 53, 2017.

[22] J. Wouters, F. Kloosterman, and A. Bertrand, "Towards online spike sorting for high-density neural probes using discriminative template matching with suppression of interfering spikes," *Journal of neural engineering*, vol. 15, no. 5, p. 056005, 2018.

[23] J. Wouters, P. Patrinos, F. Kloosterman, and A. Bertrand, "Multi-pattern recognition through maximization of signal-to-peak-interference ratio with application to neural spike sorting," *IEEE Transactions on Signal Processing*, 2020.

[24] J. S. Prentice, J. Homann, K. D. Simmons, G. Tkačik, V. Balasubramanian, and P. C. Nelson, "Fast, scalable, bayesian spike identification for multi-electrode arrays," *PloS one*, vol. 6, no. 7, p. e19884, 2011.

[25] C. Ekanadham, D. Tranchina, and E. P. Simoncelli, "A unified framework and method for automatic neural spike identification," *Journal of neuroscience methods*, vol. 222, pp. 47–55, 2014.

[26] J. H. Lee, D. E. Carlson, H. Shokri Razaghi, W. Yao, G. A. Goetz, E. Hagen, E. Batty, E. Chichilnisky, G. T. Einevoll, and L. Paninski, "YASS: Yet another spike sorter," *Advances in neural information processing systems*, vol. 30, pp. 4002–4012, 2017.

[27] C. Hurwitz, K. Xu, A. Srivastava, A. Buccino, and M. Hennig, "Scalable spike source localization in extracellular recordings using amortized variational inference," in *Advances in Neural Information Processing Systems*, pp. 4724–4736, 2019.

[28] M. Saif-ur Rehman, O. Ali, S. Dyck, R. Lienkämper, M. Metzler, Y. Parpaley, J. Wellmer, C. Liu, B. Lee, S. Kellis, *et al.*, "Spikedeep-classifier: A deep-learning based fully automatic offline spike sorting algorithm," *Journal of Neural Engineering*, 2020.

[29] Z. Li, Y. Wang, N. Zhang, and X. Li, "An accurate and robust method for spike sorting based on convolutional neural networks," *Brain Sciences*, vol. 10, no. 11, p. 835, 2020.

[30] J. Eom, I. Y. Park, S. Kim, H. Jang, S. Park, Y. Huh, and D. Hwang, "Deep-learned spike representations and sorting via an ensemble of auto-encoders," *Neural Networks*, 2020.

[31] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice-Hall, Inc., 2007.

[32] S. Theodoridis, *Machine learning: a Bayesian and optimization perspective*. Academic press, 2015.

[33] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[36] S. Ramaswamy, J.-D. Courcol, M. Abdellah, S. R. Adaszewski, N. Antille, S. Arsever, G. Atenekeng, A. Bilgili, Y. Brukau, A. Chalimourda, *et al.*, "The neocortical microcircuit collaboration portal: a resource for rat somatosensory cortex," *Frontiers in neural circuits*, vol. 9, p. 44, 2015.

[37] A. P. Buccino and G. T. Einevoll, "Mearec: a fast and customizable testbench simulator for ground-truth extracellular spiking activity," *Neuroinformatics*, pp. 1–20, 2020.

[38] N. V. Swindale and M. A. Spacek, "Spike sorting for polytrodes: a divide and conquer approach," *Frontiers in systems neuroscience*, vol. 8, p. 6, 2014.

[39] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

[42] L. Prechelt, "Early stopping-but when?," in *Neural Networks: Tricks of the trade*, pp. 55–69, Springer, 1998.

[43] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[44] J. Wouters, F. Kloosterman, and A. Bertrand, "A data-driven regularization approach for template matching in spike sorting with high-density neural probes," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 4376–4379, IEEE, 2019.

[45] N. Steinmetz, M. Carandini, and K. D. Harris, ""Single Phase3" and "Dual Phase3" Neuropixels Datasets," 3 2019.

[46] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[47] H. G. Rey, C. Pedreira, and R. Q. Quiroga, "Past, present and future of spike sorting techniques," *Brain research bulletin*, vol. 119, pp. 106–117, 2015.