

UiO : **University of Oslo**

Alessio Paolo Buccino

A computationally-assisted approach to extracellular neural electrophysiology with multi-electrode arrays

Thesis submitted for the degree of Philosophiae Doctor

Department of Informatics
Faculty of Mathematics and Natural Sciences

Center for Integrative Neuroplasticity (CINPLA)
Faculty of Mathematics and Natural Sciences
University of Oslo

Department of Biomedical Engineering
University of California San Diego



2019

© **Alessio Paolo Buccino, 2020**

*Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 2222*

ISSN 1501-7710

All rights reserved. No part of this publication may be reproduced or transmitted, in any form or by any means, without permission.

Cover: Hanne Baadsgaard Utigard.
Print production: Reprosentralen, University of Oslo.

To my girlfriend, family, and friends

Acknowledgements

The work presented in this thesis was conducted from December 2015 to October 2019. I spent the first year at the NANO group, in the Department of Informatics of the University of Oslo. In December 2016, I moved to the Center of Integrative Neuroplasticity (CINPLA) at the Department of Biosciences of the University of Oslo. During my PhD, I spent one year, from August 2017 to August 2018, at the Integrated Systems Neuroengineering Lab of the University of California in San Diego.

I would first like to thank the SUURPh programme and the University of Oslo to have given me the chance to start this wonderful journey. Thank you for believing in me and giving me all the support I needed in these four years. Thank you Andy for your effort in starting this amazing training program (and for the awesome surfing sessions). Thank you Rachel for making my PhD life way easier. Thank you Kim for taking over and keeping this training programme going.

Philipp, you have been a great supervisor. I would really like to thank you for allowing me to follow and pursue my interests, always checking that I was on track.

Gaute, thank you for showing me the wonders of the computational neuroscience world. This thesis would not have been possible without your input and supervision.

Marianne, thank you for welcoming me to CINPLA and for always believing in me. You have built something unique and amazing and it has been a real pleasure to be part of it.

Gert, thank you for making my UCSD time unforgettable. I really enjoyed our science and maths discussions. Thank you for transmitting your vision and passion, and for the great support.

Thank you Torkel for the wonderful times spent together and for your valuable guidance in the realm of neuroscience.

Aslak, the chats we had about science and the future were priceless. Thank you for your trust and support.

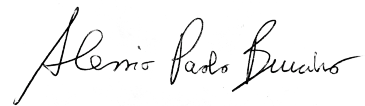
I would like to deeply thank all the CINPLA members. Working and spending time with you in these years has been a pure joy. You know you have wonderful colleagues and friends when you never have a day in which you think “I don’t wanna go to work today”.

I would like to thank the OSI H2 soccer team. Playing soccer (and partying) with you guys has represented an important part of this journey. Thank you for making me feel home and for comparing me to Pirlo.

A big thank you to all my "norwegian" friends. Alessio, Antonio, Sunniva, Stephanie, Emil, Ruben, Donato, and the rest of the crew: thank you for the great times spent together.

I would like to thank my mom, my dad, and my sister for the endless support. I know it is hard to have a family member away for such a long time, but I hope that this period will pay off in the future.

Finally, thank you Silvia for your love and immense support. I would not have been able to achieve this without you always by my side.

A handwritten signature in black ink, reading "Alessio Paolo Buccino". The signature is written in a cursive, flowing style.

Oslo, December 2019

Abstract

One of the most used techniques to study neural circuit function is extracellular electrophysiology, which enables one to measure the electrical activity of neuron communication using electrodes placed in the extracellular space of the neural tissue. The power of this technique lies in the opportunity to link single neuron and network activity to complex behaviour and cognition. Recent years have witnessed huge advancements in neurotechnology for extracellular recordings, with the advent of new neural devices capable of recording from hundreds of electrodes simultaneously and to measure the activity of hundreds of neurons with very high spatio-temporal resolution (high-density multi-electrode arrays - HD-MEAs). These novel opportunities raise challenges with regards to data analyses and how to interpret results from recordings.

In this thesis I introduce new tools and analysis methods specifically targeting HD-MEA devices. The first goal of the work was to develop a unified approach using simulations to assist in the method development to address open problems in the analysis of electrophysiological data. These include spike sorting, cell-type classification, neuron localization, and selective electrical stimulation. A secondary goal of this work was to investigate to what extent the current modeling framework is capable of replicating the measured spiking activity.

In Papers I and II I present two tools to improve the process of developing and evaluating methods for spike sorting, introducing a simulator of extracellular spiking activity and a unified framework for spike sorting evaluation.

In Papers III and IV I investigate the use of independent component analysis for spike sorting of high-density multi-electrode arrays, both in an offline and an online setting.

In Paper V I introduce a method for neuron localization and cell-type classification that combines forward modeling and deep learning techniques.

Paper VI presents a model-based optimization framework for targeted extracellular electrical stimulation of single neurons from multi-electrode arrays.

Paper VII is a modeling study which investigates the effect of the neural probe on the recorded potentials.

With the work presented in this thesis, I show that computationally-assisted methods can contribute to the state-of-the-art analysis of extracellular electrophysiological data. In combination with newly developed neurotechnologies, these methods will advance our understanding of neural circuit function.

List of Papers

Paper I

Buccino AP, and Einevoll GT.

MEARec: a fast and customizable testbench simulator for extracellular spiking activity.

In: *bioRxiv* DOI: 10.1101/691642. Submitted to *Neuroinformatics*.

Paper II

Buccino, AP*, Hurwitz C*, Magland J, Garcia S, Siegle JH, Hurwitz R, and Hennig MH.

SpikeInterface, a unified framework for spike sorting.

In: *bioRxiv* DOI: 10.1101/796599. Submitted to *eLIFE*.

Paper III

Buccino AP, Hagen E, Einevoll GT, Häfliger PD, and Cauwenberghs G.

Independent component analysis for fully automated multi-electrode array spike sorting.

In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2018), pp. 2627–2630. DOI: 10.1109/EMBC.2018.8512788.

Paper IV

Buccino AP, Hsu S-H, and Cauwenberghs, G.

Real-time spike sorting for multi-electrode arrays with online independent component analysis.

*These authors contributed equally to this work.

In: *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (2018), pp. 1–4. DOI: 10.1109/BIOCAS.2018.8584797.

Paper V

Buccino AP*, Kordovan M*, Ness TV, Merkt B, Häfliger PD, Fyhn M, Cauwenberghs G, Rotter S[†], and Einevoll GT.[†]

Combining biophysical modeling and deep learning for multielectrode array neuron localization and classification.

In: *Journal of Neurophysiology* **120-3** (2018), pp. 1212–1232. DOI: 10.1152/jn.00210.2018.

Paper VI

Buccino AP, Stöber T, Næss S, Cauwenberghs G, and Häfliger PD.

Extracellular single neuron stimulation with high-density multi-electrode array.

In: *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (2016), pp. 520–523. DOI: 10.1109/BioCAS.2016.7833846.

Paper VII

Buccino AP, Kuchta M, Jæger KH, Ness TV, Berthet P, Mardal KA, Cauwenberghs G, and Tveito A.

How does the presence of neural probes affect extracellular potentials?

In: *Journal of Neural Engineering* **16-2** (2019), pp. 026030. DOI: 10.1088/1741-2552/ab03a1.

[†]These authors also contributed equally to this work.

Contents

Abstract	v
List of Papers	vii
Contents	ix
1 Introduction	1
1.1 Objectives	4
1.2 Structure of the Thesis	5
2 The extracellular electrophysiology pipeline	7
2.1 Spike sorting	8
2.1.1 Challenges	12
2.2 Cell-type classification	14
2.3 Cell localization	15
2.4 Electrical stimulation	17
3 Computational models	19
3.1 Multi-compartment models	19
3.2 Extracellular potential	21
3.3 Models of electrical stimulation	24
3.4 Finite element methods	27
3.4.1 Hybrid approach	27
3.4.2 EMI model	27
4 Engineering solutions	31
4.1 Independent component analysis	31
4.2 Machine learning and deep learning	35
4.2.1 The basics of artificial neural networks	35
4.2.2 Deep Learning	38
4.2.3 Convolutional Neural Networks	39

Contents

4.2.4	Underfitting and overfitting: finding the right model	41
4.3	Optimization and genetic algorithms	42
4.4	Scientific programming in Python	43
5	Summary of Papers	45
6	Discussion	49
6.1	Towards next-generation electrophysiology	49
6.2	Computationally-assisted electrophysiology: benefits and limitations	51
6.3	Future developments	53
6.4	Outlook	54
	Bibliography	55
	Papers	74
I	MEArc: a fast and customizable testbench simulator for ground-truth extracellular spiking activity	75
II	SpikeInterface, a unified framework for spike sorting	109
III	Independent component analysis for fully automated multi-electrode array spike sorting	135
IV	Real-time spike sorting for multi-electrode arrays with online independent component analysis	141
V	Combining biophysical modeling and deep learning for multielectrode array neuron localization and classification	147
VI	Extracellular single neuron stimulation with high-density multi-electrode array	171
VII	How does the presence of neural probes affect extracellular potentials?	179
	Appendices	199
A	Other projects	201

Chapter 1

Introduction

“I like nonsense; it wakes up the brain cells.”

— Dr. Seuss

The brain is arguably the most fascinating and complicated of all organs.

A recent study¹ estimates that the human brain contains around 86'000'000'000, or 8.6×10^{10} , or, if the reader prefers, eighty six billion neurons. If we consider other animal species, still the numbers are incredible, with chimpanzees having around 28'000'000'000¹, cats 760'000'000², rats 200'000'000³, and honey bees around 960'000 neurons⁴.

I have always had a hard time making a sense of millions and billions. So, the day before a talk that I was preparing, I tried to come up with an analogy, both for me and my audience, to grasp the immensity of the brain:

A tennis ball has an average diameter of 6.7 cm. An soccer pitch is 100 m long and 60 m wide, give or take. Now: imagine covering the entire football field with tennis balls (let's assume for simplicity that the balls are cubes of side 6.7 cm). Once we are done with the first layer, we go on with the second layer, and so on. In order to place 86'000'000'000 tennis balls, we would have stacked around 64'000 layers, reaching an altitude of 4'300 m. Basically, we are on top of the Alps mountains.[†]

Neurons are electrical entities^{5,6}. Their structure enables them to maintain an electric potential across their membrane that is modulated by incoming signals from other neurons. When the membrane potential of a neuron crosses a threshold, an action potential is generated and it is sent to all the neurons to which that neuron's axon connects to through synapses.

Being electrical entities, one can measure neurons' activity by inserting electrodes in the brain. When an action potential is generated, electric currents

[†]when I gave the talk and made this calculation I was in the United States and I ended up on top of the Mount Everest (around 8000 m). Tennis balls are probably bigger in the US.

1. Introduction

quickly flow in and out of the neuron for around a millisecond. Extracellularly, if an electrode is close enough to a neuron *firing* an action potential, we observe a fast transition in the recorded electric potential, that we refer to as *spike*.

Recent years have witnessed an unprecedented advancement in neurotechnology⁷. Extracellular recordings have been historically performed with single microwires or bundles of microwires (e.g. four microwires in a *tetrode*), capable of recording a few tens of neurons per experiment. The use of silicon manufacturing processes for neural probes, which can provide higher density and electrode counts than microwire technology^{8–10}, has been investigated for more than two decades^{11–13}. However, the main innovation has come from the integration of electronic circuits in CMOS (Complementary Metal-Oxide-Semiconductor) technology with the neural probes. These embedded electronic circuits can perform amplification and digitization of the neural signals, which results in lower noise levels (thanks to on-site amplification) and the capability of greatly increasing the electrode counts (thanks to on-site digitization, as digital transmission requires less metal wires, which are the main bottleneck for high electrode counts)¹⁴. CMOS-based neural probes have resulted in the prototyping of several high-density custom solutions both for *in vitro*^{15–18} and *in vivo*^{19–25} recordings. Leveraging this research, neuroscientists now have access to commercial recording probes with hundreds and thousands of electrodes, for both *in vitro** and *in vivo*† applications.

High-density multi-electrode arrays, or *micro*-electrode arrays (MEAs), enable researchers to perform high-yield experiments, in which several hundreds of neurons can be recorded simultaneously. These devices offer many opportunities for next-generation electrophysiology²⁶ (as well as challenges²⁷). Apart from being able to record the activity of many more neurons than previous techniques, the high spatial density of the electrodes enables the observation of the same action potentials on many different recording contacts. The known spatial geometry of the recording sites allows one to study neurons, from extracellular signals, even at the sub-cellular level²⁸. Figure 1.1 displays a simulation with a pyramidal cell from layer 5 in the center, a tetrode on the left, and a high-density probe on the right. The tetrode sees the spike from four contacts, with no precise knowledge of the relative electrode locations (at least for self-assembled wire tetrodes). On the high-density MEA, the same spike appears on tens of adjacent electrodes, with precise knowledge of the geometry. The amount of available spatio-temporal information provided by these probes is arguably unprecedented.

How can we exploit this high spatio-temporal information to improve the current methods used in extracellular electrophysiology?

This question is the main thread of my thesis. I have an engineering

*Maxwell Biosystems – www.mxwbio.com, 3Brain – www.3brain.com, Multichannel systems – www.multichannelsystems.com

†Neuropixels – www.neuropixels.org

Microwire / Tetrode

High-density MEA

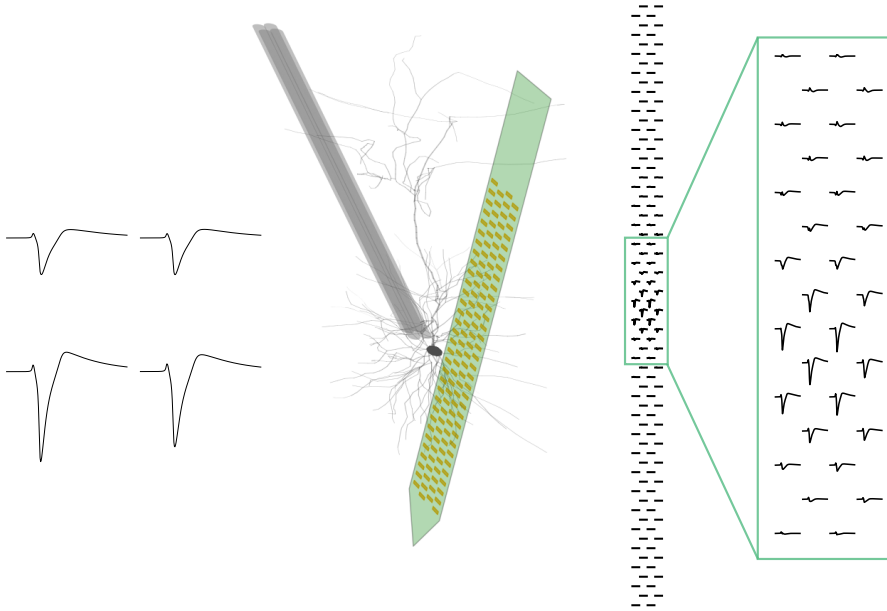


Figure 1.1: Illustration of an extracellular recording with two types of probes: a tetrode (left) and a high-density multi-electrode array (right). The traces show a single spike recorded from the pyramidal cell in the center. The amount and richness of spatial information from the MEAs can enable one to observe even sub-cellular aspects of the action potential²⁸

background and I was naturally attracted to several open questions and problems* in the extracellular electrophysiology field. So, during these (almost) four years, I tried, with the help of several colleagues, to tackle them one by one.

I am still missing something: Maths. Engineers see the world as formulas and numbers, equations to solve, parameters to estimate, or, in other words, *models*. Fortunately, the computational neuroscience field is no disappointment.

Detailed models of single neurons, or multi-compartment models (Section 3.1), have been widely used by the neuroscience community to study various aspects of single-cell dynamics²⁹⁻³¹. These models represent a valuable tool for investigating and understanding neural mechanisms, and the international community has invested huge resources in building and sharing a large variety of detailed neuronal models. As an example, the Blue Brain Project has constructed over 30'000 cell models from the somatosensory cortex of juvenile rats, with the final goal to digitally reconstruct the neocortical microcircuit³²⁻³⁵. A similar effort is being

*Yes, engineers do like to solve problems.

conducted by the Allen Institute for Brain Science, whose cell-type database³⁶ contains several hundred of cell models from mice and even humans³⁷. Moreover, we have a good understanding of how currents generated by neurons translate to recorded extracellular potentials^{38–42}.

Great, now everything is in place: an enthusiastic (now) neural engineer, advanced models of neurons, and electrophysiology problems to solve.

1.1 Objectives

The main objective of this work is to combine simulations and engineering methods to tackle several aspects of the electrophysiology pipeline (Chapter 2) for high-density MEAs. Given the availability of biophysically detailed models, that may pass a *biological* Turing test⁴³, the central idea of this project is to use modeling and simulations to assist the development of methods for electrophysiology. These techniques include spike sorting, cell localization and classification from extracellular action potentials, and targeted electrical stimulation from extracellular probes (Figure 2.1 in Chapter 2). Moreover, since a great part of this work uses simulations to drive method development, a secondary goal is to investigate to what extent the conventional modeling framework is accurate and trustworthy and what are its limitations.

While the order of the papers presented in Chapter 5 follows the natural electrophysiology pipeline (Figure 2.1), I present here the objective of the papers in a quasi-chronological order, to highlight why and how the main project evolved and the different sub-projects were conceptualized.

Paper VI *How can we exploit the high density of MEAs to make electrical stimulation more selective?*

The goal of this paper was to improve the selectivity of extracellular stimulation leveraging the high density of the electrodes, a model-based approach, and optimization techniques. We assumed that we knew the location of the neuron with respect to the probe.

Paper V *Can we accurately estimate the position of a neuron from its extracellular signals?*

The objective of this article was to improve the performance of neuronal localization and classification from extracellular action potentials by combining forward biophysical modeling and deep learning techniques. In this approach, we assumed that extracellular simulations of single spikes are biophysically accurate enough to be used as ground-truth information for training machine learning algorithms. Moreover in this contribution we assumed that inference was performed after spike sorting, on average waveforms.

Paper VII *Are our estimates of extracellular signals accurate enough?*

The aim of this article was to investigate to what extent the neural probes we use to record neural activity affect the recorded signals. In order to do so, we

used finite element methods. Moreover, we wanted to implement a more efficient method to include the presence of the probes in the calculation of extracellular potentials.

Paper III *What about spike sorting for high-density MEAs?*

In this article we aimed to investigate the use of independent component analysis (ICA) for spike sorting data from high-density MEAs. Moreover, we wanted to build a fully automatic pipeline and compare its performance with state-of-the-art algorithms. In order to benchmark the algorithm, we needed to develop a simulator of ground-truth spiking activity, wrappers for some existing spike sorters, and automated comparison routines.

Paper IV *What about real-time spike sorting for high-density MEAs?*

The goal of this paper was to investigate an online version of ICA, namely online recursive ICA – ORICA, for real-time spike sorting. We aimed to benchmark the accuracy of the ORICA model in finding spiking sources within real-time time constraints.

Paper I *Can we develop a simulator of extracellular activity to aid spike sorting development?*

This question arose from Paper III and Paper IV, in which I felt the need to include ground-truth simulations in the development of spike sorting algorithms. The objective of this paper was to create a simulator of extracellular activity and make it an accessible and usable software package that could be used by the spike sorting community. We aimed to develop a fast, easy to use, highly controllable, and biophysically detailed simulator. Additionally, we wanted the simulator to be able to reproduce features of extracellular signals that are critical for spike sorting, such as bursting, drifting, and spatio-temporal overlapping spikes.

Paper II *How can we benchmark and compare several spike sorters?*

This question as well has its origins in Paper III and Paper IV, from the tedious and time-consuming process of running several spike sorters and comparing their outcome. The goal of this paper was to build a unified software framework for spike sorting. I teamed up with international collaborators in the spike sorting community to create an open-source software to make the spike sorting process easy and accessible, even to those with very little programming background. Moreover, we wanted to include tools for the entire electrophysiology pipeline, including processing, quality control, and curation tools, as well as a comparison framework for both ground-truth and non ground-truth data.

1.2 Structure of the Thesis

The thesis is organized in three background chapters, a summary of the papers (Chapter 5), and a final discussion that contextualizes the work, highlighting its

limitations and future directions (Chapter 6).

In Chapter 2, I cover the electrophysiology pipeline, with a brief background and state-of-the-art methods for the different steps involved in the interaction between neural tissue and extracellular devices. Chapter 3 covers the basics of the computational models that I used throughout my thesis, from multi-compartment models, to finite element methods. Finally, Chapter 4 introduces the engineering solutions employed in most of the projects, focusing on independent component analysis, machine learning and deep learning, optimization strategies and genetic algorithms, as well as the use of Python for scientific software development.

Chapter 2

The extracellular electrophysiology pipeline

“You are nothing but a pack of neurons.”

— Francis Crick

When we implant a neural probe in the brain, it can be used to bi-directionally interact with the neural tissue that surrounds it. On one hand, electrodes can record the activity of neurons in the neighborhood, on the other end they can be used to inject currents in the tissue and stimulate neurons[†].

A few steps are required to go from raw recordings to selective stimulation of neurons. I will refer to these steps as the *electrophysiology pipeline*.

A representation of the electrophysiology pipeline is displayed in Figure 2.1. The neural probe inserted in the brain tissue is surrounded by neurons[‡] (A). The electrodes on the probe record the electric potential on their surface. Neural activity produces both low frequency signals, named local field potentials (LFPs), and higher frequency components, which include the spiking activity. In this work, I will focus on the latter, assuming that the recordings (B) are already filtered with a high-pass filter. The signals recorded extracellularly contain a mixture of activity from several neurons. Hence, the first processing step required in the pipeline is to identify, or *sort*, the activity of individual neurons. This step is called *spike sorting* (C). After spike sorting, one can focus on single neurons separately and use their extracellular signature, or extracellular action potential (EAP), to extract further information from the recordings. For example, one can classify if the recorded neurons are excitatory or inhibitory (D), which is important when untangling the structure and functions of neural microcircuits. In addition, EAPs can be used to triangulate or localize the 3D position of the neuron with respect to the recording electrodes (E). The locations and type

[†]not all probes allow for electrical stimulation, as stimulation requires dedicated electronic circuitry.

[‡]as well as glia cells, but let us focus on neurons for this thesis.

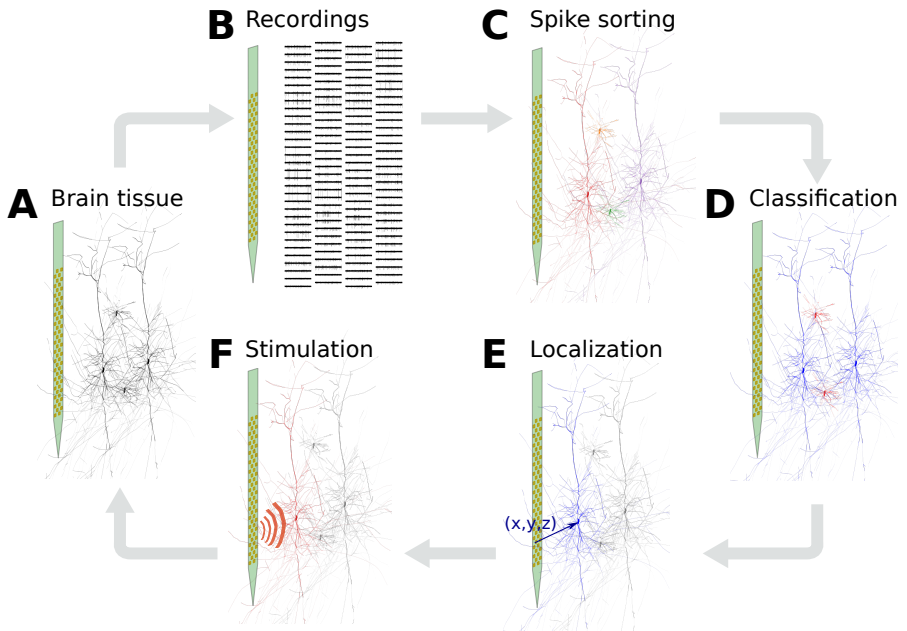


Figure 2.1: The extracellular electrophysiology pipeline.

of the recorded neurons can be used to selectively target a specific cell with electrical stimulation (F).

2.1 Spike sorting

Spike sorting is the procedure used to identify which spikes belong to the same neuron. The extracellular recordings contain in fact a mixture of activity of several neurons. The relative position between each recorded neuron and the probe, as well as the neuron morphology and electrical properties, results in different spike shapes on the electrodes. Spike sorting attempts to separate different waveforms and it outputs a set of neurons and their spike times.

Historically, spike sorting has been performed by hand. In manual spike sorting, waveforms are first detected by thresholding and aligning the signals; second, waveforms are represented in lower dimensions (either using dimensionality reduction techniques⁴⁴ or considering the amplitude of the spikes) and the electrophysiologist manually isolates (or *cuts*) clusters, i.e., tries to identify separate groups. Manual clustering has two main limitations: first it could undermine the reproducibility of the data, due to subject variability⁴⁵; second, it is not scalable to high-density probes. Manual clustering could be manageable, in terms of time, for up to a few tens of electrodes, but it quickly becomes extremely hard, if not impossible, when the number of electrodes reaches

larger numbers.

In recent years, several semi-automatic and automatic spike sorters have been developed to alleviate these problems. Spike sorting algorithms^{46–48} attempt to separate spike trains of different neurons (also called units) from the extracellular mixture of signals using a variety of different approaches. Although automatic spike sorting methods were first developed in the 80's, in recent years, probably due to the above-mentioned limitations of manual spike sorting, there has been a boost in the development and sharing of spike sorting software solutions.

In the following paragraphs, I will briefly introduce the principles of the different approaches to spike sorting. While the most widely used strategies are clustering-based and template-matching, I will also introduce a spike sorting strategy based on independent component analysis (ICA – Section 4.1), because I have used it for Paper III and Paper IV.

Clustering-based approaches Clustering-based approaches are inspired by the manual clustering steps described above (Figure 2.2A). Filtered signals are first thresholded to detect putative spikes. The detected waveforms are aligned and projected on lower dimensional space, using for example principal component analysis (PCA)⁴⁴ or wavelet features⁴⁹, and then automatic clustering algorithms are used instead of manual cutting.

While some solutions focus on low-channel count probes^{49,50}, mainly used on epileptic human subjects, other approaches attempt to scale up the algorithms for higher channel counts^{51–53}. Clustering-based approaches can suffer from over-splitting of units, i.e., the cluster corresponding to a single unit is split into multiple clusters. An automatic curation step is therefore applied by some algorithms⁵⁴ to merge putative over-split units. In addition, some solutions have mechanisms to correct for drift^{52*}. Others use an estimate of the spike location, computed for example as the center of mass of the peak amplitude, as a clustering dimension^{52,53}, in order to improve the clustering performance.

One of the main and well-known problem of clustering-based approaches is overlapping spikes. While events that are overlapping in time, but not in space, can be handled by considering the spatial location of the detected waveform^{23,53} or by applying a spatial mask to the feature set⁵¹, the occurrence of spatio-temporal overlapping events can result in a distortion of the waveforms. This is due to the spatial summation of two (or more) waveforms, and may confuse the clustering algorithm. In order to alleviate this problem, template-matching solutions have been proposed.

Template-matching approaches Although the first template-matching solutions predate clustering-based approaches⁵⁵, these methods have recently undergone a renewed interest, as high-density probes greatly increase the occurrence of spatio-temporal overlapping events.

*drift occurs when there is a relative movement between the recorded neuron and the probe, which causes a change in the spike waveforms over time

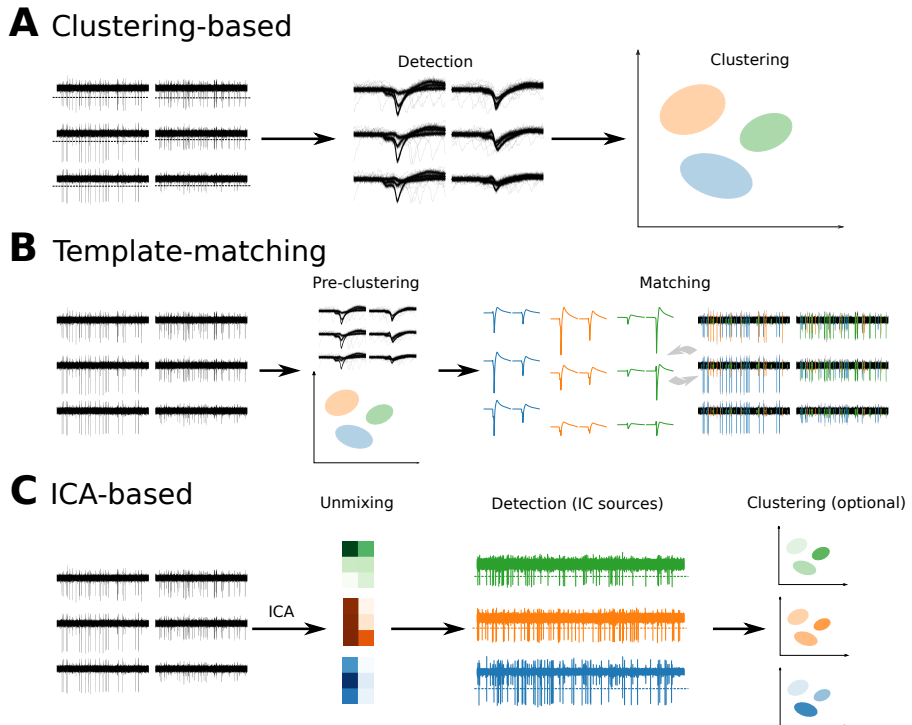


Figure 2.2: Principles of the different spike sorting approaches. The 6-channel recordings are simulated and they contain activity of three neurons. (A) Clustering-based approach. The recordings are thresholded and the detected waveforms are projected onto a lower dimensional space and clustered using automatic algorithms. (B) Template-matching approach. After a pre-clustering step, in which a subset of waveforms is clustered as described in (A), the templates of the different clusters are projected back to the time domain and recursively matched to the recordings (matching) (C) ICA-based approach. The recordings are processed with ICA, and the detection and optionally clustering is performed on the independent sources, which are tuned to separate neurons.

Template-matching approaches in general assume that the recorded signals can be described as⁴⁸:

$$\mathbf{r}(t) = \sum_{ij} a_{ij} \mathbf{T}_j(t - t_i) + \mathbf{e}(t) \quad (2.1)$$

where $\mathbf{r}(t)$ is the set of recorded signals, $\mathbf{T}_j(t - t_i)$ is the template associated to a neuron j (firing at times t_i), a_{ij} is a scalar that scales the the amplitude of the template for each spike event, and $\mathbf{e}(t)$ is additive noise.

Template-matching methods try to identify templates for each neuron, and then reconstruct the signals following Eq. 2.1 (Figure 2.2B). To achieve this,

templates are first estimated by running a pre-clustering step, which corresponds to a clustering-based approach on a subset of detected waveforms. The estimated templates are then matched to the recordings $\mathbf{s}(t)$ recursively: if a good match* is found, the template is subtracted from the recording, in order to disclose underlying overlapping spikes.

Template-matching solutions^{56–59} mainly differ in terms of the clustering step for finding the initial templates, the matching algorithm, and automatic curation steps. Finally, some of the latest algorithms attempt to handle drifting recordings⁶⁰.

ICA-based approaches A third and less explored possibility for spike sorting involves the use of ICA (see Section 4.1 for a detailed description of the algorithm). ICA is an unsupervised blind source separation method that aims to find projections that make the signals more statistically independent.

Although ICA has been suggested to spike sort tetrode signals⁶¹, in combination with clustering to increase the data dimensionality⁶², and dodecatrode recordings (12 microwires)⁶³, the main limitation of ICA has been its assumption that the number of sources, i.e., neurons, is less than the number of electrodes, which is not true for tetrode and dodecatrode recordings. The development of high-density MEAs, however, could satisfy this assumption²¹. The use of ICA and some of its variants for spike sorting of high-density probes has in fact been suggested more recently^{64,65}.

The principle of an ICA-based spike sorting pipeline is displayed in Figure 2.2C. Instead of detecting spikes or finding templates, the signals are first processed with ICA. The ICA step outputs a set of *unmixing* matrices. The projections of the initial signals on these matrices are called independent components (IC) or sources. Assuming that the signals from different neurons are fairly independent, each IC source should be *tuned* to a separate neuron. Detection and optionally clustering can be then performed directly on the IC sources^{64,65}, and this process can also be applied recursively⁶⁴, similarly to template-matching approaches. ICA-based spike sorting approaches, however, have not been fully benchmarked and compared to other available solutions for high-density MEAs, and this is the main motivation for Paper III.

With the assumption of statistical independence of signals coming from different neurons, synchronous activity could be problematic. However, this issue would be an actual problem if the spiking activity of different neurons were perfectly coincident. Due to intrinsic noise and randomness in the spiking activity, this is not fortunately the case. A second potential limitation of ICA-based spike sorting is the assumption of linearity between the sources and the signals (see Section 4.1 and Eq. 4.1), i.e., the recorded signals are assumed to be an instantaneous mix of the sources. Due to the filtering properties of dendrites, there is a phase shift between extracellular spikes recorded in the vicinity of the soma and ones closer to the dendrites. This assumption might result in finding more than one source tuned to the same neuron (duplicate sources). A

*usually defined as an improvement in a cost function

possible solution to this issue is to use convolutive ICA (cICA)⁶⁵ instead of the standard ICA model. Using cICA, the recordings are described as a filtered version of the sources. However, cICA algorithms are much more complicated and slower than instantaneous ICA approaches. In order to tackle this problem, in Paper III, we perform a post-processing step on the identified spike trains to find and remove duplicates. Finally, with the increasing number of channels in the recording devices, ICA could be a too computationally intense processing step. Nevertheless, the performance of ICA could be easily improved by applying separate ICA models to subsets of adjacent electrodes in parallel, or to estimate the ICA model using a subsample of the data instead of all the recorded signals, assuming that the statistical properties are the same.

2.1.1 Challenges

Spike sorting is a very important step in extracellular electrophysiology, but despite the huge development over the past years, there are still some open challenges.

Spike sorting validation Spike sorting is unsupervised in nature, since when recording from extracellular probes ground-truth information about the underlying spiking activity is not available.

In order to validate the spike sorting output without ground truth, several *quality control* metrics have been proposed to assess the goodness of sorted units. Some of these metrics quantify biophysical properties of the sorted units, such as refractory violations and waveform amplitude distribution⁶⁶. Others more generally quantify the isolation of the clusters with several indicators, i.e., isolation distance⁶⁷, L-ratio⁶⁸, linear discriminant analysis classification (d')⁶⁶, or nearest neighbors⁵⁴. Other metrics related to noise distribution have been proposed^{54,69}, as well as stability measures⁷⁰.

While these metrics are useful to characterize spike sorting performance, they are also unsupervised and they require users to empirically set “*good*” values. Alternatively, several groups have tried to simultaneously record units both extracellularly and using patch-clamp or juxtacellular recordings^{58,71–76}. These sets of ground-truth recordings are extremely valuable for validating spike sorting algorithms. However, their main limitation is the low yield of the experiments (only one or a few cells can be patched simultaneously), resulting in a limited validation capability.

A third validation approach relies on the use of simulated data⁷⁷. This approach can range from injecting simulated waveforms to real recordings, a so-called hybrid approach^{51,57}, to the generation of entirely simulated recordings^{78–80}. The latter approach provides full ground-truth information and it can therefore help to benchmark spike sorting algorithms. However, the question of how *realistic* these simulated recordings is still open.

Paper I and Paper II try to alleviate the problems related to spike sorting validation. In Paper I, we present a Python-based simulator of extracellular spiking activity, called MEArec. MEArec, with its easy-to-use

design, speed, controllability, and capability of reproducing challenging properties of extracellular recordings, can provide high quality ground-truth recordings for spike sorting development and evaluation.

In Paper II we introduce SpikeInterface, an open-source and unified framework for spike sorting. SpikeInterface is a powerful and comprehensive software capable of loading several file formats, running a multitude of available spike sorters, and can validate and compare the spike sorting output. Within SpikeInterface, users can compute several quality metrics available in the literature for unsupervised validation, as well as compare spike sorting output with ground-truth recordings, for example generated by MEArec.

Challenging aspects of extracellular recordings Some intrinsic aspects of spiking activity and extracellular recordings can be challenging for spike sorting algorithms.

One of the main challenges can be identifying bursting units. When neurons burst (i.e., they fire quick sequences of action potentials), the underlying dynamics of the spike generation change³⁰, causing a modulation of the spike shape, with lower amplitudes and wider waveforms^{76,79}. As a result, bursting neurons are harder to correctly identify⁴⁷, and may be over-split. However, bursting neurons can be identified *a posteriori* from their cross-correlograms, and merged using manual curation software^{81,82}. Some algorithms also implement specific steps to identify and automatically merge bursting units⁵⁴.

A second challenge is the occurrence of spatio-temporal overlapping spikes, which can distort the recorded waveforms. However, the use of template-matching approaches and the higher spatial density of probes should alleviate this problem.

A third complicated feature of extracellular recording is drift. Drift occurs when there is a relative movement between the probe and the neural tissue. Drift can result from the slow relaxation of the tissue after a probe insertion (slow drift), or from movement artifacts that cause a quicker shift of the tissue with respect to the probe (fast drift)⁶⁰. Some recent algorithms were specifically designed to identify and correct for drifting artifacts^{52,60}.

The presence of noise artifacts can also be challenging for spike sorting algorithms, especially when recording from freely moving animals. Artifacts can be related to shocks, mastication, and grooming⁸⁰ and they are usually detected as spikes because of their large amplitude. However, artifact clusters could be automatically rejected based on the above-mentioned quality metrics²³. Alternatively, due to the large availability of manually curated spike sorting outputs in which artifact-clusters are rejected, another possibility is to build machine learning systems to automatically reject noisy clusters.

In order to provide a faithful validation of spike sorting algorithm, the MEArec simulator (Paper I) is capable of reproducing bursting behavior, drift, and to precisely control the occurrence of spatio-temporal collisions between spikes.

Performance With very high density probes now reaching a channel count of several hundreds/thousands of electrodes^{15–18,20–24}, and likely to reach several thousands of simultaneously recorded channels in the near future, scalability is an essential requirement to keep in mind. Strikingly, an algorithm developed in 2016 and classified as for “large and dense” probes (up to 64 channels), has become outdated in less than 4 years⁵¹.

Recent methodologically sophisticated and innovative solutions, have been developed for highly parallel clusters^{53,54,58,59} and for graphical processing unit (GPU) accelerated hardware^{52,57,60}.

Online spike sorting While most of the available algorithms are designed for offline use^{51–54,57–59}, the capability of detecting and sorting spikes online is very powerful for closed-loop intervention with the neural tissue⁸³.

While a few solutions have been suggested for online spike sorting of low-channel probes^{63,84}, the extension to high-density probes is not trivial.

The matching phase of template-matching approaches, after an offline estimate of the templates, can be applied online^{85–87}. However, the non-stationarity of the signals, due to drifts or non-stationary activity, may require a re-estimation of the templates over the course of an experiment⁸⁵.

Alternatively, adaptive algorithms could be developed to track non-stationarities of the signals. In Paper IV, we introduce an online spike sorting method based on adaptive ICA^{88–90}, that has the potential to track non-stationarity of the signals in real-time. However, further validation for this approach is required.

2.2 Cell-type classification

In order to understand how the brain works as a whole, we first need to identify the roles of its components. To this end, the characterization of neuronal cell types based on extracellular recordings is essential to study the function of different neurons in computations.

Cell types can be defined by several different aspects, including gene expression, morphology, electrical properties, and connectivity^{32,67,91,92}. However, the main distinction between cell types happens at the connectivity level: neurons that elicit an excitatory post-synaptic potential (EPSP) are said to be *excitatory*; conversely, neurons that cause an inhibitory post-synaptic potential (IPSP) are classified as *inhibitory*. Excitatory and inhibitory cells have very distinctive roles in neural circuits. Excitatory cells mainly provide local recurrent and long-range projections⁹³, and inhibitory cells play a modulatory and balancing role⁹⁴. It is therefore important to be able to identify these different cell types from their extracellular signatures, in order to better understand the underlying neural mechanisms.

When measuring from extracellular electrodes, one can *putatively* classify cells as being excitatory or inhibitory. Based on the extracellular action potential shape, units are usually classified as regular spiking (RS) or fast spiking (FS)

cells^{95–97}. RS cells exhibit a broader action potential and are regarded as excitatory pyramidal cells; FS cells have a narrower waveform and are generally considered to be inhibitory neurons. The separation of these two classes has historically been performed by extracting waveform features from the spike sorted spikes, and then applying unsupervised clustering techniques. Some of the commonly used features computed from the waveforms include the 1) trough-to-peak width, 2) full-width half maximum, 3) half-amplitude duration, and 4) peak-amplitude asymmetry^{96–98}. Moreover, properties derived from the spike statistics such as firing rate, auto-correlogram shape, inter-spike-interval distributions, and bursting activity can exhibit differences between the two cell-types⁹⁷.

The above-mentioned methods mainly consider the waveform recorded on the electrode with the largest spike amplitude. However, the use of high-density MEAs can provide much more spatial information that can be used to refine and improve the classification. For example, in a recent study⁹⁹, several features were computed from the spatio-temporal signature of the waveforms. The authors showed that high-density silicon probes, in their case Neuropixels²¹, allow for the tracking of backpropagating action potentials, which are unique to pyramidal cells. Moreover, the use of the rich spatial information enabled the authors to identify two separate classes of RS neurons in visual cortex. In Paper V, we show that the use of features extracted from high-density probes improves the classification accuracy over conventional clustering approaches.

The validation of cell-type classification is challenging, but there are a few solutions. When measuring the activity of multiple cells from the same region, some of those might be mono-synaptically connected. In that case, the analysis of the cross-correlograms can suggest whether a cell is excitatory, inhibitory, or even if there is reciprocal excitatory-inhibitory interaction between a pair of cells^{96–98}. However, the rate of monosynaptic connections in recorded neurons is usually very low ($\sim 0.2\%$)^{96,97}. Another viable alternative for the validation of cell-type classification is the use of optotagging¹⁰⁰. This technique consists of using optogenetics¹⁰¹ to target specific sub-populations of neurons. Optogenetics allows one to make neurons express ion channels which can be activated by light (opsins). Using genetic techniques or transgenic lines, these light-sensitive channels can be expressed only in sub-populations of neurons. Shining light with wavelengths matching the activation spectrum of the opsin will activate only neurons belonging to the *tagged*, or labeled, cell types. However, optogenetics adds another invasive component to the recording approach, as it requires transgenic lines or viral injections to express the light-sensitive channels. Moreover, the presence of these additional ion channels might affect the cell dynamics, hence slightly changing the recorded waveform.

2.3 Cell localization

The recorded extracellular potentials can also be used to localize or triangulate the position of a neuron with respect to the probe. The capability to reconstruct

2. The extracellular electrophysiology pipeline

neuron locations can shed light on the microscale organization of neural circuits¹³. Moreover, precise cell localization could improve targeted stimulation¹⁰² and automatic positioning of recording electrodes for patch-clamp experiments¹⁰³.

Localization from extracellular potentials is inherently an ill-posed problem, because of the ambiguity arising from measuring the electric potential in only a limited number of locations when neuronal currents are generated from complex neuronal morphologies. While first approaches for neuron localization used tetrodes^{104,105} or low-density polytetrodes^{13,106,107}, the development of high-density MEAs can provide much higher spatial resolution, improving localization accuracy¹⁰³ and even enabling tracking sub-cellular mechanisms, such as axonal propagation¹⁰⁸ and back-propagating action potentials⁹⁹.

Most of the methods developed for localization attempt to solve the inverse problem. Given the set of recorded potentials \mathbf{V}_r (usually observed at the spike peak^{14,109,110}) and a forward model \mathcal{F} that describes the spike generation, the solution of the inverse problem tries to estimate the soma position (x_s, y_s, z_s) that minimizes the error between the recorded potentials \mathbf{V}_r and the potentials predicted by the forward model \mathbf{V}_f . Mathematically, the inverse problem can be formulated as:

$$\begin{aligned} \mathbf{V}_f &= \mathcal{F}(x_s, y_s, z_s, \mathbf{P}) \\ \underset{x_s, y_s, z_s, \mathbf{P}}{\operatorname{argmin}} & (\mathbf{V}_f - \mathbf{V}_r)^2 \end{aligned} \tag{2.2}$$

where \mathbf{P} is the set of additional parameters to x_s, y_s, z_s of the model \mathcal{F}^* .

The models chosen for the spike generation are usually very simple, in order make the solution unique. Examples of models used in previous studies are monopole current-sources^{13,109,110}, dipole current-sources^{13,104,105}, multi-pole current-sources¹⁰⁶, as well as line-source models¹⁰⁷, and more recently ball-and-stick models¹¹¹.

One of the main limitations of solving the inverse problem to tackle neuron localization is that the models chosen to solve the inverse problem are often too simple to grasp complex spike waveforms (e.g. monopolar or bipolar current-source models) or they are tuned to a specific cell types (ball-and-stick model for the pyramidal morphology¹¹¹). Therefore, in Paper V, we suggest a supervised method in which detailed simulations are used as ground truth to train deep learning models.

A challenging aspect of neural localization is arguably its validation. It is in fact experimentally very challenging to accurately measure the correct position of the soma with respect to an extracellular device^{74,75}; therefore, detailed computational neuronal models are usually used and treated as ground truth to evaluate the accuracy of the localization methods^{106,107,111}. An alternative mean for validation is to combine extracellular recordings and imaging¹¹² to precisely co-localize the cells and validate localization performance.

*for example, for a monopole current source model \mathbf{P} corresponds to the value of the monopole current.

Finally, most of the above-mentioned approaches aim to identify the neuronal soma. However, recent experimental and computational findings suggest the axon initial segment (AIS), not the soma, is the main contributor to the extracellular action potential^{113,114}. As the AIS can be tens of μm away from the soma, this discrepancy should be taken into account in developing localization methods.

2.4 Electrical stimulation

Neurons are excitable cells. Almost by definition, this means that they can be excited and one way of doing so is to use electrical stimulation from extracellular electrodes.

Stimulation of neural tissue has been successfully used for decades in several biomedical applications, including cochlear implants for hearing restoration¹¹⁵, retinal implants for vision improvements¹¹⁶, and deep brain stimulation (DBS) to treat Parkinson's disease¹¹⁷. Moreover, electrical stimulation is used to repair sensory perception in paralyzed patients¹¹⁸, and to restore walking in tetraplegic patients¹¹⁹.

Another interesting application of electrical stimulation is in closed-loop settings to facilitate neuroplasticity. Following the Hebbian principle "*neurons that fire together, wire together*", if a neuron is stimulated whenever another neuron connecting to it, directly or indirectly, fires, their synaptic connection is strengthened. This idea has inspired several studies to, for example, rewire motor neural connections^{83,120,121}, or interact with spatial memories^{122,123}

The application of extracellular potentials at a neuron's membrane can depolarize or hyperpolarize the membrane potential. If the depolarization is strong enough, this can trigger action potentials (see Section 3.3 for a modeling perspective). A multitude of experimental studies over the last century have characterized the key aspects of electrical stimulation for neural excitation¹²⁴:

- **Excitable regions:** the most excitable parts of a neuron, and therefore the most likely to be activated by electrical stimulation, are the axon initial segment and the nodes of Ranvier, where the density of sodium channels is the highest^{125,126}.
- **Current-distance relation:** the amount of current required to elicit an action potential (threshold current) in a neuron is proportional to the square of the distance from the electrode tip when simulating with a monophasic constant pulse¹²⁷.
- **Strength-duration curve:** using similar stimulation settings, the threshold current is related to the pulse duration by^{127,128}:

$$I_{th} = I_{rh} \left(1 + \frac{t_{ch}}{d} \right) \quad (2.3)$$

2. The extracellular electrophysiology pipeline

where I_{rh} is the *rheobase* current, i.e., the minimum current value capable of eliciting an action potential; d is the pulse duration; and t_{ch} is the neuron *chronaxie*, which is the pulse duration at $2I_{rh}$.

- **Charge balance:** electrical stimulation is usually applied by means of charge balanced biphasic currents¹²⁹ or balanced voltage pulses¹⁰², in order to preserve the safety of the tissue and the electrode-tissue interface¹³⁰.

Most of what we know about electrical stimulation comes from experimental studies. More recently, several groups have investigated the effect of electrical stimulation by means of computational studies, both using analytical approaches^{126,131–134} and more advanced finite element methods^{135–139}. Theoretical and numerical simulations can provide a controlled framework to investigate the effects of electrical stimulation¹²⁷.

Despite the widespread use of electrical stimulation in the clinical and academic fields, there are still some open challenges and limitations.

The first limitation is physical: when a current is injected in the brain, i.e., a conductive medium, it spreads radially in all direction. This phenomenon can result in the activation of a larger area (and more neurons) than desired^{124,127,140}. This can clearly be a problem for applications that require targeted stimulation, such as the above-mentioned closed-loop neuroplasticity experiments as well as visual and auditory prostheses applications. The use of high-density probes can in part alleviate this problem. *In vitro* studies have in fact shown that it is possible to selectively target specific regions of a neuron^{102,141}. In Paper VI, we exploit the high spatial density of MEAs for optimizing targeted spatial patterns to increase stimulation selectivity.

The second challenge is more related to technology and electronics. In order to apply electrical stimulation from the electrodes, neural probes require specialized stimulation circuits^{18,28,102}, which occupy space and consume power. Although the integration of such circuits is common for *in vitro* applications^{15,17,18}, their inclusion in active CMOS probes designed for *in vivo* experiments is still limited due to size and tissue heating constraints. Alternatively, extracellular stimulation *in vivo* is possible using passive electrodes (both silicon probes or microwires) and external systems capable of simultaneous recording and stimulation*.

*e.g http://intantech.com/stim_record_controller.html

Chapter 3

Computational models

“All models are wrong, but some are useful.”

— George Box

In this chapter, I will present a brief introduction of the computational models used in this thesis, with their main principles, applications, and limitations.

3.1 Multi-compartment models

Multi-compartment models are representations of a neuron as an electric circuit. Neuron morphologies, extracted from microscopy^{30,31} or built using simplified assumptions¹⁴², are modeled by many small segments and represented as electrical circuits. Each segment implements a membrane model, which consists of several ion channels represented as conductance-based models.

Conductance-based models describe the biophysical properties of an excitable membrane. In this model, the contribution of an ion channel type is represented as an overall conductance. For example, for an ion channel x (e.g. sodium, potassium, calcium), the current density flowing out of the membrane is:

$$i_x = g_x(V - E_x) \quad (3.1)$$

where i_x is the ionic current density generated by the x ion [$mA\ cm^{-2}$], E_x is the reversal potential for the ion, i.e., the potential at which electric and diffusion currents are balanced⁶, V is the membrane potential (the difference between intracellular and extracellular potential: $V = \phi_{in} - \phi_{ex}$), and g_x [$mS\ cm^{-2}$] is the ion channel conductance per unit of area[†].

Conductances can be passive (constant), representing leaky currents that traverse the membrane, or active. The leak current density is usually defined with respect to the resting potential V_r , i.e., $i_l = g_l(V - V_r)$. Active channels enable to reproduce the highly non-linear dynamics of neurons, such as the

[†]Note that in this derivation all spatial units are assumed to be in cm

3. Computational models

generation of an action potential. Famously, Hodgkin and Huxley⁵ were able to quantitatively reproduce an action potential by modeling active Na^{2+} and K^+ channels. For active channels, the conductance is dependent on the membrane potential ($g_x = g_x(V)$). The membrane of an excitable cell can be represented as a capacitor, as the lipid bilayer that it is made of keeps ions apart, similarly to a capacitance. The equation governing the current density of a single-compartment can be written using Kirchoff's current law:

$$c_m \frac{dV}{dt} = - \sum_x g_x(V - E_x) - g_l(V - V_r) + i_{ext} = -i_{ion} - i_{leak} + i_{ext} \quad (3.2)$$

where c_m [$mF\text{ cm}^{-2}$] is the membrane capacitance, i_{ext} [$mA\text{ cm}^{-2}$] is an external current density entering the membrane, for example a synaptic input or a stimulation current. Single-compartment models do not include morphological information as they represent a so-called *point neuron* (Figure 3.1A). In order to build a morphologically detailed version of a neuron, multi-compartment models can be used.

Multi-compartment models consist of a set of connected single-compartment models. Adjacent compartments, or segments, are connected via internal resistors, which model the resistance of the intracellular cytosol. While single-compartment models are governed by a single equation (Eq. 3.2), when assembling multiple compartments the set of equations become coupled.

Considering a non-branching neuronal segment and assuming a constant cylindrical section (Figure 3.1B), a compartment length of δx , and an axial conductivity of σ_i [$mS\text{ cm}^{-1}$], the current densities between compartments $i - 1$ and i , and between compartments i and $i + 1$ can be written as:

$$\begin{aligned} i_{i-1,i} &= \frac{\sigma_i}{\delta x} (\phi_{in,i-1} - \phi_{in,i}) \\ i_{i,i+1} &= \frac{\sigma_i}{\delta x} (\phi_{in,i} - \phi_{in,i+1}) \end{aligned} \quad (3.3)$$

Applying Kirchoff's current law, we can relate the transmembrane current density flowing out of the i -th compartment i_i to the axial currents from adjacent sections (for simplicity the additive external current i_{ext} is neglected):

$$\begin{aligned} I_i &= A_m \left[c_m \frac{\partial V}{\partial t} + i_{ion} + i_{leak} \right] = A_c (i_{i,i+1} - i_{i-1,i}) \\ &= A_c \left[\frac{\sigma_i}{\delta x} (\phi_{in,i-1} - \phi_{in,i}) - \frac{\sigma_i}{\delta x} (\phi_{in,i} - \phi_{in,i+1}) \right] \\ &= A_c \frac{\sigma_i}{\delta x} (\phi_{in,i+1} - 2\phi_{in,i} + \phi_{in,i-1}) \end{aligned} \quad (3.4)$$

where $A_c = \pi/4 d^2$ is the cross section area and $A_m = \pi d \delta x$ is the membrane area of the compartment, and d is the diameter of the neurite. Differently from Eq. 3.2, the derivative notation is partial because V is now function of time and

space. Let now $C_m = c_m A_m$ be the membrane capacitance, $G_a = A_c \sigma_i / \delta x$ the axial conductance, $I_{ion} = A_m i_{ion}$ the ionic current, and $G_l = A_m g_l$ the leak conductance of the membrane. When $\delta x \rightarrow 0$, the right hand-side of Eq. 3.4 becomes the second spatial derivative of ϕ_{in} . Since $\phi_{in} = V + \phi_{ex}$, we can reformulate as:

$$C_m \frac{\partial V}{\partial t} + I_{ion} + G_l(V - V_r) = G_a \frac{\partial^2(V + \phi_{ex})}{\partial x^2} \quad (3.5)$$

where the dependence of V , I_{ion} and ϕ_{ex} on (x, t) is not shown for clarity. To further simplify, we can rewrite with respect to $V_m = V - V_r$, define $\tau = C_m / G_l = c_m / g_l$ (membrane temporal constant) and $\lambda = \sqrt{G_a / G_l} \delta x = \sqrt{(d\sigma_i) / (4g_l)}$ (membrane spatial constant). Finally, we can assume that the extracellular potential is constant (and therefore its second derivative is null):

$$\tau \frac{\partial V_m}{\partial t} - \lambda^2 \frac{\partial^2 V_m}{\partial x^2} + \frac{i_{ion}}{g_l} + V_m = 0 \quad (3.6)$$

This partial differential equation (PDE) is the **cable equation** and it is a fundamental equation in computational neuroscience^{6,42,143}.

While Eqs. 3.5 and 3.6 assume a homogeneous neurite, axons can be covered by myelin sheets to increase the transmission speed. In this case, the continuous solution can be substituted by a discrete solution, in which the values of the membrane capacitance and resistance are fit to the properties of myelinated sections and nodes of Ranvier.

Importantly, the cable equation assumes the extracellular resistance to be 0 (in Figure 3.1 there is in fact no resistor between the extracellular nodes), i.e., the extracellular potential to be constant. This is done mainly for two reasons: on one hand the intracellular resistance is much larger than the extracellular one; secondly, assuming a null extracellular resistance simplifies extensively the solution of the cable equation. However, when a non-null resistance is incorporated, ephaptic effects, i.e., the contribution of the extracellular potential to neuronal dynamics, can be studied^{42,144-146}.

Software The solution of the cable equation for complex morphologies and conductances is far from trivial, especially considering the highly non-linear dynamics of ion channels^{5,30}. Therefore, simulators have been developed, such as NEURON^{147,148} and GENESIS¹⁴⁹, with high-level definition languages and graphical user interfaces. Moreover, NEURON has a Python Application Programming Interface (API)¹⁵⁰. NEURON, through the LFPy environment introduced in the next section, has been used extensively throughout this thesis.

3.2 Extracellular potential

After computing the internal dynamics of the neuron, we need to compute what is actually measured by neural probes, i.e., the extracellular potential.

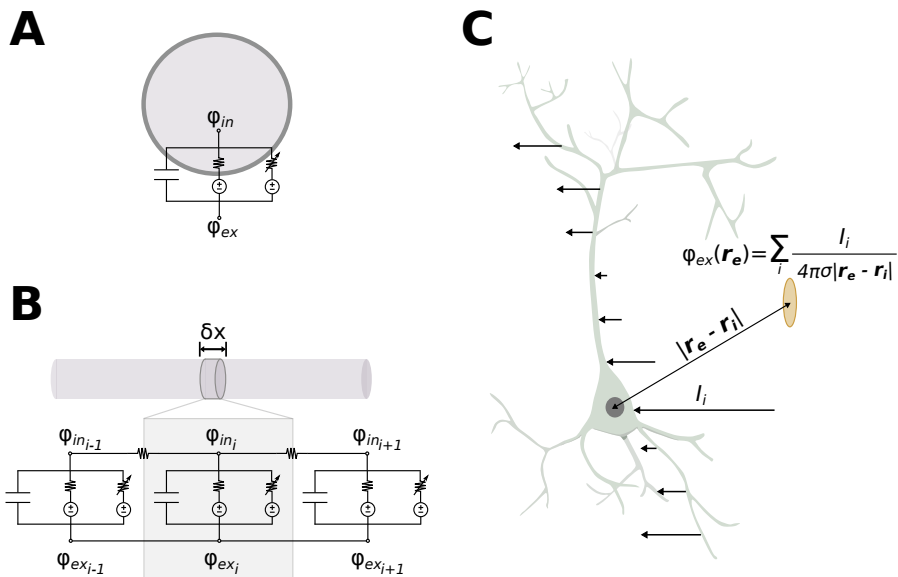


Figure 3.1: (A) Point neuron model. The membrane is modeled with a capacitance, a leaky resistor, and active ion channels represented as voltage-dependent resistors. (B) Multi-compartment model. A neuron is split in small compartments. Each compartment implements a membrane model and it is connected to adjacent compartments with resistors that model the intracellular cytosol. (C) Extracellular potentials. The extracellular potential ϕ_{ex} is computed as the sum of transmembrane currents' contributions. (The drawing of the pyramidal cell is by Federico Claudi, and it is available at scidraw.io).

From the solution of the cable equation, which is the set of membrane potentials for each neuronal compartment $V_i(t)$, one can easily obtain the transmembrane currents I_i for each compartment (Eq. 3.4).

One of the most commonly used and computationally efficient way to compute the extracellular potential generated by neurons uses *volume conduction theory*¹⁵¹. Considering a quasistatic approximation of Maxwell's equations and assuming a conductive, isotropic, homogeneous, linear, and infinite medium, the extracellular potential generated by a point current source $I_s(t)$ at position \mathbf{r}_s can be computed at any point \mathbf{r} (except for \mathbf{r}_s) as:

$$\phi_{ex}(\mathbf{r}, t) = \frac{I_s(t)}{4\pi\sigma|\mathbf{r} - \mathbf{r}_s|} \quad (3.7)$$

In this case ground ($\phi_{ex} = 0$) is set *far away* from the current and σ is the conductivity of the medium [$mS\ cm^{-1}$]. Straightforwardly, one can

then consider each transmembrane current I_i as a point source located at the center of its neuronal compartment \mathbf{r}_i , and compute the extracellular potential measured by an electrode in position \mathbf{r}_e as a linear sum of the individual current's contributions^{42,143,152}:

$$\phi_{ex}(\mathbf{r}_e, t) = \frac{1}{4\pi\sigma} \sum_i \frac{I_i(t)}{|\mathbf{r}_e - \mathbf{r}_i|} \quad (3.8)$$

Eq. 3.8 assumes that transmembrane currents are generated from a single point and it is therefore referred to as *point-source* approximation^{143,152} (Figure 3.1C). A more realistic approximation treats the neuronal segments as lines rather than points (*line-source* approximation). In this case the transmembrane currents are evenly distributed along the compartments. Integrating Eq. 3.8 over the segments' axes, the potential can be computed as:

$$\phi_{ex}(\mathbf{r}_e, t) = \frac{1}{4\pi\sigma} \sum_i I_i \int_{\mathbf{r}_{i-1/2}}^{\mathbf{r}_{i+1/2}} \frac{d\mathbf{r}}{|\mathbf{r}_e - \mathbf{r}|} \quad (3.9)$$

where $\mathbf{r}_{i-1/2}$ and $\mathbf{r}_{i+1/2}$ represent the initial and final position of each compartment.

In both Eq. 3.8 and 3.9, the electrode is represented as a three-dimensional point \mathbf{r}_e . In order to consider the spatial extent of the recording contact, one can compute the potential on several points belonging to the electrode surface and average their values (*disk-electrode* approximation¹⁵²).

The above-described formulations are based on several assumptions. First of all the conductivity of the medium is assumed to be scalar, hence neglecting capacitive properties of the tissue. This assumption seems however to be well justified for relevant frequencies in extracellular recordings^{151,152}.

Second, the medium is assumed to be isotropic, but this assumption is harder to relax. In the neural tissue, in fact, the presence of oriented pyramidal cells makes conductivity anisotropic¹⁵³. Anisotropy in the tissue can be accounted for with analytical solutions^{41,154}.

Finally, the extracellular milieu is assumed to be homogeneous (without discontinuities) and infinite. This is clearly a stronger assumption, considering that in order to measure the electric potentials generated by the neurons, we insert a probe in their vicinity. In case of *in vitro* preparations, in which cell cultures or slices lie on a planar MEAs, the effect of the electrode plane and of the discontinuity between the neural tissue and the saline solution can be modeled also analytically, using the method of images^{41,103}. For more complicated cases, one can use numerical solutions, such as finite element methods (FEM) (Section 3.4). In Paper VII, for example, we used FEM to simulate the effect of *in vivo* neural probes on the recorded potentials.

Software There are some available and open-source software to compute the extracellular potentials generated by neural activity. BIONET¹⁵⁵, developed by the Allen Institute for Brain Science, is a software for large-scale simulations.

3. Computational models

It includes a wrapper to the NEURON environment and it enables to compute extracellular potentials arising from network simulations using the line-source approximation (Eq. 3.9). In this thesis, I used LFPy, developed by our group^{152,154}, for simulating extracellular potentials. LFPy also provides a NEURON wrapper and it has a simple API for the definition of cells and electrodes. The new version of LFPy¹⁵⁴ also includes computationally-efficient forward modeling schemes for the calculation of electroencephalography (EEG) and magnetoencephalography (MEG).

3.3 Models of electrical stimulation

In order to model activation of neurons from electrical stimulation, we can combine principles from Section 3.1 and Section 3.2. The models presented in this section have been mainly derived for axons, as axons are the most excitable neuronal parts¹²⁵ (alongside with the axon initial segment - see Section 2.4).

Stimulation can trigger action potentials by imposing an extracellular potential at the neuron’s membrane. In order to compute these potentials from a stimulating electrode at position \mathbf{r}_e , we can consider, in a first approximation, the electrode as a point current source and use Eq. 3.7.

Let us now get back to the cable equation. During the derivation, we set the extracellular potential to be constant, but this assumption clearly needs to be revisited as we are attempting now to simulate the effect of extracellular potentials on the neuron dynamics. If we relax this assumption, we can rewrite Eq. 3.6 as^{129,133}:

$$\tau \frac{\partial V_m}{\partial t} - \lambda^2 \frac{\partial^2 V_m}{\partial x^2} + \frac{i_{ion}}{g_l} + V_m = \lambda^2 \frac{\partial^2 \phi_{ex}}{\partial x^2} \quad (3.10)$$

From this formulation, one can see that the *source term* or *force function* of the differential equation (the right-hand side) is proportional to the second spatial derivative of the extracellular potential ϕ_{ex} . The $\tau \frac{\partial V_m}{\partial t}$ term defines the capacitive currents, and $\lambda^2 \frac{\partial^2 V_m}{\partial x^2}$ term represents the longitudinal currents.

While Eq. 3.10 could be solved numerically, several studies tried to propose estimators for neural activation from extracellular stimulation, in order to avoid to solve Eq. 3.10 and still have “*first impression of the influence of an applied electric or magnetic field on a target neuron.*”¹²⁶.

One of the first estimator proposed in literature is the *activating function*^{126,132,156}. If we assume that the neuron is at rest, the spatial derivative of V_m (longitudinal currents) can be neglected since V_m is relatively constant. Then, the polarization at the steady state will be proportional to the second spatial derivative of the extracellular potential along the axon:

$$V_m \propto AF = \lambda^2 \frac{\partial^2 \phi_{ex}}{\partial x^2} \quad (3.11)$$

The activating function AF is convenient because it can estimate neural activation only by knowing the extracellular potential and the neuronal

morphology. Even neglecting the λ term, which might be hard to correctly estimate, the AF can tell whether a region is hyperpolarized ($AF < 0$) or depolarized ($AF > 0$). The AF has been used in Paper VI to predict neural activation in order to optimize electrical stimulation patterns involving multiple electrodes using MEAs.

It is not clear, however, if the longitudinal currents can be safely ignored, as they are scaled by λ^2 , so their contribution depends on this parameter. Secondly, the AF wrongly predicts that an axon is not activated by an applied linear potential. While this might be true for infinitely long axons¹³², in practice neurites have a finite extension and edge effects need to be considered¹³³.

Another study¹⁵⁷ suggests that when λ is large enough, then the longitudinal currents are more prominent than the capacitive currents and cannot be neglected. At the steady state, Eq. 3.10 can then be approximated as:

$$-\lambda^2 \frac{\partial^2 V_m}{\partial x^2} + V_m = \lambda^2 \frac{\partial^2 \phi_{ex}}{\partial x^2} \quad (3.12)$$

When λ is “small enough”, Eq. 3.13 coincides with the AF , but when λ is large the predictor of V_m becomes^{133,157}:

$$V_m \propto ME = -\phi_{ex} + \langle \phi_{ex} \rangle_L \quad (3.13)$$

The *mirror estimate* (ME) predicts the depolarization/hyperpolarization of a neurite as the opposite of the applied extracellular field (*mirror*) centered on its mean value ($\langle \phi_{ex} \rangle_L$)*.

Figure 3.2 shows the extracellular potential (ϕ_{ex}), AF and ME predictions for a cathodic (current drawn by the electrode - A) and an anodic stimulation (current injected by the electrode - B) for a linear and uniform axon. In this case, the extracellular potential can be computed analytically from Eq. 3.7 for a fiber oriented in the x direction and located at a distance d from the electrode:

$$\phi_{ex}(x) = \frac{I_s(t)}{4\pi\sigma\sqrt{d^2 + x^2}} \quad (3.14)$$

While AF and ME agree on the position and magnitude of the central depolarization/hyperpolarization, there are some differences between these two estimators in the shape of the depolarization and hyperpolarization regions.

Although estimators can provide a *feeling* about the effect of extracellular stimulation, one of their main limitation is that they do not take into account the time course of the stimulation, as they predict the neural activation generated by a constant monophasic pulse. However, the effect of the pulse waveform, e.g., biphasic pulses, or of the application of a train of stimulation pulses cannot be estimated. A more powerful, but computationally intense approach consists of solving Eq. 3.10 to simulate the entire dynamics of the neuron in response to an extracellular time-varying stimulation¹³⁴. Nevertheless, even using this approach

*the spatial mean can be computed analytically also for non-uniform fibers, accounting for different diameters and leaky properties^{133,157}.

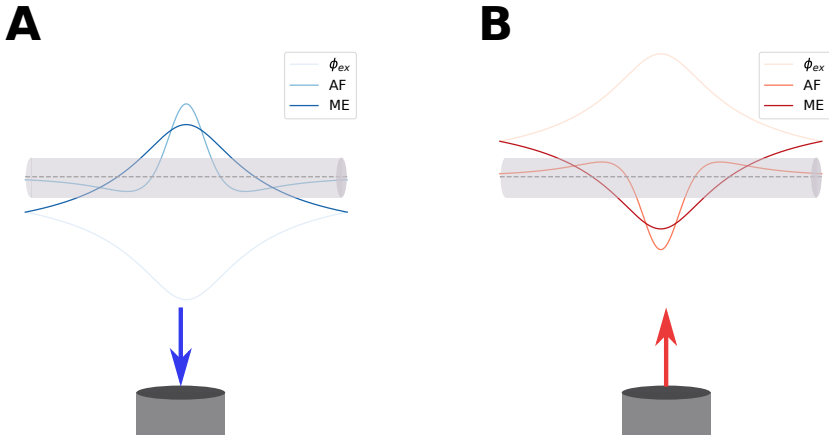


Figure 3.2: Extracellular stimulation and estimators. Extracellular potential (ϕ_{ex}), activating function (AF) and mirror estimate (ME) generated by a cathodic current (A) and by an anodic current (B) on a linear neurite.

some assumptions are made. First of all, since the cable formulation constructs a 1D problem, it assumes that the extracellular potential varies only along the main direction of the fiber, and it is constant in the orthogonal direction. Similarly, it assumes that the membrane potential is constant along the circumferential direction of the neuronal membrane. Finally, it assumes that the presence of the fiber does not affect the extracellular field^{127,139}.

Finite element methods have been used in literature for simulating extracellular potentials in highly non-homogeneous problems and pair them with the cable equation solution (hybrid approach^{42,138,139,158-160}), but the use of the cable framework cannot relax the above-mentioned assumptions. Alternatively, whole-FEM methods, with explicit representation of the extracellular and intracellular spaces, have been suggested and can be used for a detailed study of the effect of extracellular stimulation, at the cost of computational burden^{42,139,161} (see Section 3.4).

Software The NEURON framework can incorporate the effect of a non-constant extracellular potential generated by external currents using the `extracellular` mechanism, which effectively adds two layers of extracellular potentials outside the membrane and solves Eq. 3.10 instead of Eq. 3.6.

In order to compute extracellular potentials from MEAs, I developed the `MEAutility*` Python package. `MEAutility` is convenient to define MEA designs (the user can also use a wide variety of probe designs from the existing probe library), to set static or time-varying currents for each electrode, and to compute

*<https://github.com/alejoe91/MEAutility>

extracellular fields at neural locations, which can then be paired to NEURON simulations or used to compute estimators. MEAutility is used in Paper I to handle MEA designs, and in Paper VI to compute extracellular potential along the neurons.

3.4 Finite element methods

Finite element methods (FEM) are numerical methods to solve differential equations¹⁶². The basic idea is to divide the domain of interest (in our case, for example, the extracellular space and optionally the neurons) in small regions (*finite elements*) and to approximate the unknown function (e.g. the extracellular potentials) as a linear combination of simple functions defined on these regions*.

FEM can be used both to compute extracellular potentials generated by the neural activity and to simulate the effect of extracellular stimulation. Another interesting use of FEM is to validate analytical derivations with numerical simulations. For example, in Ness et al.⁴¹, a FEM simulation is used to validate the use of the method of images to analytically compute the extracellular potential for *in vitro* preparations. In Næss et al.¹⁶³ the same approach is used to correct an analytical model of the head conductivity for EEG computation.

3.4.1 Hybrid approach

The hybrid approach combines the FEM and the cable framework. It has been used both for computing extracellular potentials in non-homogeneous extracellular spaces^{138,164} and for computing the effect of stimulation, especially for the spinal cord^{158–160}.

When used for computing extracellular potentials, first neuronal dynamics are solved (for example with NEURON); then, transmembrane currents are used as force functions for a FEM simulation of the extracellular space. Conversely, when simulating the effect of external stimulation, extracellular potentials at the neuron compartments' location, generated by stimulating electrodes, are first computed using FEM; then the cable equation is solved (using the `extracellular` mechanism in NEURON).

While the hybrid approach can be useful to model complex geometries, the cable equation framework preserves its assumptions listed in Section 3.1 and Section 3.3.

3.4.2 EMI model

A second and more advanced approach consists of explicitly modeling the extracellular space, the membrane of the neuron, and the intracellular space^{139,161}. This model is also called the EMI model (Extracellular-Membrane-Intracellular)⁴²

*an extensive mathematical formulation of finite element methods is beyond the scope of this thesis.

3. Computational models

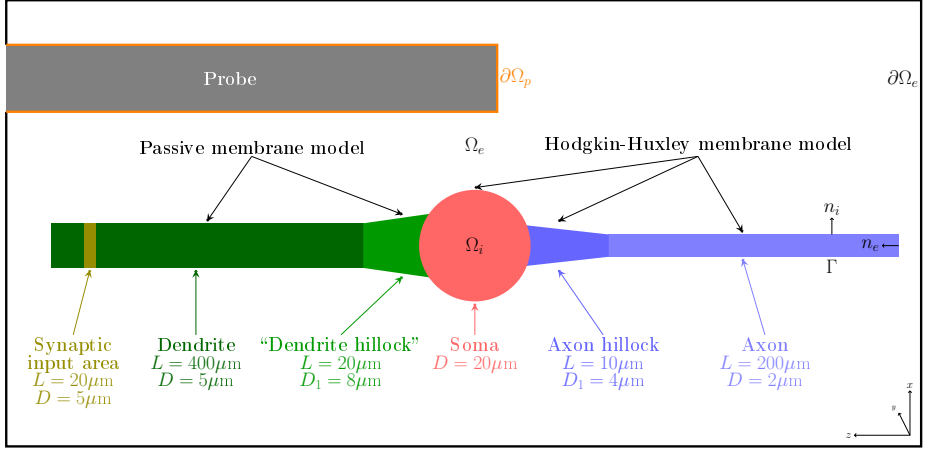


Figure 3.3: Mathematical representation of the EMI model for a simplified neuron and an extracellular probe. Figure from Paper VII.

or whole-FEM model¹³⁹. The EMI model simultaneously solves the following equations:

$$\nabla \cdot \sigma_i \nabla \phi_{in} = 0 \quad \text{in } \Omega_i, \quad (3.15)$$

$$\nabla \cdot \sigma_e \nabla \phi_{ex} = 0 \quad \text{in } \Omega_e, \quad (3.16)$$

$$\phi_{ex} = 0 \quad \text{at } \partial\Omega_e, \quad (3.17)$$

$$n_e \cdot \sigma_e \nabla \phi_{ex} = -n_i \cdot \sigma_i \nabla \phi_{in} \equiv i_m \quad \text{at } \Gamma, \quad (3.18)$$

$$\phi_{in} - \phi_{ex} = V \quad \text{at } \Gamma, \quad (3.19)$$

$$C_m \frac{\partial V}{\partial t} = -I_{ion} - I_{leak} + I_{ext} \quad \text{at } \Gamma, \quad (3.20)$$

$$(\sigma_e \nabla \phi_{ex} \cdot n_e) = 0 \quad \text{at } \partial\Omega_p). \quad (3.21)$$

With reference to Figure 3.3, Eqs. 3.15 and 3.16 are the Poisson equations for the intracellular and extracellular spaces (with conductivities σ_i and σ_e); Eq. 3.17 is the Dirichlet boundary condition to set the potential to zero *far away* from the neuron; Eq. 3.18 defines the conservation of the transmembrane current density across the membrane; Eq. 3.19 defines the membrane potential as the difference between the intracellular and extracellular potentials; Eq. 3.20 describes the membrane model (see Eq. 3.2); and Eq. 3.21, added when considering an extracellular probe in Paper VII, is the Neumann boundary condition that describes the insulating property of the probe¹³⁶.

The EMI model considers the full 3D morphology of the neuron. The EMI solution can in fact reproduce different extracellular potentials along the circumferential direction of the membrane as well as different intracellular potentials in the cross section of the intracellular space. Moreover, the EMI

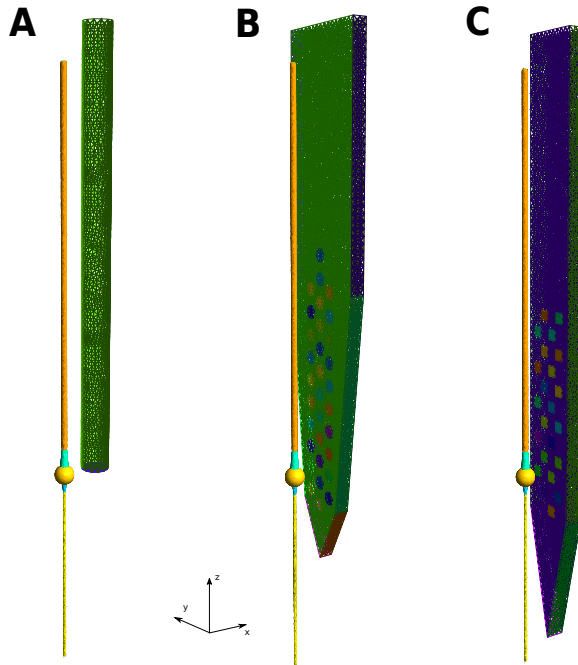


Figure 3.4: Examples of meshes for three types of extracellular probes: (A) microwire, (B) Neuronexus probe, (C) Neuropixels-like probe with 24 electrodes. Figure from Paper VII.

model does not break down the neuronal morphology into compartments, as it represents the neuron as a *continuum**. Finally, the extracellular potential is not neglected when solving the neuron dynamics, allowing for the study of ephaptic and self-ephaptic effects^{42,146}.

The main limitation of the EMI formulation, however, is that it is *much more* computationally intense than the cable approach. The hybrid approach lies somewhere in between⁴².

In Paper VII we used both the hybrid and the EMI solutions to investigate the effects of extracellular probes on the recorded potentials. Figure 3.4 shows some examples of the meshes that we used (A - microwire, B-C - MEAs) in combination with a simple ball-and-stick model of a neuron¹⁴².

Software A very common software used for FEM is COMSOL. However, it is a commercial solution and several groups have contributed with open-source software. `gmsh`¹⁶⁵ is an open-source and powerful software for generating 3D meshes of complex geometries. `FEniCS`¹⁶⁶ is a computing platform for solving

*the continuous morphology is discretized when creating the mesh, but increasing the mesh resolution can achieve higher accuracy in representing the morphology.

3. Computational models

partial differential equations in Python. It combines a high-level Python API and C++ API which makes it very accessible, versatile, and easy to use. `gms` and `FEniCS` have been used in Paper VII for generating the meshes and computing the EMI and hybrid solutions.

Chapter 4

Engineering solutions

“Engineers like to solve problems. If there are no problems handily available, they will create their own problems.”

— Scott Adams

In this chapter, I will describe the engineering and implementation solutions used in the presented papers.

4.1 Independent component analysis

Independent Component Analysis (ICA) is a signal processing method used for blind source separation and dimensionality reduction^{167–169}. In the latter case, ICA is used to find a better representation of the data making use of their statistical structure. In Paper III and Paper IV ICA has been used for blind source separation. A classic example of blind source separation is the so-called *cocktail party*.

Imagine that N guests are chatting in a bar while sipping their cocktails. Moreover, there are $M > N$ microphones located at different positions in the bar. Actually, let us simplify the problem and say that we have exactly N microphones. Each microphone records then a mixture of voices from the different guests. Let us define the speech signals coming from the guests as $\mathbf{s}(t) \in \mathcal{R}^N$ and let us assume that these signals are statistically independent, as the guests are not singing in a choir. Furthermore, we can assume that the sound propagation velocity is negligible. Therefore, we can describe the set of recorded signals $\mathbf{x}(t) \in \mathcal{R}^N$ as a linear combination, or mixture, of $\mathbf{s}(t)$:

$$\mathbf{x}(t) = A\mathbf{s}(t) \tag{4.1}$$

where A is defined as the *mixing* matrix and $\mathbf{s}(t)$ are called *sources*. Solving the ICA problem is far from trivial, as both the sources $\mathbf{s}(t)$ and the matrix A are unknown. Usually, the ICA problem is formulated as:

$$\mathbf{y}(t) \approx W\mathbf{x}(t) \quad (4.2)$$

where $\mathbf{y}(t)$ is an estimate of $\mathbf{s}(t)$ and W is the *unmixing* matrix, which is the pseudo-inverse of the *mixing* matrix A . There are different approaches for solving this problem. However, the underlying principle is to find an optimal W that maximizes the statistical independence of the estimated sources $\mathbf{y}(t)$.

In order to better explain the ICA principle, we can consider a *neuronal cocktail party*. When we insert a probe in the brain (the microphones), we will hopefully pick up the activity of several neurons (the speech signals). Let us create a simplified case to show how ICA transforms the data and how it differs from other techniques, in particular from principal component analysis (PCA). In this example we insert a two-channel probe, which records the activity of two surrounding neurons. We have to assume that the specific spike times of the two neurons are fairly independent. In the recordings, there is also some additive Gaussian background noise. For the sake of clarity, in this example we created the recordings as a linear mix of two spiking sources with some additive noise. Therefore, the recorded signals, at each time point, are instant mixes of the sources, without any phase shift. This assumption for neural signals is discussed in Section 2.1 (ICA-based approach paragraph).

Figure 4.1A shows the simulated recordings, the waveforms of the two neurons, and a scatter plot of the signals on the channel dimensions. The *directions* of the two neurons can be clearly identified.

When we apply PCA*, the signals are projected on the dimensions that maximize the variance of the data. The PCA directions are shown in Figure 4.1A as orange arrows and PCA-transformed signals, waveforms, and the scatter plot of the data on the principal directions are shown in Figure 4.1B. After the PCA transform, the data are *decorrelated*. Decorrelation implies that the covariance matrix of the variables is diagonal. Related to this, *whitening* makes the covariance matrix equal to the identity matrix (each direction is normalized). However, each principal component contains activity from both neurons, since the principal axes, displayed in Figure 4.1A as orange arrows, are *in the middle* between the two neurons' directions. This makes the PCA-transformed waveforms appear on both PCA channels.

Applying ICA to the data gives us different directions. Statistical independence is in fact different from uncorrelatedness. When two random variables x_1 and x_2 are uncorrelated, their covariance is zero:

$$E[x_1, x_2] - E[x_1]E[x_2] = Cov[x_1, x_2] = 0 \quad (4.3)$$

When two variables are statistically independent it can be shown that non-linear correlations are also zero¹⁶⁹:

$$E[h(x_1), g(x_2)] - E[h(x_1)]E[g(x_2)] = Cov[h(x_1), g(x_2)] = 0; \quad \forall h(), g() \quad (4.4)$$

*or alternatively singular value decomposition (SVD)

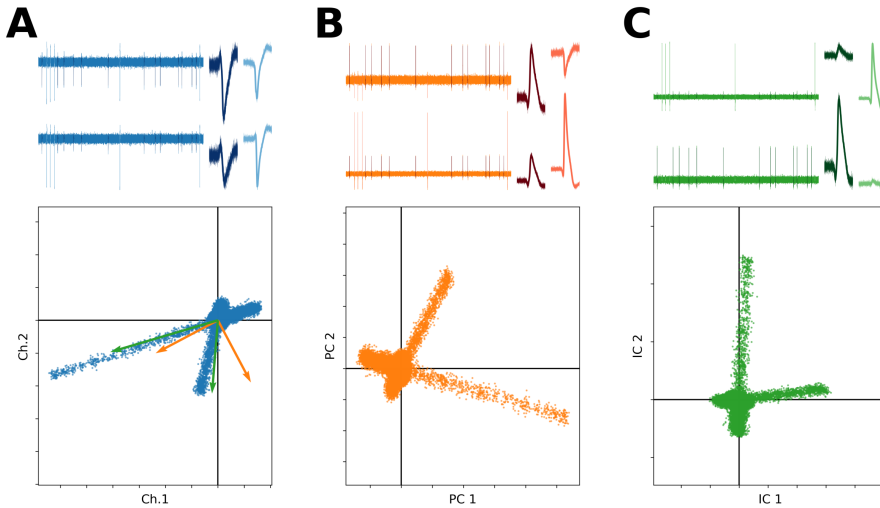


Figure 4.1: Illustration of PCA and ICA projections. (A) Top: Simulated recording and waveforms with two channels and two neurons; Bottom: data points on channel 1 - channel 2, with PCA and ICA axis in orange and green, respectively. From the waveforms and the data clouds one can see that the waveform of each unit is present on both recording channels. (B) Top: PCA-transformed signals and waveforms; Bottom: data points in the PCA space. The two units still appear on both axis. (C) Top: ICA-transformed signals and waveforms; Bottom: data points in the ICA space. The ICA axis are selectively tuned to the two units. The waveforms now mainly appear on a single ICA dimension.

Clearly, two statistically independent variables are also uncorrelated ($h(x_1) = x_1$, $h(x_2) = x_2$). ICA finds directions that are selectively *tuned* to the activity of the separate neurons in the recordings. These directions are shown as green arrows in Figure 4.1A. Figure 4.1C shows the ICA-transformed signals, waveforms, and scatter plot. In this case, each IC source is specifically tuned to one of the neuron, so that the waveform mainly appear on a single ICA direction.

There are several approaches to solve the ICA problem, i.e., to maximize the independence of the estimated sources $\mathbf{y}(t)$. ICA is solved iteratively with optimization techniques aiming to maximize a cost function that reflects the status of independence of the sources.

The independence of the sources can be computed in different ways. Independence can for instance be related to *non-gaussianity*. Due to the Central Limit Theorem, in fact, the sum of two or more independent variables makes the distribution more Gaussian. Hence, projections that maximize *non-gaussianity* also maximize independence between the variables. Measures of *non-gaussianity* include kurtosis and negentropy¹⁶⁹. The famous FastICA

4. Engineering solutions

algorithm¹⁶⁹ maximizes negentropy with a fixed-point iteration scheme, and it has been used in Paper III. Alternatively, the ICA problem can be solved by minimizing the *mutual information*, a theoretical information measure that indicates dependence between variables. The minimization of mutual information is equivalent to maximizing negentropy¹⁶⁹.

Another widely used approach for solving the ICA problem is the *infomax* principle^{168,170,171} (maximization of information flow). The learning rule for this approach using natural gradient can be written as¹⁷¹:

$$W_{n+1} = W_n + \eta [I - \mathbf{f}(\mathbf{y}_n)\mathbf{y}_n^T] W \quad (4.5)$$

where η is the learning rate and $\mathbf{f}(\cdot)$ is a non-linear function. From this formulation, one can see how the *infomax* principle is also closely related to non-linear decorrelation (Eq. 4.4), as the gradient tends to remove the non-linear correlations given by $\mathbf{f}(\mathbf{y}_n)\mathbf{y}_n^T$. It can be also proven that maximizing the information flow (in other words the joint entropy) is analogous to minimizing mutual information^{168,169}. In the *infomax* approach, the choice of the function $\mathbf{f}(\cdot)$ is important: in order to maximize the joint entropy of the sources, hence maximizing independence, the product $\mathbf{f}(\mathbf{y}_n)\mathbf{y}_n^T$ should result in a uniform distribution. Therefore, the *infomax* principle has been extended to match different distributions, such as subgaussian and supergaussian ones¹⁷¹. However, since the distribution of the sources \mathbf{y}_n is unknown *a-priori*, one can estimate it by computing the kurtosis of each source and choosing the correct non-linearity $\mathbf{f}(\cdot)$ accordingly.

From the natural gradient *infomax* learning rule in Eq. 4.5, an online recursive ICA solution – ORICA^{88–90,172} – has been developed. The adaptive learning rule reads as^{90,172}:

$$W_{n+1} = \frac{1}{(1 - \lambda_n)} W_n - \frac{\lambda_n}{(1 - \lambda_n)} \frac{\mathbf{y}_n \mathbf{f}^T(\mathbf{y}_n)}{1 + \lambda_n (\mathbf{f}^T(\mathbf{y}_n)\mathbf{y}_n - 1)} W_n \quad (4.6)$$

where λ_n is a time-varying forgetting factor. Using this online formulation, the ICA model can be estimated as the data stream is acquired. The forgetting factor enables the online algorithm to gradually forget about data acquired in the past and adjust the model to newly acquired data. This is of particular interest to track non-stationarity in the data⁸⁹. This solution has been adopted in Paper IV and it is promising for online spike sorting solutions. Neural recordings can in fact exhibit a drift due to relative movement between the tissue and the electrodes. While drift can be corrected for by some automatic offline algorithms^{52,60}, there is no current solution for handling drift online. ORICA could therefore be a valuable solution to tackle this problem for real-time applications.

Software The FastICA algorithm is available directly from the `scikit-learn` Python package¹⁷³, while other Python implementations can be run with the `MNE` package¹⁷⁴. Alternatively, several ICA implementations in `MATLAB` are available through `EEGLAB`¹⁷⁵. For Paper IV, a Python implementation of the ORICA algorithm was developed (<https://github.com/alejoe91/spyica>) based

on a MATLAB version implemented by a co-author for previous studies⁸⁸⁻⁹⁰ (<https://github.com/goodshawn12/orica>).

4.2 Machine learning and deep learning

Machine learning can be defined as a set of algorithms that are able to automatically learn from data.

Machine learning algorithms are historically divided in three different classes: supervised learning, unsupervised learning, and reinforcement learning. After a brief description of the three types of learning, I will focus on supervised learning with artificial neural networks and convolutional neural networks, because of their use in Paper V.

Supervised learning aims to learn a relation, or model, between some input and output data. In order to perform this kind of learning, one needs a *labeled* dataset, i.e., a collection of observations where both the input and corresponding output are known. The output can either be categorical or a real value. The former case is referred to as a classification problem, the latter as a regression problem. In Paper V, we used supervised learning to solve both a classification problem to predict the neuronal cell type, and a regression problem for finding 3D positions of the neurons with respect to the recording probe, using the extracellular spikes as input.

Unsupervised learning aims to learn internal structures of the data, without targeting a particular input-output relation. Unsupervised learning uses unlabeled data. A classical example of unsupervised learning is clustering: given a set of data, clustering algorithms try to find different sub-groups in the data, without knowledge of whether there are any groups and how many groups there are. Clustering is widely used in the spike sorting literature (Section 2.1 - Clustering-based approach) and it has also been used in Paper V for cell-type classification, in order to compare the unsupervised performance to the supervised approach.

Reinforcement learning aims to learn to perform the right action in a certain state to achieve a goal. It is a typical problem for autonomous agents, e.g., robots, and learning is achieved by dispensing a positive or negative reward depending on whether the chosen actions helped the agent to reach its goal.

4.2.1 The basics of artificial neural networks

Among the large variety of machine learning algorithms developed in the past decades for supervised learning, neural networks have arguably become the most popular set of algorithms.

The idea behind artificial neural networks comes from biology, as a simplification of how neurons integrate input signals. An artificial neuron, or perceptron¹⁷⁶ (Figure 4.2A), is a simple entity that transforms the sum of its weighted inputs and a bias value:

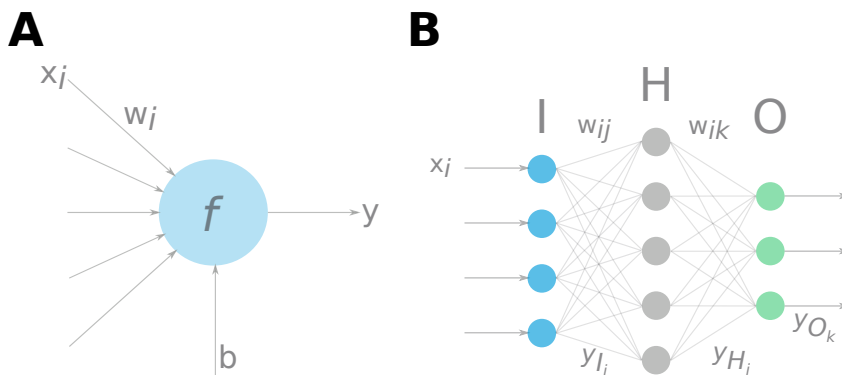


Figure 4.2: (A) Artificial neuron (or perceptron). The weighted inputs and the bias are transformed with the activation function f to generate the output. (B) Neural network and definitions for the Iris dataset example described in the text.

$$y = f \left(\sum_i w_i x_i + b \right) \quad (4.7)$$

where x_i is the i -th input to the neuron, w_i is the weight of the input x_i , b is a scalar bias, $f(\cdot)$ is the so-called activation function, and y is the output of the neuron. The function $f(\cdot)$ enables the neuron to represent non-linear transformations of the input data. Typical activation functions are sigmoids, hyperbolic tangents, or rectified linear functions. When artificial neurons are combined and connected together, the resulting network is capable of finding very complex and non-linear relations between inputs and outputs.

Let us look at a simple real-world example and see how a neural network can be constructed. A famous dataset to test machine learning algorithms is the Iris dataset¹⁷⁷. It contains observations of the sepal and petal lengths and widths of three Iris flower species, the *Iris setosa*, *Iris virginica*, and *Iris versicolor*. We would like to find a model that predicts the species from the petal and sepal morphological information. This is a typical example of classification, as the output is categorical.

We can build a three-layer neural network. The first layer is the input layer and it propagates the inputs to the all the nodes of the next layer. In the Iris case, we have four inputs (petal length, petal width, sepal length, and sepal width), so we will have four input neurons. The second layer is also called hidden layer, as it is not seen by the inputs nor the outputs. There can be many hidden layers, but in this case, for simplicity, we will only have one layer with, for example, five neurons. The hidden layers are very important to capture complex and non-linear relations in the input data, which are connected to the output. The last layer is the output layer. In this case there are three outputs (one for

each class), hence three output neurons. Figure 4.2B displays a network with these three layers. Nodes from the input layer are referred to as I_i , from the hidden layer as H_j , and from the output layer as O_k . The outputs of neurons belonging to input, hidden, and output layers are defined as y_I , y_H , and y_O , respectively. The activation function $f(\cdot)$ is a sigmoid function, whose co-domain is 0 to 1. The predicted class for an input $\mathbf{x} = [x_1, x_2, x_3, x_4]$ is the class whose output neuron is closer to 1. The output of a neuron from the output layer can be written as:

$$y_{O_k} = f \left[\sum_j w_{jk} y_{H_j} + b_{O_k} \right] = f \left[\sum_j w_{jk} f \left(\sum_i w_{ij} x_i + b_{H_j} \right) + b_{O_k} \right] \quad (4.8)$$

Now that we have a network architecture for our classification task we have to train it, i.e., to learn the parameters that better describe the input-output relation. The set of parameter that we want to optimize includes the weights from the input to the hidden layer w_{ij} (20 parameters), the ones from the hidden to the output layer w_{jk} (15 parameters), and the bias values of the hidden nodes ($b_{H_j} - 5$ parameters) and the output nodes ($b_{O_k} - 3$ parameters). In total, there are 43 parameters to be fitted*. Note that input nodes have no weights nor biases, as they propagate the inputs to the next layer without any modification. In order to fit the parameters, we have to define a cost or loss function that tells us how well the network is performing. Given an observation n , its labeled output can be encoded as an array \mathbf{t}_n : (1, 0, 0) for *Iris setosa*, (0, 1, 0) for *Iris virginica*, and (0, 0, 1) for *Iris versicolor*. Similarly, the output of the network given the input \mathbf{x}_n can be written as an array \mathbf{y}_n , which depends on the values of y_{O_1} , y_{O_2} , and y_{O_3} . For example, if y_{O_2} is the output value closest to 1, then \mathbf{y}_n will be (0, 1, 0). A cost or error function J that describes how well the network is performing over the entire dataset can be the mean of the squared errors (MSE):

$$J(\mathbf{W}) = \frac{1}{N} \sum_n \|\mathbf{t}_n - \mathbf{y}_n(\mathbf{W})\|^2 = \frac{1}{N} \sum_n \sum_v [t_{nv} - y_{nv}(\mathbf{W})]^2 \quad (4.9)$$

where \mathbf{W} is the set of all parameters and $v = (1, 2, 3)$ is the output dimension. Note that the dependence of \mathbf{y}_n on \mathbf{W} is explicit.

We can now use an optimization algorithm, for example, the gradient descent, to iteratively update the weights and minimize the cost function J . Using gradient descent, at each iteration a generic weight w is updated as follows:

$$w_{t+1} = w_t - \eta \Delta w_t = w_t - \eta \left. \frac{\partial J}{\partial w} \right|_{w=w_t} \quad (4.10)$$

where η is the learning rate. Since the computation of the gradient is effectively propagating the error function back to previous layers, this

*this is a very small network!

4. Engineering solutions

optimization strategy is also referred to as *backpropagation* in the neural network literature.

Choosing a differentiable activation function, the partial derivatives of the weights and bias values of the different layers can be computed using the chain rule. For example, the gradient of a weight that connects the hidden and the output layer (w_{jk}) can be computed as:

$$\begin{aligned}\frac{\partial J}{\partial w_{j=1,k=2}} &= \frac{\partial J}{\partial \mathbf{y}_n} \frac{\partial \mathbf{y}_n}{\partial w_{j=1,k=2}} = \frac{\partial J}{\partial \mathbf{y}_n} \frac{\partial y_{O_2}}{\partial w_{j=1,k=2}} \\ &= \frac{\partial J}{\partial \mathbf{y}_n} \frac{\partial f\left(\sum_j w_{j,k=2} y_{H_j} + b_{O_2}\right)}{\partial w_{j=1,k=2}} \\ &= -\frac{2}{N} \sum_n \sum_v [t_{nv} - y_{nv}(\mathbf{W})] f' \left(\sum_j w_{j,k=2} y_{H_j} + b_{O_2} \right) y_{H_{j=1}}\end{aligned}\tag{4.11}$$

In the first row of the equation, the only component of \mathbf{y}_n that depends on $w_{j=1,k=2}$, is indeed y_{O_2} . Similarly, gradients for biases and weights for all layers can be computed.

Here I presented an illustrative implementation of a classification task using a small neural network and some simplifications. For classification, usually the output neurons use a softmax activation function and the cross-entropy as loss function¹⁷⁸. Differently from other activation functions, softmax transforms the values of all output neurons in probabilities. Moreover, here we used all data points to compute the gradient (\sum_n), but in practice, for larger datasets, smaller batches of observations are used at each iteration. This method is referred to as stochastic gradient descent. While this example formalized a classification problem, regression can be achieved with the same principle and a few changes in the output layer. In order to predict one or multiple real values, the output neurons have no activation function, so that the value of y_{O_k} can range from $-\infty$ to $+\infty$. The mean squared error can be used as loss function also for regression problems.

4.2.2 Deep Learning

Deep learning refers to the use of much larger architectures with many more parameters, layers, and artificial neurons. Deep learning is now used ubiquitously in many applications, including image and video recognition, natural language processing, and medical diagnosis. Some of these architecture are indeed very deep. For example, the **ResNet**¹⁷⁹ architecture, used for image recognition, can contain up to 1'200 layers.

Deep learning can take several flavours. Different architectures have been in fact developed to tackle diverse classes of problems. Patterson and Gibson¹⁸⁰ identify four different classes of deep network architectures: unsupervised

pretrained networks, recurrent neural networks, recursive neural networks, and convolutional neural networks.

Unsupervised pretrained networks learn compressed representations of the data in an unsupervised manner. In order to do so, these networks are trained to reproduce the input data. In other words, the input is also used as labeled output. Belonging to this class are autoencoders, generative adversarial networks, and deep belief networks.

Recurrent neural networks are different from the type of network shown in Section 4.2.1, which is a feed-forward network, because they can have connections between the same layer (recurrent). An example of this class of architectures is the long short-term memory (LSTM) networks, which are commonly used for temporal sequence learning, such as natural language processing. Thanks to their recurrent nature, these networks are able to keep the memory, or the state, of previous samples.

Recursive neural networks are similar to recurrent networks, but they have a tree structure, which allows them to find hierarchical structures in the data.

The last class of architectures are convolutional neural networks, which have been used in Paper V and are presented in the next section.

4.2.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are biologically inspired from the information processing of the visual system^{181,182}. Their architecture is different from other configurations mainly because of the use of convolutional layers. CNNs are typically used in computer vision for image classification.

Figure 4.3 shows a sample architecture of a CNN, similar to the one used in Paper V. The input data are organized in a 2D structure, or an image I , and convolutional kernels K are convolved with the image to extract feature maps. These are then subsampled using pooling operators. Max pooling, for example, reduces the dimensions of the maps by taking the maximum of adjacent pixels. The convolution-pooling operations can be repeated multiple times (two times in Figure 4.3 and in Paper V). Finally, the outputs of convolutional-pooling layers are connected to one or more fully-connected layers, analogous to the hidden layers shown in Section 4.2.1. The last fully-connected layer is then interfaced to the output layer, which outputs the predictions of the network.

The convolution layers consist of several kernels, or filters. The number of kernels of a convolutional layer is called *depth*. Each kernel has a certain size (width and height) and it is convolved with the image. For example, if a kernel K has a 3x3 size, the input to a neuron from the convolutional layer is the convolution between the kernel and the image I :

$$F(i, j) = f \left[\sum_k \sum_l K(k, l) I(i - k, l - j) \right] \quad (4.12)$$

where $k = (-1, 0, 1)$ and $l = (-1, 0, 1)$ as the kernel size is 3x3. $F(i, j)$ is the value of the feature map at pixel position i, j . The activation function $f(x)$ is

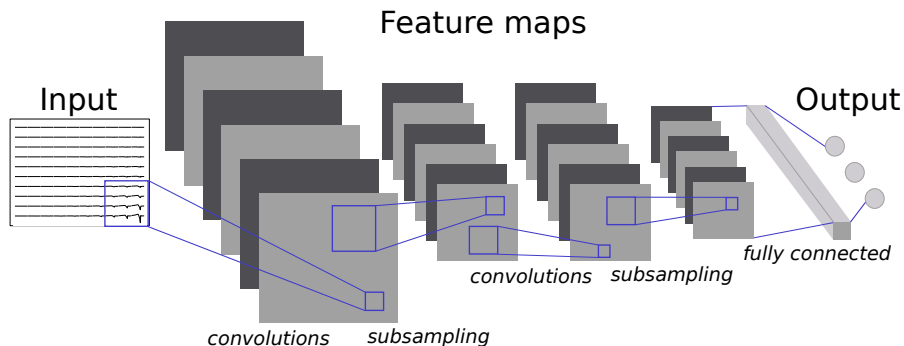


Figure 4.3: Representation of a Convolutional Neural Network. The 2D input is convolved with 2D filters to generate feature maps. Convolved maps are subsamples using the pooling layers. The last fully connected layer is connected to the output neurons.

usually chosen to be a rectified linear unit (ReLU): the output is zero if $x \leq 0$, and it is x when $x > 0$. At the edges of the image I , padding can be applied in order to maintain the same output size. Moreover, the *stride* parameter controls the amount of overlap in pixel convolutions. When the stride is set to 1, there are no gaps between convolutions and F has the same size of I (provided that padding is used at the edges).

Training is performed as described in Section 4.2.1 using backpropagation. The convolutional layers, during training, learn filters that extract relevant features from the structured data, which are used by the fully connected layer to improve the prediction accuracy.

In Paper V, we have used CNNs to predict the neuronal location and cell type from extracellular action potentials (EAPs) on dense MEAs. CNNs have been chosen because the configuration of recording sites for recent MEAs is 2D, and the recorded signals can hence be considered as electrical *images*. Therefore, the task is similar to image classification and CNNs are suited to find 2D structures in the data. However, the EAPs are actually 3D input data, because they evolve over time as well. In order to use CNNs, then, we extracted 2D feature images from the EAPs, including negative and positive peak images, peak-to-peak amplitude, peak-to-valley width, and full-width half maximum. We then fed the 2D feature images as input to the network. Alternatively, we also used 3D kernels that convolved the full 3D EAPs also in the time domain. This approach showed an increase in performance, but the training time was much larger than using 2D convolutional layers.

Due to their very large number of parameters, CNNs (and deep learning

networks in general) can be easily over-trained. When this happens, the model works very well on the data used for training, but it is not capable to generalize over new observations.

4.2.4 Underfitting and overfitting: finding the right model

One of the most important problem in the machine learning literature is to construct models that can provide good predictions of new and unseen data. In order to quantify the performance of the model over new observations, the initial dataset is usually split in two different subsets:

- *Training set*: the training set is used to perform the training and optimize the parameters of the model
- *Test set*: the test set is not used for training (it is also denoted as *holdout set*) and used to evaluate the generalization of the model

The errors on the training and test datasets are defined as training and test errors. Ideally, one wants a machine learning model to 1) have a small error on the training set and 2) to have a small validation gap, i.e., the difference between the training and test errors.

If the model is too weak or simple and it is not able to reach a high performance on the training set in the first place, then the model is *underfitting*. A typical example of underfitting is having data points drawn from a quadratic function and trying to fit it to a straight line. Alternatively, the model may perform very well on the training set, but the performance on the test set is poor. In this case, the model learned aspects of the training data that are not general, such as the noise. The model is *overfitting* the training data. A good model lies in between these two scenarios, with a good performance both on the training and test sets.

In case of deep neural networks, given a large enough architecture, overfitting is definitely more problematic than underfitting. These models have in fact a very large *capacity*¹⁷⁸, that is, the capability of fitting a large variety of functions if properly trained. One way of stemming overfitting is to use regularization techniques, which pose some constraints on the weights. Examples of regularizations are L1 and L2 regularizers, which make the weights sparser (L1) or limit their values (L2). Another popular method against overfitting is dropout¹⁸³. When using dropout, during training, a percent of the weights is randomly selected at each iteration and it is *dropped* from updating. Therefore, only a subset of weights is updated at each iteration, effectively increasing the generalization power of the model. Dropout is usually applied layer-wise, and in Paper V it is applied only to the last fully-connected layer.

Another way to improve the model performance is by hyperparameter tuning. Hyperparameters include all the parameters of the model which are not trainable, such as the number of neurons, the learning rate, and the number of training iterations. In order to choose hyperparameters, a validation set can be extracted from the training set and used as a proxy of the test set in order to assess the

generalization of the model. For example, one can monitor the validation error and stop the training when this starts to diverge from the training error (*early stopping*).

Software As machine learning and deep learning applications have become ubiquitous, several software solutions exist. For standard machine learning methods (excluding neural networks) the `scikit-learn`¹⁷³ Python package is a fantastic tool for machine learning. Specifically targeting neural networks and deep learning, both Google and Facebook have developed powerful and open source frameworks: Tensorflow¹⁸⁴ (Google) and PyTorch¹⁸⁵ (Facebook). In 2017, Tensorflow incorporated Keras¹⁸⁶ in their core API, making the creation of deep learning architectures easier and more abstract. Tensorflow has been used in Paper V.

4.3 Optimization and genetic algorithms

In several instances throughout this thesis I had to find a set of parameters to fit some data or to yield a good solution for a given problem. Generally, the problem of finding parameters that optimize a certain cost function is called an *optimization* problem. Examples of optimization problems that have been introduced in the previous sections include finding the solution of the ICA model and training artificial neural networks.

There is a large variety of strategies to solve optimization problems. Many of them are based on estimating the gradient of the cost function and using it to find the minimum of the function. For example, the *gradient descent* method follows the steepest path, and other methods add some momentum and adaptive behavior to find the minimum faster and not to get stuck in local minima^{187,188}. One drawback of gradient-based methods is that for some problems the gradient is unknown. In these cases a finite approximation in parameter space is used, but this could be time consuming in some applications.

Alternatively, a second class of optimization tools are evolutionary algorithms. Genetic algorithms are part of the evolutionary algorithms class and they are biologically-inspired, mimicking the evolution process by natural selection¹⁸⁹. Genetic algorithms are general-purpose optimization methods that can be applied to any optimization problem. Moreover, as they explore the solution space randomly and they visit several points simultaneously, they can be quite fast, but their meta-heuristic nature does not guarantee convergence to a global minimum.

The fundamental unit of genetic algorithms is a chromosome, or individual, i.e., a representation of a possible solution. At initialization, N chromosomes are randomly created to form a population P . Each chromosome's fitness is evaluated with a *fitness function*, which indicates how good the solution is. Chromosomes are then selected with a selection process and mated to create a new generation of the population P . Usually, part of the individuals with the highest fitness are kept in the new generation (*elitism*). Additionally, random

mutation is applied to the individuals, to add variability to the search space. The process is repeated until there is a convergent solution or if the solution is in stall.

Genetic algorithms have been used both in Paper V, to solve the inverse problem for localization (Eq. 2.2), and Paper VI, to optimize the electrical stimulation patterns for improving stimulation selectivity.

Software There are several available solutions for optimization and evolutionary programming in Python. `scipy`¹⁹⁰, for example, provides an excellent optimization module (`scipy.optimize`). For the implementation of genetic algorithms in Paper V and Paper VI, the DEAP¹⁹¹ (Distributed Evolutionary Algorithms in Python) package has been used, because of its simple and flexible API.

4.4 Scientific programming in Python

In the development of the methods presented in this thesis, I had to decide which programming language to use for software development. With my engineering background, I was mainly trained in MATLAB and C++. While C++ is considered too low level for standard scientific programming, despite my familiarity with MATLAB, I opted for Python for the following main reasons^{192,193}:

1. Python is open-source. It is accessible to any individual and, contrary to MATLAB, it is totally free.
2. Python can seamlessly run on several operating systems and platforms.
3. The language has a clean and versatile syntax, allowing for purely procedural coding or object-oriented approaches.
4. Python is an interpreted language (similarly to MATLAB), which allows rapid and interactive prototyping.
5. There is a huge variety of packages for scientific computing and visualization. Among those, the mostly used packages throughout this thesis are `numpy`¹⁹⁴ (numerical computations), `scipy`¹⁹⁰ (scientific computing and signal processing), `scikit-learn`¹⁷³ (machine learning), `matplotlib`¹⁹⁵ and `seaborn`¹⁹⁶ (visualization), and `pandas`¹⁹⁷ (data structures and statistics).
6. Sharing of packages in Python is very easy, using the PyPi package manager*.
7. Python can be interfaced with several tools for communication and documentation of software, such as `jupyter` notebooks¹⁹⁸ and `sphinx`[†].

*www.python.org/pypi

†<http://www.sphinx-doc.org/>

4. Engineering solutions

In addition to field-agnostic motivations, Python has gained a large popularity in the neuroscientific community over the past few years¹⁹⁹. Moreover, the field is pushing for a full commitment to open-source²⁰⁰. I strongly believe that open science, both in terms of open-source software and hardware resources and open access to research and journals, is truly beneficial for the progress of science.

Here are the source code repositories developed for the papers presented in this thesis:

- Paper I: <https://github.com/alejoe91/MEArec>
- Paper II: <https://github.com/SpikeInterface/spikeinterface>
- Paper III - Paper IV: <https://github.com/alejoe91/spyica>
- Paper V: <https://github.com/CINPLA/NeuroCNN>
- Paper VI: <https://github.com/alejoe91/MEAutility>
- Paper VII: <https://github.com/MiroK/nEuronMI>

Chapter 5

Summary of Papers

Paper I

This paper presents MEArec, an open-source Python package for simulating extracellular spiking activity. MEArec is designed to meet the requirements of ease-of-use, speed, controllability, and biophysical detail that we identify as key features of a simulator for development and validation of spike sorting software. The simulation is split in two phases. A templates generation phase builds a template library from biophysically detailed cell models. A recording generation phase selects suitable templates and combines them with random spike trains in customized convolution, capable of reproducing critical aspects of extracellular recordings such as bursting and drifting. Moreover, the user can control the rate of spatio-temporal overlapping events and there are several noise models available to test the robustness of algorithms against noise assumptions. MEArec is fully interfaced with SpikeInterface (Paper II) for benchmarking available software and it is used by SpikeForest[†], an interactive website for benchmarking spike sorting algorithms.

Paper II

In this article we introduce SpikeInterface, a unified framework for spike sorting. SpikeInterface is designed to tackle the community needs for standardization in spike sorting, which can mine reproducibility in the analysis of extracellular recordings. SpikeInterface is a collection of Python packages for: i) loading and writing spike-sorting-relevant information from several file formats, ii) running numerous available spike sorters with a standardized API (in a single line of code), iii) providing a processing toolkit for pre-, post-processing, validation of results, and automatic/manual curation, iv) performing comparison of spike sorting output with known or unknown ground-truth information; and v) visualizing every step of the electrophysiology pipeline with efficient plotting routines.

[†]spikeforest.flatironinstitute.org

5. Summary of Papers

Additionally, we implemented a graphical user interface to build analysis pipelines. In the paper, we present an overview of the framework and sample applications for analyzing experimental Neuropixels data with several sorters (resulting in large disagreements) and for benchmarking spike sorting software on a high-density MEArec (Paper I) recording.

Paper III

In this conference contribution we present a fully automated algorithm for spike sorting based on independent component analysis (ICA). While the use of ICA had been investigated before for spike sorting, new recording technologies with a very high electrode density are particularly suitable for ICA processing. In this paper, therefore, we combine ICA with a selection of spiking sources, amplitude clustering in the ICA space, and removal of duplicate spike trains in a fully automatic approach. We compare our approach with two other popular automatic spike sorting algorithms (Mountainsort⁵⁴ and SPyKING-CIRCUS⁵⁸) using initial versions of what would become MEArec (Paper I) and SpikeInterface (Paper II).

Paper IV

This conference contribution builds upon Paper III and it tackles real-time spike sorting. We approach this problem by applying an online version of the ICA method, the Online Recursive Independent Component Analysis (ORICA), which was originally developed for automatic artifact removal in EEG recordings^{88–90}. The online pipeline is semi-automatic and it is very similar to the offline method presented in Paper III. After an ORICA pre-processing step and an automatic selection of spiking sources, the user needs to set thresholds on single independent components for spike detection. Even in this case, we benchmark the approach on simulated data with known mixing matrices, in order to validate the performance of estimating the ICA model in an online setting.

Paper V

In this article we target localization and classification of neurons from extracellular recordings. We present an approach that combines biophysical forward modeling and deep learning. In brief, we simulate a large dataset of extracellular action potentials (EAPs) from around 200 cell models from the Blue Brain Project³⁵. Then, knowing the ground-truth position and type of the simulated neurons, we train convolutional neural networks (CNNs) to predict i) the 3D soma locations and ii) the cell type (excitatory-inhibitory) from the EAPs. We investigate the localization and classification performance depending on different probe designs, CNN sizes, and selected features. We compare the performance of our CNN model with state-of-the-art inverse methods for cell localization and clustering techniques for cell classification and show that

our deep learning approach is superior in both cases. Finally, we validate the trained models on simulated EAPs from other databases, e.g., the Allen Brain Institute, and on experimental data that use paired extracellular and patch clamp recordings.

Paper VI

In this conference contribution, we investigate a model-based optimization approach for selective electrical stimulation of single cells using high-density MEAs. Assuming that we can extract the position of the soma and the direction of the axon hillock from the extracellular action potentials, we use this information to optimize the spatial patterns of the currents injected by the electrodes in order to increase selectivity of the stimulation. We predict the excitation of a neuron based on the so-called activating function (Eq. 3.11) and we construct a multi-objective optimization problem that aims to i) only stimulate a target neuron (selectivity) and ii) use the least number of electrodes and the lowest stimulation currents (efficiency). We use a genetic algorithm to solve the optimization problem and compute the optimal currents. Using a Monte-Carlo approach, we show that this model-based approach outperforms standard monopolar and bipolar stimulation paradigms.

Paper VII

In this paper we investigate the effect of neural probes on the extracellular signals. While the standard modeling framework for computing extracellular potentials from neuronal currents assumes an infinite and homogeneous milieu, here we employ advanced finite element methods (FEM) to study the effect of several kind of probes on the extracellular potentials. We simulate a single action potential from a simple ball-and-stick neuron and its extracellular signature with and without a neural probe in the extracellular space. We show that for microwires/tetrodes the effect of the probe is negligible. Conversely, larger MEAs strongly affect the recorded signals due to their insulating properties. The spike peaks almost double when the probe is explicitly modeled, and the peak ratio between the potential with and without the probe is relatively constant with respect to the distance from the probe. However, the ratio is dependent on the lateral alignment and relative rotation between the probe and the neuron. Finally, we show that the probe effect is electrode-dependent and we suggest an efficient method, namely *probe correction* method, to correct for the probe effects. This method uses FEM simulations to pre-map the effect each electrode on the extracellular space and then leverages the reciprocity and superposition principles to compute extracellular potentials from neuronal currents.

Chapter 6

Discussion

“Sitting on your shoulders is the most complicated object in the known universe.”

— Michio Kaku

6.1 Towards next-generation electrophysiology

In recent years, the opportunity to conduct electrophysiological recordings has been undergoing a revolution. Developments in technology for the fabrication of neural probes have enabled the creation of next-generation devices which can record from hundreds (*in vivo*) and thousands (*in vitro*) of channels simultaneously. With these high-density multi-electrode arrays (HD-MEAs)^{15,17,18,21,22,25} we are now able to measure the activity of hundreds of neurons simultaneously, even at the sub-cellular level^{28,99,108}. However, next-generation devices introduce novel grand challenges and the need for appropriate tools to handle the rich information that can be recorded²⁷. The work presented in this thesis has therefore focused on developing and benchmarking new tools and methods for using such devices.

The literature on electrophysiological analysis targeting spiking signals has mainly focused on extracting individual spike trains from recordings (using spike sorting - Section 2.1). However, HD-MEAs pose new challenges for spike sorting and a large part of the presented work aimed to benchmark and compare existing solutions (Paper I and Paper II), as well as to develop spike sorting methods specifically targeting HD-MEAs (Paper III and Paper IV).

In addition to the high yield in terms of number of recorded neurons, the abundant spatial information available from HD-MEAs can be exploited for localizing the 3D position of the recorded neurons and accurately classifying their neuronal type (Paper V). Extracellular electrophysiology can therefore increasingly be regarded as a *functional electrical imaging* technique²⁰¹, from which one can reconstruct a 3D map of the recorded neuronal microcircuit. Additionally, precise localization and optimized stimulation strategies (Paper VI) enable advanced

closed-loop experiments, allowing for a precise bi-directional manipulation of neural circuits.

The number of neurons that can be recorded by HD-MEAs is unprecedented. For example, in a recent study²⁰², the authors report data from around 30'000 neurons acquired from 39 sessions in 10 mice. Given this immense amount of information, one question arises: is spike sorting needed at all?

Some novel approaches present spike-sorting-free techniques to infer the dynamics of the entire recorded population^{203–206}. The idea of these studies is to find neural *manifolds*, i.e., lower-dimensional representations of the ensemble neural activity, connected to behavior, hence bypassing the need for spike sorting. While the approach is interesting for specific applications, such as motor control²⁰⁷ and brain-machine interfaces²⁰⁸, I believe that the reign of spike sorting will never be overturned. In fact, spike sorting is a required step to study how single neurons respond to stimuli, movements, or encode specific memories. Without spike sorting, we would not know that neurons in visual cortex are orientation-selective²⁰⁹, that some neurons in the hippocampus only fire in a specific place²¹⁰ while others in the entorhinal cortex in a hexagonal grid pattern^{211,212}, and that some neurons like Jennifer Aniston²¹³.

Another mean for measuring neuronal activity is through imaging, which one might imagine could replace electrophysiology. Instead of inserting a probe in the brain to pick up electrical signals generated by the neurons, one can image fluorescent markers linked to sensors that (directly or indirectly) reflect neural activity. For example, two-photon calcium imaging has been used to image neural activity for decades²¹⁴. Alternatively, voltage sensitive dyes²¹⁵ (VSDs) and, more recently, genetically-encoded voltage indicators^{216–218} (GEVIs) are proxies for the neuronal membrane potential signals. While the kinetics of calcium indicators and VSDs are considered to be fairly slow with respect to the membrane potential dynamics, recently developed GEVIs can image neuronal spiking activity and sub-threshold dynamics at very high speed. Imaging techniques are capable of recording tens of thousands of neurons simultaneously²¹⁹, hence falling deservedly into the definition of next-generation tools. In addition, a combination of simultaneous imaging and manipulation of activity using optogenetics and holographic stimulation techniques²²⁰ enables a precise and bi-directional manipulation of the neural tissue with unprecedented levels of accuracy, via so-called all-optical investigations of neural circuits^{221,222}. However, imaging comes with challenges and limitations. Most importantly, current microscopes have a depth-limitation, hence deeper structures cannot be imaged. Nevertheless, multi-photon systems could alleviate this problem²²³. The second limitation of imaging techniques comes from the fact that most setups require a bulky and expensive microscope and therefore behavioral studies are mainly restricted head-fixed mice. Despite the possibility of setting up sophisticated behavioral settings, including virtual reality systems²²⁴, experiments on freely moving animals arguably represent a more natural setting to observe neuronal correlates of animal behavior. Notably, portable and wireless imaging systems exist, such as the UCLA miniscope²²⁵, but their capabilities are not comparable to fixed microscopy systems.

HD-MEA devices, instead, can be used both for head-fixed and freely moving animals, using chronic and recoverable implants^{226,227}. They enable to perform very high-yield experiments, with high spatio-temporal resolution recordings from hundreds of neurons per experimental session from several structures of the brain²¹ and they are opening a brand new era for neurophysiology. Moreover, HD-MEA devices are currently being used by large-scale international collaborations²²⁸ and will generate an unprecedented amount of neural data from all known and unknown brain regions, which will greatly increase our understanding of how the brain works.

6.2 Computationally-assisted electrophysiology: benefits and limitations

Modeling and simulations have been central in the presented work to aid the development of methods for extracellular electrophysiology.

In recent years, large international consortia have joined forces and dedicated a huge amount of resources to build detailed biophysical models of neurons. In 2015, the Blue Brain Project released the Neocortical Microcircuit Portal^{32,33}, with more than 30'000 cortical cell models. In parallel, the Allen Institute for Brain Science is constantly expanding its cell-type database³⁶ with cell models reconstructed from mice and even from human tissue⁹².

The large and growing availability of such detailed models has created a new opportunity in the development of analysis methods for neuroscience. Simulations can be used to develop model-informed, or computationally-assisted tools to solve specific problems in electrophysiology. In other words, models can provide *a-priori* and ground-truth knowledge that can be incorporated in the methods themselves. In this thesis, modeling served three main purposes: ground-truth data simulation, validation, and prediction.

In Paper V we used ground-truth simulations in combination with machine learning and showed that supervised localization and classification approaches outperform unsupervised solutions. Simulations can in fact provide a virtually infinite amount *labeled* datasets, which can be used to train machine learning algorithms.

In Paper I, a biophysically detailed simulator for extracellular spiking activity was developed specifically to aid the development and validation of spike sorting algorithms. Although also spike sorting could be tackled in a supervised manner, by learning a model that outputs spike times of different neurons using the extracellular signals as input, this approach would need to learn very complicated spatio-temporal features in the recordings, and at present it has not been attempted.

In Paper VI, modeling was used as a predictive tool for the outcome of electrical stimulation. In combination with genetic algorithms used for optimization, we were able to find optimal stimulation patterns that improved the stimulation selectivity.

Finally, a model-based approach would enable users to tune the methods to diverse applications. Neurons from different brain regions or species can be very different in terms of morphology, electrical properties, and firing patterns. Hence, analysis tools targeting specific applications, for example, hippocampus recordings in mice, could be enhanced by using cell models from the selected brain region and animal species.

There are also some limitations to the modeling framework used throughout this thesis that need to be discussed. First and foremost, simulations are not real. A mathematical representation of a real entity, no matter how complicated, should always be regarded as a simplified and limited representation of the entity. However, recent developments in computational systems and recording technologies make the simulations from cell models used in this work *almost* indistinguishable from real data. Koch and Buice⁴³, in a letter accompanying the publication of the Blue Brain Project simulation³², state that these simulations would pass a *biological Turing test*, given their ability to accurately replicate neuronal dynamics. Still, these detailed cell models have intrinsic limitations.

Most of the electrophysiological data used to construct biophysically detailed cell models comes from patch-clamp somatic voltage traces. However, neurons are more than just the soma. Neuronal dynamics are greatly affected by non-somatic cell compartments, such as dendrites and axons²²⁹. Modeling studies using data from multiple patch pipettes along the apical dendrites to fit multi-compartment models^{30,31} were capable of replicating complex active properties of the dendrites, such as backpropagating action potentials and calcium spikes, which affect the extracellular signals. However, somatic patch-clamp recordings alone are not enough to constrain cell models that replicate these features³⁰. Moreover, extracellular action potentials are largely influenced by the axonal initial segment (AIS)^{113,114}, which should therefore be carefully considered in the models. Cell models from both the Blue Brain Project and the Allen Institute, however, mainly use somatic patch-clamp recordings alone to collect data used to construct their cell models. Therefore, active dendritic properties cannot be reproduced. Moreover, in those models the entire axon is replaced by a single stereotyped axonal segment, hence the contribution of the AIS to the extracellular signals might be distorted.

Second, in the calculation of extracellular signals I have mainly used the framework described in Section 3.2 (Paper I, Paper III, Paper IV, Paper V, and Paper VI). The volume conduction theory (Eq. 3.9) assumes that the extracellular space is isotropic, homogeneous, linear, and infinite. However, neural tissue is not isotropic¹⁵³, mainly due to the preferential orientation of pyramidal cells. Anisotropy could be easily included in the calculation of extracellular potentials with analytical solutions^{41,154}. Second, the tissue surrounding the neurons is not homogeneous, because of the presence of the neural probes. In Paper VII, we therefore investigated how neural probes affect the recordings. We showed that the probe does affect the recorded potentials in a non-trivial way, and proposed an efficient solution that involves a finite element *pre-mapping* of the effect of the probe in the extracellular space. While this solution is technically feasible and could be incorporated into current simulators^{154,155}, it would require

one to embed finite element solutions in the software, making it slower and more complicated to handle. Alternatively, the finite element solutions could be approximated by analytical functions and used to efficiently pre-map the effect of different probes in the extracellular space.

Finally, regardless the above-mentioned limitations, I believe that the presented computationally-assisted approach to electrophysiology has the potential to improve the current status of extracellular electrophysiology research. Methodological improvements and the integration of new techniques for acquiring comprehensive data will make computational models more accurate and realistic. As a result, analysis tools built upon simulations will also improve their performance in tackling common problems in electrophysiology.

6.3 Future developments

Part of the work presented in this thesis is fully finalized, documented, and shared with the community. In particular, the software presented in Paper I (MEArec) and in Paper II (SpikeInterface) are already used in other projects. As an example, the SpikeForest project* is aiming to benchmark and compare most of the available spike sorters using ground-truth datasets. SpikeInterface is the engine upon which SpikeForest is running, and several MEArec-generated datasets are used as ground-truth recordings.

Paper III and Paper IV are instead at a proof of concept stage. The software, named SpyICA, is dated before the development of MEArec and SpikeInterface, which were specifically designed to ease the development, validation, and comparison of spike sorting algorithm. Therefore, a future effort will tackle the extensive characterization of SpyICA in comparison with state-of-the-art spike sorters, both for offline and online use. Moreover, as mentioned in Section 2.1, the current implementation is not capable of handling spatio-temporal overlapping units. Therefore, a hybrid ICA and template-based approach could be investigated for this purpose. In particular, adding a template-matching step to the ICA-based pipeline (2.2C) would both facilitate the correct identification of spatio-temporal collision and alleviate the problem related to duplicate units in different independent components mentioned in Section 2.1.

In Paper V, we performed a thorough analysis on the capabilities of supervised approaches in electrophysiological analysis. Although the source code is open and shared online (see Section 4.4), it consists of a series of scripts rather than a compact package. Moreover, it was developed using an old version of the Tensorflow package¹⁸⁴, which has been enormously improved and simplified over the last year. Therefore, I plan to make the software more usable and plug-and-play and to update the code to the newest Tensorflow version.

In Paper VII, we used a very simplified model of a neuron, a ball-and-stick¹⁴². While our purpose was to characterize the effect of extracellular probes, in the future, similar detailed finite element simulations could make use of more

*spikeforest.flatironinstitute.org/

complicated and realistic morphologies, which are widely available on online databases²³⁰.

Finally, the performance of spike sorting, localization, classification, and stimulation methods has been assessed using simulated data alone. Nevertheless, experimental validation is an important and required step. In this light, validation datasets can be acquired by a combination of multiple recording techniques, for example combining extracellular HD-MEAs, patch-clamp, and imaging techniques. Multi-modal setups with extracellular MEAs and imaging can provide ground-truth information on the neuronal spiking activity, location, cell type, and stimulation outcome.

6.4 Outlook

In conclusion, in this thesis I presented several methods for a computationally-assisted approach targeting neural extracellular electrophysiology for multi-electrode arrays. With the combination of state-of-the-art modeling and engineering tools, this work has introduced a series of next-generation analysis tools to handle newly developed and powerful neural devices, which will contribute to a better understanding of our fascinating brain.

Bibliography

1. Herculano-Houzel, S. The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost. *Proceedings of the National Academy of Sciences* vol. 109, 10661–10668 (2012).
2. Ananthanarayanan, R., Esser, S. K., Simon, H. D. & Modha, D. S. *The cat is out of the bag: cortical simulations with 10⁹ neurons, 10¹³ synapses* in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (2009), 63.
3. Herculano-Houzel, S. & Lent, R. Isotropic fractionator: a simple, rapid method for the quantification of total cell and neuron numbers in the brain. *Journal of Neuroscience* vol. 25, 2518–2521 (2005).
4. Menzel, R. & Giurfa, M. Cognitive architecture of a mini-brain: the honeybee. *Trends in cognitive sciences* vol. 5, 62–71 (2001).
5. Hodgkin, A. L. & Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology* vol. 117, 500–544 (1952).
6. Sterratt, D., Graham, B., Gillies, A. & Willshaw, D. *Principles of computational modelling in neuroscience* (Cambridge University Press, 2011).
7. Hong, G. & Lieber, C. M. Novel electrode technologies for neural recordings. *Nature Reviews Neuroscience*, 1 (2019).
8. Wilson, M. A. & McNaughton, B. L. Dynamics of the hippocampal ensemble code for space. *Science* vol. 261, 1055–1058 (1993).
9. Buzsáki, G. Large-scale recording of neuronal ensembles. *Nature Neuroscience* vol. 7, 446–451 (2004).
10. Voigts, J., Siegle, J. H., Pritchett, D. L. & Moore, C. I. The flexDrive: an ultra-light implant for optical control and highly parallel chronic recording of neuronal ensembles in freely moving mice. *Frontiers in systems neuroscience* vol. 7, 8 (2013).

11. Wise, K. D. Silicon microsystems for neuroscience and neural prostheses. *IEEE engineering in medicine and biology magazine* vol. 24, 22–29 (2005).
12. Bragin, A. *et al.* Multiple site silicon-based probes for chronic recordings in freely moving rats: implantation, recording and histological verification. *Journal of neuroscience methods* vol. 98, 77–82 (2000).
13. Blanche, T. J., Spacek, M. A., Hetke, J. F. & Swindale, N. V. Polytrodes: high-density silicon electrode arrays for large-scale multiunit recording. *J Neurophysiol* vol. 93, 2987–3000 (2005).
14. Ruther, P. & Paul, O. New approaches for CMOS-based devices for large-scale neural recording. *Current opinion in neurobiology* vol. 32, 31–37 (2015).
15. Frey, U., Egert, U., Heer, F., Hafizovic, S. & Hierlemann, A. Microelectronic system for high-resolution mapping of extracellular electric fields applied to brain slices. *Biosensors and Bioelectronics* vol. 24, 2191–2198 (2009).
16. Fiscella, M. *et al.* Recording from defined populations of retinal ganglion cells using a high-density CMOS-integrated microelectrode array with real-time switchable electrode selection. *Journal of neuroscience methods* vol. 211, 103–113 (2012).
17. Bertotti, G. *et al.* A capacitively-coupled CMOS-MEA with 4225 recording sites and 1024 stimulation sites in *Proceedings of the 9th International Meeting on Substrate-Integrated Microelectrode Arrays* (2014), 247–250.
18. Berdondini, L., Bosca, A., Nieuw, T. & Maccione, A. in *Nanotechnology and Neuroscience: Nano-electronic, Photonic and Mechanical Neuronal Interfacing* 207–238 (Springer, 2014).
19. Csicsvari, J. *et al.* Massively parallel recording of unit and local field potentials with silicon-based electrodes. *Journal of neurophysiology* vol. 90, 1314–1323 (2003).
20. Schröder, S. *et al.* CMOS-compatible purely capacitive interfaces for high-density in-vivo recording from neural tissue in *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (2015), 1–4.
21. Jun, J. J. *et al.* Fully integrated silicon probes for high-density recording of neural activity. *Nature* vol. 551, 232 (2017).
22. Fiáth, R. *et al.* A silicon-based neural probe with densely-packed low-impedance titanium nitride microelectrodes for ultrahigh-resolution in vivo recordings. *Biosensors and Bioelectronics* vol. 106, 86–92 (2018).
23. Chung, J. E. *et al.* High-density, long-lasting, and multi-region electrophysiological recordings using polymer electrode arrays. *Neuron* vol. 101, 21–31 (2019).
24. Angotzi, G. N. *et al.* SiNAPS: An implantable active pixel sensor CMOS-probe for simultaneous large-scale neural recordings. *Biosensors and Bioelectronics* vol. 126, 355–364 (2019).

25. Boi, F. *et al.* Multi-shanks SiNAPS Active Pixel Sensor CMOSprobe: 1024 simultaneously recording channels for high-density intracortical brain mapping. *bioRxiv*, 749911 (2019).
26. Obien, M. E. J., Deligkaris, K., Bullmann, T., Bakkum, D. J. & Frey, U. Revealing neuronal function through microelectrode array recordings. *Frontiers in neuroscience* vol. 8, 423 (2015).
27. Steinmetz, N. A., Koch, C., Harris, K. D. & Carandini, M. Challenges and opportunities for large-scale electrophysiology with Neuropixels probes. *Current opinion in neurobiology* vol. 50, 92–100 (2018).
28. Müller, J. *et al.* High-resolution CMOS MEA platform to study neurons at subcellular, cellular, and network levels. *Lab on a Chip* vol. 15, 2767–2780 (2015).
29. Mainen, Z. F., Joerges, J., Huguenard, J. R. & Sejnowski, T. J. A model of spike initiation in neocortical pyramidal neurons. *Neuron* vol. 15, 1427–1439 (1995).
30. Hay, E., Hill, S., Schürmann, F., Markram, H. & Segev, I. Models of neocortical layer 5b pyramidal cells capturing a wide range of dendritic and perisomatic active properties. *PLoS computational biology* vol. 7, e1002107 (2011).
31. Almog, M. & Korngreen, A. A quantitative description of dendritic conductances and its application to dendritic excitation in layer 5 pyramidal neurons. *Journal of Neuroscience* vol. 34, 182–196 (2014).
32. Markram, H., Muller, E., Ramaswamy, S., *et al.* Reconstruction and simulation of neocortical microcircuitry. *Cell* vol. 163, 456–492 (2015).
33. Ramaswamy, S., Courcol, J., Abdellah, M., *et al.* The neocortical microcircuit collaboration portal: a resource for rat somatosensory cortex. *Front Neural Circuits* vol. 9 (2015).
34. Migliore, R. *et al.* The physiological variability of channel density in hippocampal CA1 pyramidal cells and interneurons explored using a unified data-driven modeling workflow. *PLoS computational biology* vol. 14, e1006423 (2018).
35. *Digital Reconstruction of Neocortical Microcircuitry* <https://bbp.epfl.ch/nmc-portal/welcome/>.
36. *Allen Brain Atlas – Cell Types* <https://celltypes.brain-map.org/>.
37. Gouwens, N. W. *et al.* Systematic generation of biophysically detailed models for diverse cortical neuron types. *Nature communications* vol. 9, 710 (2018).
38. Gold, C., Henze, D. A., Koch, C. & Buzsaki, G. On the origin of the extracellular action potential waveform: a modeling study. *Journal of neurophysiology* vol. 95, 3113–3128 (2006).

39. Buzsáki, G., Anastassiou, C. A. & Koch, C. The origin of extracellular fields and currents—EEG, ECoG, LFP and spikes. *Nature reviews neuroscience* vol. 13, 407 (2012).
40. Einevoll, G. T., Kayser, C., Logothetis, N. K. & Panzeri, S. Modelling and analysis of local field potentials for studying the function of cortical circuits. *Nature Reviews Neuroscience* vol. 14, 770 (2013).
41. Ness, T. V. *et al.* Modelling and analysis of electrical potentials recorded in microelectrode arrays (MEAs). *Neuroinformatics* vol. 13, 403–426 (2015).
42. Tveito, A. *et al.* An evaluation of the accuracy of classical models for computing the membrane potential and extracellular potential for neurons. *Frontiers in computational neuroscience* vol. 11, 27 (2017).
43. Koch, C. & Buice, M. A. A biological imitation game. *Cell* vol. 163, 277–280 (2015).
44. Van Der Maaten, L., Postma, E. & Van den Herik, J. Dimensionality reduction: a comparative. *J Mach Learn Res* vol. 10, 13 (2009).
45. Wood, F., Black, M. J., Vargas-Irwin, C., Fellows, M. & Donoghue, J. P. On the variability of manual spike sorting. *IEEE Transactions on Biomedical Engineering* vol. 51, 912–918 (2004).
46. Lewicki, M. S. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems* vol. 9, R53–R78 (1998).
47. Rey, H. G., Pedreira, C. & Quiroga, R. Q. Past, present and future of spike sorting techniques. *Brain research bulletin* vol. 119, 106–117 (2015).
48. Lefebvre, B., Yger, P. & Marre, O. Recent progress in multi-electrode spike sorting methods. *Journal of Physiology-Paris* vol. 110, 327–335 (2016).
49. Quiroga, R. Q., Nadasdy, Z. & Ben-Shaul, Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation* vol. 16, 1661–1687 (2004).
50. Chaure, F. J., Rey, H. G. & Quiroga, R. A novel and fully automatic spike-sorting implementation with variable number of features. *Journal of neurophysiology* vol. 120, 1859–1871 (2018).
51. Rossant, C. *et al.* Spike sorting for large, dense electrode arrays. *Nature neuroscience* vol. 19, 634 (2016).
52. Jun, J. J. *et al.* Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *bioRxiv*, 101030 (2017).
53. Hilgen, G. *et al.* Unsupervised spike sorting for large-scale, high-density multielectrode arrays. *Cell reports* vol. 18, 2521–2532 (2017).
54. Chung, J. E., Magland, J. F., Barnett, A. H., *et al.* A fully automated approach to spike sorting. *Neuron* vol. 95, 1381–1394 (2017).

55. Gerstein, G. & Clark, W. Simultaneous studies of firing patterns in several neurons. *Science* vol. 143, 1325–1327 (1964).
56. Franke, F., Quiroga, R. Q., Hierlemann, A. & Obermayer, K. Bayes optimal template matching for spike sorting—combining fisher discriminant analysis with optimal filtering. *Journal of computational neuroscience* vol. 38, 439–459 (2015).
57. Pachitariu, M., Steinmetz, N. A., Kadir, S. N., *et al.* *Fast and accurate spike sorting of high-channel count probes with KiloSort* in *Advances in Neural Information Processing Systems* (2016), 4448–4456.
58. Yger, P. *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, e34518 (2018).
59. Diggelmann, R., Fiscella, M., Hierlemann, A. & Franke, F. Automatic spike sorting for high-density microelectrode arrays. *Journal of neurophysiology* vol. 120, 3155–3171 (2018).
60. Pachitariu, M., Steinmetz, N. A. & Colonell, J. *Kilosort2* <https://github.com/MouseLand/Kilosort2>.
61. Takahashi, S., Sakurai, Y., Tsukada, M. & Anzai, Y. Classification of neuronal activities from tetrode recordings using independent component analysis. *Neurocomputing* vol. 49, 289–298 (2002).
62. Takahashi, S., Anzai, Y. & Sakurai, Y. A new approach to spike sorting for multi-neuronal activities recorded with a tetrode—how ICA can be practical. *Neuroscience research* vol. 46, 265–272 (2003).
63. Takahashi, S. & Sakurai, Y. Real-time and automatic sorting of multi-neuronal activity for sub-millisecond interactions in vivo. *Neuroscience* vol. 134, 301–315 (2005).
64. Jäckel, D., Frey, U., Fiscella, M., *et al.* Applicability of independent component analysis on high-density microelectrode array recordings. *Journal of neurophysiology* vol. 108, 334–348 (2012).
65. Leibig, C., Wachtler, T. & Zeck, G. Unsupervised neural spike sorting for high-density microelectrode arrays with convolutive independent component analysis. *Journal of neuroscience methods* vol. 271, 1–13 (2016).
66. Hill, D. N., Mehta, S. B. & Kleinfeld, D. Quality metrics to accompany spike sorting of extracellular signals. *Journal of Neuroscience* vol. 31, 8699–8705 (2011).
67. Harris, K. D., Hirase, H., Leinekugel, X., Henze, D. A. & Buzsáki, G. Temporal interaction between single spikes and complex spike bursts in hippocampal pyramidal cells. *Neuron* vol. 32, 141–149 (2001).
68. Schmitzer-Torbert, N. & Redish, A. D. Neuronal activity in the rodent dorsal striatum in sequential navigation: separation of spatial and reward responses on the multiple T task. *Journal of neurophysiology* vol. 91, 2259–2272 (2004).

69. Pouzat, C., Mazor, O. & Laurent, G. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. *Journal of neuroscience methods* vol. 122, 43–57 (2002).
70. Barnett, A. H., Magland, J. F. & Greengard, L. F. Validation of neural spike sorting algorithms without ground-truth information. *Journal of neuroscience methods* vol. 264, 65–77 (2016).
71. Wehr, M., Pezaris, J. S. & Sahani, M. Simultaneous paired intracellular and tetrode recordings for evaluating the performance of spike sorting algorithms. *Neurocomputing* vol. 26, 1061–1068 (1999).
72. Henze, D. A. *et al.* Intracellular features predicted by extracellular recordings in the hippocampus in vivo. *Journal of neurophysiology* vol. 84, 390–400 (2000).
73. Harris, K. D., Henze, D. A., Csicsvari, J., Hirase, H. & Buzsaki, G. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *Journal of neurophysiology* vol. 84, 401–414 (2000).
74. Neto, J. P., Lopes, G., Frazão, J., *et al.* Validating silicon polytrodes with paired juxtacellular recordings: method and dataset. *Journal of Neurophysiology* vol. 116, 892–903 (2016).
75. Marques-Smith, A. *et al.* Recording from the same neuron with high-density CMOS probes and patch-clamp: a ground-truth dataset and an experiment in collaboration. *bioRxiv*, 370080 (2018).
76. Allen, B. D. *et al.* Automated in vivo patch clamp evaluation of extracellular multielectrode array spike recording capability. *Journal of neurophysiology* (2018).
77. Einevoll, G. T., Franke, F., Hagen, E., *et al.* Towards reliable spike-train recordings from thousands of neurons with multielectrodes. *Current opinion in neurobiology* vol. 22, 11–17 (2012).
78. Camuñas-Mesa, L. A. & Quiroga, R. Q. A detailed and fast model of extracellular recordings. *Neural computation* vol. 25, 1191–1212 (2013).
79. Hagen, E. *et al.* ViSAPy: a Python tool for biophysics-based generation of virtual spiking activity for evaluation of spike-sorting algorithms. *Journal of neuroscience methods* vol. 245, 182–204 (2015).
80. Mondragón-González, S. L. & Burguière, E. Bio-inspired benchmark generator for extracellular multi-unit recordings. *Scientific reports* vol. 7, 43253 (2017).
81. Rossant, C. & Harris, K. D. Hardware-accelerated interactive data visualization for neuroscience in Python. *Frontiers in neuroinformatics* vol. 7, 36 (2013).
82. Rossant, C., Kadir, S., Goodman, D., Hunter, M. & Harris, K. *Phy* <https://github.com/cortex-lab/phy>.

83. Jackson, A., Mavoori, J. & Fetz, E. E. Long-term motor cortex plasticity induced by an electronic neural implant. *Nature* vol. 444, 56–60 (2006).
84. Oliynyk, A., Bonifazzi, C., Montani, F. & Fadiga, L. Automatic online spike sorting with singular value decomposition and fuzzy C-mean clustering. *BMC neuroscience* vol. 13, 96 (2012).
85. Franke, F., Natora, M., Boucsein, C., Munk, M. H. & Obermayer, K. An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes. *Journal of computational neuroscience* vol. 29, 127–148 (2010).
86. Franke, F. *et al.* High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity. *Frontiers in neural circuits* vol. 6, 105 (2012).
87. Garcia, S. & Pouzat, C. *Tridesclous* <https://github.com/tridesclous/tridesclous>.
88. Hsu, S.-H., Mullen, T., Jung, T.-P. & Cauwenberghs, G. *Online recursive independent component analysis for real-time source separation of high-density EEG in 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (2014), 3845–3848.
89. Hsu, S.-H., Pion-Tonachini, L., Jung, T.-P. & Cauwenberghs, G. *Tracking non-stationary EEG sources using adaptive online recursive independent component analysis in 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2015), 4106–4109.
90. Hsu, S.-H., Mullen, T. R., Jung, T.-P. & Cauwenberghs, G. Real-time adaptive EEG source separation using online recursive independent component analysis. *IEEE transactions on neural systems and rehabilitation engineering* vol. 24, 309–319 (2015).
91. Markram, H. *et al.* Interneurons of the neocortical inhibitory system. *Nature reviews neuroscience* vol. 5, 793 (2004).
92. Gouwens, N. W. *et al.* Classification of electrophysiological and morphological neuron types in the mouse visual cortex. *Nature neuroscience*, 1 (2019).
93. Harris, K. D. & Shepherd, G. M. The neocortical circuit: themes and variations. *Nature neuroscience* vol. 18, 170 (2015).
94. Isaacson, J. S. & Scanziani, M. How inhibition shapes cortical activity. *Neuron* vol. 72, 231–243 (2011).
95. McCormick, D. A., Connors, B. W., Lighthall, J. W. & Prince, D. A. Comparative electrophysiology of pyramidal and sparsely spiny stellate neurons of the neocortex. *Journal of neurophysiology* vol. 54, 782–806 (1985).

96. Barthó, P. *et al.* Characterization of neocortical principal cells and interneurons by network interactions and extracellular features. *Journal of Neurophysiology* vol. 92, 600–608 (2004).
97. Peyrache, A. *et al.* Spatiotemporal dynamics of neocortical excitation and inhibition during human sleep. *Proceedings of the National Academy of Sciences* vol. 109, 1731–1736 (2012).
98. Sirota, A. *et al.* Entrainment of neocortical neurons and gamma oscillations by the hippocampal theta rhythm. *Neuron* vol. 60, 683–697 (2008).
99. Jia, X. *et al.* High-density extracellular probes reveal dendritic backpropagation and facilitate neuron classification. *Journal of neurophysiology* vol. 121, 1831–1847 (2019).
100. Lima, S. Q., Hromádka, T., Znamenskiy, P. & Zador, A. M. PINP: a new method of tagging neuronal populations for identification during in vivo electrophysiological recording. *PLoS one* vol. 4, e6099 (2009).
101. Fenno, L., Yizhar, O. & Deisseroth, K. The development and application of optogenetics. *Annual review of neuroscience* vol. 34 (2011).
102. Radivojevic, M. *et al.* Electrical identification and selective microstimulation of neuronal compartments based on features of extracellular action potentials. *Scientific reports* vol. 6, 31332 (2016).
103. Obien, M. E. J., Hierlemann, A. & Frey, U. Accurate signal-source localization in brain slices by means of high-density microelectrode arrays. *Scientific reports* vol. 9, 788 (2019).
104. Mechler, F., Victor, J. D., Ohiorhenuan, I., Schmid, A. M. & Hu, Q. Three-dimensional localization of neurons in cortical tetrode recordings. *Journal of Neurophysiology* vol. 106, 828–848 (2011).
105. Mechler, F. & Victor, J. D. Dipole characterization of single neurons from their extracellular action potentials. *Journal of Computational Neuroscience* vol. 32, 73–100 (2012).
106. Somogyvári, Z., Zalányi, L., Ulbert, I. & Érdi, P. Model-based source localization of extracellular action potentials. *Journal of Neuroscience Methods* vol. 147, 126–137 (2005).
107. Somogyvári, Z., Cserpán, D., Ulbert, I. & Érdi, P. Localization of single-cell current sources based on extracellular potential patterns: the spike CSD method. *European Journal of Neuroscience* vol. 36, 3299–3313 (2012).
108. Bakkum, D. J. *et al.* Tracking axonal action potential propagation on a high-density microelectrode array across hundreds of sites. *Nature Communications* vol. 4 (2013).
109. Chelaru, M. I. & Jog, M. S. Spike source localization with tetrodes. *Journal of Neuroscience Methods* vol. 142, 305–315 (2005).

110. Kubo, T., Katayama, N., Karashima, A. & Nakao, M. *The 3D position estimation of neurons in the hippocampus based on the multi-site multi-unit recordings with silicon tetrodes in 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (2008), 5021–5024.
111. Ruz, I. D. & Schultz, S. R. Localising and classifying neurons from high density MEA recordings. *Journal of Neuroscience Methods* vol. 233, 115–128 (2014).
112. Shew, W. L., Bellay, T. & Plenz, D. Simultaneous multi-electrode array recording and two-photon calcium imaging of neural activity. *Journal of Neuroscience Methods* vol. 192, 75–82 (2010).
113. Bakkum, D. J. *et al.* The Axon Initial Segment is the Dominant Contributor to the Neuron’s Extracellular Electrical Potential Landscape. *Advanced biosystems* vol. 3, 1800308 (2019).
114. Teleńczuk, M., Brette, R., Destexhe, A. & Teleńczuk, B. Contribution of the axon initial segment to action potentials recorded extracellularly. *eNeuro* vol. 5 (2018).
115. Clark, G. in *Speech processing in the auditory system* 422–462 (Springer, 2004).
116. Zrenner, E. Will retinal implants restore vision? *Science* vol. 295, 1022–1025 (2002).
117. Perlmutter, J. S. & Mink, J. W. Deep brain stimulation. *Annu. Rev. Neurosci.* Vol. 29, 229–257 (2006).
118. Bensmaia, S. J. & Miller, L. E. Restoring sensorimotor function through intracortical interfaces: progress and looming challenges. *Nature Reviews Neuroscience* vol. 15, 313 (2014).
119. Wagner, F. B. *et al.* Targeted neurotechnology restores walking in humans with spinal cord injury. *Nature* vol. 563, 65 (2018).
120. Moritz, C. T., Perlmutter, S. I. & Fetz, E. E. Direct control of paralysed muscles by cortical neurons. *Nature* vol. 456, 639 (2008).
121. Rebesco, J. M., Stevenson, I. H., Koording, K., Solla, S. A. & Miller, L. E. Rewiring neural interactions by micro-stimulation. *Frontiers in systems neuroscience* vol. 4, 39 (2010).
122. De Lavilléon, G., Lacroix, M. M., Rondi-Reig, L. & Benchenane, K. Explicit memory creation during sleep demonstrates a causal role of place cells in navigation. *Nature neuroscience* vol. 18, 493 (2015).
123. Diamantaki, M. *et al.* Manipulating hippocampal place cell activity by single-cell stimulation in freely moving mice. *Cell reports* vol. 23, 32–38 (2018).
124. Tehovnik, E., Tolias, A., Sultan, F., Slocum, W. & Logothetis, N. Direct and indirect activation of cortical neurons by electrical microstimulation. *Journal of neurophysiology* vol. 96, 512–521 (2006).

125. Nowak, L. & Bullier, J. Axons, but not cell bodies, are activated by electrical stimulation in cortical gray matter I. Evidence from chronaxie measurements. *Experimental brain research* vol. 118, 477–488 (1998).
126. Rattay, F. The basic mechanism for the electrical stimulation of the nervous system. *Neuroscience* vol. 89, 335–346 (1999).
127. Joucla, S., Branchereau, P., Cattaert, D. & Yvert, B. Extracellular neural microstimulation may activate much larger regions than expected by simulations: a combined experimental and modeling study. *PLoS One* vol. 7, e41324 (2012).
128. Rabinovitch, A., Braunstein, D., Biton, Y., Friedman, M. & Aviram, I. The Weiss–Lapicque and the Lapicque–Blair strength–duration curves revisited. *Biomedical Physics & Engineering Express* vol. 2, 015019 (2016).
129. Durand, D. Electric stimulation of excitable tissue. *The Biomedical Engineering Handbook* vol. 2 (2000).
130. Merrill, D. R., Bikson, M. & Jefferys, J. G. Electrical stimulation of excitable tissue: design of efficacious and safe protocols. *Journal of Neuroscience Methods* vol. 141, 171–198 (2005).
131. McNeal, D. R. Analysis of a model for excitation of myelinated nerve. *IEEE Transactions on Biomedical Engineering*, 329–337 (1976).
132. Rattay, F. Analysis of models for extracellular fiber stimulation. *IEEE Transactions on Biomedical Engineering* vol. 36, 676–682 (1989).
133. Joucla, S. & Yvert, B. Modeling extracellular electrical neural stimulation: from basic understanding to MEA-based applications. *Journal of Physiology-Paris* vol. 106, 146–158 (2012).
134. Aberra, A. S., Peterchev, A. V. & Grill, W. M. Biophysically realistic neuron models for simulation of cortical stimulation. *Journal of neural engineering* vol. 15, 066023 (2018).
135. McIntyre, C. C. & Grill, W. M. Extracellular stimulation of central neurons: influence of stimulus waveform and frequency on neuronal output. *Journal of neurophysiology* vol. 88, 1592–1604 (2002).
136. Moulin, C. *et al.* A new 3-D finite-element model based on thin-film approximation for microelectrode array recording of extracellular action potential. *IEEE Transactions on Biomedical Engineering* vol. 55, 683–692 (2008).
137. Joucla, S. & Yvert, B. Improved focalization of electrical microstimulation using microelectrode arrays: a modeling study. *PloS ONE* vol. 4, e4828 (2009).
138. Lempka, S. F. & McIntyre, C. C. Theoretical analysis of the local field potential in deep brain stimulation applications. *PloS one* vol. 8, e59839 (2013).

139. Joucla, S., Glière, A. & Yvert, B. Current approaches to model extracellular electrical neural microstimulation. *Frontiers in computational neuroscience* vol. 8, 13 (2014).
140. Dadarlat, M. C., Sun, Y. & Stryker, M. P. *Widespread activation of awake mouse cortex by electrical stimulation in 2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)* (2019), 1113–1117.
141. Ronchi, S. *et al.* Single-Cell Electrical Stimulation Using CMOS-Based High-Density Microelectrode Arrays. *Frontiers in neuroscience* vol. 13, 208 (2019).
142. Pettersen, K. H. & Einevoll, G. T. Amplitude variability and extracellular low-pass filtering of neuronal spikes. *Biophysical journal* vol. 94, 784–802 (2008).
143. Holt, G. R. & Koch, C. Electrical interactions via the extracellular potential near cell bodies. *Journal of computational neuroscience* vol. 6, 169–184 (1999).
144. Anastassiou, C. A., Perin, R., Markram, H. & Koch, C. Ephaptic coupling of cortical neurons. *Nature neuroscience* vol. 14, 217 (2011).
145. Bokil, H., Laaris, N., Blinder, K., Ennis, M. & Keller, A. Ephaptic interactions in the mammalian olfactory system. *Journal of Neuroscience* vol. 21, RC173–RC173 (2001).
146. Anastassiou, C. A. & Koch, C. Ephaptic coupling to endogenous electric field activity: why bother? *Current opinion in neurobiology* vol. 31, 95–103 (2015).
147. Carnevale, N. T. & Hines, M. L. *The NEURON book* (Cambridge University Press, 2006).
148. Hines, M. L. & Carnevale, N. T. The NEURON simulation environment. *Neural computation* vol. 9, 1179–1209 (1997).
149. Bower, J. M. & Beeman, D. *The book of GENESIS: exploring realistic neural models with the GEneral NEural SIMulation System* (Springer Science & Business Media, 2012).
150. Hines, M., Davison, A. P. & Muller, E. NEURON and Python. *Frontiers in neuroinformatics* vol. 3, 1 (2009).
151. Nunez, P. L., Srinivasan, R., *et al.* *Electric fields of the brain: the neurophysics of EEG* (Oxford University Press, USA, 2006).
152. Lindén, H., Hagen, E., Leski, S., *et al.* LFPy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons. *Frontiers in Neuroinformatics* vol. 7, 41 (2014).
153. Goto, T. *et al.* An evaluation of the conductivity profile in the somatosensory barrel cortex of Wistar rats. *Journal of neurophysiology* vol. 104, 3388–3412 (2010).

154. Hagen, E., Næss, S., Ness, T. V. & Einevoll, G. T. Multimodal Modeling of Neural Network Activity: Computing LFP, ECoG, EEG, and MEG Signals With LFPy 2.0. *Frontiers in neuroinformatics* vol. 12 (2018).
155. Gratiy, S. L. *et al.* BioNet: A Python interface to NEURON for modeling large-scale networks. *PloS one* vol. 13, e0201630 (2018).
156. Rattay, F. Analysis of models for external stimulation of axons. *IEEE transactions on biomedical engineering*, 974–977 (1986).
157. Joucla, S. & Yvert, B. The “mirror” estimate: an intuitive predictor of membrane polarization during extracellular stimulation. *Biophysical journal* vol. 96, 3495–3508 (2009).
158. Lertmanorat, Z. & Durand, D. M. A novel electrode array for diameter-dependent control of axonal excitability: a simulation study. *IEEE transactions on biomedical engineering* vol. 51, 1242–1250 (2004).
159. Capogrosso, M. *et al.* A computational model for epidural electrical stimulation of spinal sensorimotor circuits. *Journal of Neuroscience* vol. 33, 19326–19340 (2013).
160. Lubba, C. H. *et al.* PyPNS: Multiscale Simulation of a Peripheral Nerve in Python. *Neuroinformatics*, 1–19 (2018).
161. Agudelo-Toro, A. & Neef, A. Computationally efficient simulation of electrical activity at cell membranes interacting with self-generated and externally imposed electric fields. *Journal of neural engineering* vol. 10, 026019 (2013).
162. Zienkiewicz, O. C., Taylor, R. L., Nithiarasu, P. & Zhu, J. *The finite element method* (McGraw-hill London, 1977).
163. Næss, S. *et al.* Corrected four-sphere head model for EEG signals. *Frontiers in human neuroscience* vol. 11, 490 (2017).
164. Moffitt, M. A. & McIntyre, C. C. Model-based analysis of cortical recording with silicon microelectrodes. *Clinical neurophysiology* vol. 116, 2240–2250 (2005).
165. Geuzaine, C. & Remacle, J.-F. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering* vol. 79, 1309–1331 (2009).
166. Logg, A., Mardal, K.-A. & Wells, G. *Automated solution of differential equations by the finite element method: The FEniCS book* (Springer Science & Business Media, 2012).
167. Jutten, C. & Herault, J. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal processing* vol. 24, 1–10 (1991).
168. Bell, A. J. & Sejnowski, T. J. An information-maximization approach to blind separation and blind deconvolution. *Neural computation* vol. 7, 1129–1159 (1995).

-
169. Hyvärinen, A. & Oja, E. Independent component analysis: algorithms and applications. *Neural networks* vol. 13, 411–430 (2000).
 170. Lee, T.-W., Girolami, M. & Sejnowski, T. J. Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. *Neural computation* vol. 11, 417–441 (1999).
 171. Lee, T.-W., Girolami, M., Bell, A. J. & Sejnowski, T. J. A unifying information-theoretic framework for independent component analysis. *Computers & Mathematics with Applications* vol. 39, 1–21 (2000).
 172. Akhtar, M. T., Jung, T.-P., Makeig, S. & Cauwenberghs, G. *Recursive independent component analysis for online blind source separation in 2012 IEEE International Symposium on Circuits and Systems* (2012), 2813–2816.
 173. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *Journal of machine learning research* vol. 12, 2825–2830 (2011).
 174. Gramfort, A. *et al.* MNE software for processing MEG and EEG data. *Neuroimage* vol. 86, 446–460 (2014).
 175. Delorme, A. & Makeig, S. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of neuroscience methods* vol. 134, 9–21 (2004).
 176. Rosenblatt, F. Perceptron simulation experiments. *Proceedings of the IRE* vol. 48, 301–309 (1960).
 177. Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of eugenics* vol. 7, 179–188 (1936).
 178. Goodfellow, I., Bengio, Y. & Courville, A. *Deep learning* (MIT press, 2016).
 179. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition in Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770–778.
 180. Patterson, J. & Gibson, A. *Deep learning: A practitioner's approach* ("O'Reilly Media, Inc.", 2017).
 181. Krizhevsky, A., Sutskever, I. & Hinton, G. E. *Imagenet classification with deep convolutional neural networks in Advances in neural information processing systems* (2012), 1097–1105.
 182. Zeiler, M. D. & Fergus, R. *Visualizing and understanding convolutional networks in European conference on computer vision* (2014), 818–833.
 183. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* vol. 15, 1929–1958 (2014).
 184. Martin Abadi *et al.* *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015.

Bibliography

185. Paszke, A. *et al.* *Automatic Differentiation in PyTorch* in *NIPS Autodiff Workshop* (2017).
186. Chollet, F. *et al.* *Keras* <https://github.com/fchollet/keras>. 2015.
187. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
188. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).
189. Venter, G. Review of optimization techniques. *Encyclopedia of aerospace engineering* (2010).
190. Jones, E., Oliphant, T., Peterson, P., *et al.* *SciPy: Open source scientific tools for Python* 2001.
191. Fortin, F.-A., Rainville, F.-M. D., Gardner, M.-A., Parizeau, M. & Gagné, C. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research* vol. 13, 2171–2175 (2012).
192. Oliphant, T. E. Python for scientific computing. *Computing in Science & Engineering* vol. 9, 10–20 (2007).
193. Millman, K. J. & Aivazis, M. Python for scientists and engineers. *Computing in Science & Engineering* vol. 13, 9–12 (2011).
194. Van Der Walt, S., Colbert, S. C. & Varoquaux, G. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* vol. 13, 22 (2011).
195. Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in science & engineering* vol. 9, 90 (2007).
196. Michael, W., Olga, B., Drew, O., *et al.* *Seaborn: v0.8.1 (September 2017)* 2017.
197. McKinney, W. *et al.* *Data structures for statistical computing in python* in *Proceedings of the 9th Python in Science Conference* **445** (2010), 51–56.
198. Kluyver, T. *et al.* *Jupyter Notebooks – a publishing format for reproducible computational workflows* in *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (eds Loizides, F. & Schmidt, B.) (2016), 87–90.
199. Muller, E. *et al.* Python in neuroscience. *Frontiers in neuroinformatics* vol. 9, 11 (2015).
200. Gleeson, P., Davison, A. P., Silver, R. A. & Ascoli, G. A. A commitment to open source in neuroscience. *Neuron* vol. 96, 964–965 (2017).
201. Vassanelli, S. in *Nanotechnology and Neuroscience: Nano-electronic, Photonic and Mechanical Neuronal Interfacing* 239–267 (Springer, 2014).
202. Steinmetz, N., Zátka-Haas, P., Carandini, M. & Harris, K. Distributed correlates of visually-guided behavior across the mouse brain. *bioRxiv*, 474437 (2018).

203. Gallego, J. A., Perich, M. G., Miller, L. E. & Solla, S. A. Neural manifolds for the control of movement. *Neuron* vol. 94, 978–984 (2017).
204. Trautmann, E. M. *et al.* Accurate estimation of neural population dynamics without spike sorting. *Neuron* (2019).
205. Chaudhuri, R., Gerçek, B., Pandey, B., Peyrache, A. & Fiete, I. The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nature neuroscience*, 1–9 (2019).
206. Saxena, S. & Cunningham, J. P. Towards the neural population doctrine. *Current opinion in neurobiology* vol. 55, 103–111 (2019).
207. Gallego, J. A. *et al.* Cortical population activity within a preserved neural manifold underlies multiple motor behaviors. *Nature communications* vol. 9, 4233 (2018).
208. Pandarinath, C. *et al.* Latent factors and dynamics in motor cortex and their application to brain–machine interfaces. *Journal of Neuroscience* vol. 38, 9390–9401 (2018).
209. Hubel, D. H. & Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. *The Journal of physiology* vol. 160, 106–154 (1962).
210. O’Keefe, J. & Dostrovsky, J. The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research* (1971).
211. Fyhn, M., Molden, S., Witter, M. P., Moser, E. I. & Moser, M.-B. Spatial representation in the entorhinal cortex. *Science* vol. 305, 1258–1264 (2004).
212. Hafting, T., Fyhn, M., Molden, S., Moser, M.-B. & Moser, E. I. Microstructure of a spatial map in the entorhinal cortex. *Nature* vol. 436, 801 (2005).
213. Quiroga, R. Q., Reddy, L., Kreiman, G., Koch, C. & Fried, I. Invariant visual representation by single neurons in the human brain. *Nature* vol. 435, 1102 (2005).
214. Stosiek, C., Garaschuk, O., Holthoff, K. & Konnerth, A. In vivo two-photon calcium imaging of neuronal networks. *Proceedings of the National Academy of Sciences* vol. 100, 7319–7324 (2003).
215. Grinvald, A. & Hildesheim, R. VSDI: a new era in functional imaging of cortical dynamics. *Nature Reviews Neuroscience* vol. 5, 874 (2004).
216. Xu, Y., Zou, P. & Cohen, A. E. Voltage imaging with genetically encoded indicators. *Current opinion in chemical biology* vol. 39, 1–10 (2017).
217. Chamberland, S. *et al.* Fast two-photon imaging of subcellular voltage dynamics in neuronal tissue with genetically encoded indicators. *Elife* vol. 6, e25690 (2017).
218. Abdelfattah, A. S. *et al.* Bright and photostable chemigenetic indicators for extended in vivo voltage imaging. *Science* vol. 365, 699–704 (2019).

219. Stringer, C. *et al.* Spontaneous behaviors drive multidimensional, brainwide activity. *Science* vol. 364, 255–255 (2019).
220. Forli, A. *et al.* Two-photon bidirectional control and imaging of neuronal excitability with high spatial resolution in vivo. *Cell reports* vol. 22, 3087–3098 (2018).
221. Emiliani, V., Cohen, A. E., Deisseroth, K. & Häusser, M. All-optical interrogation of neural circuits. *Journal of Neuroscience* vol. 35, 13917–13926 (2015).
222. Chen, I.-W., Papagiakoumou, E. & Emiliani, V. Towards circuit optogenetics. *Current opinion in neurobiology* vol. 50, 179–189 (2018).
223. Ouzounov, D. G. *et al.* In vivo three-photon imaging of activity of GCaMP6-labeled neurons deep in intact mouse brain. *Nature methods* vol. 14, 388 (2017).
224. Dombeck, D. A., Harvey, C. D., Tian, L., Looger, L. L. & Tank, D. W. Functional imaging of hippocampal place cells at cellular resolution during virtual navigation. *Nature neuroscience* vol. 13, 1433 (2010).
225. Aharoni, D. B. & Hoogland, T. Circuit investigations with open-source miniaturized microscopes: past, present and future. *Frontiers in cellular neuroscience* vol. 13, 141 (2019).
226. Krupic, J., Bauza, M., Burton, S. & O’Keefe, J. Local transformations of the hippocampal cognitive map. *Science* vol. 359, 1143–1146 (2018).
227. Juavinett, A. L., Bekheet, G. & Churchland, A. K. Chronically-implanted Neuropixels probes enable high yield recordings in freely moving mice. *eLife* vol. 8, e47188 (2019).
228. Abbott, L. F. *et al.* An international laboratory for systems and computational neuroscience. *Neuron* vol. 96, 1213–1218 (2017).
229. Major, G., Larkum, M. E. & Schiller, J. Active properties of neocortical pyramidal neuron dendrites. *Annual review of neuroscience* vol. 36, 1–24 (2013).
230. Ascoli, G. A., Donohue, D. E. & Halavi, M. NeuroMorpho. Org: a central resource for neuronal morphologies. *Journal of Neuroscience* vol. 27, 9247–9251 (2007).
231. Buccino, A. P. *et al.* Open source modules for tracking animal behavior and closed-loop stimulation based on Open Ephys and Bonsai. *Journal of neural engineering* vol. 15, 055002 (2018).
232. Lopes, G. *et al.* Bonsai: an event-based framework for processing and controlling data streams. *Frontiers in neuroinformatics* vol. 9, 7 (2015).
233. Siegle, J. H. *et al.* Open Ephys: an open-source, plugin-based platform for multichannel electrophysiology. *Journal of neural engineering* vol. 14, 045003 (2017).

234. Hurwitz, C. L., Xu, K., Srivastava, A., Buccino, A. P. & Hennig, M. Scalable Spike Source Localization in Extracellular Recordings using Amortized Variational Inference. *QAccepted to NeurIPS 2019, arXiv preprint arXiv:1905.12375* (2019).

Papers

Paper I

**MEArec: a fast and customizable
testbench simulator for
ground-truth extracellular spiking
activity**

MEArec: a fast and customizable testbench simulator for ground-truth extracellular spiking activity

Alessio P. Buccino*¹ and Gaute T. Einevoll^{1,2}

¹Centre for Integrative Neuroplasticity (CINPLA), University of Oslo, Oslo, Norway

²Faculty of Science and Technology, Norwegian University of Life Sciences, Ås, Norway

Abstract

When recording neural activity from extracellular electrodes, both *in vivo* and *in vitro*, spike sorting is a required and very important processing step that allows for identification of single neurons' activity. Spike sorting is a complex algorithmic procedure, and in recent years many groups have attempted to tackle this problem, resulting in numerous methods and software packages. However, validation of spike sorting techniques is complicated. It is an inherently unsupervised problem and it is hard to find universal metrics to evaluate performance. Simultaneous recordings that combine extracellular and patch-clamp or juxtacellular techniques can provide ground-truth data to evaluate spike sorting methods. However, their utility is limited by the fact that only a few cells can be measured at the same time. Simulated ground-truth recordings can provide a powerful alternative mean to rank the performance of spike sorters. We present here **MEArec**, a Python-based software which permits flexible and fast simulation of extracellular recordings.

MEArec allows users to generate extracellular signals on various customizable electrode designs and can replicate various problematic aspects for spike sorting, such as bursting, spatio-temporal overlapping events, and drifting. We expect **MEArec** will provide a common testbench for spike sorting development and evaluation, in which spike sorting developers can rapidly generate and evaluate the performance of their algorithms.

keywords: Spike sorting; Benchmark data; Extracellular recordings simulator; Open-source software

Introduction

Extracellular neural electrophysiology is one of the most used and important techniques to study brain function. It consists of measuring the electrical activity of neurons from electrodes in the extracellular space, that pick up the electrical activity of surrounding neurons. To communicate with each other, neurons generate action potentials, which can be identified in the recorded signals as fast potential transients called *spikes*.

Since electrodes can record the extracellular activity of several surrounding neurons, a processing step called spike sorting is needed. Historically this has required manual curation of the data, which in addition to being time consuming also introduces human bias to data interpretations. In recent years, several automated spike sorters have been developed to alleviate this problems. Spike sorting algorithms [40, 28] attempt to separate spike trains of different neurons (units) from the extracellular mixture of signals using a variety of different approaches. After a pre-processing step that usually involves high-pass filtering and re-referencing of the signals to reduce noise, some algorithms first detect putative spikes above a detection threshold and then cluster the extracted and aligned waveforms in

*alessiob@ifl.uio.no

a lower-dimensional space [38, 41, 8, 22, 25]. Another approach consists of finding spike templates, using clustering methods, and then matching the templates recursively to the recordings to find when a certain spike has occurred. The general term for these approaches is template-matching [37, 43, 10]. Other approaches have been explored, including the use of independent component analysis [24, 3] and semi-supervised approaches [27].

The recent development of high-density silicon probes both for *in vitro* [2, 13] and *in vivo* applications [35, 26] poses new challenges for spike sorting [42]. The high electrode count calls for fully automatic spike sorting algorithms, as the process of manually curating hundreds or thousands of channels becomes more time consuming and less manageable. Therefore, spike sorting algorithms need to be capable of dealing with a large number of units and dense probes. To address these requirements, the latest developments in spike sorting software have attempted to make algorithms scalable and hardware-accelerated [37, 25, 43].

The evaluation of spike sorting performance is also not trivial. Spike sorting is unsupervised by definition, as the recorded signals are only measured extracellularly with no knowledge of the underlying spiking activity. A few attempts at providing ground-truth datasets, for example by combining extracellular and patch-clamp or juxtacellular recordings [21, 19, 35, 43, 31, 1] exist, but the main limitation of this approach is that only one or a few cells can be patched at the same time, providing very limited ground-truth information with respect to the number of neurons that can be recorded simultaneously from extracellular probes.

Biophysically detailed simulated data provide a powerful alternative and complementary approach to spike sorting validation [11]. In simulations, recordings can be built from known ground-truth data for all neurons, which allows one to precisely evaluate the performance of spike sorters. Simulators of extracellular activity should be able to replicate important aspects of spiking activity that can be challenging for spike sorting algorithms, including bursting modulation, spatio-temporal overlap of spikes, unit drifting over time, as well as realistic noise models. Moreover, they should allow users to have full control over these features and they should be efficient and fast.

In the last years, there have been a few projects aiming at developing neural simulators for benchmarking spike sorting methods [6, 18, 33]: Camunas et al. developed **NeuroCube** [6], a MATLAB-based simulator which combines biophysically detailed cell models and synthetic spike trains (a so called "hybrid approach") to simulate the activity of neurons close to a recording probe, while noise is simulated by the activity of distant neurons. **NeuroCube** is very easy to use with a simple and intuitive graphical user interface (GUI). The user has direct control of parameters to control the rate of active neurons, their firing rate properties, and the duration of the recordings. The cell models are shipped with the software and recordings can be simulated on a single electrodes or a tetrode. It is relatively fast, but the cell model simulations (using **NEURON** [7]) are re-simulated for every recording.

Hagen et al. developed **ViSAPy** [18], a Python-based simulator that uses multi-compartment simulation of single neurons to generate spikes, network modeling of point-neurons in **NEST** [9] to generate synaptic inputs onto the spiking neurons, and experimentally fitted noise. **ViSAPy** does not use a hybrid approach, as it runs a full network simulation in **NEURON** [7] and computes the extracellular potentials using **LFPy** [29, 17]. **ViSAPy** implements a Python application programming interface (API) which allows the user to set multiple parameters for the network simulation providing the synaptic input, the probe design, and the noise model generator. Cell models can be freely chosen and loaded using the **LFPy** package. Further, 1-dimensional drifting can be incorporated in the simulations by shifting the electrodes over time [12]. Learning to use the software and, in particular, tailoring the specific properties of the resulting spike trains, for example burstiness, requires some effort by the user. As the running of **NEURON** simulations with biophysically detailed neurons can be computationally expensive, the use of **ViSAPy** to generate long-duration spike-sorting benchmarking data is boosted by access to powerful computers.

Mondragon et al. developed a Neural Benchmark Simulator (**NBS**) [33] extending the **NeuroCube** software. **NBS** extends the capability of **NeuroCube** for using user-specific probes, and it combines the spiking activity signals (from **NeuroCube**), with low-frequency activity signals, and artifacts libraries shipped with the code. The user can set different weight parameters to assemble the spiking, low-

frequency, and artifact signals, but these three signal types are not modifiable.

Despite the existence of such tools for generating benchmarking data, their use in spike sorting literature has until now been limited, and the benchmarking and validation of spike sorting algorithms non-standardized and unsystematic. A natural question to ask is thus how to best stimulate the use of such benchmarking tools in the spike sorting community.

From a spike sorting developer perspective, we argue that an ideal extracellular simulator should be *i)* fast, *ii)* controllable, *iii)* biophysically detailed, and *iv)* easy to use. A fast simulator would enable spike sorter developers to generate a large and varied set of recordings to test their algorithms against and to improve their spike sorting methods. Controllability refers to the possibility to have direct control of features of the simulated recordings. The ideal extracellular spike simulator should include the possibility to use different cell models and types, to decide the firing properties of the neurons, to control the temporal and spatio-temporal synchrony of extracellular spikes, to generate recordings on different probe models, and to have full reproducibility of the simulated recordings. A biophysically detailed simulator should be capable of reproducing key physiological aspects of the recordings, including, but not limited to, bursting spikes, drifting between the electrodes and the neurons, and realistic noise profiles. Finally, to maximize the ease of use, the ideal extracellular simulator should be designed as an accessible and easy to learn software package. Preferably, the tool should be implemented with a graphical user interface (GUI), a command line interface (CLI), or with a simple application programming interface (API).

With these principles in mind, we present here **MEArec**, an open-source Python-based simulator. **MEArec** provides a fast, highly controllable, biophysically detailed, and easy to use framework to generate simulated extracellular recordings. In addition to producing benchmark datasets, we developed **MEArec** as a powerful tool that can serve as a testbench for optimizing existing and novel spike sorting methods. To facilitate this goal, **MEArec** allows users to explore how several aspects of recordings affect spike sorting, with full control of challenging features such as bursting activity, drifting, spatio-temporal synchrony, and noise effects, so that spike sorter developers can use it to help their algorithm design. Moreover, **MEArec** has an extensive documentation <https://mearec.readthedocs.io/> and the code is tested with a continuous integration platform¹.

Results

Getting started with **MEArec**: a simple tetrode dataset generation

One of the key goals of **MEArec** is to ease the simulation of extracellular recordings and make it fully reproducible. In order to demonstrate this, we first show and break down the commands used to generate a simple tetrode recording that we will to further characterize in the rest of the paper.

MEArec, at installation, comes with 13 layer 5 cortical cell models from the Neocortical Microcircuit Portal [39, 4]. This enables the user to dive into simulations without the need to download and compile cell models. On the other hand, the initial cell models can be easily extended by downloading more cell models and placing them in the cell models folder.

To generate 30 extracellular spikes (also referred as templates) per cell model recorded on a shank tetrode probe, the user can simply run this command:

```
>> mearec gen-templates -prb tetrode-mea-1 -n 30 --seed 0
...
Saved templates in path-to-templates-file.h5
```

The `-prb` option allows for choosing the probe model, `-n` controls the number of templates per cell model to generate, and the `--seed` option is used to ensure reproducibility and if it is not provided, a random seed is chosen. In both cases, the seed is saved in the **HDF5** file, so that the same templates can be perfectly replicated.

¹<https://travis-ci.org/>

Once the templates are generated, recordings can be generated as follows:

```
>> mearec gen-recordings -t path-to-templates-file.h5 -d 30 -ne 4 -ni 2
--st-seed 0 --temp-seed 1 --noise-seed 2
...
Saved recordings in path-to-recordings-file.h5
```

The `gen-recordings` command combines the selected templates from 4 excitatory cells (`-ne 4`) and 2 inhibitory cells (`-ni 2`), that usually have a more narrow spike waveform and a higher firing rate, with randomly generated spike trains. The duration of the output recordings is 30 seconds (`-d 30`). In this case, three random seeds control the spike train random generation (`--st-seed 0`), the template selection (`--temp-seed 1`), and the noise generation (`--noise-seed 2`). Figure 1 shows one second of the generated recordings (A), the extracted waveforms and the mean waveforms for each unit on the electrode with the largest peak (B), and the principal component analysis (PCA) projections of the waveforms on the tetrode channels.

MEAREC also implements a convenient Python API, which is run internally by the CLI commands. For example, the following snippet of code implements the same commands shown above for generating templates and recordings:

```
import MEAREC as mr

# generate templates
templates_params = mr.get_default_templates_params()
cell_models_folder = mr.get_default_cell_models()
templates_params['probe'] = 'tetrode-mea-1'
templates_params['n'] = 30
templates_params['seed'] = 0
tempgen = mr.gen_templates(cell_models_folder=cell_models_folder,
                           params=templates_params)
mr.save_template_generator_templates(tempgen, 'path-to-templates-file.h5')

# generate recordings
recordings_params = mr.get_default_recordings_params()
recordings_params['spiketrains']['n_exc'] = 4
recordings_params['spiketrains']['n_inh'] = 2
recordings_params['spiketrains']['duration'] = 30
recordings_params['spiketrains']['seed'] = 0
recordings_params['templates']['seed'] = 1
recordings_params['recordings']['seed'] = 2
recgen = mr.gen_recordings(params=recordings_params,
                           templates='path-to-templates-file.h5')
mr.save_recording_generator(recgen, 'path-to-recordings-file.h5')
```

Moreover, the Python API implements plotting functions to visually inspect the simulated templates and recordings. For example, Figure 1 panels were generated using the `plot_recordings()` (A), `plot_waveforms()` (B), and `plot_pca_map()` (C) functions.

MEAREC overview

After having shown how to generate a recording in MEAREC, we introduce here an overview of the software (Figure 2). The simulation is split in two phases: *templates generation* (Figure 2A) and *recordings generation* (Figure 2B).

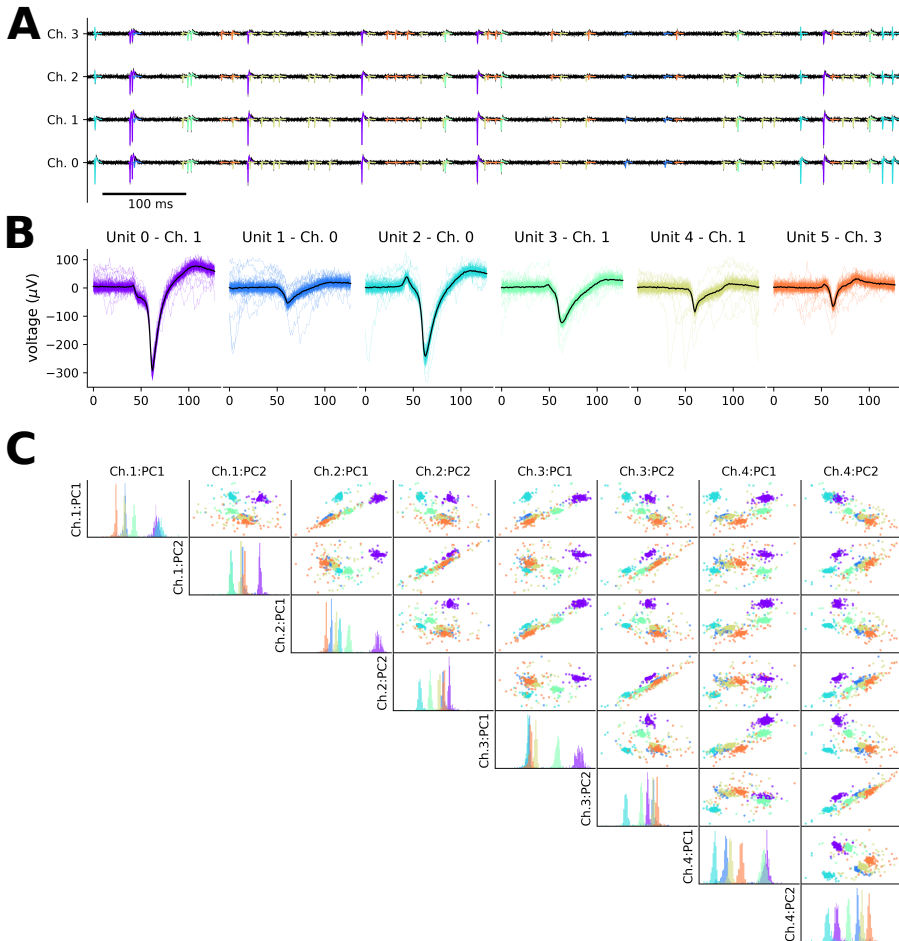


Figure 1: Example of simulated tetraode recording. (A) One second of the recording timeseries on the four tetraode channels. The templates for the different units are overlapped to the recording traces in different colors. (B) Extracted waveforms on the channel with the largest amplitude for the six units in the recordings. (C) PCA projections on the first two PC components of the four tetraode channels. Each color corresponds to a neuron. The diagonal plots display the histograms of the PC projection on the corresponding channel.

Templates (or extracellular action potentials) are generated using biophysically realistic cell models which are positioned in the surroundings of a probe model. The templates generation output is a library of a large variety of extracellular templates, which can then be used to build the recordings. The templates generation phase is the most time consuming, but the same templates library can be used to generate a virtually infinite number of different recordings.

Recordings are then generated by combining templates selected with user-defined rules (based on minimum distance between neurons, amplitudes, spatial overlaps, and cell-types) and by simulating spike trains. Selected templates and spike trains are assembled using a customized (or modulated) convolution, which can replicate interesting features of spiking activity such as bursting and drifting. After convolution, additive noise is generated and added to the recordings. Finally, the output recordings can be optionally filtered with a band-pass or a high-pass filter. For a full description of the templates and recordings generation, please refer to the Materials and Methods section.

MEAreC is designed to allow for full customization, transparency, and reproducibility of the simulated recordings. Parameters for the templates and recordings generation are accessible by the user and documented, so that different aspects of the simulated signals can be finely tuned (see Materials and Methods for a list of parameters and their explanation). Moreover, the implemented command line interface (CLI) and simple Python API, enables the user to easily modify parameters, customize, and run simulations.

Finally, MEAreC permits to manually set several random seeds used by the simulator to make recordings fully reproducible. This feature also enables one to study how separate characteristics of the recordings affect the spike sorting performance. As an example, we will show in the next sections how to simulate a recording sharing all parameters, hence with exactly the same spiking activity, but with different noise levels or drifting velocities.

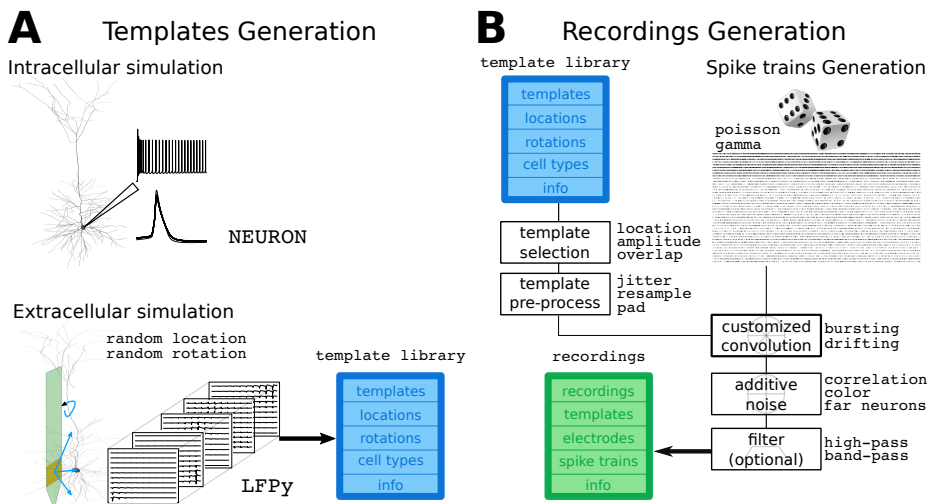


Figure 2: Overview of the MEAreC software. The simulation is divided in two phases: *templates generation* and *recordings generation*. (A) The *templates generation* phase is split in an intracellular and extracellular simulation. The intracellular simulation computes, for each available cell model, the transmembrane currents generated by several action potentials. In the extracellular simulation, each cell model is randomly moved and rotated several times and the stored currents are loaded to the model to compute the extracellular action potential, building a *template library*. (B) The *recordings generation* phase combines templates selected from the *template library* and randomly generated spike trains. Selected templates are pre-processed before a customized convolution with the spike trains. Additive noise is added to the output of the convolution, and the recordings can be optionally filtered.

Generation of realistic Multi-Electrode Array recordings

The recent development of Multi-Electrode Arrays (MEAs) enables researchers to record extracellular activity at very high spatio-temporal density both for *in vitro* [2, 13] and *in vivo* applications [35, 26]. The large number of electrodes and their high density can result in challenges for spike sorting algorithms. It is therefore important to be able to simulate recordings from these kind of neural probes.

To deal with different probe designs, MEArec uses another Python package (MEAutility - <https://meautility.readthedocs.io/>), that allows users to easily import several available probe models and to define custom probe designs. Among others, MEAutility include Neuropixels probes [26], Neuronexus commercial probes (<http://neuronexus.com/products/neural-probes/>), and a wide variety of square MEA designs with different contact densities (the list of available probes can be found in Appendix A).

Similarly to the tetrode example, we first have to generate templates for the probes. These are the commands to generate templates and recordings for a Neuropixels design with 128 electrodes (Neuropixels-128). The recordings contain 60 neurons, 48 excitatory and 12 inhibitory. With similar commands, we generated templates and recordings for a Neuronexus probe with 32 channels (A1x32-Poly3-5mm-25s-177-CM32 - Neuronexus-32) with 20 cells, and a square 10x10 MEA with 15 μ m inter-electrode-distance (SqMEA-10-15) and 50 cells.

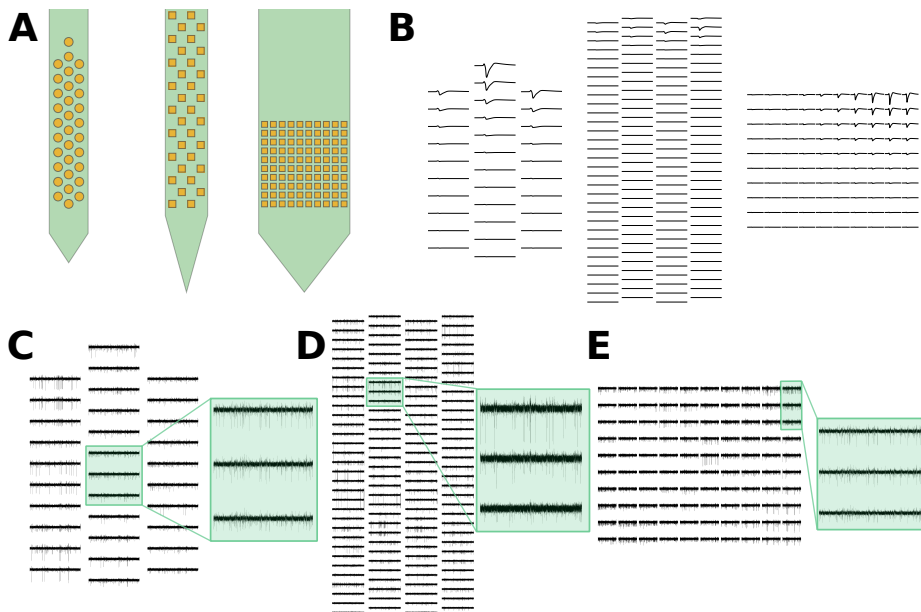


Figure 3: Generation of high-density multi-electrode array recordings. (A) Example of three available probes: a commercial Neuronexus probe (left), the Neuropixels probe (middle), and a high-density square MEA. (B) Sample templates for each probe design. (C-D-E) One-second snippets of recordings from the Neuronexus probe (C), the Neuropixels probe (D), and the square MEA probe (E). The highlighted windows display the activity over three adjacent channels and show how the same spikes are seen on multiple sites.

```
>> mearec gen-templates -prb Neuropixels-128 -n 100 --seed 0
...
Saved templates in path-to-Neuropixels-templates-file.h5

>> mearec gen-recordings -t path-to-Neuropixels-templates-file.h5 -d 30 -ne 48 -ni 12
--st-seed 0 --temp-seed 1 --noise-seed 2
...
Saved recordings in path-to-Neuropixels-recordings-file.h5
```

Figure 3 shows the three above-mentioned probes (A), a sample template for each probe design (B), and one-second snippets of the three recordings (C-D-E), with zoomed in window to highlight spiking activity.

While all the recordings shown so far have been simulated with default parameters, several aspects of the spiking activity are critical for spike sorting. In the next sections, we will show how these features, including bursting, spatio-temporal overlapping spikes, drifting, and noise assumptions can be explored with MEAREC simulations.

Bursting modulation of spike amplitude and shape

Bursting activity is one of the most complicated features of spiking activity that can compromise the performance of spike sorting algorithms. When a neuron bursts, i.e. it fires repeated and fast action potentials, the dynamics underlying the generation of the spikes changes over the bursting period [20]. While the bursting mechanism has been largely studied with patch-clamp experiments, combined extracellular-juxtacellular recordings [1] and computational studies [18] suggest that during bursting, extracellular spikes become lower in amplitude and wider in shape.

In order to simulate this property of the extracellular waveforms in a fast and efficient manner, templates are modulated both in amplitude and shape during the convolution operation, depending on the spiking history.

To demonstrate how bursting is replicated, we built a constant spike train with 10 ms inter-spike-interval (Figure 4A). A modulation value is computed for each spike and is used to modulate the convolution operation for that event. The blue dots show the default modulation, in which the modulation values are drawn from a Gaussian distribution with unitary mean. When bursting is enabled, the modulation value is computed as a sublinear power depending of the number of consecutive spikes in a burst and the inter-spike-interval (see Materials and Methods for details). The bursting events can be either controlled by the maximum number of spikes making a burst (orange - 5 spikes; green - 10 spikes) or by setting a maximum bursting duration (red - 75 ms).

The modulation value controls the level of amplitude and shape modulation of the spike event. In Figure 4B, examples of bursting templates are shown. The blue traces display templates only modulated in amplitude, i.e. the amplitude is scaled by the modulation value. The orange and green traces, instead, also present shape modulation, which is achieved by stretching the time axis using a sigmoid transform. The sigmoid transform can be adjusted to have more (green) or less (orange) shape modulation.

Figure 4C shows a one-second snippet of the tetrode recording shown previously after bursting modulation is activated. The top panel shows the spike events, the middle one displays the modulation values, and the bottom panel shows the output of the modulated convolution between one of the templates (on the electrode with the largest amplitude) and the spike train.

Figure 4D and Figure 4E show the waveform projections on the first principal component for the tetrode recording shown previously with and without bursting, respectively. In this case all neurons are bursting units and this causes a stretch in the PCA space, which is a clear complication for spike sorting algorithms.

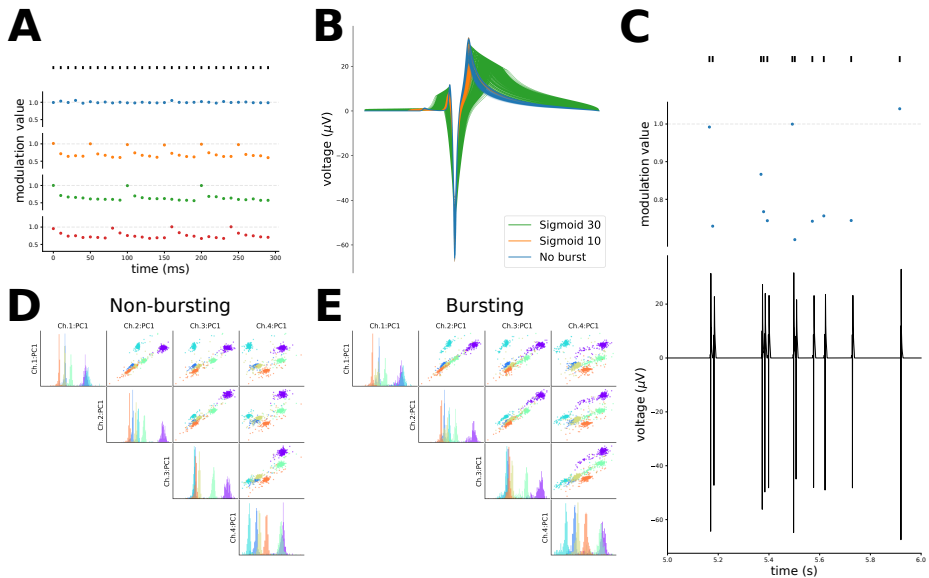


Figure 4: Bursting behavior. (A) Modulation values computation for a sample spike train of 300 ms with constant inter-spike-intervals of 10 ms. The blue dots show the modulation values for each spike when bursting is not activated: each value is drawn from a $\mathcal{N}(1, 0.05^2)$ distribution. When bursting is activated, a bursting event can be limited by the maximum number of spikes (orange - 5 spikes, green - 10 spikes), or by the maximum bursting event duration (red - 75 ms). (B) Modulated templates. The blue lines show templates modulated in amplitude only. The orange and green lines display the same templates with added shape modulation. (C) Modulation in tetraode recordings. The top panel shows spikes in a one-second period. The middle panel displays the modulation values for those spikes. The bottom panel shows the modulated template on the electrode with the largest peak after convolution. (D-E) PCA projections on the first principal component for the tetraode recordings without bursting (D) and with bursting (E) enabled. Note that the PCA projections were computed in both cases from the waveforms without bursting. The clusters, with bursting, become more spread and harder to separate than without bursting.

Controlling spatio-temporal overlaps

Another complicated aspect of extracellular spiking activity that can influence spike sorting performance is the occurrence of overlapping spikes. While temporal overlapping of events on spatially separated locations can be solved with feature masking [41], spatio-temporal overlapping can cause a distortion of the detected waveform, due to the superposition of separate spikes. Some spike sorting approaches, based on template-matching, are designed to tackle this problem [37, 43, 10].

In order to evaluate to what extent spatio-temporal overlap affects spike sorting, *MEArec* allows the user to set the number of spatially overlapping templates and to modify the synchrony rate of their spike trains. In Figure 5 we show an example of this on a *Neuronexus-32* probe (see Figure 3A). The recording was constructed with two excitatory and spatially overlapping neurons, whose templates are shown in Figure 5A (see Materials and Methods for details on spatial overlap definition). The spike synchrony rate can be controlled with the `sync_rate` parameter. If this parameter is not set

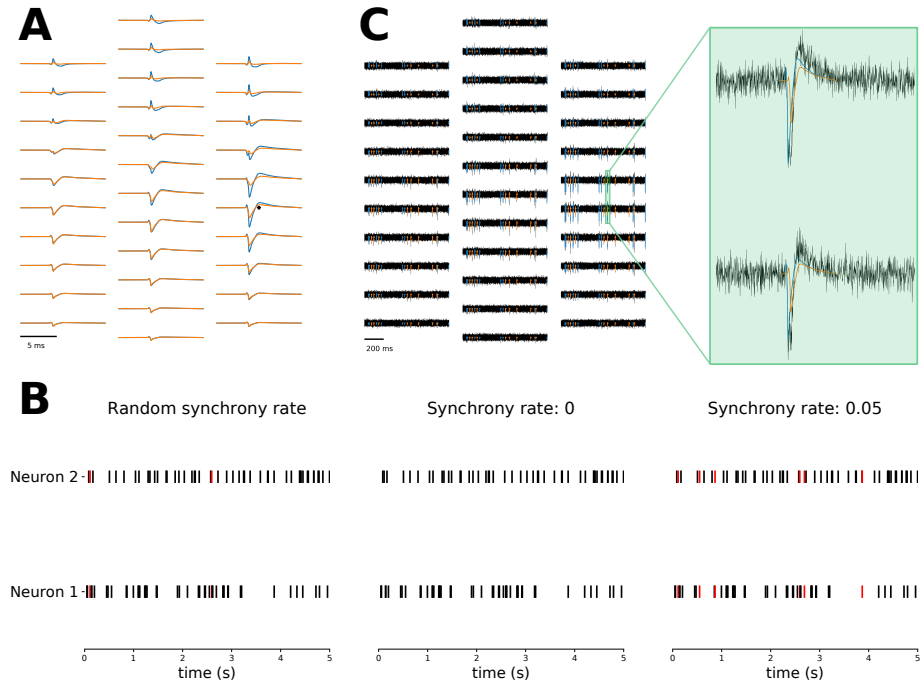


Figure 5: Controlling spatio-temporal overlapping spikes. (A) Example of two spatially overlapping templates. The two templates are spatially overlapping because on the electrode with the largest signal (depicted as an black asterisk) for template A (blue), template B has an amplitude greater than the 90% of its largest amplitude. (B) Without setting the synchrony rate, the random spike trains (left) present a few spatio-temporal collisions (red events). When setting the synchrony rate to 0 (middle), the spatio-temporal overlaps are removed. When the synchrony rate is set to 0.05 (right), spatio-temporal overlapping spikes are added to the spike trains. (C) One-second snippet of the recording with 0.05 synchrony. In the magnified window, a spatio-temporal overlapping event is shown: the collision results in a distortion of the waveform.

(Figure 5B - left), some spatio-temporal overlapping spikes are present (red events). If the synchrony rate is set to 0, those spikes are removed from the spike trains (Figure 5B - middle). If set to 0.05, i.e. 5% of the spikes will be spatio-temporal collisions, events are added to the spike trains to reach the specified synchrony rate value of spatio-temporal overlap. As shown in Figure 5C, the occurrence of spatio-temporal overlapping events affects the recorded extracellular waveform: the waveforms of the neurons, in fact, get summed and might be mistaken for a separate unit by spike sorting algorithms when the spikes are overlapping.

The possibility of reproducing and controlling this feature of extracellular recordings within MEArec could aid in the development of spike sorters which are robust to spatio-temporal collisions.

Generating drifting recordings

When extracellular probes are inserted in the brain, especially for acute experiments, the neural tissue might slowly move with respect to the electrodes. This phenomenon is known as drift.

Drifting is particularly critical for spike sorting, as the waveform shapes change over time due to the relative movement between the neurons and the probe. New spike sorting algorithms have been developed to specifically tackle the drifting problem (Kilosort2², IronClust [25]).

In order to simulate drift in the recordings, we first need to generate drifting templates:

```
>> mearec gen-templates -prb Neuronexus-32 -n 30 --drifting --seed 0
...
Saved templates in path-to-Neuronexus-drift-templates-file.h5
```

Drifting templates are generated by choosing an initial and final soma position with user-defined rules (see Materials and Methods for details) and by moving the cell along the line connecting the two positions for a defined number of drifting steps (50 by default). An example of a drifting template is depicted in Figure 6A, alongside with the drifting neuron's soma locations.

Once a library of drifting templates is generated, drifting recordings can be simulated. Depending on the drifting velocity, the drifting template is *replayed* so that, for each spike, the correct drifting template is selected for convolution. In Figure 6B, we show an example with four drifting cells, a drifting velocity of 20 $\mu\text{m/s}$, and a duration of 60 seconds. The colored arrows show the initial and final positions of the four neurons making up the recording. Note that a drifting velocity of 20 $\mu\text{m/s}$ is much larger than normal experimental drifts, and it has been chosen to illustrate the drifting phenomenon. Figure 6C shows the waveforms and the average waveforms for the four neurons on the

²e.g. <https://github.com/MouseLand/Kilosort2>

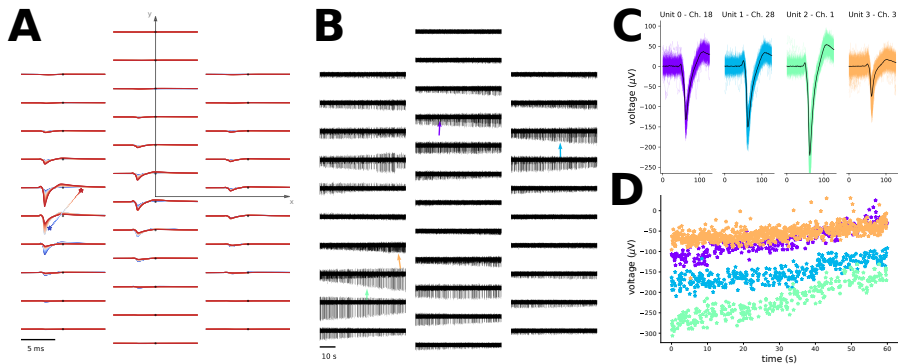


Figure 6: Drifting. (A) Example of a drifting template. The colored asterisks on the left show the trajectory from the initial (blue large asterisk) to the final (red large asterisk) neuron positions. The positions are in the x-y coordinates of the probe plane, and the electrode locations are depicted as black dots. The corresponding templates are displayed at the electrode locations with the same colormap, showing that the template peak is shifted upwards following the soma position. (B) 60-second drifting recording with four neurons moving at a velocity of 20 $\mu\text{m/s}$. The colored arrows show the initial and final soma positions for each neuron. (C) Waveforms and average waveforms on the electrode with the largest peak for each of the four neurons in the recording. (D) Amplitude of the waveforms over time recorded on the electrode with the largest initial peak. Drifting results in a slow change of amplitude over the course of the recording.

electrode with the largest peak. In this case, the variability is mainly due to the relative movement between the cell and the probe. This can be observed by visualizing the waveform amplitude for each spike over time (Figure 6C - each color is a different neuron).

Modeling experimental noise

Spike sorting performance can be greatly affected by noise in the recordings. Many algorithms first use a spike detection step to identify putative spikes. The threshold for spike detection is usually set depending on the noise standard deviation or mean average deviation [38]. Clearly, recordings with larger noise levels will result in higher spike detection thresholds, hence making it harder to robustly detect lower amplitude spiking activity. In addition to the noise amplitude, other noise features can affect spike sorting performance: some clustering algorithms, for example, assume that clusters have Gaussian shape, due to the assumption of an additive normal noise to the recordings. Moreover, the noise generated by biological sources can produce spatial correlations in the noise profiles among different channels and it can be modulated in frequency [6, 40].

To investigate how the above-mentioned assumptions on noise can affect spike sorting performance, MEArec can generate recordings with several noise models. Figure 7 shows 5-second spiking-free recordings of a tetrode probe for five different noise profiles that can be generated (A - recordings, B - spectrum, C - channel covariance, D - amplitude distribution).

The first column shows uncorrelated Gaussian noise, which presents a flat spectrum, a diagonal covariance matrix, and a symmetrical noise amplitude distribution. In the recording in the second column, spatially correlated noise was generated as a multivariate Gaussian noise with a covariance matrix depending on the channel distance. Also in this case, the spectrum (B) presents a flat profile and the amplitude distribution is symmetrical (D), but the covariance matrix shows a correlation depending on the inter-electrode distance. As previous studies showed [6, 40], the frequency content of extracellular noise is not flat, but its spectrum is affected by the spiking activity of distant neurons, which appear in the recordings as below-threshold *biological* noise. To reproduce the spectrum profile that is observed in experimental data, MEArec allows coloring the noise spectrum of Gaussian noise with a second order infinite impulse response (IIR) filter (see Materials and Methods for details). Colored noise represents an efficient way of obtaining the desired spectrum, as shown in the third and fourth columns of Figure 7, panel B. Distance correlation is maintained (panel C - fourth column), and the distribution of the noise amplitudes is symmetrical. Finally, a last noise model enables one to generate activity of distant neurons. In this case, noise is built as the convolution between many neurons (300 by default) whose template amplitudes are below an amplitude threshold (10 μ V by default). A Gaussian noise floor is then added to the resulting noise, which is scaled to match the user-defined noise level. The *far-neurons* noise profile is shown in the last column of Figure 7. While the spectrum and spatial correlation of this noise profile are similar to the ones generated with a colored, distance-correlated noise (4th column), the shape of the noise distribution is skewed towards negative values (panel D), mainly due to the negative contribution of the action potentials.

The capability of MEArec to simulate several noise models enables spike sorter developers to assess how different noise profiles affect their algorithms and to modify their methods to be insensitive to specific noise assumptions.

Testbench for spike sorting development and assessment

In the previous sections, we have shown several examples on how MEArec is capable of reproducing several aspects of extracellular recordings which are critical for spike sorting performance, in a fully reproducible way. The proposed design and its integration with a spike sorting evaluation framework called SpikeInterface³ enables developers to actively include customized simulations in the spike sorting development phase.

³<https://github.com/SpikeInterface>

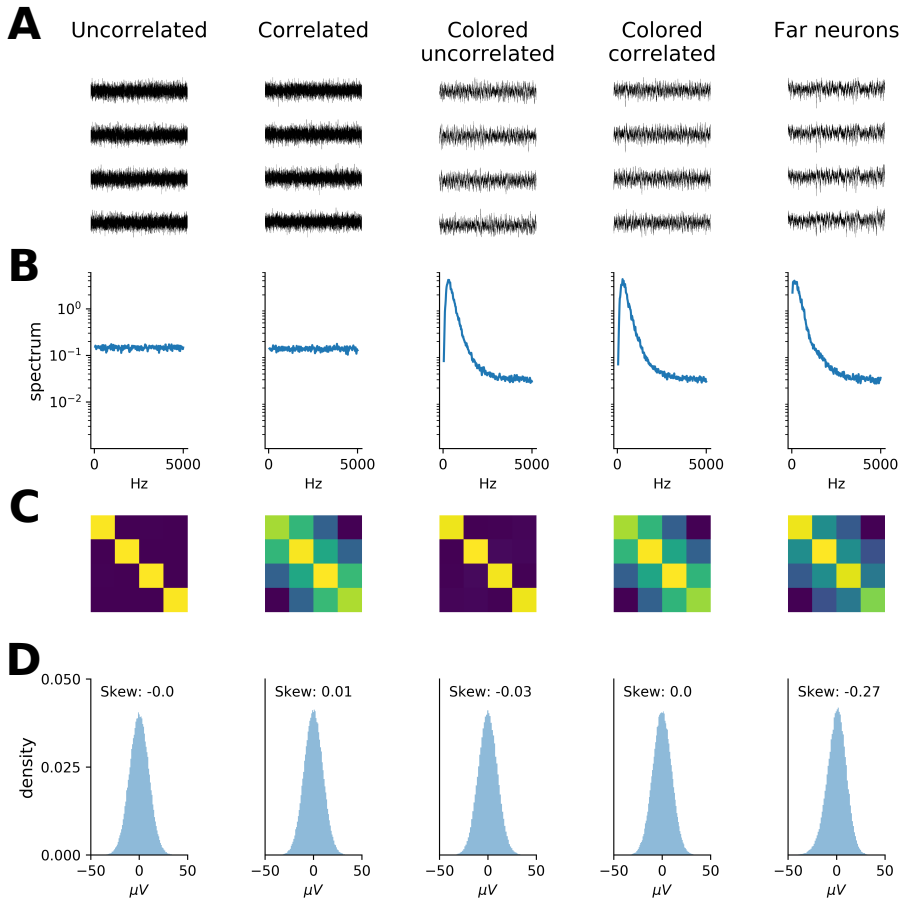


Figure 7: Noise models. The 5 columns refer to different noise models: 1) Uncorrelated Gaussian noise, 2) Distance-correlated Gaussian noise, 3) Colored uncorrelated Gaussian noise, 4) Colored distance-correlated Gaussian noise, and 5) Noise generated by distant neurons. (A) One-second spiking-free recording. (B) Spectrum of the first recording channel between 10 and 5000 Hz. (C) Covariance matrix of the recordings. (D) Distribution of noise amplitudes for the first recording channel. The different noise models vary in the spectrum, channel correlations, and amplitude distributions.

Due to its speed and controllability, we see MEArec as a *testbench*, rather than a *benchmark* tool. We provide here a couple of examples. In Figure 8A, we show a one-second section of recordings simulated on a Neuronexus-32 probe with fixed parameters and random seeds regarding template selection and spike train generation, but with four different levels of additive Gaussian noise, with standard deviations of 5, 10, 20, and 30 μV (Appendix B contains the Python code used to generate and plot these recordings). The traces show the same underlying spiking activity, so the only variability

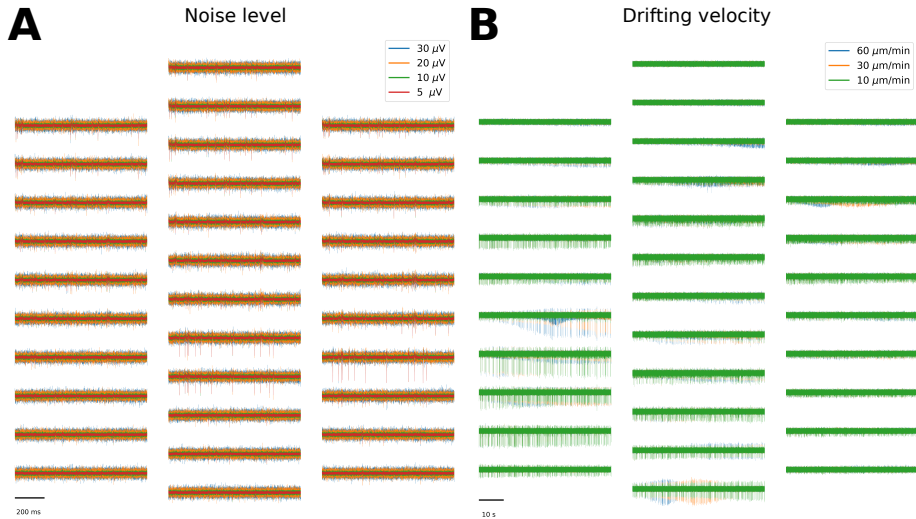


Figure 8: MEArec as testbench platform for spike sorting. (A) Four one-second snippets of recordings generated with a different noise level parameter (5 - red, 10 - green, 20 - blue, and 30 μV - blue). The underlying spiking activity is exactly the same for all recordings, and the only difference lie in the standard deviation of the underlying uncorrelated Gaussian noise. (B) Three drifting recordings generated with a different drifting velocity parameter (10 - green, 30 - blue, and 60 $\mu\text{m/s}$ - blue). Also in this case, the underlying spiking activity is the same, but it can be observed how the different speeds result in a modification of waveforms over time.

in spike sorting performance will be due to the varying noise levels. Similarly, in Figure 8B, 1-minute drifting recordings were simulated with three different drifting velocities. The recordings show that for low drifting speeds the waveform changes are almost not visible (green traces), while for faster drifts (orange and blue traces), the waveform changes over time become more important.

The capability of MEArec of reproducing such behaviors in a highly controlled manner could aid in the design of specific tests for measuring and quantifying the ability of a spike sorting software to deal with specific complexities in extracellular recordings. Other examples include simulating a recording with increasing levels of bursting in order to measure to what extent bursting units are correctly clustered, or changing the synchrony rate of spatially overlapping units to assess how much spatio-temporal collisions affect performance.

Integration with SpikeInterface We have recently developed **SpikeInterface**, a Python-based framework for running several spike sorting algorithms, comparing, and validating their results. MEArec can be easily interfaced to **SpikeInterface** so that simulated recordings can be loaded, spike sorted, and benchmarked with a few lines of code. In the following example, a MEArec recording is loaded, spike sorted with **Mountainsort4** [8] and **Kilosort2**⁴ [37], and benchmarked with respect to the ground-truth spike times available from the MEArec simulation:

```
import spikeextractors as se
import spiketoolkit as st
```

⁴<https://github.com/MouseLand/Kilosort2>

```
# loading MEArec recording
recording = se.MEArecRecordingExtractor('path-to-recording.h5')
# loading ground-truth sorting
sorting_GT = se.MEArecSortingExtractor('path-to-recording.h5')

# run several spike sorters
sorting_MS4 = st.sorters.run_mountainsort4(recording)
sorting_KS2 = st.sorters.run_kilosort2(recording)

# compare with ground-truth and get performance
cmp_GT_MS4 = st.comparison.compare_sorter_to_ground_truth(sorting_GT, sorting_MS4)
cmp_GT_KS2 = st.comparison.compare_sorter_to_ground_truth(sorting_GT, sorting_KS2)

# get and print performance
cmp_GT_MS4.get_performance()
cmp_GT_KS2.get_performance()
```

The combination of `MEArec` and `SpikeInterface` represents a powerful tool for systematically testing and comparing spike sorter performances with respect to several complications of extracellular recordings. `MEArec` simulations, in combination with `SpikeInterface`, are already being used by other groups to benchmark and compare spike sorting algorithms⁵.

Performance considerations

As a testbench tool, the speed requirement has been one of the main design principle of `MEArec`. In order to achieve high speed, most parts of the simulation process are fully parallelized. As shown in Figure 2, the simulations are split in templates and recordings generation. The templates generation phase is the most time consuming, but the same template library can be used to generate several recordings. This phase is further split in two sub-phases: the intracellular and extracellular simulations. The former only needs to be run once, as it generates a set of cell model-specific spikes that are stored and then used for extracellular simulations, which is instead probe specific.

We present here run times for the different phases of the templates generation and for the recordings generation. All simulations were run on an Ubuntu 18.04 Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz, with 16 GB of RAM.

The intracellular simulation run time for the 13 cell models shipped with the software was ~ 130 seconds (~ 10 seconds per cell model).

Run times for extracellular simulations for several probe types, number of templates in the library, and drifting templates are shown in the *Templates generation* section of Table 1. The run times for this phase mainly depend on the number of templates to be generated (N templates column), on the minimum amplitude of accepted templates (Min. amplitude column), and especially on drifting (Drifting column). When simulating drifting templates, in fact, the number of actual extracellular spikes for each cell model is N templates times N drift steps. Note that in order to generate the *far-neurons* noise model, the minimum amplitude should be set to 0, so that low-amplitude templates are not discarded. The number of templates available in the template library will be the specified number of templates (N templates) times the number of cell models (13 by default).

Recordings are then generated using the simulated template libraries. In Table 1, the *Recordings generation* section shows run times for several recordings with different probes, durations, number of cells, bursting, and drifting options. The main parameter that affects simulation times is the number of cells, as it increases the number of modulated convolutions. Bursting and drifting behavior also increase the run time of the simulations, because of the extra processing required in the convolution

⁵<https://spikeforest.flatironinstitute.org>

Templates generation						
Probe	N templates	N channels	Min. amplitude	Drifting	N drift steps	Run time (s)
tetrode-mea-1	30	4	30	No	-	169
tetrode-mea-1	100	4	30	No	-	588
tetrode-mea-1	100	4	0	No	-	236
Neuronexus-32	100	32	30	No	-	567
Neuropixels-128	100	128	30	No	-	809
SqMEA-10-15	30	100	30	No	-	1027
Neuronexus-32	30	32	30	Yes	50	2000
Recordings generation						
Probe	N cells	N channels	Duration	Bursting	Drifting	Run time (s)
tetrode-mea-1	6	4	10	No	No	2
tetrode-mea-1	6	4	600	No	No	27
Neuronexus-32	20	32	30	No	No	12
Neuronexus-32	20	32	30	Yes	No	45
Neuropixels-128	60	128	30	No	No	62
SqMEA-10-15	50	100	30	No	No	48
Neuronexus-32	4	32	60	No	Yes	20
Neuronexus-32	20	32	60	No	Yes	53

Table 1: Templates and recordings generation run times depending on several simulation parameters.

step. The simulation run times, however, range from a few seconds to a few minutes. Therefore, the speed of **MEAreC** enables users to generate numerous recordings with different parameters for testing spike sorter performances.

Discussion

In this paper we have presented **MEAreC**, a Python package for simulating extracellular recordings for spike sorting development and validation. We first showed the ease of use of the software, whose command line interface and simple Python API enable users to simulate extracellular recordings with a couple of commands or a few lines of code. We then introduced an overview of the software function, consisting in separating the *templates* and the *recordings* generation to improve efficiency and simulation speed. We explored the capability of reproducing and controlling several aspects of extracellular recordings which can be critical for spike sorting algorithms, including spikes in a burst with varying spike shapes, spatio-temporal overlaps, drifting units, and noise assumptions. We illustrated two examples of using **MEAreC**, in combination with **SpikeInterface**⁶, as a testbench platform for developing spike sorting algorithms. Finally, we benchmarked the speed performance of **MEAreC** (Table 1).

Investigating the validation section of several recently developed spike sorting algorithms [41, 37, 26, 22, 25, 27, 43], it is clear that the neuroscientific community needs a standardized validation framework for spike sorting performance. Some spike sorters are validated using a so called hybrid approach, in which well-identified units from previous experimental recordings are artificially injected in the recordings and used to compute performance metrics [41, 37]. The use of templates extracted from previously sorted datasets poses some questions regarding the accuracy of the initial sorting, as well as the complexity of the well-identified units. Alternatively, other spike sorters are validated on experimental paired ground-truth recordings [8, 43]. While these valuable datasets [19, 21, 35, 31] can certainly provide useful information, the low count of ground-truth units makes the validation incomplete and could result in biases (for example algorithm-specific parameters could be tuned to reach a higher performance for the recorded ground-truth units). A third validation method consist of using simulated ground-truth recordings [11]. While this approach is promising, in combination with experimental paired recordings, the current available simulators [6, 18, 33] present some limitations in terms of biological realism, controllability, speed, and/or ease of use (see Introduction). We therefore introduced **MEAreC**, a software package which is computationally efficient, easy to use, highly controllable, and capable of reproducing critical characteristics of extracellular recordings relevant to

⁶<https://github.com/SpikeInterface>

spike sorting, including bursting modulation, spatio-temporal overlaps, drifting of units over time, and various noise profiles.

The capability of **MEArec** to replicate complexities in extracellular recordings which are usually either ignored or not controlled in other simulators, permits the user to include tailored simulations in the spike sorting implementation process, using the simulator as a testbench platform for algorithm development. **MEArec** simulations could not only be used to test the final product, but specific simulations could be used to help implementing algorithms that are able to cope with drifting, bursting, and spatio-temporal overlap, which are regarded as the most complex aspects for spike sorting performance [40, 43].

In **MEArec**, in order to generate extracellular templates, we used a well-established modeling framework for solving the single neuron dynamics [7], and for calculating extracellular fields generated by transmembrane currents [29, 18]. These models have some assumptions that, if warranted, could be addressed with more sophisticated methods, such as finite element methods (FEM). In a recent work [5], we used FEM simulations and showed that the extracellular probes, especially MEAs, affect the amplitude of the recorded signals. While this finding is definitely interesting for accurately modeling and understanding how the extracellular potential is generated and recorded, it is unclear how it would affect the spike sorting performance. Moreover, when modeling signals on MEAs, we used the method of images [34, 5], which models the probe as a infinite insulating plane and better describes the recorded potentials for large MEA probes [5].

Secondly, during templates generation, the neuron models were randomly moved around and rotated with physiologically acceptable values [4]. In this phase, some dendritic trees might unnaturally cross the probes. We decided to not modify the cell models and allow for this behavior for sake of efficiency of the simulator. The modification of the dendritic trees for each extracellular spike generation would in fact be too computationally intense. However, since the templates generation phase is only run once for each probes, in the future we plan to both to include the probe effect in the simulations and to carefully modify the dendritic positions so that they do not cross the probes' plane.

Another limitation of the proposed modelling approach is in the replication of bursting behavior. We implemented a simplified bursting modulation that attempts at capturing the features recorded from extracellular electrodes by modifying the template amplitude and shape depending on the spiking history. However, more advanced aspects of waveform modulation caused by bursting, including morphology-dependent variation of spike shapes, cannot be modelled with the proposed approach, and their replication requires a full multi-compartment simulation [18]. Nevertheless, the suggested simplified model of bursting could be a valuable tool for testing the capability of spike sorters to deal with this phenomenon.

Finally, the current version of **MEArec** only supports cell models from the Neocortical Microcircuit Portal [30, 39], which includes models from juvenile rat somatosensory cortex. The same cell model format is also being used to build a full hippocampus model [32] and other brain regions, and therefore the integration of new models should be straightforward. Moreover, we are in the process of extending the supported cell models for the Allen Brain Institute database [16]⁷, which contains models from mice and human cells. Further, by design, the templates and recordings generation phases are split. Therefore, the recordings generation mechanism could also be used, in principle, for user-defined template libraries, either from other unsupported cell models or from units extracted from experimental recordings.

In conclusion, we introduced **MEArec**, which is a Python-based simulation framework for extracellular recordings. Thanks to its speed and controllability, we see **MEArec** to aid both the development and validation spike sorting algorithms and to help understanding the limitation of current methods, to improve their performance, and to generate new software tools for the hard and still partially unsolved spike sorting problem.

⁷<https://celltypes.brain-map.org/>

Materials and Methods

Templates generation

This section explains the templates generation phase of the simulator (Figure 2A). Table 2 shows the list of parameters involved in this phase, their default values, types, and an explanation of their function.

MEAreC is compatible with realistic multi-compartment neuronal models from the Neocortex Microcircuit Portal (NMC - [39, 30]). Upon installation, 13 cell models from layer 5 are copied in the package folder. Moreover, the user can manually download other cell models from the portal and use them for the simulation.

Intracellular simulation

The neuronal model dynamics is solved using the **NEURON** simulator [7]. The neuron's soma is stimulated with a constant current for a user-defined simulation time (1 second by default - **sim_time** parameter) and the stimulation weight is adjusted (using the **weights** parameter) so that the number of spikes in the simulation period is within a target interval (between 3 and 50 by default - **target_spikes** parameter). The stimulation starts after **delay** ms from the start of the simulation to avoid initialization artifacts. The simulation time step is defined by the parameter **dt** (default is 0.03125 ms, corresponding to 32 kHz). Single spikes are then detected by threshold crossing, aligned, and cropped (using the **cut_out** parameter). The transmembrane currents of all segments are saved to disk, so that the intracellular simulation only needs to be run once for each cell model.

Extracellular simulation

Transmembrane currents generated by the *intracellular simulation* are used to compute extracellular potentials at the electrode locations using LFPy [17]. Transmembrane currents are distributed over a line source with the length of its corresponding neural segment. Using the quasi-static approximation [36] and with the assumption of a homogeneous, isotropic, and infinite neural tissue with conductivity $\sigma = 0.3 \text{ S/m}$ [15], the contribution of each compartment i at position \mathbf{r}_i with transmembrane current $I_i(t)$ to the electric potential on an electrode at position \mathbf{r}_j reads [23, 17, 4]:

$$\phi_i(\mathbf{r}_j, t) = \frac{1}{4\pi\sigma} I_i(t) \int \frac{dr_i}{\|\mathbf{r}_j - \mathbf{r}_i\|}. \quad (1)$$

While the assumption of an infinite milieu holds for small probes, such as microwires and tetrodes, when using larger silicon probes, the use of the method of images (MoI) [34] can yield a better estimate of the extracellular potential [5]. Using MoI, the contribution of a transmembrane current to an electrode at position \mathbf{r}_j reads:

$$\phi_i(\mathbf{r}_j, t) = \frac{1}{2\pi\sigma} I_i(t) \int \frac{dr_i}{\|\mathbf{r}_j - \mathbf{r}_i\|}. \quad (2)$$

The simulated extracellular spike is obtained by summing up the contributions of all compartments. For each recording site, the electric potential can be computed on several points within the electrode area (**ncontacts** parameter - 10 points by default), that are then averaged to model the spatial filtering properties of the electrodes (disk-electrode approximation [29]).

Each cell model, during the *templates generation* phase, is used to generate several spikes (**n** parameter - 50 by default). For each extracellular action potential, the neuron is randomly moved to a position within user-defined boundaries (**xlim**, **ylim**, **zlim** parameters). If the boundary for a specific axis is set to **null**, the limits are computed as the boundary of the probe in that axis plus the **overhang** value (default 30 μm). Moreover, a random rotation of the model can be optionally added (**rot** parameter). The models can be only shifted (**norot**), rotated along a single axis (**xrot**, **yrot**, **zrot**), rotated with a physiological rotation (**physrot**), or rotated randomly along all axes (**3drot**). For

further details we refer to [4]. Extracellular spikes are included in the dataset only if their maximum amplitude is greater than a user-defined minimum amplitude (`min_amp` parameter - 30 μ V by default). In order to use the *far-neurons* noise model (Figure 7), the minimum amplitude parameter should be set to 0, so that low amplitude templates are not discarded.

Probe models

Probe models are handled using the `MEUtility` Python package (<https://meutility.readthedocs.io/>), which is automatically installed upon `MEarec` installation. The probe type can be chosen using the `probe` parameter (if not set, a random probe will be selected). `MEUtility` contains a large variety of available probe designs, e.g. commercial Neuronexus probes, Neuropixels [26], and high-density square MEA (Figure 3), and it also allow users to define new probes using a `yaml` file or a Python dictionary. The probe definition contains information about the number and arrangement of the electrodes, the electrode shape and size (used for spatial filtering), the plane in which electrodes are located, and the probe type (`wire` or `mea`), which tells the simulator whether to use the infinite assumption (Equation 1) or MoI (Equation 2) for the extracellular potential calculation. In order to list the available probes and their information, one can use the `mearec available-probes --info` command.

Drifting templates

When inserting recording probes in the brain, over time there might be relative movement between the probe and the tissue, which causes a so-called drift in the recorded action potentials. In order to incorporate this phenomenon in the simulation of the recordings, drifting templates has to be generated (when the `drifting` parameter is set to true). From an initial random position of the cell model which satisfies the requirements in terms of location (within boundaries) and amplitude (above the detection threshold) a final drifting position is found so that the same conditions are satisfied. Moreover, the user can choose preferred drifting direction by setting the `drift_xlim`, `drift_ylim`, and `drift_zlim` parameters. When the final position is selected, the cell model is moved along a straight line connecting the initial and final position and the extracellular spike is simulated for `drift_steps` equidistant points (30 points by default) along this line (Figure 6A).

The templates generation phase can be reproduced by setting the `seed` parameter, which is randomly selected if it is set to `null`.

Recordings generation

When a template library is generated, it can be used to generate many recordings, as shown in Figure 2B. Tables 3 and 4 show the list of parameters involved in the recordings generation phase, their default values, types, and an explanation of their function.

Spike trains generations

In order to obtain the spiking activity, spike trains have to be generated. All the spike train generation parameters can be found in the `spiketrains` section of the recordings parameters.

Spike trains can be generated either as Poisson or Gamma processes (`process` parameter). If the Gamma process is selected, its shape is controlled by the `gamma_shape` parameter (default is 2). The user can decide the number of excitatory (`n_exc`) and inhibitory neurons (`n_inh`) in the recordings. The average and standard deviation of the firing rates of excitatory and inhibitory neurons can be chosen (with the `f_exc`, `f_inh`, `st_exc`, and `st_inh` parameters), as well as the minimum accepted firing rate (`min_rate` - default 0.5 Hz). Alternatively, the user can define the type (E-I) and mean firing rate of all neurons in the recordings. As Poisson and Gamma processes do not have a minimum inter-spike-interval, spikes violating a refractory period (`ref_per` - 2 ms by default) are removed from the spike trains. Finally, the duration of the spike trains sets the duration of the recordings (`duration`

Parameter	Value	Type	Explanation
Intracellular simulation settings			
sim_time	1	float	intracellular simulation time in seconds
target_spikes	[3, 50]	list (int)	min-max number of spikes in sim_time
cut_out	[2, 5]	list (float)	pre-post peak cut_out in ms
dt	0.03125	float	time step in ms (default is 32 kHz)
delay	10	float	stimulation delay in ms
weights	[0.25, 1.75]	list (float)	weights to multiply stimulus amplitude if number of spikes is above (0.25) or below (1.25) target spikes
Extracellular simulation settings			
rot	physrot	string	rotation to apply to cell models (norot, xrot, yrot, zrot, physrot, 3drot)
probe	Neuronexus-32	string	extracellular probe (if null an available probe is randomly chosen)
ncontacts	10	int	number of contacts per recording site
overhang	30	float	extension in μm beyond MEA boundaries for neuron locations (if corresponding lim is null)
xlim	[10,80]	list (float)	limits (low, high) for neuron locations in the x-axis in μm
ylim	null	list (float)	limits (low, high) for neuron locations in the y-axis in μm
zlim	null	list (float)	limits (low, high) for neuron locations in the z-axis in μm
min_amp	30	float	minimum template amplitude
n	50	int	number of spikes per cell model
n_overlap_pairs	null	int	number of spatially overlapping templates
drifting	False	bool	if True, drifting templates are simulated
max_drift	100	float	maximum distance from the initial and final cell position
min_drift	30	float	minimum distance from the initial and final cell position
drift_steps	30	int	number of drift steps
drift_xlim	[-10, 10]	list (float)	limits (low, high) for neuron drift locations in the x-axis (depth)
drift_ylim	[-10, 10]	list (float)	limits (low, high) for neuron drift locations in the y-axis (depth)
drift_zlim	[20, 80]	list (float)	limits (low, high) for neuron drift locations in the z-axis (depth)
seed	null	int	random seed for positions and rotations

Table 2: Templates generation parameter list, values, types, and explanations.

parameter), and the user can set the random seed for spike train generation (`seed` parameter in the `spike trains` section). Spike trains are represented as `neo.SpikeTrain` objects [14].

Excitatory and inhibitory cell types

The `cell_types` section of the recordings parameters tells the simulator which cell types are excitatory and which are inhibitory. For all cell models in the Neocortical Microcircuit Portal [39], excitatory cells can be pyramidal cells (PC), star pyramidal cells (SP), and stellate cells (SS). The population of inhibitory cells is more diverse and it includes: axon cells (AC), bipolar cells (BP), bitufted cells (BTC), basket cells (BC), Chandelier cells (ChC), double bouquet cells (DBC), Martinotti cells (MC), and neurogliaform cells (NGC) [30]. This substrings are used to identify the cell models belonging to the excitatory and inhibitory group for the template selection process.

Template selection and pre-processing

After spike trains are generated, templates are selected from the template library and associated with each spike train. The parameters involved in the template selection and pre-processing are in the `templates` section of the recordings parameters.

Templates are chosen based on amplitude, distance, spatial overlap, and cell type. The selection algorithm discards templates with a peak amplitude below and above user-defined threshold (`min_amp` and `max_amp` parameters) and with a distance from already selected neurons below a minimum distance

(`min_dist` parameter). Moreover, the user can select specific boundaries in the x-, y-, and z-direction (`xlim`, `ylim`, and `zlim` parameter). If the boundaries are set to `null` (by default), there is no restriction on the neurons' location. Templates are chosen so that the number of excitatory and inhibitory types matches the spike trains' ones. Finally, the user can select the number of spatially overlapping template pairs in the recordings (`n_overlap_pairs` parameter). Two templates A and B are identified as spatially overlapping if the amplitude of template B on the electrode with largest amplitude for template A is above 90% (`overlap_threshold` parameter) of its maximum amplitude, and viceversa. The template selection seed can be set with the `seed` parameter in the `templates` section of the recordings parameters.

When templates are selected, they are pre-processed before the convolution operation. First, the templates are padded on both sides (by default extending the templates of 3 ms on each side - `pad_len` parameter) in order to ensure a smooth convolution operation. The template baseline is first removed, then the templates are extended in both directions by linearly interpolating their initial and final values to 0. Finally, this linearly extended template is re-interpolated with a cubic spline.

Next, to model the time variation occurring during sampling, for each template `n_jitter` versions are created (10 by default). Jittering is performed by upsampling the templates (8x by default - `upsample` parameter) and shifting them randomly in time within a sampling period, before downsampling them back to the original sampling frequency.

Recordings construction

In the `recordings` section of the recordings parameters, the user can set several parameters for the recordings generation. If not specified, the sampling frequency of the recordings (`fs` parameter) is the same as the generated templates (32 kHz by default), but the user can choose a different sampling rate. In this case the templates are resampled using a polyphase filter. If the `overlap` parameter is set to true, each spike is annotated as `NO` (no overlap), `TO` (temporal overlap), or `STO` (spatio-temporal overlap). If the `extract_waveforms` parameter is set to true, after the recordings generation the waveforms are extracted from the recordings and loaded to the spike train objects.

Overlapping spikes and spatio-temporal synchrony Spatio-temporal overlapping of spikes can make spike sorting very challenging [37, 43]. In order to control how spike sorting is affected by the rate of overlapping spikes, `MEAREC` enables users to modify the spike trains in order to introduce a controlled amount of spatio-temporal overlapping synchrony (Figure 5).

If the synchrony rate is set (`sync_rate` parameter), the spike trains of spatially overlapping templates are modified to reach the desired synchrony rate. If the chosen synchrony rate is lower than the initial rate, spatio-temporal overlapping spikes are randomly removed from the spike trains. Conversely, when the chosen synchrony rate is greater than the initial rate, additional spikes that do not violate the refractory period are randomly added to the corresponding spike trains until the desired rate is reached. The additive spikes are jittered randomly within a user-defined interval (`sync_jitt` - default ± 1 ms).

Modulated convolution Pre-processed templates and spike trains are combined with a customized (modulated) convolution. The convolution step can be performed in parallel on chunks (20 seconds by default - `chunk_conv_duration` parameter). In order to replicate the variability of spikes in experimental data and computational models [1, 18], the convolution between spike trains and templates is modulated, i.e. the template corresponding to each spike can be modified both in amplitude and in shape (Figure 4).

There are three types of amplitude modulation available: 1) *none* (no modulation), 2) *template*, 3) *electrode* modulation (default). On top of amplitude modulation, when modulation is not *none*, shape modulation can be used by setting the `shape_mod` parameter to true.

Amplitude modulation. The amplitude modulation consists of scaling the amplitude of each spike event with a modulation value. When the *template* modulation is selected, the modulation

value is the same for all the electrodes. When the *electrode* modulation is used, each electrode has a slightly different modulation value. For the *template* and *electrode* modulation types, if the `bursting` parameter is set to false, the modulation value is a random value drawn from a normal distribution $\mathcal{N}(1, \text{sdrand}^2)$, (where `sdrand` is 0.05 by default). As the distribution has mean equal 1, the average amplitude of the resulting modulated spikes is the same as the original template. When the `bursting` parameter is set to true, the modulation values are computed to reproduce the amplitude scaling due to bursting behavior (see Figure 4A). The user can choose how many units will be affected by bursting (`n_bursting` parameter). Consecutive spikes occurring within a user-defined bursting period (`max_burst_duration` parameter - default 100 ms) are scaled with a sub-linear function (up to a maximum number of consecutive spikes `n_burst_spikes` - 10 by default). The amplitude scaling for the i -th consecutive spike within a bursting event is computed as:

$$\text{mod}_i = \left(\frac{\text{avg_isi}_{0-i}}{c \cdot \text{max_burst_duration}} \right)^{\text{exp_decay}}$$

where avg_isi_{0-i} is the average inter-spike-interval (ISI) from the first bursting spike to the current spike in the bursting event, c is the number of consecutive spikes encountered up to spike i , `max_burst_duration` is the maximum bursting period (default 100 ms), and `exp_decay` is the exponent (0.1 by default). Additionally, the ISI-dependent modulation value is scaled by a random value drawn from a normal distribution at the template level (*template* modulation) or electrode level (*electrode* modulation).

Shape modulation. When `shape_mod` is set to true, spikes are also modulated in shape. Shape modulation consists of stretching the template depending on its modulation value. The stretch is achieved in the following way: first, the template time axis is centered to the template peak and scaled so that its length is equal to 1 – we will refer to this centered and normalized time axis as x_c ; second, x_c is multiplied by the `bursting_sigmoid` parameter, which controls the amount of stretch – we will refer to this transformed time axis as x_t ; then, a stretch factor s is computed for the entire template (the same factor is computed for all electrodes) as the average modulation value of all electrodes (if *electrode* modulation is used); if the stretch factor is less than 1, x_t is projected on a sigmoid function:

$$x_s = \frac{1}{(1 + \exp^{-(1-s) \cdot x_t})} - 0.5$$

x_s is now a non-linear stretched time axis. The template is interpolated on x_s with a cubic spline and transformed back to the original time axis x_c . The shape modulation with two different `bursting_sigmoid` values is shown in Figure 4B. The amplitude of the shape-modulated template is finally scaled with the modulation value to include the amplitude modulation.

Noise models and post-processing Additive noise is superimposed to the signals after the modulated convolution is finished. There are three types of noise models that can be set using the `noise_mode` parameter: *uncorrelated*, *distance-correlated*, and *far-neurons*. The uncorrelated noise model is an additive Gaussian noise with a user-defined standard deviation (`noise_level` parameter - 10 μV by default). The distance-correlated mode generates a multivariate normal noise with a covariance matrix dependent on the distance between electrodes. The covariance between electrode i and j is defined as $c_{ij} = a_h/2 \cdot d_{ij}$, where d_{ij} is the distance between the electrodes and d_h is the distance at which the covariance is 0.5 (`noise_half_distance` parameter - 30 μm by default). Finally, the *far-neurons* model generates noise as the activity of many neurons (`far_neurons_n` parameter - 300 by default) with small amplitudes (below `far_neurons_max_amp` - 10 μV by default). The population of distant neurons has an excitatory/inhibitory ratio of `far_neurons_exc_inh_ratio` (default 0.8). A random noise floor with a standard deviation of `far_neurons_noise_floor` (default 0.5) times the standard deviation of the distant neurons' spiking activity is added, in agreement with experimental data [6].

Uncorrelated and distance-correlated noise types can also be modulated in frequency to match the spectrum observed in experimental data [6, 17]. Extracellular spiking activity exhibit a peak in

Parameter	Value	Type	Explanation
Spike trains			
n_exc	7	int	number of excitatory cells
n_inh	3	int	number of inhibitory cells
f_exc	5	float	average firing rate of excitatory cells in Hz
f_inh	15	float	average firing rate of inhibitory cells in Hz
st_exc	1	float	firing rate standard deviation of excitatory cells in Hz
st_inh	3	float	firing rate standard deviation of inhibitory cells in Hz
min_rate	0.5	float	minimum firing rate in Hz
ref_per	2	float	refractory period in ms
process	poisson	string	process for spike train simulation (poisson-gamma)
gamma_shape	2	float	gamma shape (for gamma process)
duration	10	float	duration in seconds
seed	null	int	random seed for spiketrain generation
Cell types			
excitatory	['PC', 'SS', 'SP']	list (string)	Excitatory cell types
inhibitory	['AC', 'BP', 'BC', 'BTC', 'ChC', 'DBC', 'MC', 'NGC']	list (string)	Inhibitory cell types
Templates			
min_dist	25	float	minimum distance between neurons
min_amp	50	float	minimum spike amplitude in μV
max_amp	500	float	maximum spike amplitude in μV
xlim	null	list (float)	limits for neuron x in μm (min, max)
ylim	null	list (float)	limits for neuron y in μm (min, max)
zlim	null	list (float)	limits for neuron z in μm (min, max)
overlap_threshold	0.9	float	threshold to consider two templates spatially overlapping
n_jitters	10	int	number of temporal jittered copies for each template
upsample	8	int	upsampling factor to extract jittered copies
pad_len	[3, 3]	list (float)	padding of templates in ms
seed	null	int	random seed to select templates

Table 3: Recordings generation parameter list, values, types, and explanations.

frequency at around 300 Hz, a $1/f$ spectrum, and a random noise floor. Noise can be *colored* (when the `noise_color` parameter is true) with a second order infinite impulse response (IIR) peak filter and an additional gaussian noise floor. The frequency peak, quality factor, and weight of the random noise floor can be set with the `color_peak`, `color_q`, and `color_noise_floor` parameters. Note that with distance-correlated noise the correlation is slightly reduced by the color filter, as a random noise floor is added.

Optionally, the signals can be filtered (by setting the `filter` to true) with an high-pass or band-pass Butterworth filter of order `filter_order` (3 by default) and cutoff frequencies of `filter_cutoff` ([300, 6000] Hz by default).

Drifting recordings When the `drifting` parameter is set to true, drifting recordings are generated. The template library must have been generated with the `drifting` mode as well. The user can decide the number of drifting units (`n_drifting` parameter). If `n_drifting` is null, all units will be drifting.

The generation of drifting recordings is only different in the template selection and modulated convolution steps. In the template selection, in addition to the selection rules based on template amplitude, inter-neuron distance, and spatial overlap, templates are selected if the angle between the drifting direction (computed as the vector connecting the final and initial position) and a user-defined preferred direction (`preferred_dir` parameter - [0, 0, 1] by default) is within an angle tolerance (`angle_tol` parameter - 15° by default).

In the modulated convolution, the correct template among the drifting templates for each spike occurrence is selected based on the current drifting position computed as the initial position plus drifting velocity times simulation time. The drifting velocity can be modified by the user (`drift_velocity` parameter - $5 \mu\text{m}/\text{min}$ by default) and the user can decide to start the drift after `t_start_drift` seconds.

Parameter	Value	Type	Explanation
Recordings			
fs	null	int	sampling frequency in Hz (if null it is computed from the templates)
overlap	False	bool	if True, temporal and spatial overlap are computed for each spike (it may be time consuming)
extract_waveforms	False	bool	if True, waveforms are extracted from recordings
sync_rate	null	float	synchrony rate ([0-1]) for spike trains of spatially overlapping templates
sync_jitt	1	float	jitter in ms for added synchronous spikes
modulation	electrode	string	type of modulation [none template electrode] none - no modulation template - each spike instance is modulated with the same value on each electrode electrode - each electrode is modulated separately
sdrand	0.05	float	standard deviation of Gaussian modulation
bursting	False	bool	if True, spikes are modulated in amplitude depending on the ISI
exp_decay	0.1	float	(bursting) experimental decay in amplitude between consecutive spikes
n_burst_spikes	10	int	(bursting) max number of 'bursting' consecutive spikes
max_burst_duration	100	float	(bursting) duration in ms of maximum burst modulation
shape_mod	False	bool	if True waveforms are stretched in shape with a sigmoid transform depending on their modulation value
bursting_sigmoid	30	float	sigmoid range used to stretch the template
n_bursting	null	int	number of bursting units. If null all units are bursting
chunk_conv_duration	20	float	chunk duration for convolution (if running into MemoryError)
noise_level	10	float	noise standard deviation in uV
noise_mode	uncorrelated	string	[uncorrelated distance-correlated far-neurons]
noise_color	False	bool	if True noise is colored resembling experimental noise
noise_half_distance	30	float	(distance-correlated) distance between electrodes in μm for which correlation is 0.5
far_neurons_n	300	int	(far-neurons) number of far neurons to be simulated
far_neurons_max_amp	10	float	(far-neurons) maximum amplitude of far neurons
far_neurons_noise_floor	0.5	float	(far-neurons) percent of additive random noise
far_neurons_exc_inh_ratio	0.8	float	(far-neurons) excitatory / inhibitory noisy neurons ratio [0-1]
color_peak	300	float	(color) peak / cutoff frequency of resonating filter in Hz
color_q	2	int	(color) quality factor of resonating filter
color_noise_floor	0.5	float	(color) percent of additive random noise
chunk_noise_duration	0	float	chunk duration for noise addition
seed	null	int	random seed for noise generation
filter	True	bool	if True recordings are filtered
filter_cutoff	[300, 6000]	float/list	filter cutoff frequencies in Hz
filter_order	3	int	filter order
chunk_filter_duration	0	float	chunk duration for filtering
drifting	False	bool	if True drifting recordings are simulated
n_drifting	null	int	number of drifting units. If null all units are drifting
preferred_dir	[0, 0, 1]	list	preferred drifting direction ([0, 0, 1] is positive z, direction)
angle_tol	15	float	tolerance for direction in degrees
drift_velocity	5	float	drift velocity in $\mu\text{m}/\text{min}$
t_start_drift	0	float	time in seconds after which drifting starts

Table 4: (Continued) Recordings generation parameter list, values, types, and explanations.

Statistical analysis

No statistical analysis is used in this contribution.

Code availability

The presented software package is available at <https://github.com/alejoe91/MEAREC> and <https://github.com/alejoe91/MEAutility> (used for probe handling). The packages are also available on pypi: <https://pypi.org/project/MEAREC/> - <https://pypi.org/project/MEAutility/>.

Data availability

All the datasets generated for the paper and used to make figures are available on Zenodo at <https://doi.org/10.5281/zenodo.3247736>.

Acknowledgments

A.P.B. and G.T.E. are part of the Simula-UCSD-University of Oslo Research and PhD training (SU-URPh) program, an international collaboration in computational biology and medicine funded by the Norwegian Ministry of Education and Research. Moreover, we would like to thank Kristian Lensjø, Jennifer Hazen, and Mikkel Lepperød for their valuable feedback on the article.

References

- [1] B. D. Allen, C. Moore-Kochlacs, J. G. Bernstein, J. Kinney, J. Scholvin, L. Seoane, C. Chronopoulos, C. Lamantia, S. B. Kodandaramaiah, M. Tegmark, et al. Automated in vivo patch clamp evaluation of extracellular multielectrode array spike recording capability. *Journal of neurophysiology*, 2018.
- [2] L. Berdondini, K. Imfeld, A. Maccione, M. Tedesco, S. Neukom, M. Koudelka-Hep, and S. Martinoia. Active pixel sensor array for high spatio-temporal resolution electrophysiological recordings from single cell to large scale neuronal networks. *Lab on a Chip*, 9(18):2644–2651, 2009.
- [3] A. P. Buccino, E. Hagen, G. T. Einevoll, P. D. Häfliger, and G. Cauwenbergh. Independent component analysis for fully automated multi-electrode array spike sorting. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2627–2630. IEEE, 2018.
- [4] A. P. Buccino, M. Kordovan, T. V. Ness, B. Merkt, P. D. Häfliger, M. Fyhn, G. Cauwenberghs, S. Rotter, and G. T. Einevoll. Combining biophysical modeling and deep learning for multi-electrode array neuron localization and classification. *Journal of neurophysiology*, 2018.
- [5] A. P. Buccino, M. Kuchta, K. H. Jæger, T. V. Ness, P. Berthet, K. A. Mardal, G. Cauwenberghs, and A. Tveito. How does the presence of neural probes affect extracellular potentials? *Journal of neural engineering*, 2019.
- [6] L. A. Camuñas-Mesa and R. Q. Quiroga. A detailed and fast model of extracellular recordings. *Neural computation*, 25(5):1191–1212, 2013.
- [7] N. T. Carnevale and M. L. Hines. *The NEURON book*. Cambridge University Press, 2006.
- [8] J. E. Chung, J. F. Magland, A. H. Barnett, et al. A fully automated approach to spike sorting. *Neuron*, 95(6):1381–1394, 2017.
- [9] M. Diesmann and M.-O. Gewaltig. Nest: An environment for neural systems simulations. *Forschung und wissenschaftliches Rechnen, Beiträge zum Heinz-Billing-Preis*, 58:43–70, 2001.
- [10] R. Diggelmann, M. Fiscella, A. Hierlemann, and F. Franke. Automatic spike sorting for high-density microelectrode arrays. *Journal of neurophysiology*, 120(6):3155–3171, 2018.
- [11] G. T. Einevoll, F. Franke, E. Hagen, et al. Towards reliable spike-train recordings from thousands of neurons with multielectrodes. *Current opinion in neurobiology*, 22(1):11–17, 2012.

- [12] F. Franke, M. Natora, P. Meier, E. Hagen, K. H. Pettersen, H. Linden, G. T. Einevoll, and K. Obermayer. An automated online positioning system and simulation environment for multi-electrodes in extracellular recordings. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 593–597. IEEE, 2010.
- [13] U. Frey, U. Egert, F. Heer, S. Hafizovic, and A. Hierlemann. Microelectronic system for high-resolution mapping of extracellular electric fields applied to brain slices. *Biosensors and Bioelectronics*, 24(7):2191–2198, 2009.
- [14] S. Garcia, D. Guarino, F. Jaillet, T. R. Jennings, R. Pröpper, P. L. Rautenberg, C. Rodgers, A. Sobolev, T. Wachtler, P. Yger, et al. Neo: an object model for handling electrophysiology data in multiple formats. *Frontiers in neuroinformatics*, 8:10, 2014.
- [15] T. Goto, R. Hatanaka, T. Ogawa, A. Sumiyoshi, J. Riera, and R. Kawashima. An evaluation of the conductivity profile in the somatosensory barrel cortex of wistar rats. *J Neurophysiol*, 104(6):3388–3412, 2010.
- [16] N. W. Gouwens et al. Systematic generation of biophysically detailed models for diverse cortical neuron types. *Nature communications*, 9(1):710, 2018.
- [17] E. Hagen, S. Næss, T. V. Ness, and G. T. Einevoll. Multimodal modeling of neural network activity: Computing lfp, ecog, eeg, and meg signals with lfpy 2.0. *Frontiers in neuroinformatics*, 12, 2018.
- [18] E. Hagen, T. V. Ness, A. Khosrowshahi, C. Sørensen, M. Fyhn, T. Hafting, F. Franke, and G. T. Einevoll. Visapy: a python tool for biophysics-based generation of virtual spiking activity for evaluation of spike-sorting algorithms. *Journal of neuroscience methods*, 245:182–204, 2015.
- [19] K. D. Harris, D. A. Henze, J. Csicsvari, H. Hirase, and G. Buzsaki. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *Journal of neurophysiology*, 84(1):401–414, 2000.
- [20] E. Hay, S. Hill, F. Schürmann, H. Markram, and I. Segev. Models of neocortical layer 5b pyramidal cells capturing a wide range of dendritic and perisomatic active properties. *PLoS computational biology*, 7(7):e1002107, 2011.
- [21] D. A. Henze, Z. Borhegyi, J. Csicsvari, A. Mamiya, K. D. Harris, and G. Buzsaki. Intracellular features predicted by extracellular recordings in the hippocampus in vivo. *Journal of neurophysiology*, 84(1):390–400, 2000.
- [22] G. Hilgen, M. Sorbaro, S. Pirmoradian, J.-O. Muthmann, I. E. Kepiro, S. Ullo, C. J. Ramirez, A. P. Encinas, A. Maccione, L. Berdondini, et al. Unsupervised spike sorting for large-scale, high-density multielectrode arrays. *Cell reports*, 18(10):2521–2532, 2017.
- [23] G. R. Holt and C. Koch. Electrical interactions via the extracellular potential near cell bodies. *J Comput Neurosci*, 6(2):169–184, 1999.
- [24] D. Jäckel, U. Frey, M. Fiscella, et al. Applicability of independent component analysis on high-density microelectrode array recordings. *Journal of neurophysiology*, 108(1):334–348, 2012.
- [25] J. J. Jun, C. Mitelut, C. Lai, S. Gratiy, C. Anastassiou, and T. D. Harris. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *bioRxiv*, page 101030, 2017.
- [26] 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*, 551(7679):232, 2017.

- [27] J. H. Lee, D. E. Carlson, H. S. Razaghi, W. Yao, G. A. Goetz, E. Hagen, E. Batty, E. Chichilnisky, G. T. Einevoll, and L. Paninski. Yass: yet another spike sorter. In *Advances in Neural Information Processing Systems*, pages 4002–4012, 2017.
- [28] B. Lefebvre, P. Yger, and O. Marre. Recent progress in multi-electrode spike sorting methods. *Journal of Physiology-Paris*, 110(4):327–335, 2016.
- [29] H. Lindén, E. Hagen, S. Leski, et al. LFPy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons. *Frontiers in Neuroinformatics*, 7:41, 2014.
- [30] H. Markram, E. Muller, S. Ramaswamy, et al. Reconstruction and simulation of neocortical microcircuitry. *Cell*, 163(2):456–492, 2015.
- [31] A. Marques-Smith, J. P. Neto, G. Lopes, J. Nogueira, L. Calcaterra, J. Frazão, D. Kim, M. G. Phillips, G. Dimitriadis, and A. Kampff. Recording from the same neuron with high-density cmos probes and patch-clamp: a ground-truth dataset and an experiment in collaboration. *bioRxiv*, page 370080, 2018.
- [32] R. Migliore, C. A. Lupascu, L. L. Bologna, A. Romani, J.-D. Courcol, S. Antonel, W. A. Van Geit, A. M. Thomson, A. Mercer, S. Lange, et al. The physiological variability of channel density in hippocampal cal pyramidal cells and interneurons explored using a unified data-driven modeling workflow. *PLoS computational biology*, 14(9):e1006423, 2018.
- [33] S. L. Mondragón-González and E. Burguière. Bio-inspired benchmark generator for extracellular multi-unit recordings. *Scientific reports*, 7:43253, 2017.
- [34] T. V. Ness, C. Chintaluri, J. Potworowski, S. Łęski, H. Głąbska, D. K. Wójcik, and G. T. Einevoll. Modelling and analysis of electrical potentials recorded in microelectrode arrays (meas). *Neuroinformatics*, 13(4):403–426, 2015.
- [35] J. P. Neto, G. Lopes, J. Frazão, et al. Validating silicon polytrodes with paired juxtacellular recordings: method and dataset. *Journal of Neurophysiology*, 116(2):892–903, 2016.
- [36] P. L. Nunez and R. Srinivasan. *Electric fields of the brain: the neurophysics of EEG*. Oxford University Press, USA, 2006.
- [37] M. Pachitariu, N. A. Steinmetz, S. N. Kadir, et al. Fast and accurate spike sorting of high-channel count probes with kilosort. In *Advances in Neural Information Processing Systems*, pages 4448–4456, 2016.
- [38] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004.
- [39] S. Ramaswamy, J. Courcol, M. Abdellah, et al. The neocortical microcircuit collaboration portal: a resource for rat somatosensory cortex. *Front Neural Circuits*, 9, 2015.
- [40] H. G. Rey, C. Pedreira, and R. Q. Quiroga. Past, present and future of spike sorting techniques. *Brain research bulletin*, 119:106–117, 2015.
- [41] 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*, 19(4):634, 2016.
- [42] N. A. Steinmetz, C. Koch, K. D. Harris, and M. Carandini. Challenges and opportunities for large-scale electrophysiology with neuropixels probes. *Current opinion in neurobiology*, 50:92–100, 2018.

- [43] 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*, 7:e34518, 2018.

Appendix A - command line interface (CLI)

MEArc implements a command line interface (CLI) to make templates and recordings generation easy to use and to allow for scripting. In order to discover the available commands, the user can use the `--help` option:

```
>> mearec --help
```

```
Usage: mearec [OPTIONS] COMMAND [ARGS]...
```

```
MEArc: Fast and customizable simulation of extracellular recordings on
Multi-Electrode-Arrays
```

Options:

```
--help Show this message and exit.
```

Commands:

```
available-probes      Print available probes.
default-config        Print default configurations.
gen-recordings        Generates RECORDINGS from TEMPLATES.
gen-templates         Generates TEMPLATES with biophysical simulation.
set-cell-models-folder Set default cell_models folder.
set-recordings-folder Set default recordings output folder.
set-recordings-params Set default recordings parameter file.
set-templates-folder  Set default templates output folder.
set-templates-params  Set default templates parameter file.
```

Each available command can be inspected using the `--help` option:

```
>> mearec command --help
```

At installation, MEArc creates a configuration folder (`.config/mearec`) in which global settings are stored. The default paths to cell models folder, templates and recordings output folders and parameters can be set using the `set-cell-models-folder`, `set-` commands. By default, these files and folders are located in the configuration folder.

```
>> mearec default-config
```

```
{'cell_models_folder': path-to-cell_models,
 'recordings_folder': path-to-recordings-folder,
 'recordings_params': path-to-recordings-params.yaml,
 'templates_folder': path-to-templates-folder,
 'templates_params': path-to-templates-params.yaml}
```

A list of available probes can be found by running the `available-probes` command:

```
>> mearec available-probes
```

```
Neuronexus-32 ----- Neuronexus A1x32-Poly3-5mm-25s-177-CM32 probe.
                        32 circular contacts in 3 staggered columns.
```

```
Neuropixels-128 ----- Neuropixels probe.
                        128 square contacts in 4 staggered columns.
```

Neuropixels-24 ----- Neuropixels probe.
24 square contacts in 4 staggered columns.

Neuropixels-384 ----- Neuropixels probe.
384 square contacts in 4 staggered columns.

Neuropixels-64 ----- Neuropixels probe.
64 square contacts in 4 staggered columns.

Neuroseeker-128 ----- Neuroseeker probe.
128 square contacts in 4 columns.

SqMEA-10-15 ----- Square MEA. 100 square contacts in 10x10 matrix configuration
with 15um pitch.

SqMEA-15-10 ----- Square MEA. 225 square contacts in 15x15 matrix configuration
with 10um pitch.

SqMEA-5-30 ----- Square MEA. 25 square contacts in 5x5 matrix configuration
with 30um pitch.

SqMEA-6-25 ----- Square MEA. 36 square contacts in 6x6 matrix configuration
with 25um pitch.

SqMEA-7-20 ----- Square MEA. 49 square contacts in 7x7 matrix configuration
with 20um pitch.

four-tetrodes ----- 4 tetrodes on a shank with 100um inter-tetrode distance.

tetrode ----- Microwire tetrode with 4 circular contacts.

tetrode-mea-d ----- Silicon tetrode with 4 square contacts
in diamond configuration.

tetrode-mea-l ----- Silicon tetrode with 4 square contacts
in linear configuration.

tetrode-mea-s ----- Silicon tetrode with 4 square contacts
in square configuration.

Finally, examples of templates and recordings generation commands can be found in the Results section.

Appendix B - Python API example

MEarec implements a Python API for simulating both templates and recordings. The Python API is recommended for generating recordings for testbench purposes. For example, the following script is used to generate four recordings with varying noise level, shown in Figure 8A:

```
import MEarec as mr
import matplotlib.pyplot as plt
import numpy as np
```

```
# load template file as a TemplateGenerator object
template_file = 'path-to-template-file.h5'
tempgen = mr.load_templates(template_file)

# load default recording parameters
params = mr.get_default_recordings_params()

# set seeds for spike trains and template selection
params['spiketrains']['seed'] = 0
params['templates']['seed'] = 1

# list of noise levels
noise_levels = [30, 20, 10, 5]

# generate recordings
recordings_noise = []
for n in noise_levels:
    print('Noise level:', n)
    params['recordings']['noise_level'] = n
    params['recordings']['seed'] = np.random.randint(1000)
    recgen = mr.gen_recordings(tempgen=tempgen, params=params)
    recordings_noise.append(recgen)

# plot recordings on the same axis
colors = plt.rcParams['axes.prop_cycle'].by_key()['color']
fig = plt.figure()
ax = fig.add_subplot(111)
for i, rec in enumerate(recordings_noise):
    ax = mr.plot_recordings(rec, colors=colors[i], ax=ax)
```

For additional examples, please refer to the Github page <https://github.com/alejoe91/MEarec>.

Paper II

SpikeInterface, a unified framework for spike sorting



SpikeInterface, a unified framework for spike sorting

Alessio P. Buccino^{¶1}, Cole L. Hurwitz^{¶*2}, Jeremy Magland³, Samuel Garcia⁴,
Joshua H. Siegle⁵, Roger Hurwitz⁶, and Matthias H. Hennig²

¹Centre for Integrative Neuroplasticity (CINPLA), University of Oslo, Oslo, Norway

²School of Informatics, University of Edinburgh, United Kingdom

³Flatiron Institute, New York City, NY, United States

⁴Centre de Recherche en Neurosciences de Lyon, CNRS, Lyon, France

⁵Allen Institute for Brain Science, Seattle, WA, United States

⁶Independent Researcher, Portland, Oregon, USA

[¶] These authors contributed equally to this work.

Abstract

Given the importance of understanding single-neuron activity, much development has been directed towards improving the performance and automation of spike sorting. These developments, however, introduce new challenges, such as file format incompatibility and reduced interoperability, that hinder benchmarking and preclude reproducible analysis. To address these limitations, we developed SpikeInterface, a Python framework designed to unify preexisting spike sorting technologies into a single codebase and to standardize extracellular data file operations. With a few lines of code and regardless of the underlying data format, researchers can: run, compare, and benchmark most modern spike sorting algorithms; pre-process, post-process, and visualize extracellular datasets; validate, curate, and export sorting outputs; and more. In this paper, we provide an overview of SpikeInterface and, with applications to both real and simulated extracellular datasets, demonstrate how it can improve the accessibility, reliability, and reproducibility of spike sorting in preparation for the widespread use of large-scale electrophysiology.

1 Introduction

Extracellular recording is an indispensable tool in neuroscience for probing how single neurons (and populations of neurons) encode and transmit information. When analyzing extracellular recordings, most researchers are interested in the spiking activity of individual neurons, which must be extracted from the raw voltage traces through a process called *spike sorting*. Many laboratories perform spike sorting using fully manual techniques (e.g. XClust [49], MClust [65], SimpleClust [71], Plexon Offline Sorter [7]), but such approaches are nearly impossible to standardize due to inherent operator bias [73]. To alleviate this issue, spike sorting has seen decades of algorithmic and software improvements to increase both the accuracy and automation of the process [58]. This progress has accelerated in the past few years as high-density devices [24, 11, 25, 9, 51, 75, 42, 36, 21, 8], capable of recording

*cole.hurwitz@ed.ac.uk

from hundreds to thousands of neurons simultaneously, have made manual intervention impractical, increasing the demand for both accurate and scalable spike sorting algorithms [61, 55, 40, 19, 74, 33, 35].

Along with these exciting advances, however, come unintended complications. Over the years, dozens of new file formats have been introduced, a multitude of data processing and evaluation methods have been developed, and an enormous amount of software, written in a variety of different programming languages, has been made available for general-use. In an ideal world, standards and best-practices for spike sorting would naturally arise from using these tools in experimental and clinical settings. However, due to the high complexity of spike sorting and a lack of interoperability among its corresponding technologies, no clear standards exist for how it should be performed or evaluated [58, 10, 17]. Furthermore, the importance of publication for career development rewards proof-of-concept breakthroughs at the expense of long-term maintenance of software tools. The lack of best practices, the scarcity of well-maintained code, the dearth of rigorous benchmarking, and the high barrier to entry of using a new spike sorter or file format all contribute to many laboratories either continuing to use manual spike sorting techniques or arbitrarily settling on one automated algorithm and its corresponding suite of analysis tools and supported file formats. Reproducibility, data provenance, and data sharing become increasingly difficult as different laboratories adopt different spike sorting solutions [20].

Recent work to alleviate these issues has focused on tackling file format incompatibilities in electrophysiology. This has led to progress in creating a common description of neurophysiological data both with new software tools and file formats [72, 27, 70, 69, 67, 68]. Despite progress in defining a common standard, many different file formats are still widely used in electrophysiology, with more being developed continuously [22]. Along with attempts to define common standards, much work has been put into creating open-source analysis tools that make extracellular analysis and spike sorting more accessible [23, 15, 32, 26, 30, 12, 41, 13, 53, 39, 44, 14, 57, 76, 52]. These software frameworks, while valuable tools in electrophysiology, implement spike sorting as a small step in a larger extracellular analysis pipeline, leading to undersupported, incomplete, and outdated spike sorting functionality. Given the ever-increasing amount of spike sorting software and file formats, there is an urgent need for an open-source analysis framework that is up-to-date with modern spike sorting methods, is agnostic to the underlying file formats of the extracellular datasets, and is extendable to new technologies.

In this paper, we introduce SpikeInterface, the first open-source, Python¹ framework designed exclusively to encapsulate all steps in the spike sorting pipeline. SpikeInterface overcomes both file format incompatibilities and software interoperability in spike sorting with a intuitive application program interface (API), and with a unified, extendable codebase of modern analysis tools. Using SpikeInterface, researchers can: run, compare, and benchmark most modern spike sorters; pre-process, post-process, and visualize extracellular datasets; validate, curate, and export sorting outputs; and more. This can all be done regardless of the underlying data format as SpikeInterface addresses file format compatibility issues within spike sorting pipelines without creating yet another file format. For developers, SpikeInterface enables easy integration and evaluation of their spike sorting software, allowing for accelerated development and a constantly expanding codebase. We also introduce a graphical user interface (GUI) based on SpikeInterface that allows for straightforward construction of spike sorting pipelines without any Python programming knowledge. To illustrate the advantages of a unified framework for spike sorting, we utilize SpikeInterface's Python API and GUI to build a complex spike sorting pipeline. We also use SpikeInterface to run, compare, and evaluate six modern spike sorters on both a sample Neuropixels recording and a simulated, ground-truth recording. With these three use cases, we demonstrate how SpikeInterface can help alleviate long-standing challenges in spike sorting. To clarify, the main contribution of this work is a novel framework for running and comparing spike sorting pipelines, not an exhaustive comparison of current spike sorting algorithms. All code for SpikeInterface is open-source and can be found on GitHub².

¹We utilize Python as it is open-source, free, and increasingly popular in the neuroscience community [50, 29].

²<https://github.com/SpikeInterface>

2 Design Principles

SpikeInterface is designed to efficiently encapsulate all aspects of a spike sorting pipeline. To this end, we apply a set of design principles. These principles inform both the project's overall structure and the implementation of specific functionalities.

Focused. SpikeInterface was designed to unify all operations related to the spike sorting of extracellular recordings. Therefore, we did not attempt to incorporate any metadata from the underlying experiments (stimulus information, behavioral readouts, etc.) as such a task is beyond the scope of our problem statement. Also, we did not incorporate any analysis steps unrelated to spike sorting and did not attempt to handle any other electrophysiological data such as intracellular or electroencephalographic recordings. Keeping this narrow focus makes SpikeInterface light-weight, scalable, easy to use, and extendable.

Comprehensive. To do justice to years of research and development into spike sorting, we incorporated many existing extracellular file formats and the most current, semi-automatic spike sorters into SpikeInterface. We also incorporated common pre- and post-processing methods, quality metrics, evaluation and curation tools, and data visualization widgets. The broad range of methods and technologies that are supported makes SpikeInterface the most expansive spike sorting toolbox currently available by a wide margin. For an overview of the current file formats and spike sorters that are supported in SpikeInterface, see Table 1.

Modularized. The SpikeInterface codebase is separated into multiple, distinct modules which encapsulate individual processing steps shared across *all* spike sorting pipelines. In Section 3, we explain this modularized and conceptualized structure and show how it can be utilized to build robust and flexible spike sorting workflows. This design also makes SpikeInterface easily extendable, allowing new formats, methods, and tools to be added rapidly. We encourage the community to contribute to its further development.

Efficient. As we aim to support the analysis of large-scale extracellular recordings, much consideration has been put into making SpikeInterface as memory- and computation-efficient as possible. For instance, file input/output (I/O) operations are generally memory-mapped (only the data needed for a computation are loaded into memory) and processing is parallelized where feasible. We also made sure that running a spike sorter *in* our framework adds little to no extra computational cost in comparison to running the same spike sorter *outside* of our framework.

Reproducible. Although spike sorting is an essential step in extracellular analysis, it is often difficult to reproduce due to the variety of complex (and sometimes stochastic) processing steps performed on the underlying dataset. We designed SpikeInterface to make spike sorting and all associated computation as reproducible as possible with a unified codebase, fixed random seeds, a standard API, and a careful version control system. Laboratories using SpikeInterface can share and process extracellular datasets with a guarantee that they get identical results for the same functions.

3 Overview of SpikeInterface

SpikeInterface consists of five main Python packages designed to handle different aspects of the spike sorting pipeline: (i) `spikeextractors`, for extracellular recording, sorting output, and probe file I/O; (ii) `spiketoolkit` for low level processing such as pre-processing, post-processing, validation, curation; (iii) `spikesorters` for spike sorting algorithms and job launching functionality; (v) `spikecomparison`

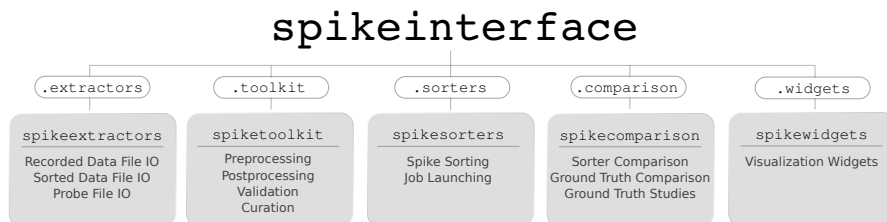


Figure 1: Overview of SpikeInterface’s Python packages, their different functionalities, and how they can be accessed by our meta-package, `spikeinterface`.

for sorter comparison, ground-truth comparison, and ground-truth studies; and (iv) `spikewidgets`, for data visualization.

These five packages can be used individually or installed and used together with the `spikeinterface` metapackage, which contains stable versions of all five packages as internal modules (see Figure 1). With these five packages (or our meta-package), users can build, run, and evaluate full spike sorting pipelines in a reproducible and standardized way. In the following subsections, we present an overview of, and a code snippet for, each main package.

3.1 SpikeExtractors

The `spikeextractors` package³ is designed to solve issues of file format incompatibility within spike sorting without creating yet another file format. This goal is met by standardizing data retrieval rather than data storage. By standardizing access to data from *all* spike sorting related files, whether extracellular recordings or sorting outputs, we eliminate the need for shared file formats and can allow for new tools and packages to directly interface with our framework instead. We distinguish between three data files in spike sorting: the extracellular recording, the sorting output, and the probe configuration. Being able to efficiently and easily interface with these three data file types is essential for running and evaluating any spike sorting pipeline. To this end, we developed two Python objects that can provide all the functionality required to access these data: the `RecordingExtractor` and the `SortingExtractor`.

The `RecordingExtractor` directly interfaces with an extracellular recording and can query it for four primary pieces of information: (i) the extracellular recorded traces; (ii) the sampling frequency; (iii) the number of frames, or duration, of the recording; and (iv) the channel indices of the recording electrodes. These data are shared across all extracellular recordings, allowing standardized access. In addition, a `RecordingExtractor` may store extra information about the recording device as "channel properties" which are key–value pairs. This includes properties such as "location", "group", and "gain" which are either provided by certain extracellular file formats, loaded manually by the user, or loaded automatically with our built-in probe file (.prb or .csv) reader. Taken together, the `RecordingExtractor` is an object representation of an extracellular recording and the associated probe configuration.

The `SortingExtractor` directly interfaces with a sorting output and can query it for two primary pieces of information: (i) the unit indices; and (ii) the spike train of each unit. Again, these data are shared across all sorting outputs. A `SortingExtractor` may also store extra information about the sorting

³<https://github.com/SpikeInterface/spikeextractors>

output as either "unit properties" or "unit spike features", key-value pairs which store information about the individual units or the individual spikes of each unit, respectively. This extra information is either loaded from the sorting output, loaded manually by the user, or loaded automatically with built-in post-processing tools (discussed in Section 3.2). In sum, the `SortingExtractor` is an object representation of a sorting output along with any associated post-processing.

Critically, `Extractors` only query the underlying datasets for information as it is required, reducing their memory footprint and allowing their use for long, large-scale recordings. All extracted data is converted into either native Python data structures or into numpy arrays for immediate use in Python.

The following code snippet illustrates how to return a 2D numpy array of raw data (channels×time) from an extracellular recording and a list of unit indices from a sorting output:

```
import spikeinterface.extractors as se
recording = se.MyFormatRecordingExtractor(file_path='myrecording')
sorting = se.MyFormatSortingExtractor(file_path='mysorting')
traces = recording.get_traces()
unit_ids = sorting.get_unit_ids()
```

Along with using `Extractors` for single files, it is possible to access data from multiple files or portions of files with the `MultiExtractors` and `SubExtractors`, respectively. Both have identical functionality to normal `Extractors` and can be used and treated in the same ways, simplifying, for instance, the combined analysis of a recording split into multiple files.

As of this moment, we support 15 extracellular recording formats, 11 sorting output formats, and 2 probe file formats. Although this covers many popular formats in extracellular analysis, we expect this number to grow with future versions as supporting a new format is as simple as making a new `Extractor` subclass for it. Also, we plan to integrate Neo's [27] I/O system into `spikeextractors` which would allow our framework to support many more open-source and proprietary file formats in extracellular electrophysiology without changing any functionality.

3.2 SpikeToolkit

The `spiketoolkit` package⁴ is designed for efficient pre-processing, post-processing, validation, and curation of extracellular datasets and sorting outputs. It contains four modules that encapsulate each of these functionalities: `preprocessing`, `postprocessing`, `validation`, and `curation`.

3.2.1 Pre-processing

The `preprocessing` module provides functions to process raw extracellular recordings before spike sorting. To pre-process an extracellular recording, the user passes a `RecordingExtractor` to a pre-processing function which returns a new "processed" `RecordingExtractor`. This new `RecordingExtractor`, which can be used in exactly the same way as the original extractor, implements the processing in a *lazy* fashion so that the actual computation is performed only when data is requested. As all pre-processing functions take in and return a `RecordingExtractor`, they can be naturally chained together to perform multiple pre-processing steps on the same recording.

⁴<https://github.com/SpikeInterface/spiketoolkit>

Pre-processing functions range from commonly used operations, such as bandpass filtering, notch filtering, re-referencing signals, and removing channels, to more advanced procedures such as clipping traces depending on the amplitude, or removing artifacts arising, for example, from electrical stimulation.

The following code snippet illustrates how to chain together a few common pre-processing functions to process a raw extracellular recording:

```
import spikeinterface.spiketoolkit as st
recording = st.preprocessing.bandpass_filter(recording, freq_min=300, freq_max=6000)
recording_1 = st.preprocessing.remove_bad_channels(recording, bad_channels=[5])
recording_2 = st.preprocessing.common_reference(recording_1, reference='median')
```

In this code snippet, `recording_2` is still a `RecordingExtractor`. However, the extracted data when using `recording_2` will have channel 5 removed and the underlying extracellular traces bandpass filtered and common median referenced to remove noise.

Overall, the pre-processing functions in `SpikeInterface` represent a wide range of tools that are used in modern spike sorting applications and, since implementing a new pre-processor is straightforward, we expect more to be added in future versions.

3.2.2 Post-processing

The `postprocessing` module provides functions to compute and store information about an extracellular recording given an associated sorting output. As such, post-processing functions are designed to take in both a `RecordingExtractor` and a `SortingExtractor`, using them in conjunction to compute the desired information. These functions include, but are not limited to: extracting unit waveforms and templates, as well as computing principle component analysis projections.

One essential feature of the `postprocessing` module is that it provides the functionality to export a `RecordingExtractor/SortingExtractor` pair into the `Phy` format for manual curation later. `Phy` [59, 61] is a popular manual curation GUI that allows users to visualize a sorting output with several views and to curate the results by manually merging or splitting clusters. `Phy` is already supported by several spike sorters (including `klusta`, `Kilosort`, `Kilosort2`, and `SpyKING-CIRCUS`) so our exporter function extends `Phy`'s functionality to all `SpikeInterface`-supported spike sorters. After manual curation is performed in `Phy`, the curated data can be re-imported into `SpikeInterface` using the `PhySortingExtractor` for further analysis.

The following code snippet illustrates how to retrieve waveforms for each sorted unit, compute principal component analysis (PCA) features for each spike, and export to `Phy` using `SpikeInterface`:

```
import spikeinterface.toolkit as st
waveforms = st.postprocessing.get_unit_waveforms(recording, sorting)
pca_scores = st.postprocessing.compute_unit_pca_scores(recording, sorting, n_comp=3)
st.postprocessing.export_to_phy(recording, sorting_MS4, output_folder='phy_folder')
```

3.2.3 Validation

The `validation` module allows users to automatically evaluate spike sorting results in the absence of ground truth with a variety of quality metrics. The quality metrics currently available are a compilation

of historical and modern approaches that were re-implemented by researchers at Allen Institute for Brain Science⁵ and by the SpikeInterface team. All quality metrics can be computed for the entire duration of the recording or for specific time periods (epochs) specified by the user.

The quality metrics that have been implemented so far include:

- spike count: the total spike count for a sorted unit.
- SNR: the signal-to-noise ratio (SNR) of the sorted units.
- firing rate: the average firing rate in a time period.
- presence ratio: the fraction of a time period in which spikes are present.
- ISI violations: the rate of inter-spike-interval (ISI) refractory period violations
- amplitude cutoff: an estimate of the miss rate based on an amplitude histogram.
- isolation distance: the Mahalanobis distance from a specified unit within as many spikes belong to the specified unit as to other units [31].
- L-ratio: the Mahalanobis distance and χ^2 inverse cumulative density function (under the assumption that the spikes in the unit distribute normally in each dimension) are used to find the probability of unit membership for each spike [64].
- d' : the classification accuracy between units based on linear discriminant analysis (LDA) [34].
- nearest-neighbors: a non-parametric estimate of unit contamination using nearest-neighbor classification [19].
- silhouette score: a standard metric for quantifying cluster overlap [62].
- maximum drift: the maximum change in spike position throughout a recording.
- cumulative drift: The cumulative change in spike position throughout a recording.

To compute quality metrics with SpikeInterface, the user can instantiate and use a `MetricCalculator` object. The `MetricCalculator` utilizes a `RecordingExtractor/SortingExtractor` pair to generate and cache all data needed to run the quality metrics (amplitudes, principal components, etc.) and also to calculate any (or all) of the quality metrics. We also allow for quality metrics to be computed with a functional interface. All quality metric function calls internally utilize a `MetricCalculator`, concealing the object representation from the user.

The following code snippet demonstrates how to compute a single quality metric (SNR) and all quality metrics with two function calls:

```
import spikeinterface.toolkit as st
snr_metric = st.validation.compute_snrs(sorting, recording)
all_metrics = st.validation.compute_metrics(sorting, recording)
```

⁵https://github.com/AllenInstitute/ecephys_spike_sorting

3.2.4 Curation

The `curation` module allows users to quickly remove units from a `SortingExtractor` based on computed quality metrics. To curate a sorted dataset, the user passes a `SortingExtractor` to a curation function which returns a new "curated" `SortingExtractor` (similar to how pre-processing works). This new `SortingExtractor` can be used in exactly the same way as the original extractor. As all curation functions take in and return a `SortingExtractor`, they can be naturally chained together to perform multiple curation steps on the same sorting output.

Currently, all implemented curation functions are based on excluding units given a threshold that is specified by the user (we provide sensible default values). If passed a `MetricCalculator`, curation functions will threshold units based on the *cached* quality metrics that are stored in the `MetricCalculator`. Otherwise, curation functions will recompute the associated quality metric and then threshold the dataset accordingly.

The following code snippet demonstrates how to chain together two curation functions that are based on different quality metrics and apply a "less than" threshold to the underlying units:

```
import spikeinterface.toolkit as st
sorting = st.curation.threshold_firing_rate(sorting, threshold=2.3,
                                           threshold_sign='less')
sorting_1 = st.curation.threshold_snr(sorting, recording, threshold=10,
                                     threshold_sign='less')
```

In this code snippet, `sorting_1` is still a `SortingExtractor`. However, when queried about the underlying units, `sorting_1` will return only the units that had firing rates higher than 2.3 Hz and SNRs greater than 10.

As of this moment, we support thresholding of all basic quality metrics including: spike count; SNR; firing rate; presence ratio; and ISI violations. We plan to include curation tools for all implemented quality metrics and for more complicated curation steps (merging, splitting, etc.) in future versions.

3.3 SpikeSorters

The `spikesorters`⁶ package provides a straightforward interface for running spike sorting algorithms supported by SpikeInterface. Modern spike sorting algorithms are built and deployed in a variety of programming languages including C, C++, MATLAB, and Python. Along with variability in the the underlying program language, each sorting algorithm may depend on external technologies like CUDA or command line interfaces (CLIs), complicating standardization. To unify these disparate algorithms into a single codebase, `spikesorters` provides Python-wrappers for each supported spike sorting algorithm. These spike sorting wrappers use a standard API for running the corresponding algorithms, internally handling intrinsic complexities such as automatic code generation for MATLAB- and CLI-based algorithms.

To allow for a simple, overarching API despite inherent differences between the sorting algorithms, each sorting wrapper is implemented as a subclass of a `BaseSorter` class. To run a spike sorting algorithm in SpikeInterface, the user passes a `RecordingExtractor` object to the wrapper and sets parameters for the underlying algorithm. Internally, each spike sorter wrapper creates and modifies the configuration based on these user-defined parameters and then runs the sorter on the dataset

⁶<https://github.com/SpikeInterface/spikesorters>

encapsulated by the `RecordingExtractor`. Once the spike sorting algorithm is finished, the sorting output is saved and a corresponding `SortingExtractor` is returned for the user. Spike sorters can be invoked either by using the wrapper directly or by using a simple function call.

In the following code snippet, `Mountainsort4` and `Kilosort2` are used to sort an extracellular recording. Running each algorithm (and setting its associated parameters) can be done using a single function:

```
import spikeinterface.sorters as ss
sorting_MS4 = ss.run_mountainsort4(recording, adjacency_radius=50)
sorting_KS2 = ss.run_kilosort2(recording, detect_threshold=5)
```

Along with running each sorting algorithm normally, our spike sorting wrappers allow for users to sort specific "groups" of channels in the recording separately (and in parallel, if specified). This can be very useful for multiple tetrode recordings where the data are all stored in one file, but the user wants to sort each tetrode separately. For large-scale analyses where the user wants to run many different spike sorters on many different datasets, `spikesorters` also provides a launcher function which handles any internal complications associated with running multiple sorters and returns a nested dictionary of `SortingExtractors` corresponding to each sorting output.

Currently, `SpikeInterface` supports 9 semi-automated spike sorters which are listed in Table 1. We encourage developers to contribute to this expanding list in future versions. We provide comprehensive documentation on how to do so⁷.

3.4 SpikeComparison

The `spikecomparison` package⁸ provides a variety of tools that allow users to compare and benchmark sorting outputs. Along with these comparison tools, `spikecomparison` also provides the functionality to run systematic performance comparisons of multiple spike sorters on multiple ground-truth recordings.

Within `spikecomparison`, there exist three core comparison functions:

1. `compare_two_sorters` - Compares two sorting outputs.
2. `compare_multiple_sorters` - Compares multiple sorting outputs.
3. `compare_sorter_with_ground_truth` - Compares a sorting output to ground truth.

Each of these comparison functions takes in multiple `SortingExtractors` and uses them to compute agreement scores among the underlying spike trains. The agreement score between two spike trains is defined as:

$$score = \frac{\#n_{matches}}{\#n_1 + \#n_2 - \#n_{matches}} \quad (1)$$

where $\#n_{matches}$ is the number of "matched" spikes between the two spike trains and $\#n_1$ and $\#n_2$ are the number of spikes in the first and second spike train, respectively. Two spikes from two different

⁷<https://spikeinterface.readthedocs.io/en/latest/contribute.html>

⁸<https://github.com/SpikeInterface/spikecomparison>

spike trains are "matched" when they occur within a certain time window of each other (this window length can be adjusted by the user and is 0.4 ms by default).

When comparing two sorting outputs (`compare_two_sorters`), a linear assignment based on the Hungarian method [38] is used. With this assignment method, each unit from the first sorting output can be matched to at most one other unit in the second sorting output. The final result of this comparison is then the list of matching units (given by the Hungarian method) and the agreement scores of the spike trains.

The multi-sorting comparison function (`compare_multiple_sorters`) can be used to compute the agreement among the units of many sorting outputs at once. Internally, pair-wise sorter comparisons are run for all of the sorting output pairs. A graph is then built with the sorted units as nodes and the agreement scores among the sorted units as edges. With this graph implementation, it is straightforward to query for units that are in agreement among multiple sorters. For example, if three sorting outputs are being compared, any units that are in agreement among all three sorters will be part of a subgraph with large weights.

For a ground-truth comparison (`compare_sorter_with_ground_truth`), either the Hungarian or the best-match method can be used. With the Hungarian method, each tested unit from the sorting output is matched to at most a single ground-truth unit. With the best-match method, a tested unit from the sorting output can be matched to multiple ground-truth units (above an adjustable agreement threshold) allowing for more in-depth characterizations of sorting failures.

Additionally, when comparing a sorting output to a ground-truth sorted result, each spike can be optionally labeled as:

- true positive (*tp*): spike found both in the ground-truth spike train and tested spike train.
- false negative (*fn*): spike found in the ground-truth spike train, but not in the tested spike train.
- false positive (*fp*): spike found in the tested spike train, but not in the ground-truth spike train.

Using these labels, the following performance measures are computed:

- accuracy: $\frac{\#tp}{(\#tp+\#fn+\#fp)}$
- recall: $\frac{\#tp}{(\#tp+\#fn)}$
- precision: $\frac{\#tp}{(\#tp+\#fp)}$
- miss rate: $\frac{\#fn}{(\#tp+\#fn)}$
- false discovery rate: $\frac{\#fp}{(\#tp+\#fp)}$

Based on the matching results and the scores, the units of the sorting output are classified as *well-detected*, *false positive*, *redundant*, and *over-merged*. Well-detected units are matched units with an agreement score above 0.8. False positive units are unmatched units or units which are matched with an agreement score below 0.2. Redundant units have agreement scores above 0.2 with only one ground-truth unit, but are not the best matched tested units (redundant units can either be oversplit or duplicate units). Over-merged units have an agreement score above 0.2 with two or more ground-truth units. All threshold scores are adjustable by the user.

The following code snippet shows how to perform all three types of spike sorter comparisons:

```
import spikeinterface.comparison as sc
comp_type_1 = sc.compare_two_sorters(sorting1, sorting2)
comp_type_2 = sc.compare_multiple_sorters([sorting1, sorting2, sorting3])
comp_type_3 = sc.compare_sorter_with_ground_truth(gt_sorting, tested_sorting)
```

Along with the three comparison functions, `spikecomparison` also includes a `GroundTruthStudy` class that allows for the systematic comparison of multiple spike sorters on multiple ground-truth datasets. With this class, users can set up a study folder (in which the recordings to be tested are saved), run several spike sorters and store their results in a compact way, perform systematic ground-truth comparisons, and aggregate the results in `pandas` dataframes [48]. An example ground-truth study is shown in Section 5.2.

3.5 SpikeWidgets

The `spikewidgets` package⁹ implements a variety of widgets that allow for efficient visualization of different elements in a spike sorting pipeline.

There exist four categories of widgets in `spikewidgets`. The first one only needs a `RecordingExtractor` for its visualization. This category includes widgets for time series, electrode geometry, signal spectra, and spectrograms. The second category only needs a `SortingExtractor` for its visualization. These widgets include displays for raster plots, auto-correlograms, cross-correlograms, and inter-spike-interval distributions. The third category utilizes both a `RecordingExtractor` and a `SortingExtractor` for its visualization. These widgets include visualizations of unit waveforms, amplitude distributions for each unit, amplitudes of each unit over time, and PCA features. The fourth, and final, category needs comparison objects from the `spikecomparison` package for its visualization. These widgets allow the user to visualize confusion matrices, agreement scores, spike sorting performance metrics (e.g. accuracy, precision, recall) with respect to a unit property (e.g. SNR), and the agreement between multiple sorting algorithms on the same dataset.

The following code snippet demonstrates how `SpikeInterface` can be used to visualize ten seconds of both the extracellular traces and the corresponding raster plot:

```
import spikeinterface.widgets as sw
sw.plot_timeseries(recording, channel_ids=[0,1,2,3], trange=[0,10])
sw.plot_rasters(sorting, unit_ids=[0,1,3], trange=[0,10])
```

The widget class is easily extendable, and will likely grow rapidly as new visualization tools are added. We also plan to introduce interactive widgets that allow the user to quickly explore the underlying elements of a spike sorting pipeline.

4 Building a Spike Sorting Pipeline

So far, we have given an overview of each of the main packages in isolation. In this section, we illustrate how these packages can be combined, using both the Python API and the `Spikely` GUI, to build a

⁹<https://github.com/SpikeInterface/spikewidgets>

robust spike sorting pipeline. The spike sorting pipeline that we construct using SpikeInterface is depicted in Figure 2A and consists of the following analysis steps:

1. Loading an Open Ephys recording [66].
2. Loading a probe file.
3. Applying a bandpass filter.
4. Applying common median referencing to reduce the common mode noise.
5. Spike sorting with Mountainsort4.
6. Removing clusters with less than 100 events.
7. Exporting the results to Phy for manual curation.

Traditionally, implementing this pipeline is challenging as the user has to load data from multiple file formats, interface with a probe file, memory-map all the processing functions, prepare the correct inputs for Mountainsort4, and understand how to export the results into Phy. Even if the user manages to implement all of the analysis steps on their own, it is difficult to verify their correctness or reuse them without proper unit testing and code reviewing.

4.1 Using the Python API

Using SpikeInterface's Python API to build the pipeline shown in Figure 2A is straightforward. Each of the seven steps is implemented with a single line of code (as shown in Figure 2B). Additionally, data visualizations can be added for each step of the pipeline using the appropriate widgets (as described in Section 3.5). Unlike handmade scripts, SpikeInterface has a wide range of unit tests and has been carefully developed by a team of researchers. Users, therefore, can have increased confidence that the pipelines they create are correct and reusable.

4.2 Using the `spikely` GUI

Along with our Python API, we also developed `spikely`¹⁰, a PyQt-based GUI that allows for simple construction of complex spike sorting pipelines. With `spikely`, users can build workflows that include: (i) loading a recording and a probe file; (ii) performing pre-processing on the underlying recording with multiple processing steps; (iii) running any spike sorter supported by SpikeInterface on the processed recording; (iv) automatically curating the sorter's output; and (v) exporting the final result to a variety of file formats, including Phy. At its core, `spikely` utilizes SpikeInterface's Python API to run any constructed spike sorting workflow. This ensures that the functionality of `spikely` grows organically with that of SpikeInterface.

Figure 2C shows a screenshot from `spikely` where the pipeline in Figure 2A is constructed. Each stage of the pipeline is added using drop-down lists, and all the parameters (which were not left at their default values) are set in the right-hand panel. Once a pipeline is constructed in `spikely`, the user can save it using the built-in save functionality and then load it back into `spikely` at a later date. Since `spikely` is cross-platform and user-friendly, we believe it can be utilized to increase the accessibility and reproducibility of spike sorting.

¹⁰<https://github.com/SpikeInterface/spikely>

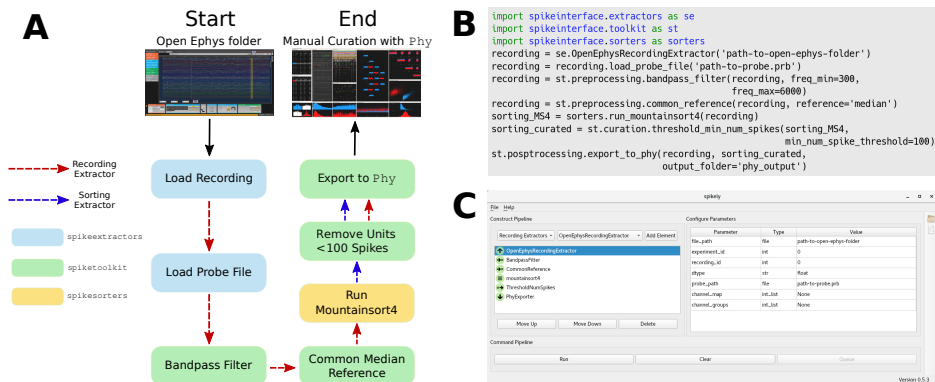


Figure 2: Sample spike sorting pipeline using SpikeInterface. (A) A diagram of a sample spike sorting pipeline. Each processing step is colored to represent the SpikeInterface package in which it is implemented and the dashed, colored arrows demonstrate how the **Extractors** are used in each processing step. (B) How to use the Python API to build the pipeline shown in (A). (C) How to use the GUI to build the pipeline shown in (A).

5 Applications

We present two applications of the SpikeInterface framework in this section. In Application 1, we sort a Neuropixels dataset with six popular spike sorters. After sorting, we quantify and visualize the agreement among the spike sorters. In Application 2, we sort a simulated, ground-truth dataset with the same six spike sorters. Afterwards, we systematically evaluate and visualize the performance of each sorter (based on their default parameters). These applications demonstrate the advantages of using SpikeInterface for spike sorting analysis and highlight unsolved issues in the field. All analysis is done with PyPI version 0.9.0 of `spikeinterface`.

5.1 Application 1: Comparing Spike Sorters on Neuropixels Data

In this application, we utilize SpikeInterface to sort a dense *in vivo* recording with many different spike sorters. After sorting and without ground-truth information, we use SpikeInterface to assess the level of agreement between spike sorters.

The dataset we use in this application is a recording from a rat cortex using the Neuropixels probe ([47, 46] – recording `c1`). It has a duration of 270 seconds, 384 channels, and a sampling frequency of 30 kHz. The raw data are first pre-processed with a bandpass filter (highpass cutoff 300 Hz – lowpass cutoff 6000 Hz) and are subsequently pre-processed with a common median reference filter.

For this analysis, we choose to run six different spike sorters: HerdingSpikes2 [33], Kilosort2 [54], IronClust [35], SpykingCircus [74], Tridesclous [28], and Mountainsort4 [19]¹¹. As each of these algorithms are semi-automatic, we fix their parameters to default values to allow for straightforward comparison. We do not include Klusta [61], WaveClus [18], and Kilosort [55] in this analysis as Klusta can only

¹¹The versions for each spike sorter are as follows: SpykingCircus==0.8.2, Tridesclous==1.2.2, Mountainsort4==0.3.2, HerdingSpikes2==0.3.2, IronClust==4.8.8, Kilosort2==GitHub commit 2a39926.

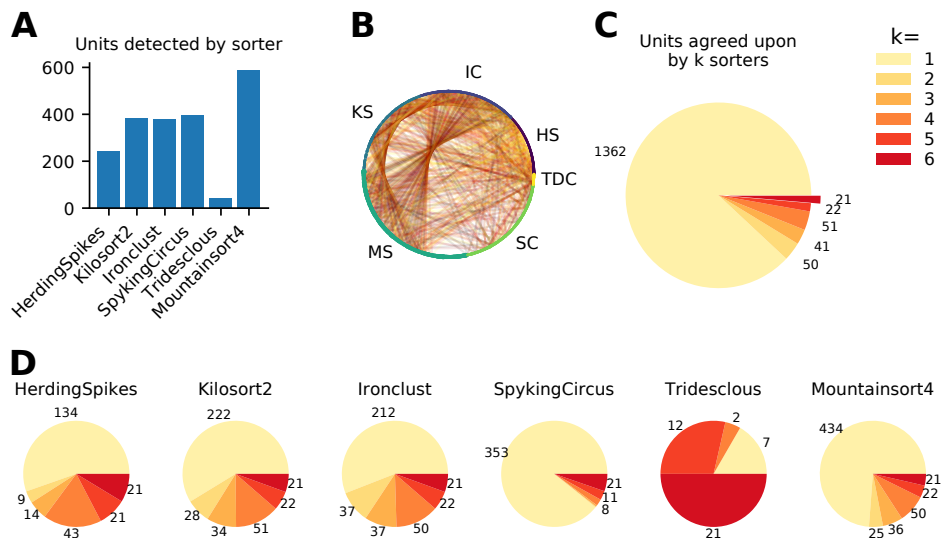


Figure 3: Analysis of Neuropixels recording with six spike sorters. (A) Number of units found by each spike sorter. (B) Network representation of the comparison between multiple sorters: each node is a unit and edges connect agreed upon units (edge color indicates agreement score). (C) Total number of units where k out of six sorters agree at a level of at least 0.5. (D) Number of units found and their agreement levels for each spike sorter.

handle up to 64 channels, WaveClus is designed for probes with a low channel count, and Kilosort is superseded by Kilosort2.

By quickly comparing the outputs of all six sorters, large discrepancies are immediately apparent. Figure 3A shows the number of units found by each sorter. While four of the sorters find between 200 and 400 putative units, Tridesclous only identifies 42, and Mountainsort4 almost 600.

Next, we use the `compare_multiple_sorters` function of the `comparison` module to explore these differences in more depth. As explained in Section 3.4, this function builds a weighted graph in which each node is a unit detected by a sorter and in which each edge is the best-match between a pair of units from different sorters. The edges are weighted by the respective agreement scores (Eq. 1; Figure 3B), and only edges with a score of at least 0.5 are kept. Once constructed, this graph can be interrogated to extract the units agreed upon by different sorters.

Figure 3C shows the overall agreement statistics. Surprisingly, out of a total of 1547 units, 1362 (~88%) are not matched at all, i.e. they are only found by a single sorter. The number of units found by all six sorters is just 21 which is only the 1.36% of the total number of units. Panel 3D breaks this result down for each spike sorter. For HerdngSpikes, Kilosort2, and IronClust, about half of the units are not matched by any other sorters. For SpykingCircus and Mountainsort4, an overwhelming majority of their units are not matched to another sorter (~90% for SpykingCircus, ~74% for Mountainsort4). Tridesclous is more conservative, as it finds very few units, but ~83% of them are matched by at least three sorters.

As units with little agreement are potentially noisy or very low-SNR units, we suggest a consensus-based

strategy for removing them. Using the multiple sorting comparison function, the units in agreement can automatically be extracted from the output of a sorter. This leads, potentially, to a subset of the putative units that are well-isolated and suited for downstream analysis. In future work, we plan to better understand low agreement units and to explore this consensus-based curation method.

5.2 Application 2: Benchmarking Spike Sorters on Simulated Data

In this application, we utilize SpikeInterface to evaluate and benchmark multiple spike sorters on a simulated, ground-truth dataset. We then illustrate that a popular, unsupervised quality metric for evaluating sorting outputs, SNR, can be correlated with a spike sorter's accuracy on the underlying ground-truth units. To be clear, the main goal of this application is to illustrate the capabilities of SpikeInterface to perform such comparisons and not to thoroughly analyze and benchmark the performance of different sorters which may be improved using different parameter sets or dedicated curation tools.

We use a simulated dataset¹² created with the MEArec Python package [16]. The probe is a square MEA with 100 channels, organized in a 10x10 configuration with an inter-electrode distance of 15 μm . The recording contains spiking activity from 50 neurons (from the Neocortical Micro Circuit Portal [56, 45]) that exhibit independent Poisson firing patterns. The recording also has an additive Gaussian noise with 10 μV standard deviation. For preprocessing, the recording is bandpass filtered (highpass cutoff 300 Hz - lowpass cutoff 6000 Hz).

For this analysis, we choose to benchmark the same six sorters as used in Application 1. We use the `GroundTruthStudy` class of the `comparison` module to run and benchmark all the algorithms in a systematic manner (as described in Section 3.4). Again, we use the default parameters of each sorter to allow for straightforward comparison.

As the full ground-truth information is available, we are able to thoroughly quantify the performance of each sorter with a variety of metrics. Figure 4A shows swarm plots of the accuracy, precision, and recall (these terms are defined in Section 3.4) for each sorter on all 50 ground-truth units. This type of analysis provides a good first insight into the strengths of each sorter, but does not tell the whole story. In this analysis, Kilosort2 appears to be the best performing sorter with a mean accuracy of 0.88 and the least variability across each of the metrics.

While assessing the accuracy of each sorter on the ground-truth units is important, it is also critical to analyze **all** the units found by the sorters, not just the well-detected ones. Figure 4B shows the number of well detected, redundant, false positive, and over-merged units for each sorter (these terms are defined in Section 3.4). From this analysis we can see that although Kilosort2 finds many well-detected units (43), it also returns a large number of false positive (58), redundant (6), and over-merged (3) units. Other sorters, in contrast, display a more conservative behavior. IronClust, HerdingSpikes, and Tridesclous, for example, find fewer well-detected units (30, 26, and 26, respectively), but also significantly fewer false positives, redundant, and over-merged units. This suggests that there may be a trade-off between unit isolation and reliability, a factor that has to be taken into account in subsequent analysis of sorted spike trains.

Additionally, SpikeInterface records the runtime of each sorter (Figure 4C). The spike sorters specifically designed to deal with high-density probes (HerdingSpikes, Kilosort2, and IronClust), as expected, have a lower computation time than more general-purpose software (Tridesclous, SpykingCircus, and Mountainsort4). All spike sorters were run on an Ubuntu 18.04 machine, an Intel(R) Core(TM) i7-8700 CPU 3.20GHz processor, and 64 GB of RAM. Additionally, IronClust and Kilosort2 were run using a

¹²<https://doi.org/10.5281/zenodo.3260283>

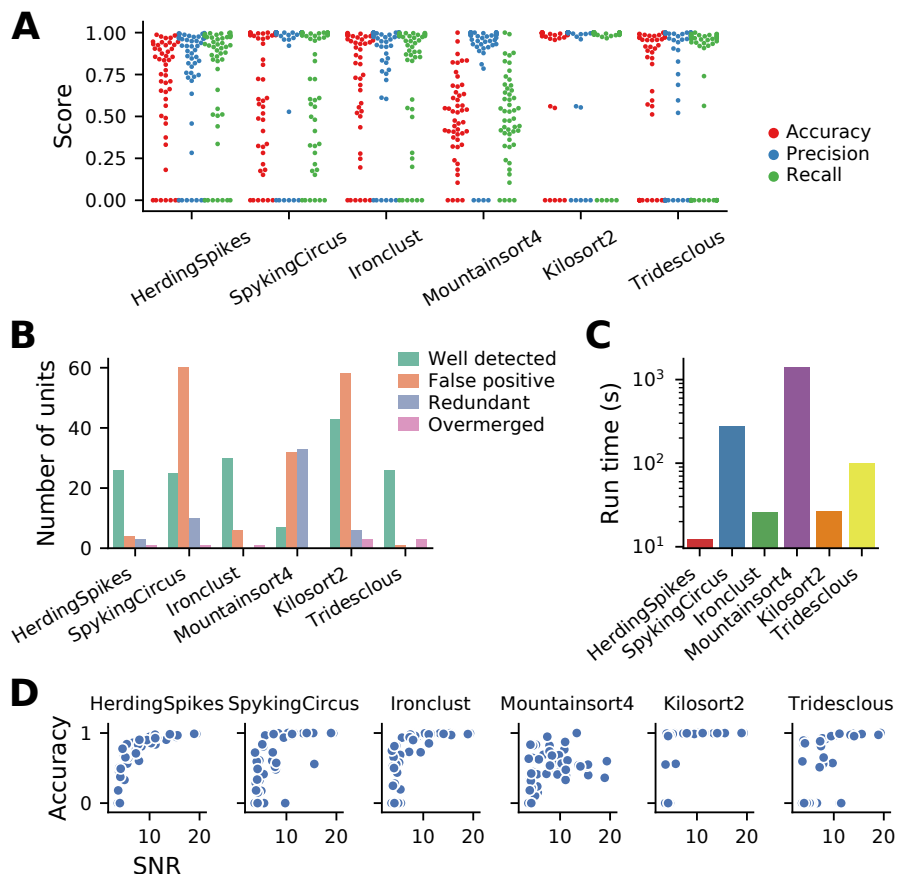


Figure 4: Analysis of a simulated ground-truth dataset. (A) Run times for each spike sorter. (B) Number of well detected, false positive, redundant, and over-merged units for each spike sorter. (C) Accuracy, precision, and recall for all ground-truth units for each spike sorter. (D) Accuracy on ground-truth units with respect to the SNR for each spike sorter.

GeForce RTX 2080 Ti GPU.

Finally, performance metrics can be also related to unsupervised quality metrics. In Figure 4D, for example, we plot the accuracy of each unit with respect to its SNR for each sorter. This plot illustrates that spike sorters generally are capable of isolating units with strong signals, but may differ in their ability to separate units with signals closer to the background noise level.

6 Discussion

We have introduced SpikeInterface, a Python framework designed to consolidate a complex ecosystem of software tools and file formats and to enhance the accessibility, reliability, and reproducibility of spike sorting. To highlight the modularity and careful design of SpikeInterface, we provided an overview of, and code examples for, each of the five main packages (Figure 1). To demonstrate how SpikeInterface can be used to construct flexible spike sorting workflows, we implemented an example pipeline (Figure 2A) using both the Python API (Figure 2B) and the `spikely` GUI (Figure 2C). Finally, to demonstrate potential applications of SpikeInterface, we evaluated the results of six spike sorters on both a NeuroPixels and a simulated recording.

6.1 Supported File Formats and Spike Sorters

The file formats and spike sorters currently supported by SpikeInterface are summarized in Table 1. We expect this list to grow in future versions as both spike sorting developers and the general neuroscience community contribute to the growth of SpikeInterface. In order to facilitate contributions to SpikeInterface, we provide documentation on how to add a `RecordingExtractor`, a `SortingExtractor`, or a spike sorter¹³ to our framework. At present, several `Extractors` have already been developed by or in collaboration with external contributors (SpikeGLX, Neurodata Without Borders, MCS H5, MaxOne, NIX, and Neuroscope).

Raw File Formats	Sorted File Formats	Sorters
Klusta	Klusta	Klusta [61]
Mountainsort (MDA)	Mountainsort (MDA)	Mountainsort4 [36]
Phy/Kilosort/Kilosort2 [59, 55, 60]	Phy/Kilosort/Kilosort2	Kilosort [55]
SpyKING Circus	Spyking Circus	Kilosort2 [54]
Exdir [22]	Exdir	SpyKING Circus [74]
MEArec [16]	MEArec	HerdingSpikes2 [33]
SpikeGLX [37]	HerdingSpikes2	Tridesclous [28]
Open Ephys [66]	Tridesclous	IronClust [35]
Intan [2]	NPZ (numpy zip)	Wave clus [18]
Neurodata Without Borders (NWB) [70]	Neurodata Without Borders (NWB)	
NIX [5]	NeuroScope [6]	
MaxOne [3]		
MCS H5 [4]		
Neuroscope [32]		
Biocam HDF5 [1]		
Binary		

Table 1: In this table, we show SpikeInterface’s currently supported file formats and spike sorting algorithms. With the help of the neuroscience community, we plan to expand these lists in future versions.

6.2 Comparison to Other Frameworks

As mentioned in the introduction, many software tools have attempted to improve the accessibility and reproducibility of spike sorting. Here we review the four most recent tools that are in use (to our

¹³<https://spikeinterface.readthedocs.io/en/latest/contribute.html>

knowledge) and compare them to SpikeInterface.

`NeV2lkit` [14] is a cross-platform, C++-based GUI designed for the analysis of recordings from multi-shank multi-electrode arrays (Utah arrays). In this GUI, the spike sorting step consists of PCA for dimensionality reduction and then `klustakwik` for automatic clustering [61]. As `NeV2lkit` targets low-density probes where each channel is spike sorted separately, it is not suitable for the analysis of high-density recordings. Also, since it implements only one spike sorter, users cannot utilize any consensus-based curation or exploration of the data. The software is available online¹⁴, but it lacks version-control and automated testing with continuous integration platforms.

`SigMate` [44] is a MATLAB-based toolkit built for the analysis of electrophysiological data. `SigMate` has a large scope of usage including the analysis of electroencephalography (EEG) signals, local field potentials (LFP), and spike trains. Despite its large scope, or because of it, the spike sorting step in `SigMate` is limited to `Wave clus` [18], which is mainly designed for spike sorting recordings from a few channels. This means that both major limitations of `NeV2lkit` (as discussed above) also apply to `SigMate`. The software is available online¹⁵, but again, it lacks version-control and automated testing with continuous integration platforms.

Regalia et al. [57] developed a spike sorting framework with an intuitive MATLAB-based GUI. The spike sorting functionality implemented in this framework includes 4 feature extraction methods, 3 clustering methods, and 1 template matching classifier (O-Sort [63]). These "building blocks" can be combined to construct new spike sorting pipelines. As this framework targets low-density probes where signals from separate electrodes are spike sorted separately, its usefulness for newly developed high-density recording technology is limited. Moreover, this framework only runs with a specific file format (MCD format from Multi Channel Systems [4]). The software is distributed upon request.

Most recently, Nasiotis et al. [52] implemented `IN-Brainstorm`, a MATLAB-based GUI designed for the analysis of invasive neurophysiology data. `IN-Brainstorm` allows users to run three spike sorting packages, (`Wave clus` [18], `UltraMegaSort2000` [34], and `Kilosort` [55]). Recordings can be loaded and analyzed from six different file formats: Blackrock, Ripple, Plexon, Intan, NWB, and Tucker Davis Technologies. `IN-Brainstorm` is available on GitHub¹⁶ and its functionality is documented¹⁷. `IN-Brainstorm` does not include the latest spike sorting software [61, 74, 19, 35, 54, 33], however, and it does not cover any post-sorting analysis such as validation, curation, and sorting output comparison.

`SpikeInterface` overcomes all limitations of the aforementioned analysis frameworks by following rigorous design principles. As the scope of `SpikeInterface` is **focused** on spike sorting only, we were able to provide a **comprehensive** framework that encompasses all the functionality required for spike sorting. This includes interfacing with a wide range of commonly used file formats for extracellular recordings and sorting outputs, handling probe file information, pre-processing, spike sorting, post-processing, validation, curation (automatic or manual with `Phy`), comparison, and visualization. The **modularized** and object-oriented design of `SpikeInterface` enables users to build custom analysis pipelines using the Python API or the `spikely` GUI and for the codebase to expand gracefully with community contributions of new `Extractors` and spike sorters. Since `SpikeInterface` is **efficient** and already implements 9 modern spike sorters, it can be used to analyze large-scale recordings from next-generation multi-electrode arrays as shown in Section 5. Finally, `SpikeInterface` allows users to implement **reproducible** analysis pipelines with careful version control, fixed random seeds, and a standardized API. All source code is open-source, version-controlled, and tested with a continuous integration platform¹⁸.

¹⁴<http://nev2lkit.sourceforge.net/>

¹⁵<https://sites.google.com/site/muftimahmud/codes>

¹⁶<https://github.com/brainstorm-tools/brainstorm3>

¹⁷<https://neuroimage.usc.edu/brainstorm/e-phys/Introduction>

¹⁸<https://travis-ci.org/>

6.3 Outlook

As it stands, spike sorting is still an open problem. No step in the spike sorting pipeline is completely solved and no spike sorter can be used for all applications. With SpikeInterface, researchers can quickly build, run, and evaluate many different spike sorting workflows on their specific datasets and applications, allowing them to determine which will work best for them. Once a researcher determines an ideal workflow for their specific problem, it is straightforward to share and re-use that workflow in other laboratories studying similar problems. We envision that many laboratories will use SpikeInterface to satisfy their spike sorting needs.

Along with its applications to extracellular analysis, SpikeInterface is also a powerful tool for developers looking to create new spike sorting algorithms and analysis tools. Developers can test their methods using our efficient and comprehensive comparison functions. Once satisfied with their performance, developers can integrate their work into SpikeInterface, allowing them access to a large-community of new users and providing them with automatic file I/O and software deployment. For developers who work on projects that use spike sorting, SpikeInterface can be used out-of-the-box, providing more reliability and functionality than handmade spike sorting scripts. We envision that many developers will be excited to use and integrate with SpikeInterface.

Already, SpikeInterface is being used in a variety of applications. In one application, SpikeInterface is being used as the engine of a related project called SpikeForest [43]. SpikeForest is an interactive website for benchmarking and tracking the accuracy of publicly available spike sorting algorithms. At present, it includes ten sorting algorithms and more than 300 extracellular recordings with ground-truth firing information. These recordings include both simulations and paired recordings where ground-truth is obtained from juxtacellular signals.

Overall, we hope that SpikeInterface can become a standard tool in neuroscience and can help foster a stronger relationship between spike sorting users and developers. To this end, we are maintaining an open forum¹⁹ that can be a common space for the community to discuss any and all spike-sorting-related topics. We look forward to sharing and growing SpikeInterface over the years to come.

Competing interests

The authors declare no competing interests.

Acknowledgements

This work was supported by the Wellcome Trust grant 214431/Z/18/Z (MHH). APB is a doctoral fellow in the Simula-UCSD-University of Oslo Research and PhD training (SUURPh) program, an international collaboration in computational biology and medicine funded by the Norwegian Ministry of Education and Research. CLH is supported by the Thouron Award and by the Institute for Adaptive and Neural Computation, University of Edinburgh. JHS wishes to thank the Allen Institute founder, Paul G. Allen, for his vision, encouragement and support. We would also like to thank Shangmin Guo for his recent contributions to debugging and improving the codebase.

¹⁹www.spikeforum.org

References

- [1] Biocam. <https://www.3brain.com/biocamx.html>.
- [2] Intan technologies. <http://intantech.com/>.
- [3] Maxwell biosystems. <https://www.mxwbio.com/>.
- [4] Multi channel systems. <https://www.multichannelsystems.com/>.
- [5] Neuroscience information exchange format - nix. <http://g-node.github.io/nix/>.
- [6] Neuroscope. <http://neurosuite.sourceforge.net/>.
- [7] Plexon offline sorter. <https://plexon.com/products/offline-sorter/>.
- [8] G. N. Angotzi, F. Boi, A. Lecomte, E. Miele, M. Malerba, S. Zucca, A. Casile, and L. Berdoncini. Sinaps: An implantable active pixel sensor cmos-probe for simultaneous large-scale neural recordings. *Biosensors and Bioelectronics*, 126:355–364, 2019.
- [9] M. Ballini, J. Muller, P. Livi, Y. Chen, U. Frey, A. Stettler, A. Shadmani, V. Viswam, I. L. Jones, D. Jackel, M. Radivojevic, M. K. Lewandowska, W. Gong, M. Fiscella, D. J. Bakkum, F. Heer, and A. Hierlemann. A 1024-channel CMOS microelectrode array with 26,400 electrodes for recording and stimulation of electrogenic cells in vitro. *IEEE Journal of Solid-State Circuits*, 49(11):2705–2719, 2014.
- [10] A. H. Barnett, J. F. Magland, and L. F. Greengard. Validation of neural spike sorting algorithms without ground-truth information. *Journal of neuroscience methods*, 264:65–77, 2016.
- [11] L. Berdoncini, P. D. van der Wal, O. Guenat, N. F. de Rooij, M. Koudelka-Hep, P. Seitz, R. Kaufmann, P. Metzler, N. Blanc, and S. Rohr. High-density electrode array for imaging in vitro electrophysiological activity. *Biosensors & Bioelectronics*, 21(1):167–74, jul 2005.
- [12] H. Bokil, P. Andrews, J. E. Kulkarni, S. Mehta, and P. P. Mitra. Chronux: a platform for analyzing neural signals. *Journal of neuroscience methods*, 192(1):146–151, 2010.
- [13] L. L. Bologna, V. Pasquale, M. Garofalo, M. Gandolfo, P. L. Baljon, A. Maccione, S. Martinoia, and M. Chiappalone. Investigating neuronal activity by spycode multi-channel data analyzer. *Neural Networks*, 23(6):685–697, 2010.
- [14] M. Bongard, D. Micol, and E. Fernandez. Nev2lkit: a new open source tool for handling neuronal event files from multi-electrode recordings. *International journal of neural systems*, 24(04):1450009, 2014.
- [15] M. P. Bonomini, J. M. Ferrandez, J. A. Bolea, and E. Fernandez. Data-means: an open source tool for the classification and management of neural ensemble recordings. *Journal of neuroscience methods*, 148(2):137–146, 2005.
- [16] A. P. Buccino and G. T. Einevoll. Mearec: a fast and customizable testbench simulator for ground-truth extracellular spiking activity. *bioRxiv*, page 691642, 2019.
- [17] D. Carlson and L. Carin. Continuing progress of spike sorting in the era of big data. *Current opinion in neurobiology*, 55:90–96, 2019.
- [18] F. J. Chaure, H. G. Rey, and R. Quian Quiroga. A novel and fully automatic spike-sorting implementation with variable number of features. *Journal of neurophysiology*, 120(4):1859–1871, 2018.

- [19] J. E. Chung, J. F. Magland, A. H. Barnett, et al. A fully automated approach to spike sorting. *Neuron*, 95(6):1381–1394, 2017.
- [20] M. Denker, G. T. Einevoll, F. Franke, S. Grün, E. Hagen, J. N. D. Kerr, M. P. Nawrot, T. V. Ness, R. Ritz, L. S. Smith, T. Wachtler, and D. K. Wójcik. 1st incf workshop on validation of analysis methods. 2018.
- [21] G. Dimitriadis, J. P. Neto, A. Aarts, A. Alexandru, M. Ballini, F. Battaglia, L. Calcaterra, F. David, R. Fiath, J. Frazao, et al. Why not record from every channel with a cmos scanning probe? *bioRxiv*, page 275818, 2018.
- [22] S.-A. Dragly, M. Hobbi Mobarhan, M. E. Lepperød, S. Tennøe, M. Fyhn, T. Hafting, and A. Malthe-Sørensen. Experimental directory structure (exdir): An alternative to hdf5 without introducing a new file format. *Frontiers in neuroinformatics*, 12:16, 2018.
- [23] U. Egert, T. Knott, C. Schwarz, M. Nawrot, A. Brandt, S. Rotter, and M. Diesmann. Mea-tools: an open source toolbox for the analysis of multi-electrode data with matlab. *Journal of neuroscience methods*, 117(1):33–42, 2002.
- [24] B. Eversmann, M. Jenkner, F. Hofmann, C. Paulus, R. Brederlow, B. Holzapfl, P. Fromherz, M. Merz, M. Brenner, M. Schreiter, R. Gabl, K. Plehnert, M. Steinhauser, G. Eckstein, D. Schmitt-landsiedel, and R. Thewes. A 128 128 CMOS Biosensor Array for Extracellular Recording of Neural Activity. *IEEE Journal of Solid-State Circuits*, 38(12):2306–2317, 2003.
- [25] U. Frey, J. Sedivy, F. Heer, R. Pedron, M. Ballini, J. Mueller, D. Bakkum, S. Hafizovic, F. D. Faraci, F. Greve, K. U. Kirstein, and A. Hierlemann. Switch-matrix-based high-density micro-electrode array in CMOS technology. *IEEE Journal of Solid-State Circuits*, 45(2):467–482, 2010.
- [26] S. Garcia and N. Fourcaud-Trocmé. Openelectrophy: an electrophysiological data-and analysis-sharing framework. *Frontiers in neuroinformatics*, 3:14, 2009.
- [27] S. Garcia, D. Guarino, F. Jaillet, T. R. Jennings, R. Pröpper, P. L. Rautenberg, C. Rodgers, A. Sobolev, T. Wachtler, P. Yger, et al. Neo: an object model for handling electrophysiology data in multiple formats. *Frontiers in neuroinformatics*, 8:10, 2014.
- [28] S. Garcia and C. Pouzat. Tridesclous. <https://github.com/tridesclous/tridesclous>.
- [29] P. Gleeson, A. P. Davison, R. A. Silver, and G. A. Ascoli. A commitment to open source in neuroscience. *Neuron*, 96(5):964–965, 2017.
- [30] D. H. Goldberg, J. D. Victor, E. P. Gardner, and D. Gardner. Spike train analysis toolkit: enabling wider application of information-theoretic techniques to neurophysiology. *Neuroinformatics*, 7(3):165–178, 2009.
- [31] K. D. Harris, H. Hirase, X. Leinekugel, D. A. Henze, and G. Buzsáki. Temporal interaction between single spikes and complex spike bursts in hippocampal pyramidal cells. *Neuron*, 32(1):141–149, 2001.
- [32] L. Hazan, M. Zugaro, and G. Buzsáki. Klusters, neuroscope, ndmanager: a free software suite for neurophysiological data processing and visualization. *Journal of neuroscience methods*, 155(2):207–216, 2006.
- [33] G. Hilgen, M. Sorbaro, S. Pirmoradian, J.-O. Muthmann, I. E. Kepiro, S. Ullo, C. J. Ramirez, A. P. Encinas, A. Maccione, L. Berdondini, et al. Unsupervised spike sorting for large-scale, high-density multielectrode arrays. *Cell reports*, 18(10):2521–2532, 2017.
- [34] D. N. Hill, S. B. Mehta, and D. Kleinfeld. Quality metrics to accompany spike sorting of extracellular signals. *Journal of Neuroscience*, 31(24):8699–8705, 2011.

- [35] J. J. Jun, C. Mitelut, C. Lai, S. Gratiy, C. Anastassiou, and T. D. Harris. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *bioRxiv*, page 101030, 2017.
- [36] J. J. Jun, N. A. Steinmetz, J. H. Siegle, D. J. Denman, M. Bauza, B. Barbarits, A. K. Lee, C. A. Anastassiou, A. Andrei, C. Aydın, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232, 2017.
- [37] B. Karsh. SpikeGLX. <https://billkarsh.github.io/SpikeGLX/>.
- [38] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
- [39] K. Y. Kwon, S. Eldawlatly, and K. Oweiss. Neuroquest: a comprehensive analysis tool for extracellular neural ensemble recordings. *Journal of neuroscience methods*, 204(1):189–201, 2012.
- [40] J. H. Lee, D. E. Carlson, H. S. Razaghi, W. Yao, G. A. Goetz, E. Hagen, E. Batty, E. Chichilnisky, G. T. Einevoll, and L. Paninski. Yass: yet another spike sorter. In *Advances in Neural Information Processing Systems*, pages 4002–4012, 2017.
- [41] X.-q. Liu, X. Wu, and C. Liu. Spktool: An open source toolbox for electrophysiological data processing. In *2011 4th International Conference on Biomedical Engineering and Informatics (BMEI)*, volume 2, pages 854–857. IEEE, 2011.
- [42] C. M. Lopez, S. Mitra, J. Putzeys, B. Raducanu, M. Ballini, A. Andrei, S. Severi, M. Welkenhuyesen, C. Van Hoof, S. Musa, et al. 22.7 a 966-electrode neural probe with 384 configurable channels in 0.13 μm soi cmos. In *Solid-State Circuits Conference (ISSCC), 2016 IEEE International*, pages 392–393. IEEE, 2016.
- [43] J. Magland, J. Jun, E. Lovero, L. Greengard, A. Barnett, et al. SpikeForest, 2019. <https://spikeforest.flatironinstitute.org>.
- [44] M. Mahmud, A. Bertoldo, S. Girardi, M. Maschietto, and S. Vassanelli. Sigmate: a matlab-based automated tool for extracellular neuronal signal processing and analysis. *Journal of neuroscience methods*, 207(1):97–112, 2012.
- [45] H. Markram, E. Muller, S. Ramaswamy, et al. Reconstruction and simulation of neocortical microcircuitry. *Cell*, 163(2):456–492, 2015.
- [46] A. Marques-Smith, J. P. Neto, G. Lopes, J. Nogueira, L. Calcaterra, J. Frazão, D. Kim, M. G. Phillips, G. Dimitriadis, and A. Kampff. Simultaneous patch-clamp and dense cmos probe extracellular recordings from the same cortical neuron in anaesthetized rats. data available from <http://dx.doi.org/10.6080/K0J67F4T>.
- [47] A. Marques-Smith, J. P. Neto, G. Lopes, J. Nogueira, L. Calcaterra, J. Frazão, D. Kim, M. G. Phillips, G. Dimitriadis, and A. Kampff. Recording from the same neuron with high-density cmos probes and patch-clamp: a ground-truth dataset and an experiment in collaboration. *bioRxiv*, page 370080, 2018.
- [48] W. McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.
- [49] H.-J. Mucha. Xclust: clustering in an interactive way. In *XploRe: an Interactive Statistical Computing Environment*, pages 141–168. Springer, 1995.
- [50] E. Muller, J. A. Bednar, M. Diesmann, M.-O. Gewaltig, M. Hines, and A. P. Davison. Python in neuroscience. *Frontiers in neuroinformatics*, 9:11, 2015.

- [51] J. Müller, M. Ballini, P. Livi, Y. Chen, M. Radivojevic, A. Shadmani, V. Viswam, I. L. Jones, M. Fiscella, R. Diggelmann, A. Stettler, U. Frey, D. J. Bakkum, A. Hierlemann, J. Müller, M. Ballini, P. Livi, Y. Chen, M. Radivojevic, A. Shadmani, V. Viswam, I. L. Jones, M. Fiscella, R. Diggelmann, A. Stettler, U. Frey, D. J. Bakkum, and A. Hierlemann. High-resolution CMOS MEA platform to study neurons at subcellular, cellular, and network levels. *Lab on a Chip*, 15(13):2767–2780, 2015.
- [52] K. Nasiotis, M. Cousineau, F. Tadel, A. Peyrache, R. M. Leahy, C. C. Pack, and S. Baillet. Integrated open-source software for multiscale electrophysiology. *BioRxiv*, page 584185, 2019.
- [53] R. Oostenveld, P. Fries, E. Maris, and J.-M. Schoffelen. Fieldtrip: open source software for advanced analysis of meg, eeg, and invasive electrophysiological data. *Computational intelligence and neuroscience*, 2011:1, 2011.
- [54] M. Pachitariu, N. A. Steinmetz, and J. Colonell. Kilosort2. <https://github.com/MouseLand/Kilosort2>.
- [55] M. Pachitariu, N. A. Steinmetz, S. N. Kadir, et al. Fast and accurate spike sorting of high-channel count probes with kilosort. In *Advances in Neural Information Processing Systems*, pages 4448–4456, 2016.
- [56] S. Ramaswamy, J. Courcol, M. Abdellah, et al. The neocortical microcircuit collaboration portal: a resource for rat somatosensory cortex. *Front Neural Circuits*, 9, 2015.
- [57] G. Regalia, S. Coelli, E. Biffi, G. Ferrigno, and A. Pedrocchi. A framework for the comparative assessment of neuronal spike sorting algorithms towards more accurate off-line and on-line microelectrode arrays data analysis. *Computational intelligence and neuroscience*, 2016, 2016.
- [58] H. G. Rey, C. Pedreira, and R. Q. Quiroga. Past, present and future of spike sorting techniques. *Brain research bulletin*, 119:106–117, 2015.
- [59] C. Rossant and K. D. Harris. Hardware-accelerated interactive data visualization for neuroscience in python. *Frontiers in neuroinformatics*, 7:36, 2013.
- [60] C. Rossant, S. Kadir, D. Goodman, M. Hunter, and K. Harris. Phy. <https://github.com/cortex-lab/phy>.
- [61] 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*, 19(4):634, 2016.
- [62] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.
- [63] U. Rutishauser, E. M. Schuman, and A. N. Mamelak. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of neuroscience methods*, 154(1-2):204–224, 2006.
- [64] N. Schmitzer-Torbert and A. D. Redish. Neuronal activity in the rodent dorsal striatum in sequential navigation: separation of spatial and reward responses on the multiple t task. *Journal of neurophysiology*, 91(5):2259–2272, 2004.
- [65] L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: clustering, classification and density estimation using gaussian finite mixture models. *The R journal*, 8(1):289, 2016.
- [66] J. H. Siegle, A. C. López, Y. A. Patel, K. Abramov, S. Ohayon, and J. Voigts. Open ephys: an open-source, plugin-based platform for multichannel electrophysiology. *Journal of neural engineering*, 14(4):045003, 2017.

- [67] A. Sobolev, A. Stoewer, A. Leonhardt, P. L. Rautenberg, C. J. Kellner, C. Garbers, and T. Wachtler. Integrated platform and api for electrophysiological data. *Frontiers in neuroinformatics*, 8:32, 2014.
- [68] A. Stoewer, C. J. Kellner, J. Benda, T. Wachtler, and J. Grewe. File format and library for neuroscience data and metadata.
- [69] J. Teeters, J. Benda, A. Davison, S. Eglén, R. C. Gerkin, J. Grethe, J. Grewe, K. Harris, C. Kellner, Y. L. Franc, et al. Requirements for storing electrophysiology data. *arXiv preprint arXiv:1605.07673*, 2016.
- [70] J. L. Teeters, K. Godfrey, R. Young, C. Dang, C. Friedsam, B. Wark, H. Asari, S. Peron, N. Li, A. Peyrache, et al. Neurodata without borders: creating a common data format for neurophysiology. *Neuron*, 88(4):629–634, 2015.
- [71] J. Voigts. Simpleclust. <https://jvoigts.scripts.mit.edu/blog/simpleclust-manual-spike-sorting-in-matlab/>.
- [72] M. Weeks, M. Jessop, M. Fletcher, V. Hodge, T. Jackson, and J. Austin. The carmen software as a service infrastructure. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1983):20120080, 2013.
- [73] F. Wood, M. J. Black, C. Vargas-Irwin, M. Fellows, and J. P. Donoghue. On the variability of manual spike sorting. *IEEE Transactions on Biomedical Engineering*, 51(6):912–918, 2004.
- [74] 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*, 7:e34518, 2018.
- [75] X. Yuan, S. Kim, J. Juyon, M. D’Urbino, T. Bullmann, Y. Chen, A. Stettler, A. Hierlemann, and U. Frey. A microelectrode array with 8,640 electrodes enabling simultaneous full-frame readout at 6.5 kfps and 112-channel switch-matrix readout at 20 ks/s. In *VLSI Circuits (VLSI-Circuits), 2016 IEEE Symposium on*, pages 1–2. IEEE, 2016.
- [76] B. Zhang, J. Dai, and T. Zhang. Neoanalysis: A python-based toolbox for quick electrophysiological data processing and analysis. *Biomedical engineering online*, 16(1):129, 2017.

Paper III

**Independent component analysis
for fully automated multi-electrode
array spike sorting**



Paper IV

Real-time spike sorting for multi-electrode arrays with online independent component analysis



IV

Paper V

**Combining biophysical modeling
and deep learning for
multielectrode array neuron
localization and classification**

V

Combining biophysical modeling and deep learning for multielectrode array neuron localization and classification

 Alessio P. Buccino,^{1,2*} Michael Kordovan,^{3,4*} Torbjørn V. Ness,⁵ Benjamin Merkt,^{3,4} Philipp D. Häfziger,¹ Marianne Fyhn,¹ Gert Cauwenberghs,² Stefan Rotter,^{3,4*} and  Gaute T. Einevoll^{1,5*}

¹Center for Integrative Neuroplasticity (CINPLA), Faculty of Mathematics and Natural Sciences, University of Oslo, Oslo, Norway; ²Department of Bioengineering, University of California, San Diego, California; ³Bernstein Center Freiburg, Freiburg, Germany; ⁴Faculty of Biology, University of Freiburg, Freiburg, Germany; and ⁵Faculty of Science and Technology, Norwegian University of Life Sciences, Ås, Norway

Submitted 28 March 2018; accepted in final form 29 May 2018

Buccino AP, Kordovan M, Ness TV, Merkt B, Häfziger PD, Fyhn M, Cauwenberghs G, Rotter S, Einevoll GT. Combining biophysical modeling and deep learning for multielectrode array neuron localization and classification. *J Neurophysiol* 120: 1212–1232, 2018. First published May 30, 2018; doi:10.1152/jn.00210.2018.—Neural circuits typically consist of many different types of neurons, and one faces a challenge in disentangling their individual contributions in measured neural activity. Classification of cells into inhibitory and excitatory neurons and localization of neurons on the basis of extracellular recordings are frequently employed procedures. Current approaches, however, need a lot of human intervention, which makes them slow, biased, and unreliable. In light of recent advances in deep learning techniques and exploiting the availability of neuron models with quasi-realistic three-dimensional morphology and physiological properties, we present a framework for automatized and objective classification and localization of cells based on the spatiotemporal profiles of the extracellular action potentials recorded by multielectrode arrays. We train convolutional neural networks on simulated signals from a large set of cell models and show that our framework can predict the position of neurons with high accuracy, more precisely than current state-of-the-art methods. Our method is also able to classify whether a neuron is excitatory or inhibitory with very high accuracy, substantially improving on commonly used clustering techniques. Furthermore, our new method seems to have the potential to separate certain subtypes of excitatory and inhibitory neurons. The possibility of automatically localizing and classifying all neurons recorded with large high-density extracellular electrodes contributes to a more accurate and more reliable mapping of neural circuits.

NEW & NOTEWORTHY We propose a novel approach to localize and classify neurons from their extracellularly recorded action potentials with a combination of biophysically detailed neuron models and deep learning techniques. Applied to simulated data, this new combination of forward modeling and machine learning yields higher performance compared with state-of-the-art localization and classification methods.

Supplemental Material for this article is available online at the Journal website.

* A. P. Buccino and M. Kordovan contributed equally to this work. S. Rotter and G. T. Einevoll share senior authorship.

Address for reprint requests and other correspondence: A. P. Buccino, Postboks 1080 Blindern, 0316 Oslo, Norway (e-mail: alessio@ifi.uio.no).

convolutional neural networks; deep learning; extracellular action potentials; multielectrode arrays; neuron localization and classification

INTRODUCTION

The neural circuits of the brain involve the interplay of many different types of neurons. On the most superficial level, neurons are classified by their effect on the neurons they project to as either excitatory or inhibitory. Extracellular recordings enable us to measure the activity of neurons as electrical potential deflections mainly due to ionic transmembrane currents. In recent years, many groups have been developing high-density multielectrode arrays (MEAs) for in vitro and in vivo applications, which allow measurements of neuronal activity with high spatiotemporal resolution (Berndtini et al. 2014; Müller et al. 2015; Neto et al. 2016; Schröder et al. 2015; Welkenhuysen et al. 2016). These probes provide something close to a functional electrical imaging (Vassanelli 2014) of the neural activity, and this high information density can potentially be exploited to obtain a better understanding of the neural circuits under study. Specifically, it might be possible to extract information that could be used to localize the individual neurons and to classify them into their respective cell types. The latest developments in manufacturing of high-density neural probes call for novel analysis methods to exploit the richness and detail in the recordings.

Neural localization from extracellular action potentials (EAPs) recorded on a MEA is an ill-posed problem, since solutions might not be unique for complex neural morphologies. Current approaches for localization assume simple neuronal models to facilitate the inverse problem and make the solution unique. Examples of models used in previous studies are monopole current-source models (Blanche et al. 2005; Chelaru and Jog 2005; Kubo et al. 2008), dipole current-source models (Blanche et al. 2005; Mechler et al. 2011; Mechler and Victor 2012), as well as line-source models (Somogyvári et al. 2012). More recently, Delgado Ruz and Schultz (2014) used a modified ball-and-stick model (Pettersen and Einevoll 2008) to predict somatic locations. In these approaches, the soma posi-

tion is estimated by minimizing the error between the electrical potential on the MEA sites predicted by the chosen model and the recorded potential. However, it is experimentally challenging to measure the correct position of the soma (Neto et al. 2016); therefore, detailed computational neuronal models are usually used and treated as simulated ground truth to evaluate the accuracy of the localization methods (DeLgado Ruz and Schultz 2014; Somogyvári et al. 2005, 2012). The main limitations regarding neuron localization are that the models chosen to solve the inverse problem are often too simple to grasp complex spike waveforms (e.g., monopolar or bipolar current-source models) or are tuned to specific cell types (ball-and-stick model for pyramidal morphology).

Regarding classification of neurons, unsupervised clustering techniques are commonly applied to differentiate EAP shapes (Barthó et al. 2004; McCormick et al. 1985; Peyrache et al. 2012): narrow waveforms are considered to be fast-spiking inhibitory neurons and broad waveforms excitatory neurons. Also in this case, it is experimentally challenging, especially *in vivo*, to validate the classification methods. One approach is to measure a multitude of neurons and find putative monosynaptic connections based on the shape of spike train cross-correlograms. However, the rate of recorded monosynaptic connections is usually very low ($\sim 0.2\%$; Barthó et al. 2004; Peyrache et al. 2012), resulting in a small number of observations useful for validation. In neural classification, the complexity of spike shapes across the multiple recording sites is usually compressed by extracting spike widths (such as peak-to-peak and full-width half-maximum widths; Barthó et al. 2004; Peyrache et al. 2012) only from the electrode with the highest recorded amplitude.

In this study, we apply a powerful machine learning technique, namely, convolutional neural networks (CNNs), to clas-

sify excitatory and inhibitory neurons and to localize their somata from simulated EAPs. This approach—being a supervised learning algorithm—demands for a large amount of labeled data, in this case EAPs in combination with soma position and cell type of the neuron evoking the EAPs. The proposed method is schematically depicted in Fig. 1. First, compartmental cell simulations are performed (Fig. 1A) that yield EAP data sets with known simulated ground truth (forward modeling) (Fig. 1B). Relying on the simulations, CNNs are trained (Fig. 1C) on these data sets to predict position and cell type (Fig. 1D) of the neuron generating the simulated EAP. If the method is applied to experimental data (Fig. 1E), a spike-triggered average EAP (average waveform) serves as input to a CNN previously trained on simulated data to predict soma position and cell type. In addition to binary classification, we attempt to distinguish 11 different morphological types (m-type classification). The performance of the CNNs depending on different characteristics extracted from the EAP, different MEA designs, and different CNN configurations is explored. Finally, we evaluate the effect of varying neuron alignments with respect to the recording MEA. To put our approach into context, we compare its performance with established methods of cell localization and classification.

CNNs perform supervised machine learning and require a large data set to be trained. It would be experimentally challenging, if not unfeasible, to gather the required number of recordings of exact known position (used for localization) and morphology (used for classification) information. Therefore, we rely on detailed compartmental cell models to provide detailed simulated recordings and simulated ground-truth information. Forward biophysical modeling of extracellular potentials has been developed and refined over the last 30 years (Gold et al. 2006; Holt and Koch 1999; Lindén et al. 2014;

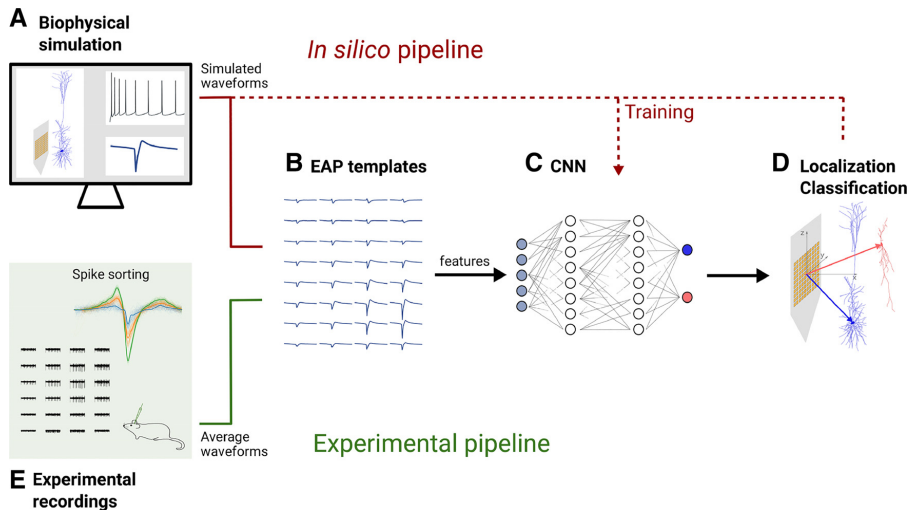


Fig. 1. Graphical representation of the presented method. Red arrows show our approach for training (dashed lines indicate error backpropagation) and validating the convolutional neural network (CNN) on simulated data. Green arrow shows how the method is applied within an experimental pipeline. Starting with the red path, biophysical simulations (A) are used to generate extracellular action potential (EAP) templates (B), from which features (e.g., amplitude and width; see *Classification and Localization Features*) are extracted and fed to a CNN (C) to localize and classify excitatory (blue) and inhibitory (red) neurons (D). When applied to experimental data (green arrow), recordings are first spike sorted (E), then features are extracted from spike-triggered average waveforms (B), and finally the CNN trained on simulated data (C) is used to localize and classify spike-sorted neurons (D).

Pettersen and Einevoll 2008; Rall and Shepherd 1968), and it is a still growing field. Therefore, we expect that computer simulations will become closer to real recordings as the detail and fidelity of neural models increase.

The aim of this study was to investigate the capability and feasibility of the proposed deep learning approach and to explore the large parameter space to guide future developments and experiments. At this stage, the method is a proof of concept, as we used some simplifications along the way. However, valuable information can be obtained and used in the future. Simulation software scripts are available at <https://github.com/CINPLA/NeuroCNN>.

MATERIALS AND METHODS

Cell Models

We first generated simulated spike recordings from various neuronal cell types (Fig. 1A). The neuronal models have been adopted from the Neocortical Microcircuit Collaboration (NMC) Portal (Markram et al. 2015; Ramaswamy et al. 2015, <https://bbp.epfl.ch/nmc-portal>), a database containing cell models from juvenile rat somatosensory cortex. We focused on neurons in layer (L)5, but the methods described can be applied to a larger variety of neuronal models. The cell models were used unmodified and are based on experimentally reconstructed morphologies, with up to 13 different types of active ion channels, depending on the cell type. The dynamics of these active ion channels have been fitted to reproduce specific features of somatic current-injection experiments, such as the amplitude and width of the evoked action potentials, the depth and timing of the following afterhyperpolarization, the mean interspike interval, spiking adaptations, and the amplitude of the backpropagating action potential at different positions along the apical dendrite of pyramidal cells (PC) (Markram et al. 2015). Importantly, this means that the source of the cell type-specific variability that we observe in, e.g., the spike widths is grounded in experimental data. For a more detailed description of the cell models the reader is referred to Markram et al. (2015). Note also that each of the cell models used can be explored interactively online through the NMC Portal (Ramaswamy et al. 2015, <https://bbp.epfl.ch/nmc-portal/microcircuit#/layer/L5>).

In principle, the data set contains 13 types of morphologies (m-types) in L5, in accordance with the NMC Portal. However, because of the limited data variety in the case of bipolar and neuroglial cells (BP and NGC), we excluded them from the analyses unless elsewhere specified. APPENDIX A describes the data set in detail and explains how we manipulated the original data set to extract unbiased sub-data sets for training and validating the results. A single-cell morphology of each m-type is displayed in Fig. 2, divided into inhibitory and excitatory neurons. Axons are not depicted because only their initial tract is included in the simulations.

Simulated Recordings

Extracellular action potential computation. Each of the multicompartment neuronal models was simulated separately, and extracellular potentials were calculated in two steps. First, transmembrane currents were computed by solving the cable equation (Holt and Koch 1999) with LFPy (Lindén et al. 2014) running on NEURON 7.4 (Carnevale and Hines 2006; Hines et al. 2009). A constant depolarizing current was applied to the soma to get the neuron firing at least 10 times (and not more than 30 times) in a simulation period of 1.2 s. Multiple spikes were simulated to account for spike-to-spike variations due to electrophysiological properties (e-types). All transmembrane currents for each compartment were stored within a time window $t = -2$ ms and $t = 5$ ms, where $t = 0$ is the time of the intracellular membrane voltage peak considered as spike time. Simulations were run with a time resolution of $dt = 2^{-5}$ ms, i.e., with a sampling frequency of 32 kHz, so that a single spike window of 7-ms duration (2 ms + 5 ms) consists of 224 samples.

Second, transmembrane currents were used within LFPy to calculate the extracellular potential at the recording site. Each transmembrane current, including the somatic one, was distributed over a line source with the length of its corresponding neural segment. With quasi-static approximation (Nunez and Srinivasan 2006) and assuming a homogeneous and isotropic neural tissue with conductivity $\sigma = 0.3$ S/m (Goto et al. 2010), the contribution of each compartment i at position \mathbf{r}_i with transmembrane current $I_i(t)$ to the electric potential on an electrode at position \mathbf{r}_j reads (Holt and Koch 1999; Lindén et al. 2014; Pettersen and Einevoll 2008)

$$\phi_e(\mathbf{r}_j, t) = \frac{1}{4\pi\sigma} I_i(t) \int \frac{d\mathbf{r}_i}{\|\mathbf{r}_j - \mathbf{r}_i\|} \quad (1)$$

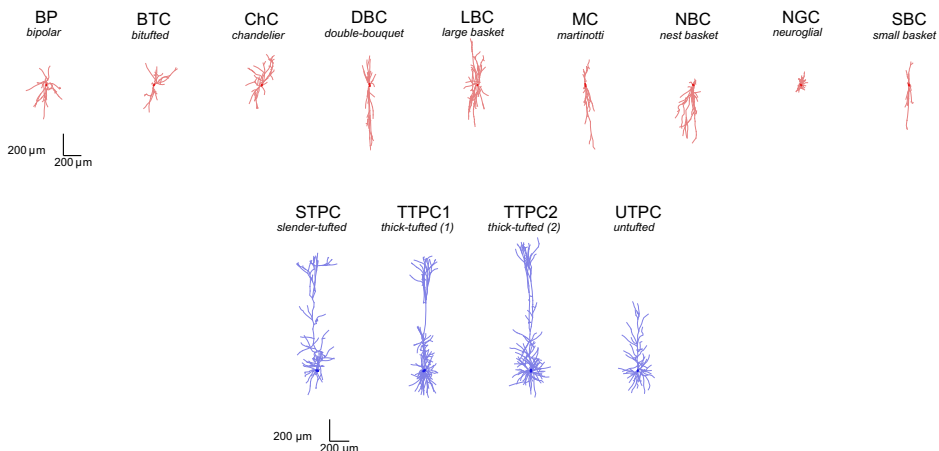


Fig. 2. One representative cell for each different morphology type in layer 5 [data from Neocortical Microcircuit Collaboration Portal (Ramaswamy et al. 2015)]. *Top*: 9 inhibitory cells (red). *Bottom*: 4 excitatory cells (blue). Dendrites are colored in lighter shades, and the soma is indicated with a darker circle. The same red/blue convention is used throughout article. For all cells C stands for cell, and for excitatory cells the P represents pyramidal.

The simulated EAP was obtained by summing up the contributions of all compartments. For each recording site, the electric potential was computed at a single point corresponding to the center of the recording electrode. These EAPs are associated with the templates in Fig. 1B.

Only spikes generating a notable waveform with a peak-to-peak amplitude exceeding $30 \mu\text{V}$ on at least one of the electrodes were included in the data set. The detection threshold of $30 \mu\text{V}$ was chosen to mimic experimental settings, where noise in the recordings due to equipment electronics and background neural signals does not allow detection of low-amplitude action potentials.

The coordinate system was fixed in reference to the MEA plane. Each recording site (different MEA designs are explained in *MEA designs*) lay within the yz -plane, and neuron somata were located within the semispace of the positive x -axis (the x -coordinate is thus the distance from the MEA). For each neuron, 1,000 EAP recordings above the detection threshold were simulated by randomly choosing one of the generated spikes and by placing the soma at random locations with distances x between $10 \mu\text{m}$ and $80 \mu\text{m}$. The y and z boundaries were different for each MEA, and a neuron could exceed the y and/or z boundary of the MEA by a maximum of $30 \mu\text{m}$. For the EAP, we considered the contributions of all dendritic compartments, including those crossing the MEA. This was done to force the sum of transmembrane currents to be zero (Pettersen and Einevoll 2008). In other words, we did not clip neurites entering the probe, but we made sure that their position did not coincide with a recording site within LFPy.

MEA designs. We investigated the performance of seven different MEA probes. Five of these were based on the prototype described in Schröder et al. (2015), in which the recording sites are arranged in a 16×16 matrix on a penetrating shank. Instead of considering a single interelectrode distance, we investigated five different pitches (i.e., distance between the centers of adjacent electrodes), namely, 10, 15, 20, 25, and $30 \mu\text{m}$. The probe models were built in a way that they roughly covered the same area on the shank.

Hence, we ended up with the following list of squared probes: SqMEA-15-10: squared MEA with 15×15 electrodes and $10\text{-}\mu\text{m}$

pitch; SqMEA-10-15: squared MEA with 10×10 electrodes and $15\text{-}\mu\text{m}$ pitch; SqMEA-7-20: squared MEA with 7×7 electrodes and $20\text{-}\mu\text{m}$ pitch; SqMEA-6-25: squared MEA with 6×6 electrodes and $25\text{-}\mu\text{m}$ pitch; and SqMEA-5-30: squared MEA with 5×5 electrodes and $30\text{-}\mu\text{m}$ pitch.

Moreover, we simulated recordings on the Poly3-25s probe (Neuronexus Technologies), which has 32 electrodes in three columns with a hexagonal arrangement, a y -pitch of $18 \mu\text{m}$, and a z -pitch of $22 \mu\text{m}$. Another probe becoming popular is the NeuroPixels probe (Jun et al. 2017), with recording sites arranged in a checkerboard pattern with a y -pitch of $32 \mu\text{m}$ and a z -pitch of $20 \mu\text{m}$. Although the probe has 384 recording channels, for convenience we decided to trim it to 128 channels. Finally, we constructed a model of the NeuroSeeker probe (<http://www.neuroseeker.eu>; used in Neto et al. 2016), a MEA with 128 recording sites arranged in a 4×32 configuration and a regular interelectrode distance of $22.5 \mu\text{m}$. Figure 3 shows all the probes in the yz -plane.

The CNNs we used required a rectangular shape of the input data. The two dimensions of the data array correspond to the number of electrode sites N_y and N_z in y - and z -directions, respectively. Therefore, we cut the Neuronexus-32 MEA probe to a Neuronexus-30, which is shown in the fourth position from the left in Fig. 3.

Neuron-MEA alignment. We investigated different neuron-MEA alignments (or rotations) of neurons. Some neurons, such as pyramidal cells (PC) or bipolar cells (BP), have morphologies that follow a specific orientation (see Fig. 2) that might affect the classification and localization performance. For this reason, we generated three rotational data sets:

Norot: The orientation of the cells (e.g., the apical dendrite of PCs) was along the z -axis (same orientation as in Delgado Ruz and Schultz 2014 and Somogyvári et al. 2012).

Physrot: Neurons with a preferential orientation were randomly rotated such that after rotation their axis from white matter toward the pia pointed into a cone around the z -direction with an opening angle of 15° (the puncture point on the unit sphere is uniformly distributed in this spherical cap). Neurons without a preferential orientation were rotated randomly in the three-dimensional (3D) space. We considered all

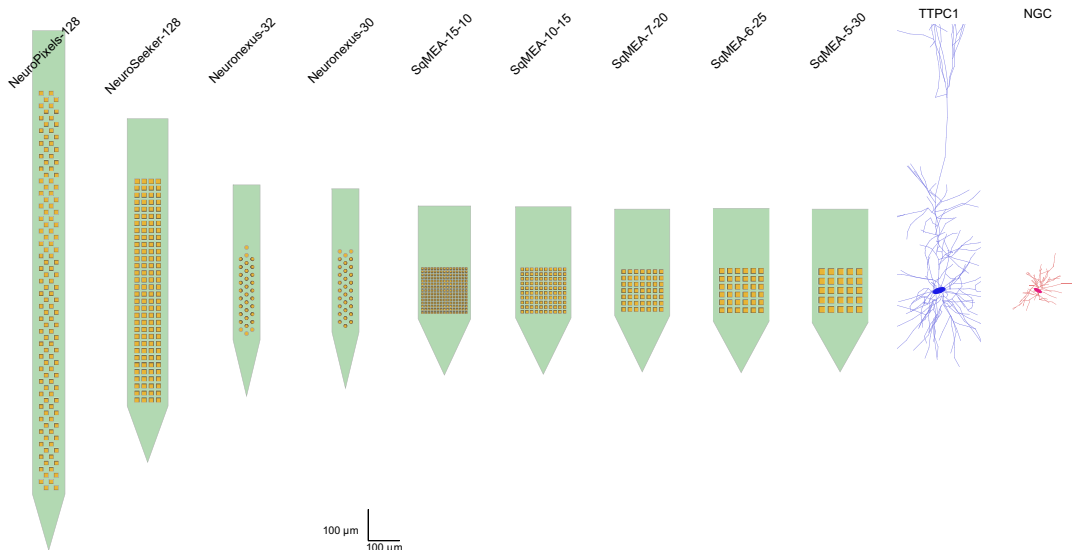
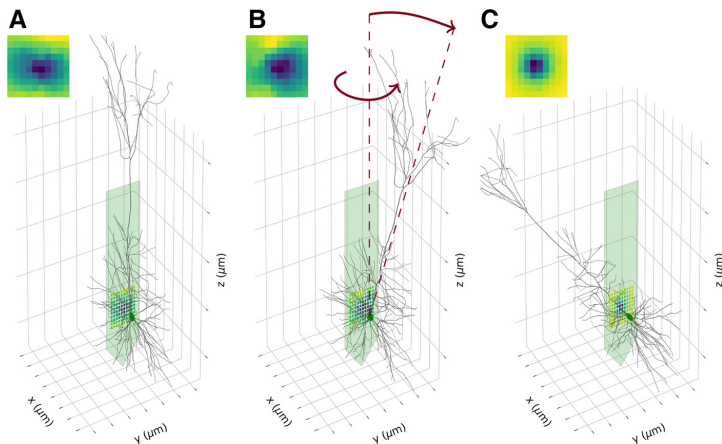


Fig. 3. Multielectrode array models used in the study. *Right:* plots show an excitatory cell [thick-tufted pyramidal cell (PC) with late bifurcating apical tuft (TTPC1)] and an inhibitory cell [neuroglial cell (NGC)] to compare probe and neuron sizes. PCs are on average much larger than inhibitory cells, and only a portion of the neuron is located directly in front of the probe (the apical dendrite is not fully shown, and it can be seen in Fig. 2).

Fig. 4. Neuron-multielectrode array (MEA) alignments: in each panel we show a sample orientation of a pyramidal cell [thick-tufted pyramidal cell with late bifurcating apical tuft (TTPC1)] placed at a fixed position of (50 μm, 0, 0). The MEA is the SqMEA-10-15, and the colors of each recording site (displayed as an image at top left of each plot) show the qualitative sodium peak image (Na image, explained in *Classification and Localization Features*). **A:** Norot: the pyramidal cell main axis is aligned to the z-direction. **B:** Physrot: the TTPC1 is rotated around the z-axis, and the apical dendrite is tilted at maximum 15° inside a cone around the z-axis. **C:** 3drot: the neuron can be rotated along all axes and might end up with its apical dendrite entering the MEA probe, as depicted in the plot. While the Na image is similar for A and B, it can become quite different in C, when the neuron is 3D-rotated.



neurons apart from nest basket cells (NBC), small basket cells (SBC), and NGC to have a preferential orientation (DeFelipe et al. 2006; Markram et al. 2004; Overstreet-Wadiche and McBain 2015; Wang et al. 2002, 2004; Woodruff and Yuste 2008). NBC, SBC, and NGC were assumed to have no preferential axis.

3drot: Neurons were rotated randomly around all axes.

The soma positions corresponded to the intersection point of rotation axes and were shifted randomly inside the observation volume in all cases. Figure 4 displays a sample orientation with respect to the MEA of a PC [thick-tufted PC with late bifurcating apical tuft (TTPC1)] in each of the three data sets, Norot (Fig. 4A), Physrot (Fig. 4B), and 3drot (Fig. 4C).

Classification and Localization Features

We extracted features from the EAPs as input variables to a CNN for training. Since classification and localization of neurons from extracellular recordings are quite different tasks, we used different sets of features from the simulated spikes. The pipeline for extracting feature images is described in Fig. 5. First, neurons with known cell type and position were simulated and the spike traces on the MEA probe were obtained. Then, for each spike, a set of features was computed and these features were rearranged in a 2D shape according to the MEA arrangement, i.e., the feature image. In the following paragraphs N , N_y , and N_z are the total number of electrodes, the number of electrodes in y-direction, and the number of electrodes in z-direction, respectively.

Localization features. The goal of localization is to estimate the soma position with respect to the probe. Therefore, we considered only the EAP negative peak and the positive peak time points, during which negative and positive transmembrane currents are larger in proximity of the soma (Delgado Ruz and Schultz 2014; Gold et al. 2007; Somogyvári et al. 2005, 2012). For simplicity, we refer to the EAP negative peak as Na peak because, close to the soma, it is mainly attributed to the sodium currents flowing into the soma. The positive peak is referred to as Rep because it is associated with the cell repolarization phase. The peak values were computed with respect to a reference of 0 μV, i.e., the baseline, as follows:

NA. For each spike recording on N electrodes, the spike with the largest negative peak amplitude was identified. At the time instant when the minimum peak occurred (t_{min}) the recorded potential on all N electrodes was used to build the Na image (the amplitude values are sampled at the same time instant t_{min}).

REP. The time instant of the repolarization peak (t_{max}) was extracted from the spike trace with the largest negative trough (same electrode as Na) and a Rep image was built by probing all N electrodes at t_{max} .

Overall, the localization-specific (N_y , N_z)-dimensional sets of features are Na, Rep, and NaRep, where the last is a stacked version of both features having dimension ($N_y, N_z, 2$).

Classification features. From each spike, we extracted features that are commonly used for cell classification (Barthó et al. 2004; Peyrache et al. 2012): peak-to-peak width (W), full-width half-maximum (F), and peak-to-peak amplitude (A). The peak-to-peak amplitude A , despite not being one of the commonly used features for classifying neurons from extracellular recordings, was selected as a feature in combination with F and W because spike widths increase with increasing recording distance (Anastassiou et al. 2015; Hagen et al. 2015; Pettersen and Einevoll 2008) and therefore with decreasing amplitude.

The following is a list including a detailed description of the features:

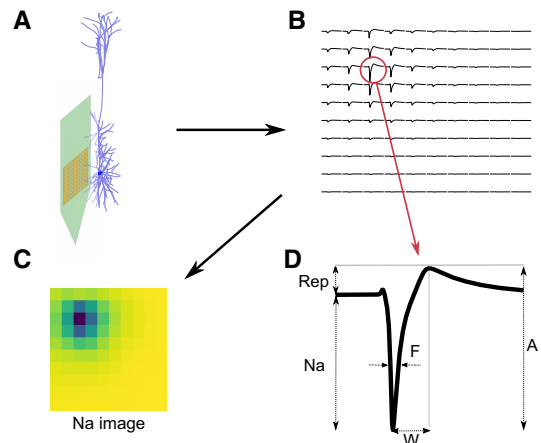


Fig. 5. Feature extraction pipeline: first, neuronal models (a pyramidal cell here) are simulated (A) and extracellular action potentials (EAPs; B) are computed on the multielectrode array (MEA) probe (SqMEA-10-15 here). Features (D) are then extracted from EAPs. Localization features are sodium negative peak (Na) and repolarization (positive) peak (Rep), and classification features are peak-to-peak amplitude (A), spike width (W), and full-width half-maximum width (F). The feature images (C) are finally used as input for convolutional neural networks.

A: For each recording site, the peak-to-peak amplitude was extracted as the absolute difference between the negative peak and the following positive repolarization peak. If the amplitude value of a recording site was less than a detection threshold of $5 \mu\text{V}$, then the amplitude for that electrode was set to zero.

W: For each electrode site, the width was computed as the time difference between the negative peak and the following positive repolarization peak. If the amplitude of the corresponding electrode was below the detection threshold (i.e., when the amplitude feature was set to 0), the width was set to the duration of the spike window, which was 7 ms.

F: For each electrode site, the full-width half-maximum was computed as the width at 50% of the negative maximum amplitude (refer to Fig. 5 for a graphical visualization and further explanation). In this case, the reference voltage was the initial voltage on the selected electrode site at beginning of the spike window. If the amplitude of the corresponding electrode was below the detection threshold (when the amplitude feature was set to 0), F was set to the duration of the spike window, which was 7 ms.

For classification, we considered the feature combinations $AW(N_s, N_z, 2)$, $FW(N_s, N_z, 2)$, and $AFW(N_s, N_z, 3)$, where the shapes of input arrays to the CNN are indicated in parentheses.

Waveform. We also investigated the performance using the entire spike waveform as input to the CNNs for localization and classification. While localization and classification features focused on amplitudes at specific time points (e.g., Na, Rep) or on extracting significant spike shape parameters (A, F, W), here we took into account the evolution of the action potential in time. As the additional third dimension ($2D + \text{time}$) increased the training time significantly, we downsampled the initial spike waveforms from 224 to 14 samples, i.e., with a downsampling factor of 16. We denoted this feature set, with a shape of $(N_s, N_z, 14)$, as *Waveform*.

Convolutional Neural Network

CNNs are biologically inspired artificial neural networks, and they differ from standard artificial neural networks mainly by the use of convolutional layers. The biological inspiration originates from the information processing in the visual system (Krizhevsky et al. 2012; Zeiler and Fergus 2014). For our implementation, we used the open-source software TensorFlow to train and evaluate the network (see Abadi et al. 2016; software available from <https://www.tensorflow.org>). All computations were done on the HPC clusters NEMO in Freiburg and ABEL in Oslo.

Configuration. We investigated the performance of CNNs of different sizes, all having the same underlying configurations (except for *Waveform* input features, whose CNN morphology is explained at the end of this section). Five CNN sizes (XS, S, M, L, XL) were used, and they differ in the size k_i of convolutional kernels (the index $i \in \{1, 2\}$ specifies the convolutional layer), convolutional layer depths d_i , and the number of nodes in the fully connected layer n_{FC} . The values used for different sizes are listed in Table B1 in APPENDIX B.

Feature images of dimension (N_s, N_z) were input to a d_1 -deep convolutional layer with rectified-linear units that filter the input image with (k_1, k_1) kernels and a stride of 1. Then max-pooling was applied, and the image was shrunk to a $(n_s, n_z) = (N_s/2, N_z/2)$ footprint. Another d_2 -deep convolutional layer with rectified-linear units applied (k_2, k_2) kernels, and a second max-pooling operation reduced the output image features to a $(m_s, m_z) = (n_s/2, n_z/2)$ size. The (m_s, m_z) features were input to a fully connected layer with n_{FC} nodes. The fully connected nodes were connected to the nodes in the output layer (see *Output layer and optimization*).

The *Waveform* feature set differed from the classification- and localization-specific sets as it included time as a dimension. Although some feature images for localization and classification were concatenated and thus had a 3D shape [for example, NaRep had a shape of $(N_s, N_z, 2)$ and *AFW* is $(N_s, N_z, 3)$ -dimensional], the optimized kernels

were the same for two or three dimensions. For the *Waveform* feature set, a 3D CNN was used, i.e., convolutional kernels were 3D with a shape of (k_1, k_1, k_1) and (k_2, k_2, k_2) . Max-pooling was also applied in all three dimensions. For the *Waveform* feature set, we used a CNN with $k_i = 4$.

Output layer and optimization. The output layer of the network was different depending on whether localization or classification was performed. In case of localization, three output nodes linearly summed the fully connected node inputs and biases to output the x -, y -, and z -coordinates. Optimization minimized the mean squared error between the predicted x -, y -, and z -coordinates and the true soma positions of the training spikes. For classification, there were instead two output nodes in case of excitatory/inhibitory classification. For the m -type classification, we used 11 output nodes, 1 for each cell type under consideration (see *Cell Models* for a list of m -types). During training, softmax cross entropy was minimized (Goodfellow et al. 2016).

For both localization and classification we used an Adam optimizer (learning rate = 0.0005) (see Kingma and Ba 2014), and we ran 2,000 training epochs. At each iteration, 10% of the training observations were randomly sampled and used to update network weights with backpropagation. To limit overfitting, we used dropout on the fully connected layer (Srivastava et al. 2014) with a dropout rate of 0.3 (during training 30% randomly chosen fully connected nodes were dropped and not considered for updating the network weights).

Validation strategy. To estimate the accuracy of the CNNs, we divided the input data into a training set, used to estimate the CNN parameters, and a validation set, upon which the trained CNN's accuracy was tested. Before the training-validation set division, we preprocessed the data set so that morphologies in the training and validation sets were unique (APPENDIX A). Then, we balanced the occurrence of observations of the same cell type (m -type) by random undersampling. For excitatory/inhibitory classification, we further subsampled the inhibitory neuron observations to match the excitatory ones (in the data set, there are 7 inhibitory cell types—not counting BP and NGC—and 4 excitatory types). The class balancing was performed for training and validation sets separately. For localization (and m -type classification), the training and validation data sets consisted of 44,000 and 11,000 instances, respectively. For classification, we used 32,000 observations for training and 8,000 for validation.

Comparison with Other Models

Localization. In previous work on neural localization, the underlying idea has been to solve the inverse problem by choosing a generative model and minimizing the error between the true extracellular potential and the one predicted by the chosen model. The soma position has been among the model parameters that have been optimized. Several models were used in previous studies: monopole and dipole current-source models (Blanche et al. 2005), line-source models (Somogyvári et al. 2012), and ball-and-stick models (Delgado Ruz and Schultz 2014). We compared our localization approach to inverse problem solutions solved with the EAP negative peak (i.e., Na image) with the following models (σ denotes the extracellular conductivity):

MONOPOLAR CURRENT SOURCE. A negative monopolar current-source I_s placed at position r_s evokes a potential $\phi(r)$ at position r according to

$$\phi(r) = \frac{I_s}{4\pi\sigma|r - r_s|} \quad (2)$$

The somatic current and soma position are the only parameters to be optimized. The predicted soma position is r_s .

BIPOLAR CURRENT SOURCE. Placing a negative current-source I_s at position r_{neg} and its positive counterpart (absolute value of I_s) at r_{pos} , the potential at position r reads

$$\phi(\mathbf{r}) = \frac{I_s}{4\pi\sigma} \left(\frac{1}{\|\mathbf{r} - \mathbf{r}_{\text{neg}}\|} - \frac{1}{\|\mathbf{r} - \mathbf{r}_{\text{pos}}\|} \right). \quad (3)$$

In this case, the estimated soma position corresponds to the negative current-source location, which is \mathbf{r}_{neg} . This model is equivalent to the two-monopole model in Pettersen and Einevoll (2008).

BALL AND STICK. The ball-and-stick model combines a somatic point-like constant current-source I_s at position \mathbf{r}_s with a dendritic stick of length d_{len} pointing in direction \mathbf{d} . We used a modified version of the ball-and-stick model described in Pettersen and Einevoll (2008), since we do not assume net currents to be zero (Delgado Ruz and Schultz 2014). The current along the stick $I(x)$ is assumed to decay exponentially, as confirmed by experimental data (Foust et al. 2010; Goldberg and Yuste 2005; Golding et al. 2001; Gulledge and Stuart 2003; Migliore et al. 1999; Sasaki et al. 2012; Waters et al. 2005). With initial negative value I_{dend} at \mathbf{r}_s , the current distribution along the stick reads

$$I(x_d) = I_{\text{dend}} \times \exp\left(-\frac{x_d}{d_\tau}\right) \quad (4)$$

where d_τ denotes the decay constant and $x_d \in [0, d_{\text{len}}]$ is the position along the dendritic segment (discretized in $N_{\text{seg}} = 50$ uniformly distributed points along the stick of length d_{len}). The predicted soma location corresponds to \mathbf{r}_s . The potential at position \mathbf{r} is given by the summation of the somatic and dendritic contributions:

$$\phi(\mathbf{r}) = \phi_{\text{soma}}(\mathbf{r}) + \phi_{\text{dend}}(\mathbf{r}) = \frac{I_s}{4\pi\sigma\|\mathbf{r} - \mathbf{r}_s\|} + \sum_{i=1}^{N_{\text{seg}}} \frac{1}{4\pi\sigma} I(x_i) \int \frac{d\mathbf{r}_i}{\|\mathbf{r} - \mathbf{r}_i\|} \quad (5)$$

where each segment is modeled as a line current source (see Eq. 1). Table 1 summarizes the parameters to be estimated for each described model.

GENETIC OPTIMIZATION. To estimate the model parameters, we minimized the sum of squared errors at each recording site between the extracellular potential predicted by the model and the extracted Na feature image of the true simulated extracellular potential. Optimization was performed with a genetic algorithm implemented with the Distributed Evolutionary Algorithms in Python (DEAP) package (Fortin et al. 2012). We used the $(\mu + \lambda)$ evolution strategy, which selects the next parents from the common set of the current parents (μ individuals) and the offspring (λ individuals). More precisely, the algorithm was implemented with the `deap.algorithms.eaMuPlusLambda` function. We used $\mu = 100$, $\lambda = 200$, crossover probability $p_{\text{cx}} = 0.8$, and mutation probability $p_{\text{mut}} = 0.2$. Furthermore, tournament selection (`deap.tools.selTournament`) and blend crossover (`deap.tools.cxBlend`) were used for selecting and mating individuals, respectively. Mutation was performed with a random Gaussian mutation (`deap.tools.mutGaussian`). When an individual was selected for mutation with probability p_{cx} , each of its elements was individually mutated with an individual probability of $p_{\text{mut}} = 0.3$. Gaussian means for all parameters were set to zero, and standard deviations (SDs) were

Table 1. Inverse model parameters

Model	Parameters	No. of Parameters
Monopolar	$I_s, r_{s_x}, r_{s_y}, r_{s_z}$	4
Bipolar	$I_s, r_{\text{pos}_x}, r_{\text{pos}_y}, r_{\text{pos}_z}, r_{\text{neg}_x}, r_{\text{neg}_y}, r_{\text{neg}_z}$	7
Ball and stick	$I_s, r_{s_x}, r_{s_y}, r_{s_z}, d_x, d_y, d_z, I_{\text{dend}}, d_{\text{len}}, d_\tau$	10

Summary of parameters for the different inverse models involved in the study. I_s , current source; r_s , predicted soma position; r_{pos} , position of positive I_s ; r_{neg} , position of negative I_s ; I_{dend} , dendritic current; d_{len} , dendritic length; d_τ , decay constant.

Table 2. Model parameter summary

Parameter	Range	Gaussian σ
I_s, I_{dend}	(−100, 0) nA	1 nA
x positions	(10, 80) μm	10 μm
y, z positions	(−180, 180) μm	10 μm
d_x, d_y, d_z	(−1, 1)	0.1
d_{len}	(1, 500) μm	50 μm
d_τ	(0.1, 500) μm	20 μm

Range for initialization and constraint and standard deviation σ for mutation Gaussian for the different parameters. I_s , current source; I_{dend} , dendritic current; d_{len} , dendritic length; d_τ , decay constant.

different depending on the parameter. The parameter values are summarized in Table 2, and we constrained the optimization to solutions within biophysically acceptable boundaries (shown in Table 2).

Classification. Besides applying a CNN, the problem of classifying neurons according to their EAP can be done by several other methods (Barthó et al. 2004; Delgado Ruz and Schultz 2014; Peyrache et al. 2012). It is a well-established observation that pyramidal excitatory cells present a broad spike waveform, while inhibitory cells have a narrow one (Barthó et al. 2004). Therefore, a standard way of classifying between the two classes is to plot spike width W and full-width half-maximum F (see *Classification and Localization Features* for feature extraction details) of the EAP with the maximum amplitude and then cluster the data points in this 2D space (Barthó et al. 2004; Peyrache et al. 2012). Once F and W were computed for the electrode with the maximum peak-to-peak amplitude, we applied two different clustering techniques to the point cloud: k -means and a mixture of Gaussians (MoG) clustering (Pedregosa et al. 2011). While k -means clustering iteratively assigns points to K clusters based on their distances to the cluster means and then recomputes the cluster means with new assignments until convergence, the MoG estimates K Gaussians to fit the data and then labels the data points based on the Gaussian shape. In this case, since the goal is excitatory/inhibitory classification, we set $K = 2$.

Statistical Analysis

For localization errors, statistical tests were run on the 11,000 validation observations: since all distributions did not satisfy the normality assumption, nonparametric tests were run (1-sided Mann-Whitney U -test; Mann and Whitney 1947). When sample sizes are large, statistical tests are prone to indicate that there is a significant difference (effect) between distributions, resulting in low P values. To characterize whether such difference is relevant, a measure of its magnitude, or effect size, should be included (Sullivan and Feinn 2012). To quantify the effect size, we used Cohen's d coefficient (Cohen 1992), i.e., the difference between population means normalized by the pooled SD. We considered significant differences (low P values) to be negligible (effect size < 0.2), small (effect size ≈ 0.2), medium (effect size ≈ 0.5), or large (effect size ≈ 0.8). Test results are shown in APPENDIX B, divided by group (data set rotation, Table B3; CNN size, Table B4; feature type, Table B5; probe type, Table B6; and localization method, Table B7). Each entry of the tables shows the Cohen's d coefficient (rounded to 2 decimals) and the significance of the Mann-Whitney U -test with the alternative hypothesis that column group $<$ row group.

RESULTS

In the following sections, we show localization and excitatory/inhibitory classification results only of m-type cells included in the training data set. Therefore, unless otherwise specified, BP and NGC are excluded from the analysis. The performance measures were different for localization and clas-

sification. In case of localization we used the average total error and for classification the average classification accuracy (ratio between correctly classified observations and total number of observations). Moreover, we analyzed the cell-specific accuracy to get more insight on the classification performance. The average localization error in the x -, y -, and z -directions can be interpreted in the following way. Assuming normally distributed errors, the true soma location is with 32% probability inside a box centered at the predicted soma position with edge lengths of twice the average localization error in the corresponding dimension. The probability rises to 87% with a box edge length of four times the average localization error in the corresponding dimension.

Dependence on Neuron-MEA Alignment

The first question we investigate is how the neuron-MEA alignment affects the localization and classification performance. Three data sets were built, Norot, Physrot, and 3drot. To focus on the effects of alignments, we use fixed feature sets (NaRep for localization, FW for classification; see *Classification and Localization Features* for definitions), MEA probe (SqMEA-15-10), and CNN size L.

Localization. First, we show the mean and SD of the localization errors along the three axes as well as the total error in Table B2. Each row displays the performance of a rotational data set. Average errors and SDs are $7.3 \pm 5.7 \mu\text{m}$ for Norot, $7.8 \pm 6.3 \mu\text{m}$ for Physrot, and $8.9 \pm 8.2 \mu\text{m}$ for 3drot.

The Norot data set results in significantly lower errors with respect to the 3drot data set (effect size = 0.21; Table B3). Negligible differences are found in the comparison between the Norot errors and the Physrot errors (effect size = 0.09) and between the Physrot and 3drot data sets (effect size = 0.13). Taking into account the finite size of the soma (~ 10 – $15 \mu\text{m}$ of diameter), we consider the resulting error values to be a sufficient performance for most applications. The errors along the three axes appear to be isotropically distributed, as they show similar values in all directions (but the observations in the x -direction are not uniformly distributed—as shown below in Fig. 9C—since we only considered spikes above $30 \mu\text{V}$ and spike amplitude decreases with distance).

In Fig. 6, A–C, we show the errors along x -, y -, and z -axes with respect to the x -, y -, and z -coordinates for the three rotational data sets. In these plots, we bin the true x , y , and z neuron positions in seven bins and treat them as categorical data (i.e., the positions can have discrete values depending on the bin they belong to). The data points are the mean of the error grouped by bin and rotation type and the error bar is the SD. Figure 6A shows that errors in the distance estimation (x -direction) tend to increase as the distance of the neuron increases for all three data sets, similarly to Delgado Ruz and Schultz (2014). Regarding the y - and z -dimensions (Fig. 6, B and C, respectively), it is interesting to note how the errors have a convex shape, meaning that errors tend to increase when the neuron is at the border of

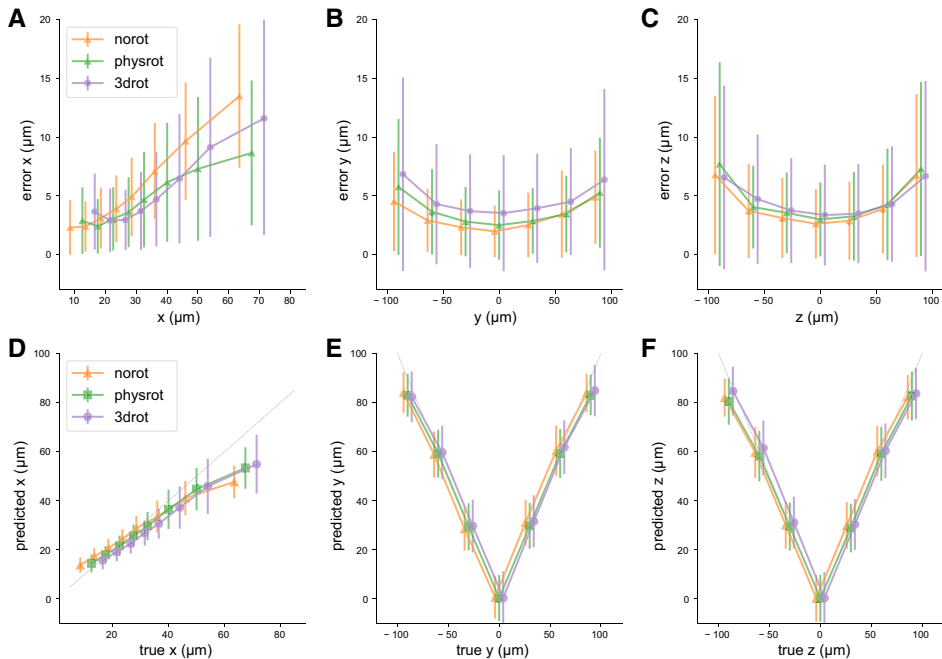


Fig. 6. *A*: x errors with respect to the x -coordinate (distance). *B*: y errors with respect to the y -coordinate. *C*: z errors with respect to the z -coordinate. *D*: x predictions with respect to the true x -coordinate (distance). *E*: absolute value of the y predictions with respect to the true y -coordinate. *F*: absolute value of the z predictions with respect to the true z -coordinate. All values are in μm . Orange lines indicate the *Norot* data set, green lines the *Physrot* data set, and purple lines the *3drot* data set. Gray lines correspond to a perfect prediction. Data are binned in 7 bins along x -, y -, and z -directions: points and error bars display the average errors and their SDs for each bin and each data set.

the probe, in which case only partial information about the spike is available.

When looking at the distribution of the predicted x -, y -, and z -coordinates with respect to the true coordinates in Fig. 6, $D-F$, one can note how the errors observed in Fig. 6, $A-C$, are caused by an underestimation of the soma distance in all dimensions: at large distances (x -direction) from the probe, neurons are predicted to be closer to the MEA; when they are close to the y and z borders of the probe, the predicted position is closer to the center of the MEA.

Next, we consider the variability of localization performance depending on cell types and alignment. In Fig. 7A the bar plots show the average total errors and their SDs grouped by neuron morphology type (11 training morphologies + BP and NGC; see Fig. 2 for representative cells) for the Norot, Physrot, and 3drot data sets. The range and distribution of distances taken into consideration are the same as in Fig. 6. Focusing on the Physrot data set, the minimum error of $4.7 \pm 3.8 \mu\text{m}$ is obtained for the SBC morphology, while the worst performance is $15.9 \pm 9.1 \mu\text{m}$ for slender-tufted PC (STPC) morphology. The difference in prediction performance with respect to cell type does not seem to be depending on excitatory/inhibitory morphologies (i.e., pyramidal and nonpyramidal cells), nor do they look to be clustered depending on morphological subclasses; for instance, among the different basket cells (names ending with BC) there is some variability among large basket cells (LBC), NBC, and SBC, and the same holds for pyramidal cells [STPC, TTPC1, thick-tufted PC with early bifurcating apical tuft (TTPC2), and untufted PC (UTPC)]. The performance of BP and NGC (Fig. 7A), which were not used for training, is in line with other cell types, with errors of $8.8 \pm 2.2 \mu\text{m}$ and $9.4 \pm 5.1 \mu\text{m}$, respectively. This result confirms that the method is capable of dealing with diverse morphologies, as long as the training set contains a large representation of cell types.

Classification. The accuracy analysis of excitatory/inhibitory classification is based on the FW feature set. Table B8 in APPENDIX B shows the classification accuracies for each cell morphology plus the average accuracy and the SD for the

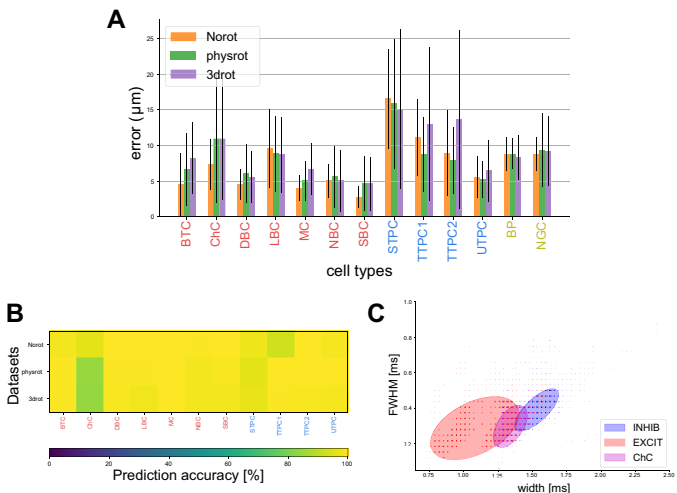
different data sets. The cell-specific accuracies are also visualized by color coding in Fig. 7B.

The average classification accuracy is equally high for the Norot and Physrot data sets ($98.1 \pm 2.4\%$ and $98.0 \pm 3.9\%$, respectively) and lower for the 3drot data set ($97.6 \pm 3.9\%$). This is because neurons are rotated with more degrees of freedom; nevertheless, on average the accuracy remains very high in all cases. A closer examination of this result reveals that the main reason for the drop in classification accuracy was misclassification of the chandelier cells (ChCs). The lowest value is the ChC accuracy in the 3drot data set (84.1%). In Fig. 7C, we show the spike shapes in the FW -plane of the electrode site with largest amplitude. Inhibitory neurons mainly lie in the lower left part (narrow spikes). Excitatory neurons are almost perfectly classified as excitatory cells, as shown in Table B8 and Fig. 7B. The spike shapes of ChC in the FW -plane mainly lie at the interface with the excitatory neurons. This might explain why they are harder to classify correctly with respect to the other cell types.

Effect of CNN Size

We investigated how localization and classification performances vary with network size. The results shown in this section were obtained with the SqMEA-10-15 probe, NaRep features for localization, and FW for classifications. For the remaining analyses, boxplots and cumulative distribution functions (cdfs) are used to represent the performance of the localization models. In all boxplots, the box is the interquartile range (IQR), i.e., the 25th and 75th percentiles, the horizontal lines inside the box show the medians, and the red diamonds display the means. The whiskers (horizontal black lines) represent the highest and lowest data values within 1.5 times the IQR. Data points outside the whiskers are plotted as black dots and are regarded as outliers. We obtained the cdfs by sorting the sample and pairing each data point with its normalized rank (percentile). Hence, the point where the cdf crosses 0.5 represents the median of the localization error.

Fig. 7. A: localization error grouped by cell type for all rotational data sets. Bars are the average errors, and error bars show the SDs in μm . Orange bars display the *Norot* data set, green bars display the *Physrot* data set, and purple bars display the *3drot* data set. Ticks on the x -axis show the cell types: red ticks are inhibitory cells, blue ticks are excitatory, and yellow ticks are bipolar (BP) and neuroglial (NGC) cells (not used for training). B: excitatory/inhibitory classification accuracy in color code grouped by rotation and cell type. Each element of the matrix is the accuracy of a specific cell type (red, inhibitory neurons; blue, excitatory neurons) in the different rotational data sets (rows). For explicit values see Table B8. C: spike shapes for maximum peak electrode sites are plotted in the FW -plane. Red dots are inhibitory neurons, and they lie in the lower left part of the plot. Blue dots show excitatory cells, in the upper right part of the plane. The magenta dots are chandelier cells (ChC), and they lie at the intersection between excitatory and inhibitory cells. The opacity represents the number of occurrences; more opaque dots correspond to more frequent observations. The ellipses are built by following the principal axis of each distribution, and the lengths of their major and minor axes are proportional to the eigenvalues of the covariance matrix. FWHM, full-width half-maximum.



Localization. Figure 8A shows the localization errors grouped by CNN size (XS, S, M, L, and XL). Increasing the size of the network improves the performance significantly (Table B4), but for sizes L and XL the average localization error is almost the same [$7.8 \pm 6.3 \mu\text{m}$ for L and $7.3 \pm 5.8 \mu\text{m}$ for XL (effect size = 0.09)]. If not stated otherwise, networks of size L have been chosen, as they provide a good compromise between performance and time required for training.

Classification. Table B9 in APPENDIX B shows the performance in classification into excitatory and inhibitory neuron types. The highest accuracy (98.6 \pm 1.1%) is reached with a network of size M, while all others show a slightly lower performance. A possible explanation for the lower score of the XL network is overfitting to the training set because of the large number of parameters.

Feature Selection

In the previous sections, we have presented results with fixed feature sets (NaRep for localization and FW for classification), eliminating the effects caused by other factors, such as alignment, cell type and CNN size. The following results were obtained on SqMEA-10-15 probes using CNNs of size M (because of the long training time required by 3D CNNs for Waveform feature).

Localization. In Fig. 8, C and D, we display the boxplots and cdf of the errors with varying feature sets for localization. In other studies, either the sodium peak is the only feature used (Blanche et al. 2005; Delgado Ruz and Schultz 2014; Mechler et al. 2011; Mechler and Victor 2012; Somogyvári et al. 2005)

or the entire spike time course is modeled (Somogyvári et al. 2012). Here we show that all CNNs relying on peak input show roughly the same performance: average errors \pm SDs are $8.8 \pm 7.1 \mu\text{m}$ for Na, $8.8 \pm 7.9 \mu\text{m}$ for Rep, and $8.8 \pm 7.7 \mu\text{m}$ for NaRep. Negligible differences are found when comparing Na, Rep, and NaRep, with effect sizes close to zero (Table B5).

The Waveform CNN results in a lower average prediction error of $6.9 \pm 6.5 \mu\text{m}$, which is significantly better in comparison with Na (effect size = 0.28), Rep (effect size = 0.26), and NaRep (effect size = 0.27; Table B5). We speculate that the performance of the Waveform approach is only slightly increased (by $\sim 2 \mu\text{m}$) for the following reason: when considering the peaks only, transmembrane currents are mainly concentrated around the soma (Delgado Ruz and Schultz 2014; Gold et al. 2007; Somogyvári et al. 2005, 2012); therefore, the peak features contain almost all information the CNN needs for soma location.

Classification. Classification performances are listed in Table B10 in APPENDIX B. The AFW feature set, with an accuracy of 98.6%, performs better than AW and FW, with accuracies of 98.1% and 97.0%, respectively. The Waveform feature set, which uses a downsampled version of the entire spike, performs almost perfectly on the classification task (accuracy 99.7%). Given these results, the Waveform feature set is better than the other approaches, at the expense of more computationally demanding training procedures.

Performance with Different MEA Probes

We built simulated spikes using eight different MEA models: five of them are square arrays with varying pitch, and the

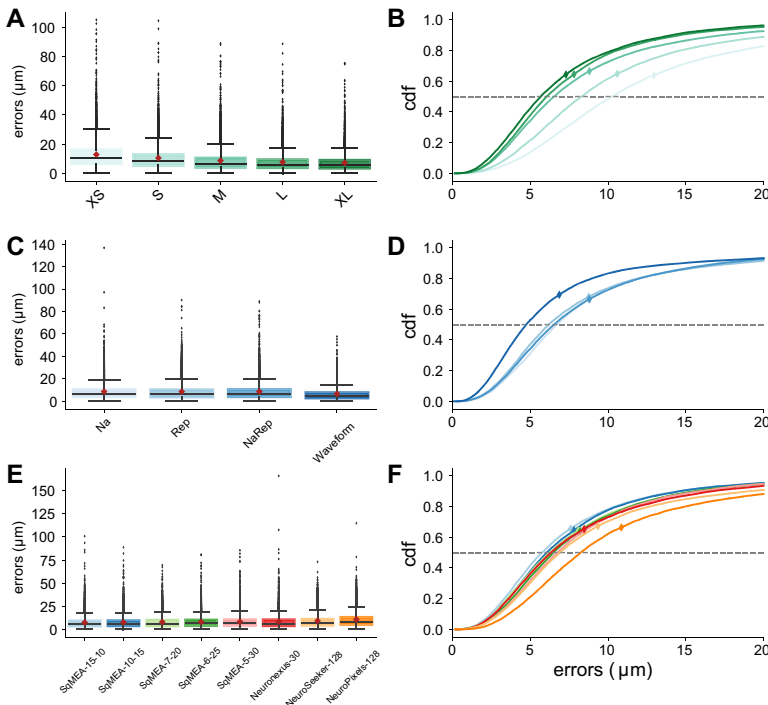


Fig. 8. A, C, and E: error boxplots grouped by convolutional neural network (CNN) size (A), feature type (C), and probe type (E). Red diamonds represent means, and black diamonds denote outliers. B, D, and F: cumulative distribution function (cdf) grouped by CNN size (B), feature type (D), and probe type (F). Gray dashed horizontal line at 0.5 defines the median. Diamonds on each curve show the means (their x values are the average error; the y values are the percentile of their occurrence). The error values are clipped to $20 \mu\text{m}$ to zoom in the distributions. Statistical analyses are shown in Table B4 (for A and B), Table B5 (for C and D), and Table B6 (for E and F).

other three are the NeuroSeeker (Neto et al. 2016), NeuroPixels (Jun et al. 2017) (trimmed to 128 channels), and Neuronexus-32 (clipped to 30 electrodes to make it rectangular) probes. In the following paragraphs, we present the capabilities in terms of neuron localization and classification for the different probes. All simulations shown in this section make use of CNNs of size L, NaRep features for localization, and FW for classification.

Localization. Figure 8, *E* and *F*, show localization errors for the eight different probes (boxplots and cdf). Although an error reduction can be observed from square MEA with 30- μm pitch to 10- μm pitch, as expected, even with a relatively low density (30- μm pitch) a CNN can learn localization models with an average error as low as $8.4 \pm 6.4 \mu\text{m}$ for the SqMEA-5-30. As a comparison, the average error for the probe with 10- μm pitch (SqMEA-15-10) is $7.6 \pm 6.4 \mu\text{m}$. The errors are in the same order also for the Neuronexus probe (mean of $8.5 \pm 7.2 \mu\text{m}$) and for the NeuroSeeker probe (mean of $9.3 \pm 7.7 \mu\text{m}$). When evaluating the performance on 128 sites with the arrangement of the NeuroPixels probe, the average error is $10.8 \pm 8.5 \mu\text{m}$. One may note that even though the NeuroSeeker and Neuronexus probes have lower pitch (NeuroSeeker: $22.5 \mu\text{m}$ in *y*- and *z*-axes; Neuronexus: $18 \mu\text{m}$ in *y*-axis and $25 \mu\text{m}$ in *z*-axis) compared with SqMEA-5-30, their localization error is higher. The reason for this discrepancy might be in the arrangement of the electrodes: while the SqMEA-5-30 has an effective width (considering point electrode contacts) of $120 \mu\text{m}$, for the NeuroSeeker the effective width (considering point electrode contacts) is $67.5 \mu\text{m}$ and for the Neuronexus it is $36 \mu\text{m}$. Hence on the NeuroSeeker and Neuronexus probes there is less spatial information in the *y*-direction, which may explain the reduced localization accuracy.

In general, most comparisons show negligible differences (effect size < 0.2), except for the NeuroSeeker and NeuroPixels probes. The NeuroSeeker probe performs worse than the high-density square MEAs (SqMEA-15-10: effect size = 0.25, SqMEA-10-15: effect size = 0.22), while the NeuroPixels errors show effect sizes above 0.2 in all comparisons (ranging from 0.43 compared with SqMEA-15-10 to 0.3 compared with Neuronexus) except for the comparison with the NeuroSeeker probe (effect size = 0.19). In case of the NeuroPixels probe the checkerboard arrangement might pose additional difficulties, resulting in even lower performance.

Classification. Table B11 in APPENDIX B shows accuracies for classification with different probes. The average accuracies are very high and almost the same for all probes, from a minimum of 96.6% (SqMEA-6-25, SqMEA-15-10) to a maximum of 98.6% (NeuroPixels-128).

Comparison with Other Approaches

In this section, we compare the CNN approach to other state-of-the-art methods. For localization, we used the monopolar, bipolar, and ball-and-stick models described in *Comparison with Other Models* to solve the inverse problem on our simulated data sets. Hence the results obtained for other methods might be different with respect to ones in the literature because the number of cell models, the utilized probes, and the neuron-MEA alignments vary. For characterization of excitatory and inhibitory neurons, we compared with commonly used clustering techniques.

Localization. For localization, we use the validation data set on the SqMEA-10-15 probe. The CNN errors displayed in the plots are obtained with the NaRep feature set and a network of size L. In Fig. 9, we show the errors of the simplified models described in *Comparison with Other Models* and for the CNN method.

We found that the CNN performs significantly better than the inverse approach in all cases, with an average error and SD of $7.8 \pm 6.3 \mu\text{m}$. The large differences between the CNN and the other methods' error distributions are confirmed by the effect sizes: 0.9 for the monopolar approach, 0.68 for the bipolar approach, and 0.87 compared with the ball-and-stick approach (Table B7). Among the models used to solve the inverse problem, the monopolar has a mean error and SD of $21.7 \pm 20.9 \mu\text{m}$, the bipolar model of $15.6 \pm 15.2 \mu\text{m}$, and the ball-and-stick model of $22.6 \pm 23.2 \mu\text{m}$. The better performance of the bipolar model with respect to the monopolar model (and ball-and-stick model) can be due to the fact that it is the only model capable of predicting negative and positive potential values on the MEA. Dendritic branches act as current sources when the soma is depolarized, causing positive deflections in the extracellular potential (Pettersen and Einevoll 2008).

Studying the probability density function (pdf) of the predicted coordinates by different models in Fig. 9, *C–E*, the monopolar model tends to underestimate the distance (*x*-coordinate) from the MEA (note sharp peak in the distribution in Fig. 9C). In the *y*- and *z*-axes, instead the predictions are closer to the center of the MEA when observations lie outside the boundary of the probe (note the different steepness and shape of the monopolar pdf with respect to the true pdf in Fig. 9, *D* and *E*, close to $-100 \mu\text{m}$ and $100 \mu\text{m}$). Similarly, the bipolar model also underestimates distances in the *x*-direction, but the underestimation is less severe. In the *y*- and *z*-directions it nicely follows the true distribution. The ball-and-stick model has distributions very similar to the monopolar model in all three directions. The CNN approach, on the other hand, is the closest match to the true distribution in all three dimensions. Note that the distribution in the *x*-direction is not as uniform as in the *y*- and *z*-directions (the density decreases with increased distances) because we discarded spikes with a peak-to-peak amplitude below $30 \mu\text{V}$ here.

Classification. For excitatory/inhibitory classification we compared the performance of our CNN approach to standard clustering techniques in the FW space (*F*: full-width half-maximum, *W*: width). In Fig. 10A, we show the validation data with the SqMEA-10-15 probe and the excitatory/inhibitory balanced data sets (8,000 observations). Each point is computed from the recording site with largest amplitude. Although it is true that inhibitory cells cover the bottom left part of the cloud (narrower width and full-width half-maximum) and excitatory cells the top right (wider spike shape), we can observe that there is some overlap between the two groups. When we apply *k*-means clustering (Fig. 10B), the algorithm correctly assigns the bottom left part to inhibitory neurons and the top right part to excitatory neurons, but the overlap is mainly assigned to the excitatory class. This yields an accuracy of 99.9% for the excitatory class but only 60.7% for the inhibitory one, with an average of 80.3%. When it comes to the MoG, the data are fit to two multivariate Gaussians and labels are assigned based on the probability of an observation to belong

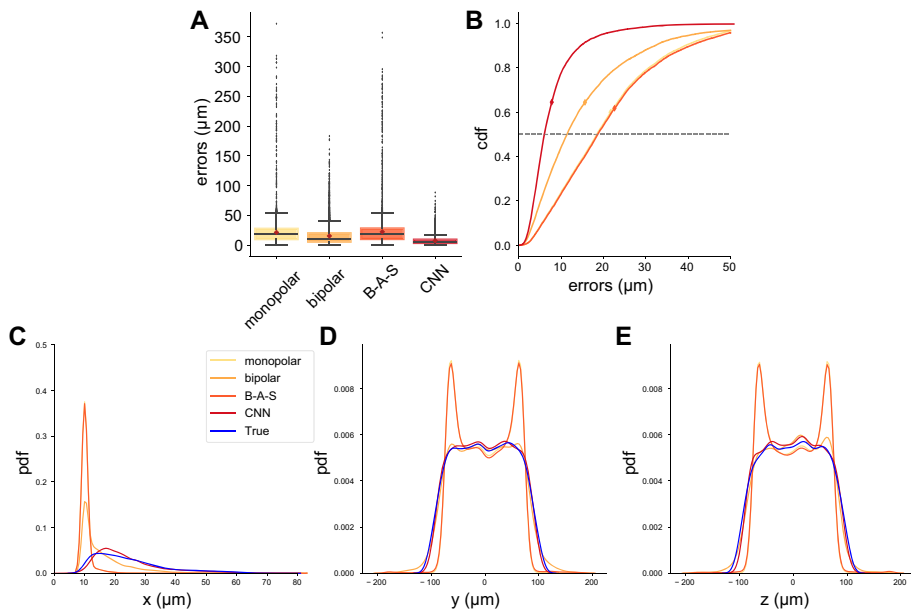


Fig. 9. *A*: error boxplots grouped by model used to solve the inverse problem. Red diamonds are means; black diamonds are outliers. B-A-S, ball-and-stick; CNN, convolutional neural network. *B*: cumulative distribution function (cdf) grouped by localization method. Gray dashed horizontal line at 0.5 defines the median. Diamonds on each curve show the means (their x values are the average error; the y values are the percentile of their occurrence). The error values are clipped to $50 \mu\text{m}$ to zoom in the distributions. *C-E*: probability density functions (pdfs) of predicted values by different models and true values in the x (*C*)-, y (*D*)-, and z (*E*)-directions. Statistical analysis is shown in Table B7.

to the two distributions. Figure 10C shows the estimated Gaussians (ellipses) and the labeling of the points. Although the MoG is capable of describing the diagonal shape of the excitatory cloud, the overlap between the observations cannot be untangled, resulting in an accuracy of 98.8% for the excitatory class and 54.2% for the inhibitory one, with an average of 76.5%. The CNN method, instead, is able to discern the overlap in the FW space. This is certainly due to its higher complexity, due both to the method itself and to the use of all electrodes' information, not only those with highest amplitude. The CNN result shown here (FW feature set, size L) allows us to correctly predict excitatory cells in 98.9% of the cases and inhibitory cells with an accuracy of 97.1%. This makes it the best-performing method among those compared here, with an overall accuracy of 98.0%. It could be argued that the comparison was somewhat unfair, as our CNN approach considers F and W images (computed on all recording sites), while the clustering is performed with values computed from the electrode with highest amplitude only. Nevertheless, it is not common practice to consider waveforms on all electrodes but only on the one with highest amplitude (Barthó et al. 2004; Peyrache et al. 2012).

m-Type Classification

In addition to separating excitatory cells from inhibitory ones by trained CNNs, we tried to make a finer subdivision and classify cells into morphology classes (m -type) based on the EAP. The approach is similar to that for excitatory/inhibitory classification, but instead of only 2 output classes we take the

11 m -type classes (cells of m -type BP and NGC are excluded, since only 1 morphology is available in the data set). We use a CNN of size L and consider Waveform features (in this case with a downsampling factor of 8, i.e., a sampling frequency of 4 kHz) on the SqMEA-10-15 with the Physrot data set. The resulting confusion matrix E , in which each entry E_{ij} represents the amount of observations of the true cell type i predicted as cell type j , is depicted in Fig. 11. We do not observe a striking diagonal, indicating that full identification of all cell types is not feasible from EAPs. But it is noteworthy that there is some block structure dividing excitatory neurons from inhibitory neurons. This division is learned intrinsically by the network, and inhibitory cells are classified within the inhibitory block in 100.0% of the cases and excitatory cells within the excitatory block in 95.7%. Concerning the mixing of TTPC1 and TTPC2, we do not expect to be able to differentiate between these two types because their only difference is the distance of the bifurcation point of the apical dendrite to the soma. Since the MEA is located close to the somatic region, recordings might not be sensitive to this delicate difference. Disregarding this mixing, the m -type classification performs well (chance would be 9.1%) on excitatory cells and inhibitory Martinotti cells (MC) (80.5%). Note that these well-classified cells make up a large proportion of cortical cells. The overall accuracy of 34.0% illustrates that the morphological details are partially resolved by the CNN. In cases in which the CNN is not able to extract the information about the morphological details, it is unclear whether the information is present at all in the EAP or an increased number of cell models could solve the problem. In

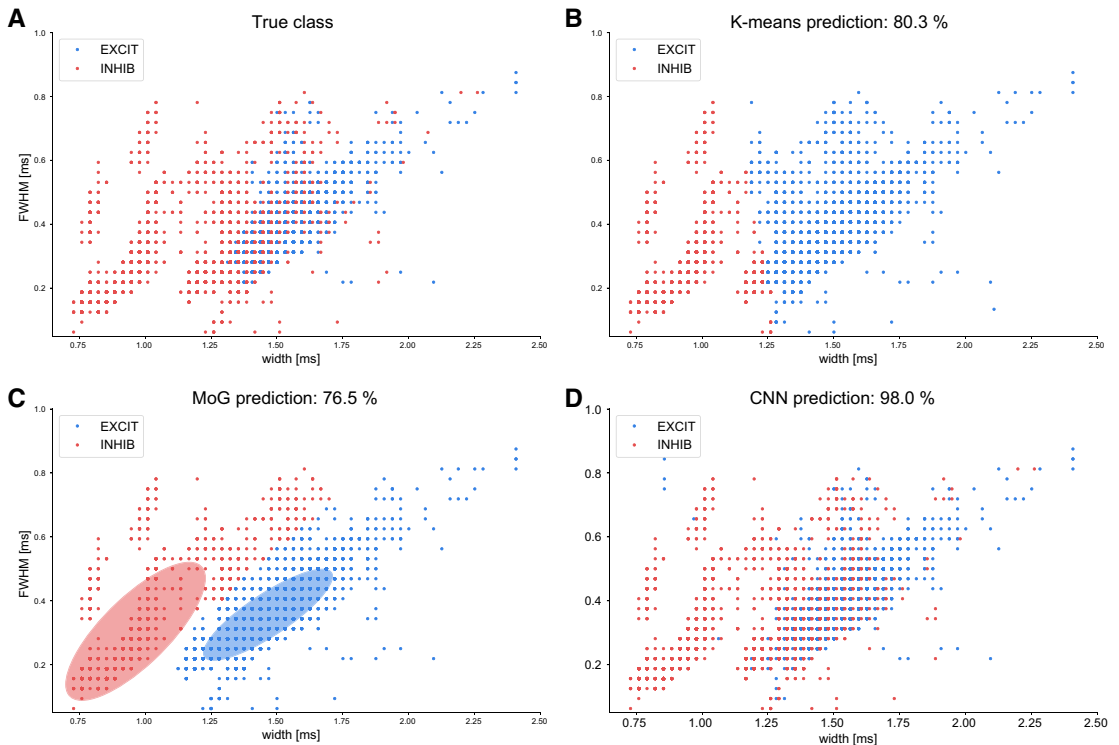


Fig. 10. Comparison with standard clustering approaches for excitatory/inhibitory classification. Each point is 1 spike in the *FW*-plane; red dots are inhibitory cells, and blue dots are excitatory cells. *A*: true excitatory/inhibitory classes: it is evident that there is some overlap between the groups in the center of the plot. *B*: *k*-means prediction: *k*-means clustering splits the data in 2 groups and cannot untangle the overlap (accuracy: 80.3%). *C*: mixture of Gaussians (MoG) prediction: MoG clustering correctly finds the diagonal shape of the excitatory population, but it cannot separate the overlap either (accuracy: 76.5%). *D*: convolutional neural network (CNN) prediction: the CNN output is able to correctly predict almost all the observations, with an accuracy of 98.0%. FWHM, full-width half-maximum.

conclusion, the results show promise for a more refined classification than only distinguishing excitatory cells from inhibitory cells.

Validation on Different Models

To investigate how general the trained CNN models are, we tested the performance of localization and classification on simulated EAPs from other neuronal models, namely, the cell model from Hay et al. (2011) and the models from the Allen Brain Institute (ABI) cell type database (Gouwens et al. 2018; <http://celltypes.brain-map.org>). For the following results, we used the SqMEA-10-15 probe, CNNs of size L, and NaRep and *FW* feature sets for localization and classification, respectively.

Hay model. The Hay cell models a neocortical pyramidal cell from L5b, and the techniques used to build the models were similar to the models from the NMC Portal. Therefore, we expect a relatively good performance in localization and classification with the CNNs trained on our standard NMC data sets. We built a Physyrot data set of Hay cells consisting of 1,000 observations at random locations around the probe as described in *Simulated Recordings*, and we then evalu-

ated the performance of the CNNs in localization and classification.

For localization, the average error on the Hay data set is $8.7 \pm 6.6 \mu\text{m}$, perfectly in line with the average errors of TTPC models in the NMC validation data set (Fig. 7A). The average error over all cell types in the NMC validation data set is $7.8 \pm 6.3 \mu\text{m}$. For classification, we obtain an average accuracy of 76.4%, while the accuracy on the NMC validation set is 98.0%. The lower accuracy could be due to the fact that the Hay model includes other types of mechanisms, such as active calcium channels in the apical dendrites, that are not modeled in the NMC cell models.

Allen Brain Institute models. The cell models from the ABI that we selected are quite different from the NMC cell models at least for two reasons. First, the ABI neurons are from mice, whereas the NMC cells are from juvenile rats. Second, they are from visual cortex (19 cells) and postprimal area (1 cell), whereas the NMC models are from somatosensory cortex. With CNNs trained on NMC data are expected to have lower accuracy when applied to the ABI data. We generated 1,000 EAPs for each of the 20 ABI cell models, according to the description in *Simulated Record-*

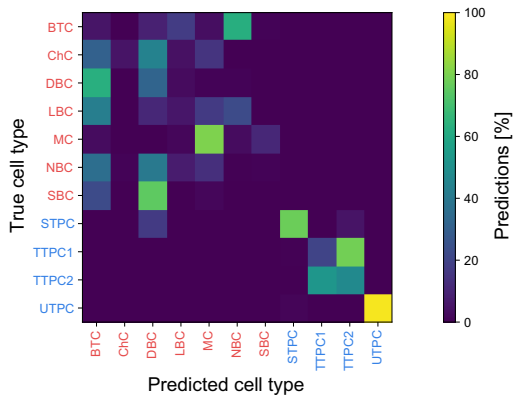


Fig. 11. Normalized confusion matrix of m-type classification based on Waveform features (here with downsampling factor 8, i.e., 28 sample points) for Physrot data set. The convolutional neural network size is L and extracellular action potentials are taken for the SqMEA-10-15 probe. BTC, bitufted cells; ChC, chandelier cells; DBC, double bouquet cells; LBC, large basket cells; MC, Martinotti cells; NBC, nest basket cells; SBC, small basket cells; STPC, slender-tufted pyramidal cells (PC); TTPC1, thick-tufted PC with late bifurcating apical tuft; TTPC2, thick-tufted PC with early bifurcating apical tuft; UTPC, untufted PC.

ings. We used the 3drot alignment because of the variability in the ABI cell models' orientation (for details about selection of cell models see APPENDIX A).

We ran the 3drot CNN for localization on the ABI data set, obtaining an average error of $19.3 \pm 11.5 \mu\text{m}$, larger than the $8.9 \pm 8.2 \mu\text{m}$ obtained on the NMC validation set as expected. For classification, we distinguished excitatory and inhibitory cells in the ABI data set based on mouse transgenic lines (details in APPENDIX A). With the CNN for classification trained on NMC models, the average accuracy is 76.9%, while it is 97.6% on the NMC validation data set.

Since the cell models of the ABI come from a different species and are from a different cortical region, we trained a CNN on this data set only—16 models are used for training and 4 for validation (APPENDIX A). We used a CNN of size L and NaRep features, obtaining a localization average error of $5.9 \pm 4.5 \mu\text{m}$, which is in line with the performance we obtained on NMC models only. We did not run classification with so few models (only 20 cell models in total), because the CNNs need a larger diversity to find general features related to excitatory-inhibitory types (using the NMC data we trained on 192 cell models).

Test on Experimental Data

Although the method proposed in this report is at a proof-of-concept stage, we tested some CNNs trained with simulated data on experiments at least for plausibility.

We decided to use data (publicly available at <http://www.kampff-lab.org/validating-electrodes>) from paired juxtacellular and extracellular recordings (Neto et al. 2016) where, to a certain extent, the ground-truth location is known. The extracellular signals are measured with either the Neuronexus or the NeuroSeeker probe. Taking the amplitude threshold of our CNN training simulation (peak-to-peak amplitude of $30 \mu\text{V}$ on at least 1 electrode) into account and considering only

cells in front of the MEA, we were left with 10 data sets (see APPENDIX A for further details). After performing juxtacellular-triggered averaging, we fed the average EAP waveform into the CNN and predicted the soma position. The CNNs were trained with simulated data having the appropriate geometric alignment (MEA probes are rotated by -48.2° along the y-axis). On average the prediction error is $42.2 \pm 16.8 \mu\text{m}$, assuming the true soma position is the tip of the juxtacellular probe. The experimentally determined positions for the x-, y-, and z-coordinates range from $27 \mu\text{m}$ to $129 \mu\text{m}$, $-48 \mu\text{m}$ to $6 \mu\text{m}$, and $-121 \mu\text{m}$ to $21 \mu\text{m}$, respectively. Neto et al. (2016) report a distance uncertainty of $10.5 \pm 5.2 \mu\text{m}$. This uncertainty only applies to the tip position of the juxtacellular probe with respect to the MEA, but it is a drastic assumption to consider this position equal to the soma position (center of the soma). Without neglecting the soma diameter, one might favor a larger distance error by adding a soma radius uncertainty of $10\text{--}20 \mu\text{m}$. Furthermore, the uncertainty of $10.5 \pm 5.2 \mu\text{m}$ reflects misalignment errors caused by the manipulators used in the experiment and is investigated under free conditions, i.e., without entering any brain tissue. Additional misalignment originating from entering brain tissue with the probes or from the brain's pulsation due to breathing of the mouse are not taken into account.

Figure 12 shows predicted vs. true coordinates. Figure 12A demonstrates that the predicted soma distances (x-coordinate) are in a plausible range. Overall, the distance x is predicted with a mean error of $20.3 \pm 16.1 \mu\text{m}$. The y- and z-positions are in the same range of precision ($22.3 \pm 13.1 \mu\text{m}$ and $22.9 \pm 15.7 \mu\text{m}$, respectively, on average). Note that the horizontal error bars in the plots only represent the uncertainty due to the misalignment of the manipulators as reported by Neto et al. (2016) and all other previously mentioned uncertainties (which are not quantified) are not considered.

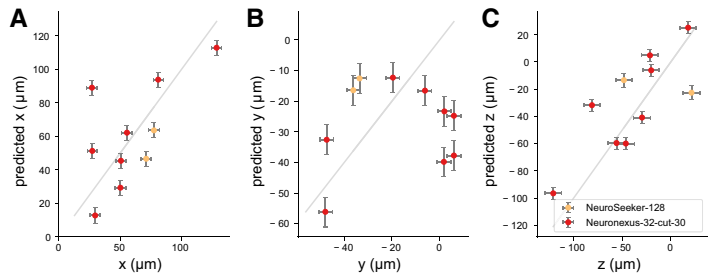
DISCUSSION

This work provides a deep learning approach for neuron localization and classification based on MEA recordings. We simulated in vivo-equivalent EAPs and built data sets for various probe designs, using a multitude of cell models from the NMC Portal (205 cell models from Ramaswamy et al. 2015). CNN models trained on these simulated spikes predict the soma position of the neuron and characterize whether it is excitatory or inhibitory. The accuracy depends on the neuron-MEA alignment, the specific cell types, the CNN size itself, and the input feature sets. For completeness, we compared the proposed method with existing strategies regarding both localization and classification of recorded spikes, we validated on cell models from other databases, and we tested the models on publicly available experimental data.

Localization

We showed that the CNN method is robust and accurate in predicting the 3D soma location from spikes generated by neurons with a physiological neuron-MEA alignment (Physrot, defined in *Neuron-MEA alignment*). The average errors are on the order of $7.6\text{--}11.7 \mu\text{m}$ for all probes involved in the study (Fig. 8E). We demonstrated the CNN approach to be robust with different cell models and to be able to generalize among cell types not used for training (BP and NGC). Finally, local-

Fig. 12. Soma position predictions of a convolutional neural network (CNN) based on experimental extracellular action potential (EAP) recordings. Experimental data are from paired juxtacellular-extracellular recordings (Neto et al. 2016) where the position of the soma is associated with the tip position of the juxtacellular probe. The CNN (size L, NaRep feature) is trained on simulated (3drot) EAP signals. Error bars are CNN prediction uncertainties for the predicted coordinates and $4.2 \mu\text{m}$, $2.8 \mu\text{m}$, and $8.5 \mu\text{m}$ (misalignment uncertainties reported by the experimenters of Neto et al. 2016) for true x -, y -, and z -coordinates, respectively.



ization performances achieved with our approach are significantly better than solving the inverse problem with various generative models. With a SqMEA-10-15 data set the total error in three dimensions was $21.7 \pm 20.9 \mu\text{m}$ for the monopolar current source, $15.6 \pm 15.2 \mu\text{m}$ for the bipolar current source, and $22.6 \pm 23.2 \mu\text{m}$ for the ball-and-stick model, whereas with our CNN approach we obtained an error of $7.8 \pm 6.3 \mu\text{m}$ (as shown in Fig. 9).

In a recent study (Delgado Ruz and Schultz 2014), a NeuroNexus-32 probe was used (shown in Fig. 3) with a modified ball-and-stick model to solve the inverse problem. For the five cell types considered in Delgado Ruz and Schultz (2014), they reached average errors of $6.26 \pm 6.10 \mu\text{m}$, $6.03 \pm 7.68 \mu\text{m}$, and $2.58 \pm 4.75 \mu\text{m}$ along the x -, y -, and z -axes, respectively. Using the same probe on our Physrot data set, we obtained with CNNs average errors of $4.1 \pm 4.5 \mu\text{m}$, $4.3 \pm 4.7 \mu\text{m}$, and $4.3 \pm 5.3 \mu\text{m}$ for the x -, y -, and z -axes, respectively (with NaRep feature set and size L).

Classification

The deep learning method was applied to excitatory/inhibitory classification with accuracies above 96.6% for all employed MEA models using the *FW* feature set and a CNN size L. An almost perfect outcome of 99.7% was obtained with the Waveform features on the SqMEA-10-15 probe. Compared with standard strategies using spike widths extracted from the spike shape, it showed a significant improvement (k -means clustering: 80.3%, MoG: 76.5%; *Comparison with Other Approaches*). We also attempted to distinguish among 11 cell morphologies (m-type classification). The overall accuracy of 34.0% is substantially better than the chance level of 9.1%. It is interesting to see that m-type classification performs a sort of unsupervised learning, as inhibitory cells were classified as inhibitory in 100.0% of the cases and excitatory cells as excitatory 95.7% of the time.

Overfitting and Stability

When evaluating the predictions of our CNNs on the validation data set, we observed a drop in accuracy compared with training accuracy. The drop is in an acceptable range for excitatory/inhibitory classification (0–3% with respect to the training accuracy) and localization (up to $3.7 \mu\text{m}$ prediction error increase). In case of m-type classification, the validation accuracy drops ~65% compared with training accuracy, clearly indicating overfitting. Since we do not have enough diversity in cell model data to build a third data set for implementing early-stopping regularization (i.e., stop training as soon as the

generalization error increases), we tracked the evaluation accuracies depending on the number of training epochs. In most cases, they reached a plateau after roughly 2,000 training epochs and did not decrease significantly afterwards, while training accuracies still increased. Therefore, we decided to stop training after 2,000 training epochs, assuming that the CNN has extracted most of the generalizable information provided by the EAPs at that point. Moreover, we tried to quantify the stability of the performance depending on different initial weights before the optimization process. To do so we ran the CNN training for localization and classification (on the SqMEA-10-15 probe, CNN size L, and with NaRep and *FW* features, respectively) six times with different random seeds. We obtained an average mean error of $7.6 \pm 0.1 \mu\text{m}$ with an average SD of $6.3 \pm 0.2 \mu\text{m}$ for localization (including the BP and NGC models) and an average mean accuracy of 97.9% with a SD of 0.2% for classification, indicating that performance is not dependent on the initial conditions of network weights and the convergence is robust.

Model-Based Approach

The findings presented in this study are based on simulations. Although this might be regarded as a limitation, we want to stress that the proposed method makes use of highly detailed cell models (Markram et al. 2015) and the complexity of such models is maintained and learned by CNNs. Previous approaches to localization and/or classification relied on simple forward models to solve the inverse models—monopolar, bipolar, ball-and-stick models, etc. (Blanche et al. 2005; Delgado Ruz and Schultz 2014; Somogyvári et al. 2012). We showed that CNNs outperform these models in estimating the soma positions. Another point that plays in favor of the use of neural simulations is the difficulty in gathering ground-truth data experimentally. Localizing and classifying neurons in real recordings requires advanced and highly accurate equipment, and the recorded labeled data would most likely still not be sufficient to train data-hungry machine learning algorithms such as CNNs. Nevertheless, validation on experimental data is definitely a required step and will be based on combined approaches with paired electrophysiological recordings and standard microscopy (Neto et al. 2016), or even involving more sophisticated and precise imaging techniques, such as two-photon imaging (Göbel and Helmchen 2007), which was paired with electrophysiological recordings *in vivo* in Shew et al. (2010). Paired electrophysiology and two-photon microscopy data, possibly in combination with intracellular voltage monitoring through patch clamping or voltage-sensitive dyes, could also represent a valuable tool to further validate and

improve the forward modeling schemes, providing morphological, intracellular, and extracellular recordings simultaneously.

Another advantage of using forward modeling is that the performance of the machine learning algorithm could be improved by building case-specific data sets that better match the real experimental scenarios. In this work, we assumed that the simulated probe was inserted in L5 of somatosensory cortex of a juvenile rat with a vertical insertion angle. However, somatosensory cortex can present large differences with respect to other brain regions (e.g., hippocampus, cerebellum, or other cortical regions) but also among animal species. Therefore, we do not envision a single universal model to localize and classify neurons but species- and brain area-specific CNNs to accurately deal with variability in neuronal types and functions. For example, when we fed mouse data from the ABI database, the localization CNN trained on rat data performed relatively poorly ($19.3 \pm 11.5 \mu\text{m}$; *Allen Brain Institute models*), but trained on mouse models the performance is in line with what we obtained on rat data ($5.9 \pm 4.5 \mu\text{m}$).

Effect of Probe Design

Regarding neural probes, a forward modeling-based approach can give important insights for the design and manufacturing of next-generation probes. For example, our results showed that even relatively low-density probes, such as the SqMEA-5-30, despite performing slightly worse than higher-density probes, still yield high accuracy in localizing and classifying neurons. Potentially, the pursuit of extremely high-density probes, which makes the design complicated and the data throughput very high, is not required for classification and localization tasks [although it might still be important for spike sorting (Franke et al. 2012; Rossant et al. 2016)]. However, for such simulation-driven MEA design, the simulations lack a more accurate electrode model considering finite size recording sites (in this work we used an ideal point electrode), electrode impedances, and transfer functions.

Future Extensions

The generative model for spike simulations could be improved in various ways. A straightforward improvement to obtain more accurate simulations could be including the MEA scar in the data generation, by clipping or bending neuronal branches in the proximity of the probe before simulating the recordings. Another refinement might be to take into account the finite size effects of the electrode contacts by means of the disk-electrode approximation (Lindén et al. 2014), which was shown to be appropriate for current sources positioned at distances larger than the contact radius (Ness et al. 2015). Moreover, here we assumed a tissue with homogeneous and isotropic electrical properties, but experimental findings suggest that in the cortex the conductive properties of the extracellular space are anisotropic (Goto et al. 2010). Anisotropy could be easily taken into account for the simulation of spikes (Ness et al. 2015; Pettersen et al. 2012). As the proposed approach strongly relies on high-fidelity simulations that reliably describe the neuron dynamics and volume conduction, another strategy could be using finite element method-based models, as in Agudelo-Toro and Neef (2013), Pods et al. (2013), and Tveito et al. (2017), which would result in more detailed simulations at the cost of a much higher computational

cost for data generation. Another layer of modeling is the electrode-tissue interface. The generated data should include electrical properties of the electrodes, such as the impedance, and account for their variability in experimental scenarios. In this work, we used polytrodes with a relevant size with respect to the neuron: although we assumed a homogeneous medium, the presence of the probe itself represents an obstacle for electrical signal propagation and can be modeled with either finite element method or analytical simplifications, such as the method of images (Ness et al. 2015).

In this work, we did not include any noise in the simulated recordings. The rationale behind this choice is that sorted spikes can be cleaned by applying spike-triggered averaging. With spike-triggered averaging, additive random noise is reduced by a factor of \sqrt{N} , where N is the number of occurrences of the sorted unit. Moreover, a common problem in spike sorting is electrode drift, in which the relative position between a neuron and the recording electrodes changes during the experiment. If drifting is detected from the spike sorting algorithm, one could feed different averaged EAPs computed in separate time windows and evaluate the drift over time, similarly to Delgado Ruz and Schultz (2014), in which windows of 5 min were used to compute the mean EAP.

Furthermore, the recording site area affects the amount of noise in the recordings, as the recording area is related to the impedance of the electrode. Here we assumed perfectly sorted spikes, from which a clean EAP can be computed. Clearly, with experimental data errors in spike sorting would affect the performance of localization and classification due to distorted waveforms from wrong assignments.

Outlook

Precise neural localization and classification from in vivo extracellular recordings has the potential of making electrophysiology an even more powerful technique to interact with neural tissue. Rather than only extracting spike trains, we could build a 3D representation of the recorded units and perform functional electrical imaging to study the spatial interactions among different cell types in neural microcircuits. On top of this, a precise localization of neuronal somata might enable the use of highly selective electrical stimulation patterns (Buccino et al. 2016) and represent an advancement in single-neuron stimulation from extracellular probes.

We strongly believe that computational approaches must go hand in hand with experimental ones, and an extension of this work might include the simulations of the entire pipeline from simulated MEA recordings, for example, with VISAPy (Hagen et al. 2015), to electrical stimulation including spike sorting, localization, classification, electrical stimulation, and evaluation of its effect on detailed neural morphologies.

APPENDIX A: DATA SELECTION

Neocortical Microcircuit Collaboration Portal Data Set

In this appendix we discuss the data set and the modifications that we applied to make sure that that training and validation set are completely disjointed.

In the original data set (<https://bbp.epfl.ch/nmc-portal/welcome>; L5 cells) there are nine inhibitory neuron types: BP, bitufted cells (BTC), ChC, double bouquet cells (DBC), LBC, MC, NBC, NGC, and SBC.

The four excitatory types, i.e., the PCs, are grouped into STPC, TTPC1, TTPC2, and UTPC. While belonging to the same m-type, neurons can have different electrophysiology properties (e-type) based on their firing patterns (Markram et al. 2015). In L5 the e-types are categorized into continuous accommodating (cAC), continuous stuttering (cSTUT), burst accommodating (bAC), burst stuttering (bSTUT), continuous nonaccommodating (cNAC), delayed stuttering (dSTUT), burst nonaccommodating (bNAC), continuous irregular (cIR), delayed nonaccommodating (dNAC), burst irregular (bIR), and continuous adapting (cAD). Since not all m-types express all e-types, the combination of morphological and electrical type gives rise to 52 morpho-electrical types (me-types) in L5. For each me-type, the NMC database contains five cell models; therefore, there are a total of 260 cell models in the data set.

In Markram et al. (2015), to extend the number of reconstructed models, an algorithm is used to clone morphologies: neural compartments are randomly scaled and rotated with respect to each other. Moreover, morphologies are also stretched and shrunk to make up new morphologies. We identified 54 different morphologies in the data set, listed in the Supplemental Material for this article. Although the cloned and/or scaled morphologies are indeed different than the original ones, their shape is quite similar.

The use of CNNs, which are among the most powerful machine learning algorithms, pushed us to pay particular attention in the training-validation splitting so that no information of the validation set is present in the training set (leakage). Hence, the presence of a cloned/scaled version of the same morphology in both training and validation has been avoided. We selected training and validation cell models so that all morphologies in the validation set are unique. In doing so, we had to remove all instances of BP and NGC from the training set, as all the models are derived from the same reconstructed morphology. For localization and excitatory/inhibitory classification, we kept a BP and an NGC model in the additional validation set.

After the manipulation, the training set consists of 192 cell models, while the validation set only contains 11 cell models, one for each m-type. Moreover, we use one BP and one NGC model, not used for training, as further validation. In total, we included 205 neuronal models out of the available 260. The cell models are listed in the Supplemental Material.

Allen Brain Institute Data Set

From the Allen Brain Institute cell type portal (<http://celltypes.brain-map.org/data>), we selected cell models according to three criteria: 1) cells were from mice, 2) cells were from L5 (to maintain consistency with the data from the NMC Portal), and 3) cells had an all-active model. This search reduced the number of cell models available to 42. During the simulation process, we further discarded 22 models based on two extra rules: 1) if adjusting the current-clamp amplitude to the soma could not induce a number of spikes between 10 and 30 in 10 iterations (in which the weight was multiplied by 0.75 if the number of intracellular spikes was >30 and by 1.25 when <10 spikes were detected) and 2) if <5 EAP peaks had a peak-to-peak threshold of $30 \mu\text{V}$ in 500 random positioning of the neuron around the probe (meaning that the EAPs were mainly below the defined detection threshold). After this pruning, 20 cell models are left. To distinguish between excitatory and inhibitory cells, we used the transgenic line information: *Pvalb*, *Sst*, *Htr3*, and *Gad2* lines were considered inhibitory; *Rbp4*, *Senn*, and *Rorb* were considered excitatory (Gouwens et al. 2018). After this division there were 11 inhibitory and 9 excitatory cell types.

To avoid overfitting, we randomly selected 4 models, 2 excitatory and 2 inhibitory, and we set them aside for validation, while we used the remaining 16 neuronal models for training.

The cell models are listed in the Supplemental Material.

Kampff Laboratory Data Set

Accompanying their article on paired juxtacellular-extracellular recordings (Neto et al. 2016), the laboratory of Adam Kampff publicly offers the data on <http://www.kampff-lab.org/validating-electrodes>. To extract an averaged extracellular waveform for each cell that can be fed into a trained CNN, some data processing was necessary. First, we detected spikes in the juxtacellular probe by thresholding the signal to get the cell's spike times. Second, we high-pass filtered the extracellular MEA recording with a third-order Butterworth filter in forward-backward mode with a band pass of 100–14,250 Hz. Afterwards, we averaged the EAP in windows of 7 ms around the spike times (2 ms preceding and 5 ms after the peak). This average waveform was then referred to as the juxtacellular-triggered average and was used as input for CNN predictions. After this preprocessing, 10 of the 29 available data sets fulfilled the criteria of having a peak-to-peak amplitude of $30 \mu\text{V}$ on at least one electrode and being in front of the extracellular probe (2014_03_26: Pair 2.0, 2014_03_20: Pair 3.0, 2014_03_26: Pair 2.1, 2014_10_17: Pair 1.1, 2014_10_17: Pair 1.0, 2014_11_25: Pair 3.0, 2014_11_25: Pair 1.0, 2014_11_25: Pair 2.0, 2015_09_04: Pair 5.0, 2015_09_03: Pair 6.0). These 10 data sets were used in *Test on Experimental Data* to test our deep learning approach.

APPENDIX B: ADDITIONAL INFORMATION

This appendix contains additional information on parameters and results.

Table B1 shows the specific CNN parameters for different network sizes.

The average localization errors and SDs for different rotational data sets are contained in Table B2, and the corresponding statistical analysis is depicted in Table B3.

Further results on significant differences and effect sizes of localization performances for different CNN sizes, features, MEA probes, and localization methods are listed in Tables B4, B5, B6, and B7, respectively.

The excitatory/inhibitory classification accuracies grouped by cell type for different rotational data sets, CNN sizes, feature sets, and MEA probes are shown in Tables B8, B9, B10, and B11, respectively.

ACKNOWLEDGMENTS

We thank Joana Neto and colleagues from the Kampff laboratory for helping with analysis of the experimental data.

GRANTS

The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through Grant INST 39/963-1 FUGG. Furthermore, the research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement 600925 (NeuroSeeker) and the European Union Horizon 2020 Research and Innovation Programme under

Table B1. CNN sizes

Size	k_1	d_1	k_2	d_2	n_{FC}
XS	2	4	2	8	128
S	2	8	2	16	256
M	3	16	3	32	512
L	3	32	3	64	1,024
XL	3	64	3	128	2,048

Convolutional neural network (CNN) parameters of the different network sizes: layer 1 convolutional kernel size k_1 , layer 1 convolutional kernel depth d_1 , layer 2 convolutional kernel size k_2 , layer 2 convolutional kernel depth d_2 , and nodes in fully connected layer n_{FC} .

Table B2. Localization errors grouped by rotational data set

Data Set	x Error	y Error	z Error	Total Error
Norot	3.9 ± 3.9	3.0 ± 3.2	3.9 ± 4.6	7.3 ± 5.7
Physrot	3.8 ± 3.9	3.5 ± 3.8	4.3 ± 5.1	7.8 ± 6.3
3drot	4.4 ± 4.9	4.5 ± 5.7	4.4 ± 5.6	8.9 ± 8.2

Values (in μm) are average \pm SD errors along x, y, and z dimensions and total errors grouped by rotational data sets. The average of total error is computed over the 3-dimensional distances and is not derived from the mean x, y, and z errors.

Grant Agreement 720270 [Human Brain Project (HBP) SGA1]. In addition, this work was promoted by the German Academic Exchange Service (DAAD), funded by the Federal Ministry of Education and Research (BMBF; Grant BFNT 01GQ0830), and supported by the Carl Zeiss Stiftung. A. P. Buccino (PhD fellow) and M. Fyhn, P. D. Häfliger, G. Cauwenberghs, and G. T. Einevoll (principal investigators) are part of the Simula-UCSD-University of Oslo Research and PhD training (SUURPh) program, an international collaboration in computational biology and medicine funded by the Norwegian Ministry of Education and Research.

DISCLOSURES

No conflicts of interest, financial or otherwise, are declared by the authors.

AUTHOR CONTRIBUTIONS

A.P.B., M.K., T.V.B.N., S.R., and G.T.E. conceived and designed research; A.P.B., M.K., and T.V.B.N. performed experiments; A.P.B. and M.K. analyzed data; A.P.B. and M.K. interpreted results of experiments; A.P.B. and M.K. prepared figures; A.P.B. and M.K. drafted manuscript; A.P.B., M.K., T.V.B.N., B.M., P.D.H., M.F., G.C., S.R., and G.T.E. edited and revised manuscript; A.P.B., M.K., T.V.B.N., B.M., P.D.H., M.F., G.C., S.R., and G.T.E. approved final version of manuscript.

REFERENCES

- Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mane D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viegas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. TensorFlow: large-scale machine learning on heterogeneous distributed systems (Preprint). ArXiv 1603.04467, 2016.
- Agudelo-Toro A, Neef A. Computationally efficient simulation of electrical activity at cell membranes interacting with self-generated and externally imposed electric fields. *J Neural Eng* 10: 026019, 2013. doi:10.1088/1741-2560/10/2/026019.
- Anastassiou CA, Perin R, Buzsáki G, Markram H, Koch C. Cell type- and activity-dependent extracellular correlates of intracellular spiking. *J Neurophysiol* 114: 608–623, 2015. doi:10.1152/jn.00628.2014.
- Barthó P, Hirase H, Monconduit L, Zugaro M, Harris KD, Buzsáki G. Characterization of neocortical principal cells and interneurons by network interactions and extracellular features. *J Neurophysiol* 92: 600–608, 2004. doi:10.1152/jn.01170.2003.
- Berdondini L, Bosca A, Nius T, Maccione A. Active pixel sensor multi-electrode array for high spatiotemporal resolution. In: *Nanotechnology and Neuroscience: Nano-electronic, Photonic and Mechanical Neuronal Inter-*

Table B3. Localization by data set rotation: statistical analysis

>	Norot	Physrot	3drot
Norot		ns	ns
Physrot	0.09***		ns
3drot	0.21***	0.13***	

Statistical analysis for localization errors grouped by rotational data set. Each entry shows Cohen's *d* and significance of the test column group < row group. ****P* < 0.001. ns, Not significant.

Table B4. Localization by CNN size: statistical analysis

>	XS	S	M	L	XL
XS		0.26***	0.48***	0.63***	0.71***
S	ns		0.23***	0.38***	0.47***
M	ns	ns		0.14***	0.22***
L	ns	ns	ns		0.09***
XL	ns	ns	ns	ns	

Statistical analysis for localization errors grouped by convolutional neural network (CNN) size. Each entry shows Cohen's *d* and significance of the test column group < row group. ****P* < 0.001. ns, Not significant.

- facing, edited by Di Vittorio M, Martiradonna L, Assad J. New York: Springer, 2014, p. 207–238. doi:10.1007/978-1-4899-8038-0_7.
- Blanche TJ, Spacke MA, Hetke JF, Swindale NV. Polytrodes: high-density silicon electrode arrays for large-scale multiunit recording. *J Neurophysiol* 93: 2987–3000, 2005. doi:10.1152/jn.01023.2004.
- Buccino AP, Stöber T, Næss S, Cauwenberghs G, Häfliger P. Extracellular single neuron stimulation with high-density multi-electrode array. *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. Shanghai, China, October 17–19, 2016, p. 520–523.
- Carnevale NT, Hines ML. *The NEURON Book*. Cambridge, UK: Cambridge Univ. Press, 2006. doi:10.1017/CBO9780511541612.
- Chelaru MI, Jog MS. Spike source localization with tetrodes. *J Neurosci Methods* 142: 305–315, 2005. doi:10.1016/j.jneumeth.2004.09.004.
- Cohen J. A power primer. *Psychol Bull* 112: 155–159, 1992. doi:10.1037/0033-2909.112.1.155.
- DeFelipe J, Ballesteros-Yáñez I, Inda MC, Muñoz A. Double-bouquet cells in the monkey and human cerebral cortex with special reference to areas 17 and 18. *Prog Brain Res* 154: 15–32, 2006. doi:10.1016/S0079-6123(06)54002-6.
- Delgado Ruz I, Schultz SR. Localising and classifying neurons from high density MEA recordings. *J Neurosci Methods* 233: 115–128, 2014. doi:10.1016/j.jneumeth.2014.05.037.
- Fortin FA, De Rainville FM, Gardner MA, Parizeau M, Gagné C. DEAP: evolutionary algorithms made easy. *J Mach Learn Res* 13: 2171–2175, 2012.
- Foust A, Popovic M, Zecovic D, McCormick DA. Action potentials initiate in the axon initial segment and propagate through axon collaterals reliably in cerebellar Purkinje neurons. *J Neurosci* 30: 6891–6902, 2010. doi:10.1523/JNEUROSCI.0552-10.2010.
- Franke F, Jäckel D, Dragas J, Müller J, Radivojevic M, Bakkum D, Hierlemann A. High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity. *Front Neural Circuits* 6: 105, 2012. doi:10.3389/fncir.2012.00105.
- Göbel W, Helmchen F. In vivo calcium imaging of neural network function. *Physiology (Bethesda)* 22: 358–365, 2007.
- Gold C, Henze DA, Koch C. Using extracellular action potential recordings to constrain compartmental models. *J Comput Neurosci* 23: 39–58, 2007. doi:10.1007/s10827-006-0018-2.
- Gold C, Henze DA, Koch C, Buzsáki G. On the origin of the extracellular action potential waveform: a modeling study. *J Neurophysiol* 95: 3113–3128, 2006. doi:10.1152/jn.00979.2005.

Table B5. Localization by feature type: statistical analysis

>	Na	Rep	NaRep	Waveform
Na		0.0***	0.0**	0.28***
Rep	ns		ns	0.26***
NaRep	ns	0.0***		0.27***
Waveform	ns	ns	ns	

Statistical analysis for localization errors grouped by feature type. Each entry shows Cohen's *d* and significance of the test column group < row group. Na, extracellular action potential (EAP) negative peak (mainly attributed to sodium currents flowing into soma); Rep, EAP positive peak (associated with cell repolarization phase); NaRep, stacked version of Na and Rep; Waveform, downsampled EAP waveforms. ****P* < 0.001, ***P* < 0.01, ns, Not significant.

Table B6. Localization by MEA probe: statistical analysis

>	Sq15-10	Sq10-15	Sq7-20	Sq6-25	Sq5-30	Neuronexus	NeuroSeeker	NeuroPixels
Sq15-10		ns	ns	ns	ns		ns	ns
Sq10-15	0.04***		ns	ns	ns	ns	ns	ns
Sq7-20	0.08***	0.04**		ns	ns	ns	ns	ns
Sq6-25	0.1***	0.06***	0.02***		ns	ns	ns	ns
Sq5-30	0.13***	0.1***	0.06***	0.04***		0.0***	ns	ns
Neuronexus	0.13***	0.1***	0.06**	ns	ns		ns	ns
NeuroSeeker	0.25***	0.22***	0.18***	0.16***	0.13***	0.12***		ns
NeuroPixels	0.43***	0.41***	0.37***	0.36***	0.32***	0.3***	0.19***	

Statistical analysis for localization errors grouped by probe type. Each entry shows Cohen's *d* and significance of the test column group < row group. MEA, multielectrode array. ****P* < 0.001, ***P* < 0.01. ns, Not significant.

Goldberg JH, Yuste R. Space matters: local and global dendritic Ca²⁺ compartmentalization in cortical interneurons. *Trends Neurosci* 28: 158–167, 2005. doi:10.1016/j.tins.2005.01.005.

Golding NL, Kath WL, Spruston N. Dichotomy of action-potential back-propagation in CA1 pyramidal neuron dendrites. *J Neurophysiol* 86: 2998–3010, 2001. doi:10.1152/jn.2001.86.6.2998.

Goodfellow I, Bengio Y, Courville A. *Deep Learning*. Cambridge, MA: MIT Press, 2016.

Goto T, Hatanaka R, Ogawa T, Sumiyoshi A, Riera J, Kawashima R. An evaluation of the conductivity profile in the somatosensory barrel cortex of Wistar rats. *J Neurophysiol* 104: 3388–3412, 2010. doi:10.1152/jn.00122.2010.

Gouwens NW, Berg J, Feng D, Sorensen SA, Zeng H, Hawrylycz MJ, Koch C, Arkhipov A. Systematic generation of biophysically detailed models for diverse cortical neuron types. *Nat Commun* 9: 710, 2018. doi:10.1038/s41467-017-02718-3.

Gulledge AT, Stuart GJ. Action potential initiation and propagation in layer 5 pyramidal neurons of the rat prefrontal cortex: absence of dopamine modulation. *J Neurosci* 23: 11363–11372, 2003. doi:10.1523/JNEUROSCI.23-36-11363.2003.

Hagen E, Ness TV, Khosrowshahi A, Sørensen C, Fyhn M, Hafting T, Franke F, Einevoll GT. ViSAPy: a Python tool for biophysics-based generation of virtual spiking activity for evaluation of spike-sorting algorithms. *J Neurosci Methods* 245: 182–204, 2015. doi:10.1016/j.jneumeth.2015.01.029.

Hay E, Hill S, Schürmann F, Markram H, Segev I. Models of neocortical layer 5b pyramidal cells capturing a wide range of dendritic and perisomatic active properties. *PLOS Comput Biol* 7: e1002107, 2011. doi:10.1371/journal.pcbi.1002107.

Hines ML, Davison AP, Muller E. NEURON and Python. *Front Neuroinform* 3: 1, 2009. doi:10.3389/neuro.11.001.2009.

Holt GR, Koch C. Electrical interactions via the extracellular potential near cell bodies. *J Comput Neurosci* 6: 169–184, 1999. doi:10.1023/A:1008832702585.

Jun JJ, Steinmetz NA, Siegle JH, Denman DJ, Bauza M, Barbaris B, Lee AK, Anastassiou CA, Andrei A, Aydin C, Barbic M, Blanche TJ, Bonin V, Couto J, Dutta B, Gratiy SL, Gutnisky DA, Häusser M, Karsh B, Ledochowitsch P, Lopez CM, Mitelut C, Musa S, Okun M, Pachitariu M, Putzeys J, Rich PD, Rossant C, Sun WL, Svoboda K, Carandini M, Harris KD, Koch C, O'Keefe J, Harris TD. Fully integrated silicon probes for high-density recording of neural activity. *Nature* 551: 232–236, 2017. doi:10.1038/nature24636.

Table B7. Localization with different methods: statistical analysis

>	Monopolar	Bipolar	B-A-S	CNN
Monopolar		0.33***	ns	0.9***
Bipolar	ns		ns	0.68***
B-A-S	0.04*	0.36***		0.87***
CNN	ns	ns	ns	

Statistical analysis for localization errors grouped by localization method. Each entry shows Cohen's *d* and significance of the test column group < row group. B-A-S, ball and stick; CNN, convolutional neural network. ****P* < 0.001, **P* < 0.05. ns, Not significant.

Kingma D, Ba J. Adam: a method for stochastic optimization (Preprint). ArXiv 1412.6980, 2014.

Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems 25*, edited by Pereira F, Burges CJ, Bottou L, Weinberger KQ. La Jolla, CA: NIPS, 2012, p. 1097–1105.

Kubo T, Katayama N, Karashima A, Nakao M. The 3D position estimation of neurons in the hippocampus based on the multi-site multi-unit recordings with silicon tetrodes. *Conf Proc IEEE Eng Med Biol Soc* 2008: 5021–5024, 2008. doi:10.1109/IEMBS.2008.4650341.

Lindén H, Hagen E, Leški S, Norheim ES, Pettersen KH, Einevoll GT. LFPy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons. *Front Neuroinform* 7: 41, 2014. doi:10.3389/fninf.2013.00041.

Mann HB, Whitney DR. On a test of whether one of two random variables is stochastically larger than the other. *Ann Math Stat* 18: 50–60, 1947. doi:10.1214/aoms/1177730491.

Markram H, Muller E, Ramaswamy S, Reimann MW, Abdellah M, Sanchez CA, Ailamaki A, Alonso-Nanclares L, Antille N, Arsever S, Kahou GA, Berger TK, Bilgili A, Buncic N, Chalimourda A, Chindemi G, Courcol JD, Delalandre F, Delattre V, Druckmann S, Dumusc R, Dynes J, Eilemann S, Gal E, Gevaert ME, Ghojiri JP, Gidon A, Graham JW, Gupta A, Haenel V, Hay E, Heinis T, Hernandez JB, Hines M, Kanari L, Keller D, Kenyon J, Khazen G, Kim Y, King JG, Kisvarday Z, Kumbhar P, Lasserre S, Le Bé JV, Magalhães BR, Merchán-Pérez A, Meystre J, Morrice BR, Muller J, Muñoz-Céspedes A, Muralidhar S, Muthurasa K, Nachbaur D, Newton TH, Nolte M, Ovcharenko A, Palacios J, Pastor L, Perin R, Ranjan R, Riachi I, Rodríguez JR, Riquelme JL, Rössert C, Sfyarakis K, Shi Y, Shillock JC, Silberberg G, Silva R, Tauheed F, Telefont M, Toledo-Rodríguez M, Tränkle T, Van Geit W, Díaz JV, Walker R, Wang Y, Zaninetta SM, DeFelipe J, Hill SL, Segev I, Schürmann F. Reconstruction and simulation

Table B8. Classification accuracies by rotation

	Norot	Physrot	3drot
BTC	98.4	99.3	99.5
ChC	96.0	84.8	84.1
DBC	100.0	99.3	99.1
LBC	100.0	99.5	97.9
MC	100.0	100.0	100.0
NBC	99.5	98.1	98.2
SBC	100.0	98.9	98.4
STPC	97.4	96.0	96.5
TTPC1	92.6	100.0	99.5
TTPC2	100.0	100.0	99.1
UTPC	98.7	99.8	98.7
Average	98.1	98.0	97.6
SD	2.4	3.9	3.9

Values are excitatory/inhibitory classification accuracy (in %) grouped by rotation and cell type. BTC, bitufted cells; ChC, chandelier cells; DBC, double bouquet cells; LBC, large basket cells; MC, Martinotti cells; NBC, nest basket cells; SBC, small basket cells; STPC, slender-tufted pyramidal cells (PC); TTPC1, thick-tufted PC with late bifurcating apical tuft; TTPC2, thick-tufted PC with early bifurcating apical tuft; UTPC, un-tufted PC.

Table B9. CNN size classification performance

	XS	S	M	L	XL
BTC	96.8	98.4	99.3	99.3	98.8
ChC	73.6	79.5	97.0	84.8	87.4
DBC	97.4	99.1	98.9	99.3	98.6
LBC	96.1	98.1	99.6	99.5	98.6
MC	99.5	100.0	99.8	100.0	100.0
NBC	94.2	96.5	96.1	98.1	97.5
SBC	96.7	98.2	97.9	98.9	98.9
STPC	100.0	100.0	98.7	96.0	88.5
TTPC1	100.0	99.8	99.4	100.0	99.9
TTPC2	99.8	99.4	99.6	100.0	99.9
UTPC	97.6	97.9	97.8	99.8	99.8
Average	96.4	97.4	98.6	98.0	97.1
SD	6.6	5.1	1.1	3.9	4.5

Values are excitatory/inhibitory classification accuracy (in %) grouped by size and cell type. BTC, bitufted cells; ChC, chandelier cells; DBC, double bouquet cells; LBC, large basket cells; MC, Martinotti cells; NBC, nest basket cells; SBC, small basket cells; STPC, slender-tufted pyramidal cells (PC); TTPC1, thick-tufted PC with late bifurcating apical tuft; TTPC2, thick-tufted PC with early bifurcating apical tuft; UTPC, un-tufted PC.

of neocortical microcircuitry. *Cell* 163: 456–492, 2015. doi:10.1016/j.cell.2015.09.029.

Markram H, Toledo-Rodriguez M, Wang Y, Gupta A, Silberberg G, Wu C. Interneurons of the neocortical inhibitory system. *Nat Rev Neurosci* 5: 793–807, 2004. doi:10.1038/nrn1519.

McCormick DA, Connors BW, Lighthall JW, Prince DA. Comparative electrophysiology of pyramidal and sparsely spiny stellate neurons of the neocortex. *J Neurophysiol* 54: 782–806, 1985. doi:10.1152/jn.1985.54.4.782.

Mechler F, Victor JD. Dipole characterization of single neurons from their extracellular action potentials. *J Comput Neurosci* 32: 73–100, 2012. doi:10.1007/s10827-011-0341-0.

Mechler F, Victor JD, Ohiorhenuan I, Schmid AM, Hu Q. Three-dimensional localization of neurons in cortical tetrode recordings. *J Neurophysiol* 106: 828–848, 2011. doi:10.1152/jn.00515.2010.

Migliore M, Hoffman DA, Magee JC, Johnston D. Role of an A-type K⁺ conductance in the back-propagation of action potentials in the dendrites of hippocampal pyramidal neurons. *J Comput Neurosci* 7: 5–15, 1999. doi:10.1023/A:1008906225285.

Müller J, Ballini M, Livi P, Chen Y, Radivojevic M, Shadmani A, Viswam V, Jones IL, Fiscella M, Diggelmann R, Stettler A, Frey U, Bakkum DJ, Hierlemann A. High-resolution CMOS MEA platform to study neurons at subcellular, cellular, and network levels. *Lab Chip* 15: 2767–2780, 2015. doi:10.1039/C5LC00133A.

Ness TV, Chintaluri C, Potworowski J, Łęski S, Głowska H, Wójcik DK, Einevoll GT. Modelling and analysis of electrical potentials recorded in microelectrode arrays (MEAs). *Neuroinformatics* 13: 403–426, 2015. doi:10.1007/s12021-015-9265-6.

Neto JP, Lopes G, Frazão J, Nogueira J, Lacerda P, Baião P, Aarts A, Andrei A, Musa S, Fortunato E, Barquinha P, Kampff AR. Validating silicon polytrodes with paired juxtacellular recordings: method and dataset. *J Neurophysiol* 116: 892–903, 2016. doi:10.1152/jn.01013.2016.

Nunez PL, Srinivasan R. *Electric Fields of the Brain: the Neurophysics of EEG*. New York: Oxford Univ. Press, 2006. doi:10.1093/acprof:oso/9780195050387.001.0001.

Overstreet-Wadiche L, McBain CJ. Neurogliaform cells in cortical circuits. *Nat Rev Neurosci* 16: 458–468, 2015. doi:10.1038/nrn3969.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in Python. *J Mach Learn Res* 12: 2825–2830, 2011.

Petersen KH, Einevoll GT. Amplitude variability and extracellular low-pass filtering of neuronal spikes. *Biophys J* 94: 784–802, 2008. doi:10.1529/biophysj.107.111179.

Petersen KH, Lindén H, Dale AM, Einevoll GT. Extracellular spikes and CSD. In: *Handbook of Neural Activity Measurement*, edited by Brette R, Destexhe A. Cambridge, UK: Cambridge Univ. Press, 2012, p. 92–135. doi:10.1017/CBO9780511979958.004.

Peyrache A, Dehghani N, Eskandar EN, Madsen JR, Anderson WS, Donoghue JA, Hochberg LR, Halgren E, Cash SS, Destexhe A. Spatio-temporal dynamics of neocortical excitation and inhibition during human sleep. *Proc Natl Acad Sci USA* 109: 1731–1736, 2012. doi:10.1073/pnas.1109895109.

Pods J, Schönte J, Bastian P. Electrodiffusion models of neurons and extracellular space using the Poisson-Nernst-Planck equations—numerical simulation of the intra- and extracellular potential for an axon model. *Biophys J* 105: 242–254, 2013. [Erratum in *Biophys J* 106: 769–770, 2014.] doi:10.1016/j.bpj.2013.05.041.

Rall W, Shepherd GM. Theoretical reconstruction of field potentials and dendrodendritic synaptic interactions in olfactory bulb. *J Neurophysiol* 31: 884–915, 1968. doi:10.1152/jn.1968.31.6.884.

Ramaswamy S, Courcol JD, Abdellah M, Adaszewski SR, Antille N, Arsever S, Atenekeng G, Bilgili A, Brukay Y, Chalimourda A, Chindemi G, Delalandre F, Dumusc R, Eilemann S, Gevaert ME, Gleeson P, Graham JW, Hernando JB, Kanari L, Katkov Y, Keller D, King JG, Ranjan R, Reimann MW, Rössert C, Shi Y, Shillcock JC, Telefont M, Van Geit W, Diaz JV, Walker R, Wang Y, Zaninetta SM, DeFelipe J, Hill SL, Muller J, Segev I, Schürmann F, Muller EB, Markram H. The neocortical microcircuit collaboration portal: a resource for rat somatosensory cortex. *Front Neural Circuits* 9: 44, 2015. doi:10.3389/fncir.2015.00044.

Rossant C, Kadir SN, Goodman DF, Schulman J, Hunter ML, Saleem AB, Grosmark A, Belluscio M, Denfield GH, Ecker AS, Tolias AS, Solomon S, Buzsáki G, Carandini M, Harris KD. Spike sorting for large, dense electrode arrays. *Nat Neurosci* 19: 634–641, 2016. doi:10.1038/nn.4268.

Sasaki T, Matsuki N, Ikegaya Y. Effects of axonal topology on the somatic modulation of synaptic outputs. *J Neurosci* 32: 2868–2876, 2012. doi:10.1523/JNEUROSCI.5365-11.2012.

Schröder S, Cecchetto C, Keil S, Mahmud M, Brose E, Dogan Ö, Bertotti G, Wolanski D, Tillack B, Schneidewind J, Gargouri H, Arens M, Bruns J, Szyzka B, Vassaneli S, Thewes R. CMOS-compatible purely capacitive interfaces for high-density in-vivo recording from neural tissue. *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. Atlanta, GA, October 22–24, 2015, p. 1–4.

Shew WL, Bellay T, Pleniz D. Simultaneous multi-electrode array recording and two-photon calcium imaging of neural activity. *J Neurosci Methods* 192: 75–82, 2010. doi:10.1016/j.jneumeth.2010.07.023.

Somogyvári Z, Cserpán D, Ulbert I, Érdi P. Localization of single-cell current sources based on extracellular potential patterns: the spike CSD method. *Eur J Neurosci* 36: 3299–3313, 2012. doi:10.1111/j.1460-9568.2012.08249.x.

Somogyvári Z, Zálányi L, Ulbert I, Érdi P. Model-based source localization of extracellular action potentials. *J Neurosci Methods* 147: 126–137, 2005. doi:10.1016/j.jneumeth.2005.04.002.

Table B10. Feature classification performance

	AW	FW	AFW	Waveform
BTC	99.8	98.4	99.3	100.0
ChC	91.6	79.3	97.0	100.0
DBC	97.7	98.4	98.9	100.0
LBC	99.3	97.7	99.6	100.0
MC	100.0	100.0	99.8	100.0
NBC	91.9	93.7	96.1	100.0
SBC	99.6	98.4	97.9	99.8
STPC	99.9	95.6	98.7	98.1
TTPC1	99.5	100.0	99.4	100.0
TTPC2	99.7	99.9	99.6	100.0
UTPC	97.0	99.8	97.8	100.0
Average	98.1	97.0	98.6	99.7
SD	2.8	5.3	1.1	0.6

Values are excitatory/inhibitory classification accuracy (in %) grouped by feature set and cell type. W, peak-to-peak width; F, full-width half-maximum; A, peak-to-peak amplitude; BTC, bitufted cells; ChC, chandelier cells; DBC, double bouquet cells; LBC, large basket cells; MC, Martinotti cells; NBC, nest basket cells; SBC, small basket cells; STPC, slender-tufted pyramidal cells (PC); TTPC1, thick-tufted PC with late bifurcating apical tuft; TTPC2, thick-tufted PC with early bifurcating apical tuft; UTPC, un-tufted PC.

Table B11. MEA probe classification performance

	Sq15-10	Sq10-15	Sq7-20	Sq6-25	Sq5-30	Neuronexus	NeuroSeeker	NeuroPixels
BTC	98.9	99.3	98.4	98.9	99.1	99.1	99.6	98.9
ChC	83.0	84.8	81.8	84.4	89.1	93.3	91.4	91.2
DBC	97.7	99.3	97.4	97.9	99.1	99.5	99.1	99.1
LBC	97.9	99.5	98.9	98.1	98.9	99.3	98.9	98.9
MC	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
NBC	95.3	98.1	93.5	93.7	98.1	99.1	96.3	96.7
SBC	97.0	98.9	97.7	97.5	99.8	99.1	98.6	99.1
STPC	90.6	96.0	93.4	90.6	84.1	83.6	97.0	98.4
TTPC1	100.0	100.0	100.0	99.7	99.8	100.0	99.8	100.0
TTPC2	100.0	100.0	100.0	99.6	99.6	99.6	99.8	100.0
UTPC	99.7	99.8	99.9	99.7	99.8	98.5	99.7	99.6
Average	96.6	98.0	96.9	96.6	96.7	97.0	98.4	98.6
SD	4.8	3.9	4.8	4.6	5.4	5.3	2.3	2.2

Values are excitatory/inhibitory classification accuracy (in %) grouped by probe and cell type. MEA, multielectrode array; BTC, bitufted cells; ChC, chandelier cells; DBC, double bouquet cells; LBC, large basket cells; MC, Martinotti cells; NBC, nest basket cells; SBC, small basket cells; STPC, slender-tufted pyramidal cells (PC); TTPC1, thick-tufted PC with late bifurcating apical tuft; TTPC2, thick-tufted PC with early bifurcating apical tuft; UTPC, untufted PC.

Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R.

Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15: 1929–1958, 2014.

Sullivan GM, Feinn R. Using effect size—or why the P value is not enough. *J Grad Med Educ* 4: 279–282, 2012. doi:10.4300/JGME-D-12-00156.1.

Tveito A, Jæger KH, Lines GT, Paszkowski L, Sundnes J, Edwards AG, Mäki-Marttunen T, Halmes G, Einevoll GT. An evaluation of the accuracy of classical models for computing the membrane potential and extracellular potential for neurons. *Front Comput Neurosci* 11: 27, 2017. doi:10.3389/fncom.2017.00027.

Vassanelli S. Multielectrode and multitransistor arrays for in vivo recording.

In: *Nanotechnology and Neuroscience: Nano-electronic, Photonic and Mechanical Neuronal Interfacing*, edited by De Vittorio M, Martiradonna L, Assad J. New York: Springer, 2014, p. 239–267. doi:10.1007/978-1-4899-8038-0_8.

Wang Y, Gupta A, Toledo-Rodriguez M, Wu CZ, Markram H. Anatomical, physiological, molecular and circuit properties of nest basket cells in the developing somatosensory cortex. *Cereb Cortex* 12: 395–410, 2002. doi:10.1093/cercor/12.4.395.

Wang Y, Toledo-Rodriguez M, Gupta A, Wu C, Silberberg G, Luo J,

Markram H. Anatomical, physiological and molecular properties of Martinotti cells in the somatosensory cortex of the juvenile rat. *J Physiol* 561: 65–90, 2004. doi:10.1113/jphysiol.2004.073353.

Waters J, Schaefer A, Sakmann B. Backpropagating action potentials in neurones: measurement, mechanisms and potential functions. *Prog Biophys Mol Biol* 87: 145–170, 2005. doi:10.1016/j.pbiomolbio.2004.06.009.

Welkenhuysen M, Hoffman L, Luo Z, De Proft A, Van den Haute C, Baekelandt V, Debyser Z, Gielen G, Puers R, Braeken D. An integrated multi-electrode-optrode array for in vitro optogenetics. *Sci Rep* 6: 20353, 2016. doi:10.1038/srep20353.

Woodruff A, Yuste R. Of mice and men, and chandeliers. *PLoS Biol* 6: e243, 2008. doi:10.1371/journal.pbio.0060243.

Zeiler MD, Fergus R. Visualizing and Understanding Convolutional Networks. In: *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings Part I*, edited by Fleet D, Pajdla T, Schiele B, Tuytelaars T. Cham, Switzerland: Springer International, 2014, p. 818–833.

Errata corrige Paper V

In Figure 7C the INHIB and EXCIT labels are reversed.

Paper VI


**Extracellular single neuron
stimulation with high-density
multi-electrode array**

VI

Paper VII

How does the presence of neural probes affect extracellular potentials?

How does the presence of neural probes affect extracellular potentials?

Alessio Paolo Buccino^{1,2} , Miroslav Kuchta³ , Karoline Horgmo Jæger⁴ ,
Torbjørn Veffestad Ness^{1,5} , Pierre Berthet¹ , Kent-Andre Mardal^{3,4} ,
Gert Cauwenberghs²  and Aslak Tveito⁴ 

¹ Center for Integrative Neuroplasticity (CINPLA), Faculty of Mathematics and Natural Sciences, University of Oslo, Oslo, Norway

² Department of Bioengineering, University of California San Diego, San Diego, CA, United States of America

³ Department of Mathematics, University of Oslo, Oslo, Norway

⁴ Simula Research Laboratory, Oslo, Norway

⁵ Faculty of Science and Technology, Norwegian University of Life Sciences, Ås, Norway

E-mail: alessiob@ifi.uio.no

Received 27 March 2018, revised 24 January 2019

Accepted for publication 31 January 2019

Published 26 February 2019



CrossMark

Abstract

Objective. Mechanistic modeling of neurons is an essential component of computational neuroscience that enables scientists to simulate, explain, and explore neural activity. The conventional approach to simulation of extracellular neural recordings first computes transmembrane currents using the cable equation and then sums their contribution to model the extracellular potential. This two-step approach relies on the assumption that the extracellular space is an infinite and homogeneous conductive medium, while measurements are performed using neural probes. The main purpose of this paper is to assess to what extent the presence of the neural probes of varying shape and size impacts the extracellular field and how to correct for them. **Approach.** We apply a detailed modeling framework allowing explicit representation of the neuron and the probe to study the effect of the probes and thereby estimate the effect of ignoring it. We use meshes with simplified neurons and different types of probe and compare the extracellular action potentials with and without the probe in the extracellular space. We then compare various solutions to account for the probes' presence and introduce an efficient probe correction method to include the *probe effect* in modeling of extracellular potentials. **Main results.** Our computations show that microwires hardly influence the extracellular electric field and their effect can therefore be ignored. In contrast, multi-electrode arrays (MEAs) significantly affect the extracellular field by magnifying the recorded potential. While MEAs behave similarly to infinite insulated planes, we find that their effect strongly depends on the neuron-probe alignment and probe orientation. **Significance.** Ignoring the *probe effect* might be deleterious in some applications, such as neural localization and parameterization of neural models from extracellular recordings. Moreover, the presence of the probe can improve the interpretation of extracellular recordings, by providing a more accurate estimation of the extracellular potential generated by neuronal models.



Original content from this work may be used under the terms of the [Creative Commons Attribution 3.0 licence](https://creativecommons.org/licenses/by/3.0/). Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

Keywords: neural simulation, extracellular action potential, neural probes effect, finite element modeling, EMI model

(Some figures may appear in colour only in the online journal)

1. Introduction

Huge efforts have been invested in computational modeling of neurophysiology over the last decades. This has led to the development and public distribution of a large array of realistic neuron models, for example from the Blue Brain Project (bbp.epfl.ch [1, 2]), the Allen-Brain Institute brain cell database (celltypes.brain-map.org [3]), and the Neuromorpho database (neuromorpho.org [4, 5]). As experimental data become available, these models become both more elaborate and more accurate. However, some of the assumptions underlying the most commonly used models may not allow the accuracy necessary to obtain good agreements between models and experiments. For instance, it was pointed out in Tveito *et al* [6] that assumptions underlying the classical cable equation and the associated method for computing the extracellular potential, lead to significant errors both in the membrane potential and the extracellular potential. In the present paper we investigate whether the classical modeling techniques used in computational neurophysiology are sufficiently accurate to reflect measurements obtained by different types of probes, such as microwires/tetrodes, and larger silicon multi-electrode arrays (MEAs). Traditionally, these devices are not represented in the models describing the extracellular field, and our aim is to see if this omission introduces significant errors and how this mismatch could be accounted for in modeling of extracellular activity.

The most widely accepted and used modeling framework for computing the electrophysiology of neurons is the *cable equation* [7–12], which is used to find current and membrane potentials at different segments of a neuron. One straightforward and computationally convenient way to model the extracellular electric potential generated by neural activity is to sum the individual contributions of the transmembrane currents (computed for each segment) considering them as point current sources or line current sources [7, 11] using volume conductor theory. Although this approach represents the gold standard in computational neuroscience, there are some essential assumptions that need to be discussed. First, (i) the neuron is represented as a cable of discrete nodes and the continuous nature of its membrane is not preserved. Second, (ii) when solving the cable equation, the extracellular potential is neglected, but the extracellular potential is computed *a posteriori*. Third, and foremost, (iii) when computing extracellular potentials, the tissue in which the neuron lies is modeled as an infinite medium with homogeneous properties. The validity of these assumptions must be addressed in light of the specific application under consideration. The first assumption (i) can be justified by increasing the number of nodes in the model, but assumption (ii) is harder to relax since it means that the

model ignores ephaptic effects. Therefore, this assumption has gained considerable attention [6, 13–18]. However, the main focus of the present paper is assumption (iii). More specifically our aim is to study the effect of the physical presence of a neural probe on the extracellular signals. Can it be neglected in the mathematical model, or should it be included as a restriction on the extracellular domain? Specifically, is the conventional modeling framework, ignoring the effect of the probes, sufficient to yield reliable prediction of extracellular potentials? Finally, what can modelers do in order to represent and include the effect of recording probes?

In order to investigate this question, we have used the extracellular-membrane-intracellular (EMI) model [6, 19, 20]. The EMI model allows for explicit representation of both the intracellular space of the neuron, the cell membrane and the extracellular space surrounding the neuron. Therefore, the geometry of neural probes can be represented accurately in the model. We have run finite element simulations of simplified pyramidal cells combined with different types of probes, such as microwires/tetrodes, and larger silicon multi-electrode arrays (MEAs).

Our computations strongly indicate that the effect of the probe depends on several factors; small probes (microwires) have little effect on the extracellular potential, whereas larger devices (such as multi-electrode arrays, MEAs) change the extracellular potential quite dramatically, resembling the effect of a non-conductive infinite plane in the proximity of the neuron. The effect, however, depends on the neuron-probe alignment and orientation. We then compare the EMI results with conventional cable equation-based techniques, such as the current summation approach [11, 20], the hybrid solution [20–23], and the method of images [24, 25] and introduce the probe correction method, which allows to reach a hybrid solution accuracy leveraging on a pre-mapping of the probe-specific effect and the reciprocity principle.

The results may aid in understanding experimental data recorded with MEAs, it may improve accuracy when extracellular potentials are used to parameterize membrane models as advocated in [26], and to localize and classify neurons from MEA recordings [27, 28].

The rest of the article is organized as follows: in section 2 we describe the methods used throughout the paper, with particular focus on the EMI model (section 2.1), the meshes (section 2.2), the finite element framework (section 2.3), and modeling approaches used for comparison (section 2.4). In section 3 we present our findings related to the effect of probes of different geometry on the extracellular recordings (section 3.1), the variability of our simulations depending on geometrical parameters of the mesh (section 3.2), before comparing them with results obtained from other computational

approaches (section 3.3) and the relative computational costs of these methods (section 3.4). Finally, we discuss and contextualize the work in section 4.

2. Methods

In this section we introduce the modeling frameworks used to investigate the effect of the probes on the extracellular potential. In particular we first describe the EMI model, the meshes, and the membrane and finite element modeling. Then, we describe the conventional modeling based on the cable equation solution: the current summation approach (CS), the hybrid solution (HS) and the method of images (Mol). Finally, we introduce the probe correction method (PC), which reaches the hybrid solution accuracy in a more efficient and computationally-cheap way.

2.1. The extracellular-membrane-intracellular model

The purpose of the present report is to estimate the effect of introducing a probe in the extracellular domain on the extracellular potential. This can be done using a model discussed in [6, 19, 29–31] referred to as the EMI model. In the EMI model the extracellular space surrounding the neuron, the membrane of the neuron and the intracellular space of the neuron are all explicitly represented in the model. The model takes the form

$$\nabla \cdot \sigma_i \nabla u_i = 0 \quad \text{in } \Omega_i, \quad (1)$$

$$\nabla \cdot \sigma_e \nabla u_e = 0 \quad \text{in } \Omega_e, \quad (2)$$

$$u_e = 0 \quad \text{at } \partial\Omega_e, \quad (3)$$

$$\sigma_e \nabla u_e \cdot n_e = 0 \quad \text{at } \partial\Omega_p, \quad (4)$$

$$n_e \cdot \sigma_e \nabla u_e = -n_i \cdot \sigma_i \nabla u_i \stackrel{\text{def}}{=} I_m \quad \text{at } \Gamma, \quad (5)$$

$$u_i - u_e = v \quad \text{at } \Gamma, \quad (6)$$

$$\frac{\partial v}{\partial t} = \frac{1}{C_m} (I_m - I_{\text{ion}}) \quad \text{at } \Gamma. \quad (7)$$

In the simplified geometry sketched in figure 1, Ω denotes the total computational domain consisting of the extracellular domain Ω_e and the intracellular domain Ω_i , and the cell membrane is denoted by Γ . n_i and n_e are the vectors normal to Γ pointing out of the intra- and extracellular domains, respectively. u_i and u_e denote the intra- and extracellular potentials, and $v = u_i - u_e$ denotes the membrane potential defined at the membrane Γ . The intra- and extracellular conductivities are given respectively by σ_i and σ_e and in this work we assume that the quantities are constant scalars. The cell membrane capacitance is given by C_m , and the ion current density is given by I_{ion} . I_m is the total current current escaping through the membrane.

The EMI model is here considered with grounding (Dirichlet) boundary conditions, i.e. $u_e = 0$, on the boundary of the extracellular domain ($\partial\Omega_e$) while insulating (Neumann)

boundary conditions, i.e. $\sigma_e \nabla u_e \cdot n_e = 0$, were prescribed at the surface of the probe ($\partial\Omega_p$). Note that the latter is a suitable boundary condition also for the conducting surfaces of the probe [25, 32]. The resting potential (see table 1) is used as initial condition for v .

2.2. Meshes

In order to implement the EMI model described above, the computational domain was discretized by unstructured tetrahedral meshes generated by gmsh [33]. We used a simplified neuron model similar to a ball-and-stick model [34, 35], with a spherical soma with $20 \mu\text{m}$ diameter—whose center is in the origin of the axis—an apical dendrite of length $L_d = 400 \mu\text{m}$ and diameter $D_d = 5 \mu\text{m}$ in the positive z direction and an axon of length $L_d = 200 \mu\text{m}$ and diameter $D_d = 2 \mu\text{m}$ in the negative z direction. Both the axon and the dendrites are connected to the soma via a tapering in the geometry. On the dendritic side, the diameter at the soma is $8 \mu\text{m}$ and it linearly reduces to $5 \mu\text{m}$ in a $20 \mu\text{m}$ portion. On the axonal side, the axon hillock has a diameter of $4 \mu\text{m}$ at the soma and it is tapered to $2 \mu\text{m}$ in $10 \mu\text{m}$.

The neuron was placed in a box with and without neural probes to study the effect of the recording device on the simulated signals. We used three different types of probes:

Microwire: the first type of probe represents a microwire type of probe (or tetrode). For this kind of probes we used a cylindrical insulated model with $30 \mu\text{m}$ diameter. The extracellular potential, after the simulations, was estimated as the average of the electric potential measured at the tip of the cylinder. The microwire probe is shown in figure 2(A) alongside with the simplified neuron.

Neuronexus (MEA): the second type of probe model represents a commercially available silicon MEA (A1x32-Poly3-5mm-25s-177-CM32 probe from Neuronexus Technologies), which has 32 electrodes in three columns (the central column has 12 recording sites and first and third columns have 10) with hexagonal arrangement, a y -pitch of $18 \mu\text{m}$, and a z -pitch of $22 \mu\text{m}$. The electrode radius is $7.5 \mu\text{m}$. This probe has a thickness of $15 \mu\text{m}$ and a maximum width of $114 \mu\text{m}$, and it is shown in figure 2(B).

Neuropixels (MEA): the third type of probe model represents the Neuropixels silicon MEA [36]. The original probe has more than 900 electrodes over a 1cm shank, it is $70 \mu\text{m}$ wide and $20 \mu\text{m}$ thick. In our mesh, shown in figure 2(C) we used $24 \times 12 \times 12 \mu\text{m}$ recording sites arranged in the chessboard configuration with an inter-electrode-distance of $25 \mu\text{m}$ [36].

In order to evaluate the effect of the described probes depending on the relative distance to the neuron (x direction), we generated several meshes in which the distance between the contact sites and the center of the neuron was $17.5, 22.5, 27.5, 37.5, 47.5,$ and $77.5 \mu\text{m}$. Note that these distances refer to the beginning of the microwire tip (which extends in the x direction for $30 \mu\text{m}$) and to the MEA $y - z$ plane (for the MEA

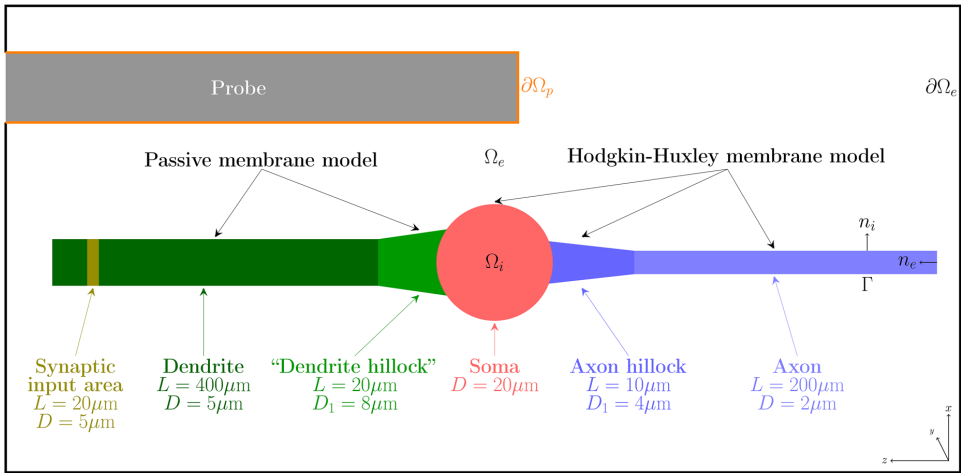


Figure 1. Sketch of the simplified neuron geometry and its surroundings. The intracellular domain is denoted by Ω_i , the cell membrane is denoted by Γ , and the extracellular domain is denoted by Ω_e . The boundary of the probe is denoted by $\partial\Omega_p$ and the remaining boundary of the extracellular domain is denoted by $\partial\Omega_e$. The normal vector pointing out of Ω_i is denoted by n_i , and n_e denotes the normal vector pointing out of Ω_e . L and D are the length and diameter of neural segments, respectively, and D_1 is the diameter of the hillocks in correspondence of the soma. In our simulations, we consider three types of probe geometry (see figure 2). Note that the probe interior is not part of the computational domain.

probes the recording sites do not extend in the x direction). When not specified, instead, the distance for the microwire probe was $25\ \mu\text{m}$, $32.5\ \mu\text{m}$ for the Neuronexus MEA probe, and $30\ \mu\text{m}$ for the Neuropixels probe (center of the probe tip at $40\ \mu\text{m}$).

To investigate if and how the bounding box size affects the simulation, since the electric potential is set to zero at its surface, we generated meshes with five different box sizes. Defining dx , dy , and dz as the distance between the extremity of the neuron and the box in the x , y , and z directions, the three box sizes were:

- size 1:** $dx = 80\ \mu\text{m}$, $dy = 80\ \mu\text{m}$, and $dz = 20\ \mu\text{m}$
- size 2:** $dx = 100\ \mu\text{m}$, $dy = 100\ \mu\text{m}$, and $dz = 40\ \mu\text{m}$
- size 3:** $dx = 120\ \mu\text{m}$, $dy = 120\ \mu\text{m}$, and $dz = 60\ \mu\text{m}$
- size 4:** $dx = 160\ \mu\text{m}$, $dy = 160\ \mu\text{m}$, and $dz = 100\ \mu\text{m}$
- size 5:** $dx = 200\ \mu\text{m}$, $dy = 200\ \mu\text{m}$, and $dz = 150\ \mu\text{m}$

Moreover, we evaluated the solution convergence depending on the resolution by generating meshes with four different resolutions. Defining r_n , r_p , and r_{ext} as the resolutions/typical mesh element sizes for the neuron volume and membrane, for the probe, and for the bounding box surface, respectively, the four degrees of coarseness were:

- coarse 0:** $r_n = 2\ \mu\text{m}$, $r_p = 5\ \mu\text{m}$, and $r_{ext} = 7.5\ \mu\text{m}$
- coarse 1:** $r_n = 3\ \mu\text{m}$, $r_p = 6\ \mu\text{m}$, and $r_{ext} = 9\ \mu\text{m}$
- coarse 2:** $r_n = 4\ \mu\text{m}$, $r_p = 8\ \mu\text{m}$, and $r_{ext} = 12\ \mu\text{m}$
- coarse 3:** $r_n = 4\ \mu\text{m}$, $r_p = 10\ \mu\text{m}$, and $r_{ext} = 15\ \mu\text{m}$

At the interface between two resolutions, the mesh size was determined as their minimum. Further, having instructed gmsh to not allow hanging nodes the mesh in the surroundings of the neuron and probe is gradually coarsened to r_{ext} resolution.

Table 1. Model parameters used in the simulations. The parameters of the Hodgkin–Huxley model are given in [37].

Parameter	Value	Parameter	Value
C_m	$1\ \mu\text{F cm}^{-2}$	g_{syn}	$10\ \text{mS cm}^{-2}$
σ_i	$7\ \text{mS cm}^{-1}$	v_{eq}	$0\ \text{mV}$
σ_e	$3\ \text{mS cm}^{-1}$	t_0	$0.01\ \text{ms}$
g_L	$0.06\ \text{mS cm}^{-2}$	α	$2\ \text{ms}$
v_{rest}	$-75\ \text{mV}$		

For each of the mesh configuration with varying probe model, box size, and coarseness we simulated the extracellular signals with and without the probe in the extracellular space and sampled the electric potential at the recording site locations (even when the probe is absent).

2.3. Membrane model and finite element implementation

On the membrane of the soma and the axon, the ionic current density, I_{ion} , is computed by the Hodgkin–Huxley model with standard parameters as given in [37]. On the membrane of the dendrite, we apply a passive membrane model with a synaptic input current of the form

$$I_{\text{ion}} = I_{\text{leak}} + I_{\text{syn}}, \quad (8)$$

$$I_{\text{leak}} = g_L(v - v_{\text{rest}}), \quad (9)$$

$$I_{\text{syn}} = g_s(\mathbf{x})e^{-\frac{t-t_0}{\alpha}}(v - v_{\text{eq}}), \quad (10)$$

where

$$g_s(\mathbf{x}) = \begin{cases} g_{\text{syn}}, & \text{for } \mathbf{x} \text{ in the synaptic input area,} \\ 0, & \text{elsewhere.} \end{cases} \quad (11)$$

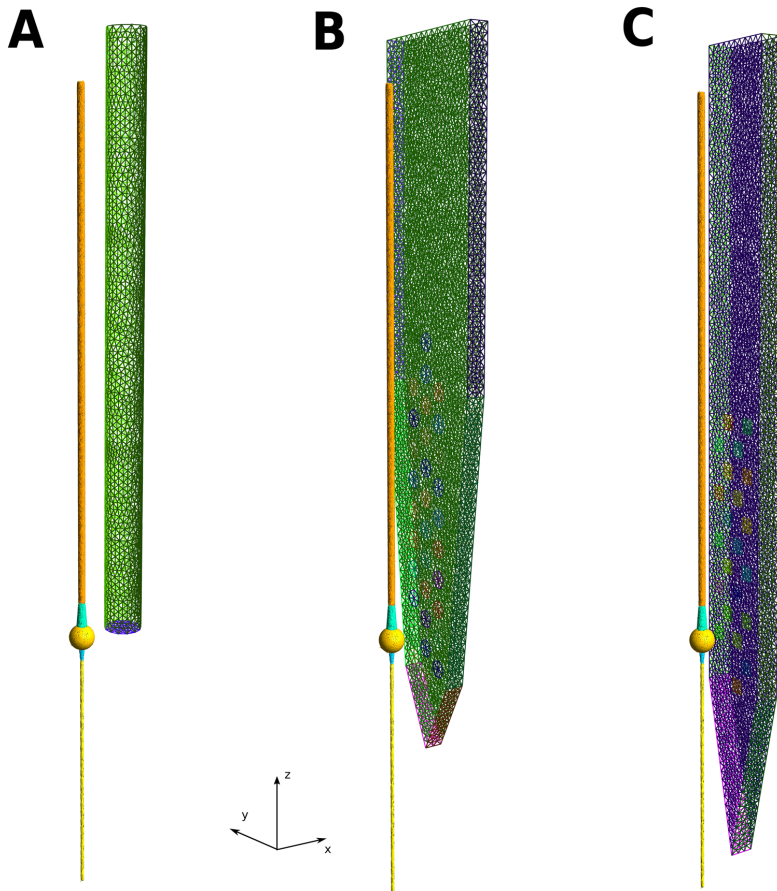


Figure 2. Visualization of simplified neuron and probe meshes. (A) Microwire: the probe has a $15\ \mu\text{m}$ radius and it is aligned to the neuronal axis (z direction) and the center of the probe tip is at $(40, 0, 0)\ \mu\text{m}$ (the soma center is at $(0, 0, 0)\ \mu\text{m}$). The axon and soma of the neuron are depicted in yellow, the dendrite is orange, and the axon and dendritic hillock are in cyan. (B) Neuronexus MEA: the probe represents a Neuronexus A1x32-Poly3-5mm-25s-177-CM32 with recording sites facing the neuron. The MEA is $15\ \mu\text{m}$ thick and the center of the bottom vertex is at $(40, 0, -100)\ \mu\text{m}$. The maximum width of the probe is $114\ \mu\text{m}$, which makes it almost four times larger than the microwire probe. (C) Neuropixels MEA: this probe [36] has a width of $70\ \mu\text{m}$, a thickness of $20\ \mu\text{m}$, and the center of the bottom vertex is at $(40, 0, -100)\ \mu\text{m}$. All meshes represented here are built with the finest coarseness described in the text (*coarse 0*).

The parameters of the dendrite model are given in table 1, and the synaptic input area is defined as a section of the dendrite of length $20\ \mu\text{m}$ located $350\ \mu\text{m}$ from the soma, as illustrated in figure 1.

The EMI model (1)–(7) is solved by the operator splitting scheme and the $H(\text{div})$ discretization proposed in [20]. In this scheme a single step of the EMI model consists of two sub-steps. First, assuming the current membrane potential v is known, the ordinary differential equations (ODE) of the membrane model are solved yielding a new membrane state and the value of v . Next, equation (7), discretized in time with I_{ion} set to zero, is solved together with equations (1)–(6) using the computed value of v as input. This step yields the new values of intra/extra-cellular potentials u_i, u_e and the transmembrane potential v . The $H(\text{div})$ approach then means that the EMI model is transformed by introducing unknown electrical fields

$\sigma_i \nabla u_i$ and $\sigma_e \nabla u_e$ in addition to the potentials u_i, u_e and v . Thus more unknowns are involved, however, the formulation leads to more accurate solutions, see [20, section 3].

In our implementation the ODE solver for the first step of the operator splitting scheme is implemented on top of the computational cardiac electrophysiology framework `cbc.beat` [38]. For the second step, the $H(\text{div})$ formulation of the EMI model, see [20, section 2.3.3], is discretized by the finite element method (FEM) using the `FEniCS` library [39]. More specifically, the electrical fields are discretized by the lowest order Raviart–Thomas elements [40] while the potentials use piecewise constant elements. The linear system due to implicit/backward-Euler temporal discretization in equation (7) and FEM is finally solved with the direct solver `MUMPS` [41] which is interfaced with `FEniCS` via the `PETSc` [42] linear algebra library.

2.4. Other modeling approaches

2.4.1. *Current summation (CS), method of images (MoI), and scaled current summation (SCS).* The cable equation [43–45] is of great importance in computational neuroscience, and it reads,

$$C_m \frac{\partial v}{\partial t} + I_{ion} = \eta \frac{\partial^2 v}{\partial x^2}, \quad (12)$$

where v is the membrane potential of the neuron, C_m is the membrane capacitance, I_{ion} is the ion current density and $\eta = \frac{h\sigma_i}{4}$, where h is the diameter of the neuron, and σ_i denotes the intracellular conductivity of the neuron [43].

This equation is used to compute the membrane potential of a neuron and the solution is commonly obtained by dividing the neuron into compartments and replacing the continuous model (12) by a discrete model [43]. In order to compute the associated extracellular potential, it is common to use the solution of the cable equation to compute the transmembrane currents densities in every compartment, and then invoke the classical summation formula,

$$u_e(x, y, z) = \frac{1}{4\pi\sigma_e} \sum_k \frac{I_k}{|\mathbf{r} - \mathbf{r}_k|}. \quad (13)$$

Here, σ_e is the constant extracellular conductivity (in all the implemented models, the milieu is assumed to be linear by using a constant σ_e), \mathbf{r}_k is the center of the k th compartment of the neuron, $|\mathbf{r} - \mathbf{r}_k|$ denotes the Euclidean distance from $\mathbf{r} = \mathbf{r}(x, y, z)$ to the point \mathbf{r}_k , and I_k denotes the transmembrane current of each compartment. This solution assumes that the extracellular milieu is purely conductive, infinite, and homogeneous. We denote this method as current summation approach (CS) [6].

As the silicon probes are made of insulated material, they could be approximated with the method of images (MoI) [12, 24, 25]. With the MoI the probe is assumed to be an infinite insulating plane, effectively increasing the extracellular potential by a factor of 2. Using the MoI, the factor 2 can be explained as follows: for each current source, an *image* current source is introduced in the mirror position with respect to the insulating plane, effectively doubling the potential in proximity of the plane and canceling current densities normal to the plane. For the MoI, the summation formula (equation 13) reads:

$$u_e(x, y, z) = \frac{1}{2\pi\sigma_e} \sum_k \frac{I_k}{|\mathbf{r} - \mathbf{r}_k|}. \quad (14)$$

As will be shown section 3.1, the peak scaling factor (1 and 2 for the CS and MoI solutions, respectively) of the modeled probes is modulated by the neuron-probe alignment, rotation, and by the probe type and it can be a value between 0 and 2 depending on these factors. Therefore, we also propose and compare a third current summation-based approach, namely scaled current summation (SCS), in which the scale factor is set to match the peak ratio between the hybrid solution (section 2.4.2) and the CS solution on the electrode with largest amplitude (e.g. 1.65 is used in section 3.3.1).

We implemented the same simulations presented in section 2.1 using the conventional modeling approach described above (CS) to compare them with the EMI simulations. We used LFPy [11, 12], running upon NEURON 7.5 [9, 10], to solve the cable equation and compute extracellular potentials using equation (13). As morphology, we used a ball-and-stick model with an axon with the same geometrical properties described in section 2.2. Similarly to the EMI simulations, we used a synaptic input in the middle of the dendritic region activated in the EMI simulation ($z = 360 \mu\text{m}$) to induce a single spike and we observed the extracellular potentials on the recording sites. The synaptic weight was adjusted so that the extracellular largest peak was coincident in time with the one from the EMI simulation. To model the spatial extent of the electrodes, we randomly drew 50 points within a recording site and we averaged the extracellular potential computed at these points [11]. We used the same parameters shown in table 1 (note that in NEURON conductances are defined in S cm^{-2} so we set $g_L = g_{\text{pas}} = 0.06 \cdot 10^{-3} \text{ S cm}^{-2}$) and we used an axial resistance R_a of $150 \Omega \text{ cm}^{-1}$. The `fixed_length` method was used as discretization method with a fixed length of $1 \mu\text{m}$, yielding 658 segments (23 somatic, 422 dendritic, and 213 axonal). Transmembrane currents were considered as current point sources in their contributions to the extracellular potential, following equation (13) (using LFPy `point-source` argument of the `RecExtElectrode` class). The MoI and SCS solutions were calculated by multiplying the CS solution by a factor 2 and 1.65 (optimized scale factor using the hybrid solution).

2.4.2. *Hybrid solution (HS).* The hybrid solution (HS) [21–23] combines the transmembrane currents for each neural segments computed with the cable equation and a finite element modeling for the extracellular space. The transmembrane currents are used as source terms in a finite element solution of the Poisson Equation in the extracellular space (equation (2)), using an iterative solver for the Poisson problem, specifically, preconditioned conjugate gradients with algebraic multigrid preconditioning). With this approach, the probe can be explicitly modeled using insulating (Neumann) boundary conditions at the surface of the probe (equation (5)) and the differences between the HS and the EMI solution lie in differences regarding the modeling of the neuron dynamics, such as the self-eaphaptic effect. The HS requires that a FEM simulation is run for each timestep of the transmembrane currents, each time setting the source terms with the currents at the specific timestep. This makes it computationally expensive, especially, for long simulations. Alternatively, one could run a single FEM simulation for each neural segment with a unitary test current and then use the potentials computed at the recording sites as a static map for summing the contribution of all currents at each timestep. The latter approach can be also computationally complex, as the number of segments in the multi-compartment simulation can be quite high and it would require to store in memory a large number of finite element solutions.

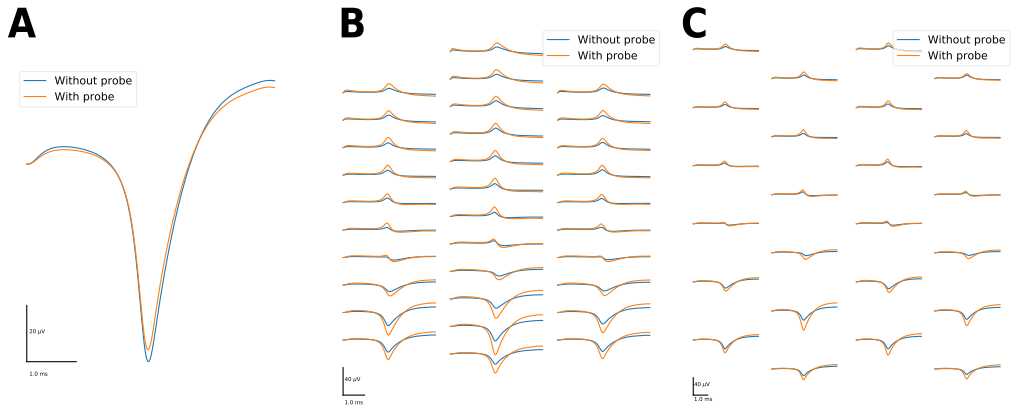


Figure 3. Extracellular action potentials (EAPs). (A) EAPs without (blue) and with (orange) the microwire probe (single recording site) in the extracellular space. The amplitude difference in the largest peak is only 1.03 μV , which is negligible for most applications. (B) Same as (A) but with the Neuronexus MEA probe. For this probe, the difference in amplitude is 20.17 μV (the solution with the MEA is almost twice as large as the one without the MEA in the extracellular space). (C) Same as (A) but with the Neuropixels MEA probe. For this probe, the difference in amplitude is 23.16 μV .

2.4.3. Probe correction (PC). The hybrid solution is a good and widely used approach to model a non-homogeneous extracellular space, especially in the peripheral nervous system literature [21–23]. However, it requires to run a finite element simulation for every neuron simulation, as transmembrane currents are located in different positions for different neurons.

In order to overcome this issue, we designed the probe correction method (PC) that relies on the reciprocity principle [46] and the principle of superimposition (given the assumption of linearity of the milieu expressed in section 2.4.1). The reciprocity principle states that if a current I_1 in a position (x_1, y_1, z_1) generates a potential u_1 in a second position (x_2, y_2, z_2) , then the same current I_1 placed in (x_2, y_2, z_2) will result in a potential u_1 in (x_1, y_1, z_1) ⁶. Using this principle, we first simulated with a finite element method the extracellular potential generated by a test current (1 nA) from each electrode i of a specific probe (e.g. Neuronexus) in any point of the extracellular space and define it as $u_i(x_i, y_i, z_i)$, where (x_i, y_i, z_i) is the relative position with respect to the electrode i . Also in this case we used an iterative solver for the Poisson problem (preconditioned conjugate gradients with algebraic multigrid preconditioning). Then, leveraging on the reciprocity and superimposition principles, we mapped the contribution of each transmembrane current to the potential at each electrode i as: $u_{ik} = I_k u_i(x_k, y_k, z_k)$, where (x_k, y_k, z_k) is now the relative position between the k th neural segment and the electrode i , and I_k is the transmembrane current for the k th neural segment. The potential at each electrode i can be computed as:

$$u_i = \sum_k u_{ik} = \sum_k I_k u_i(x_k, y_k, z_k).$$

The PC method allows to pre-compute the effect of a probe in the extracellular space and then use this mapping for any

⁶The reciprocity principle was originally derived for static charges and extended here to static currents.

neural model, without the need to run a full FEM simulation. The number of FEM solutions that need to be computed and stored during the pre-mapping is equal to the number of electrodes in the probe.

3. Results

In this section we present results of numerical simulations which quantify the effect of introducing probes in the extracellular domain on the extracellular potential. We show how this effect depends on the distance between the neuron and the probe, their lateral alignment, and the probe rotation. The evaluation of the *probe effect* (section 3.1) is carried out using the EMI simulation framework. Furthermore, we evaluate the numerical variability of the EMI solutions (section 3.2), we compare with other modeling schemes (section 3.3), and finally report CPU-efforts for the simulations (section 3.4).

3.1. The probe effect

3.1.1. The geometry of the probe affects the recorded signals. The first question that we investigated is whether the probes have an effect and, if so, how substantial this effect is and if it depends on the probe geometry. In order to do so we analyzed the extracellular action potential (EAP) traces with and without placing the probe in the mesh.

In figure 3 we show the EAP with and without the microwire probe (A), the Neuronexus probe (B), and the Neuropixels probe (C). The blue traces are the extracellular potentials computed at the recording sites when the probe was removed, while the orange traces show the potential when the probe is present in the extracellular space. In this case the probe tip was placed 40 μm from the soma center, we used a box of size 2 and coarse 2 resolution. It is clear that the *probe effect*

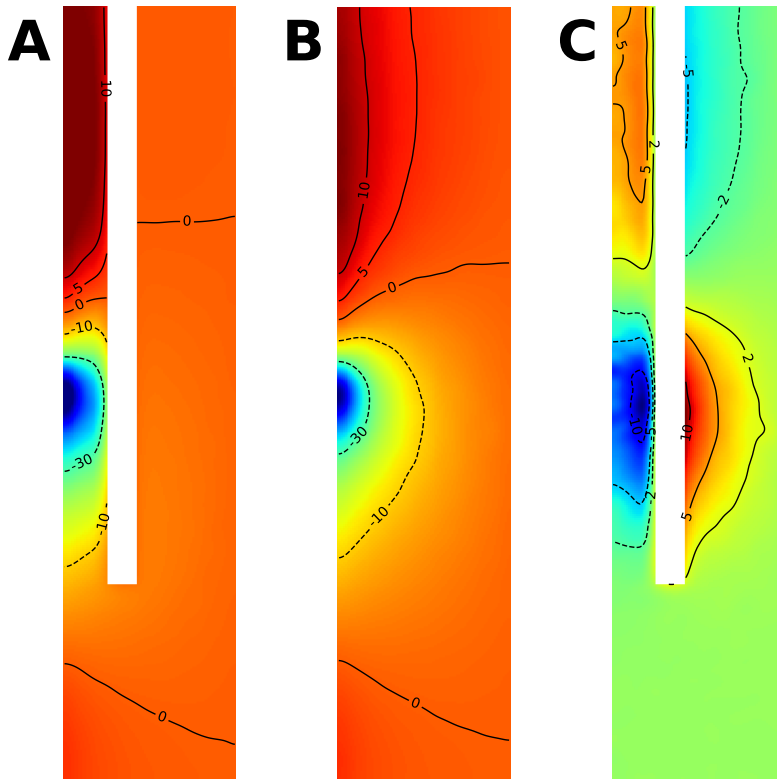


Figure 4. Extracellular potential distribution on the $x - z$ plane with the Neuronexus MEA probe (A) without the probe (B), and their difference (C). The images were smoothed with a gaussian filter with standard deviation of $4 \mu\text{m}$. The color code for panel A and B is the same. The isopotential lines show the potential in μV . The probe (white area) acts as an insulator, effectively increasing the extracellular potential (in absolute value) in the area between the neuron and the probe (panel C, blue colors close to the soma and red close to the dendrite) and decreasing it behind the probe of several μV . The effect is smaller at the tip of the probe (the green color represents a $0 \mu\text{V}$ difference).

is more prevalent for the MEA probes than for the microwire, suggesting that the physical size and geometry of the probe play an important role. In particular, for the Neuronexus probe the minimum peak without the probe is $-21.09 \mu\text{V}$ and with the probe it is $-41.26 \mu\text{V}$: the difference is $20.17 \mu\text{V}$. For the Neuropixels probe the peak with no probe is $-21.2 \mu\text{V}$, with the probe it is $-44.36 \mu\text{V}$ and the difference is $23.16 \mu\text{V}$. In case of the microwire type of probe, the effect is minimal: the minimum peak without the probe is $-16.85 \mu\text{V}$, with the probe it is $-15.82 \mu\text{V}$, and the difference is about $1.03 \mu\text{V}$ (the peak without the probe is even larger than the one with the probe). Note that the values for the microwire are slightly lower than the MEAs because even if the microwire tip center is at the same distance ($40 \mu\text{m}$), it extends for $30 \mu\text{m}$ in the x direction, effectively lowering the recorded potential due to the fast decay of the extracellular potential with distance. The recording sites of the MEAs, instead, lie on the $y - z$ plane, at a fixed distance.

The MEAs, electrically speaking, are like insulating walls that do not allow currents to flow in. The insulating effect can

be appreciated in figure 4, in which the extracellular potential at the time of the peak is computed in the $[10, 100] \mu\text{m}$ interval in the x direction and in the $[-200, 200] \mu\text{m}$ interval in the z direction (the origin is the center of the soma). Panel A shows the extracellular potential with the probe (Neuronexus) and panel B without the probe. The currents are deflected due to the presence of the probe, and this causes an increase (in absolute value) in the extracellular potential between the neuron and the probe, as shown in panel C, where the difference of the extracellular potential with and without probe is depicted. The substantial effect using the MEA probe probably also depends on the arrangement of the recording sites: while for the MEAs, the electrodes *face* the neuron (they lie on the $y - z$ plane) and currents emitted by the membrane cannot flow in the x direction due to the presence of the probe, for the microwire, the electrode is at the tip of the probe (at $z = 0$, extending in the $x - y$ plane—see figure 2) and currents can *naturally* flow downwards in the x direction, yielding a little effect (figure 4(C) shows that the effect at the tip of the MEA probe is almost null).

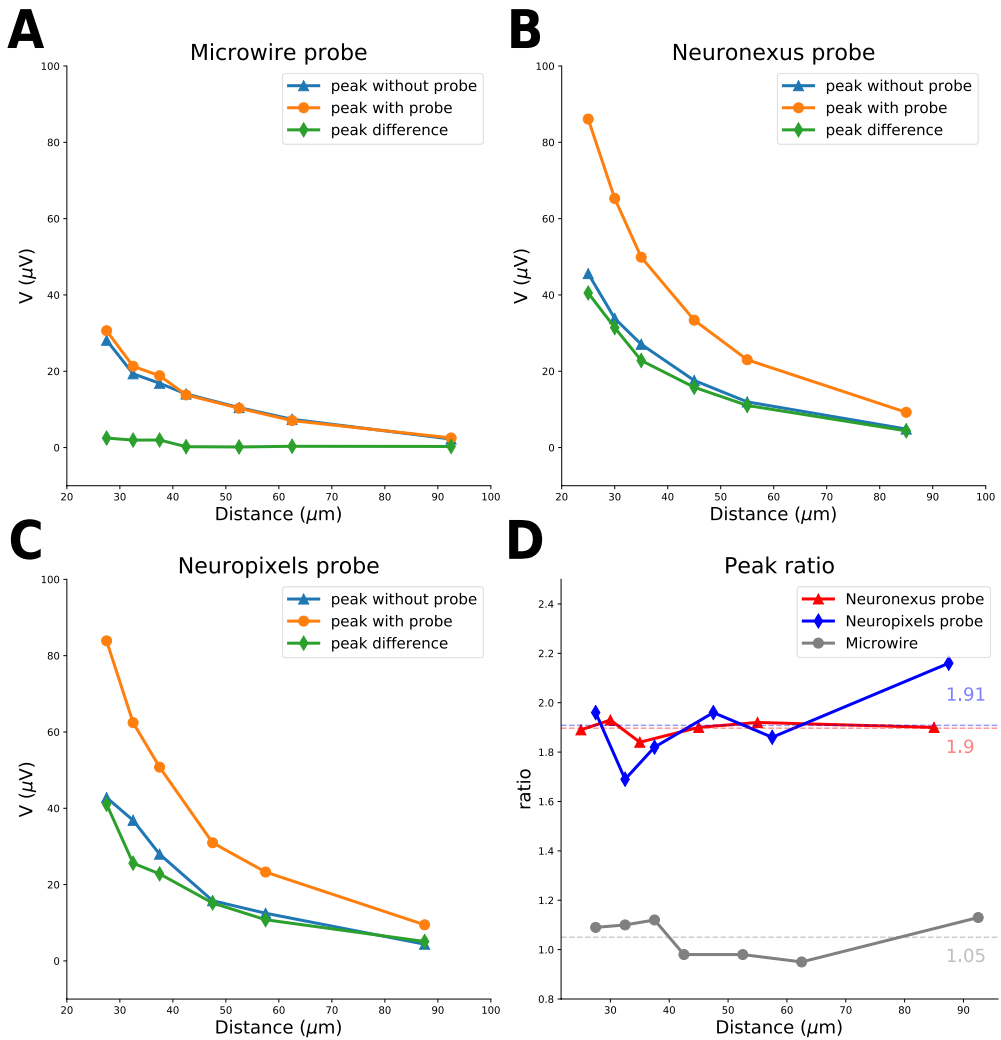


Figure 5. Differences in EAP maximum absolute value peak with and without probe depending on distance. (A) Microwire probe: maximum peak without probe (blue), with probe (orange), and their difference (green). The difference is small even when the probe is close to the neuron. (B) Neuronexus MEA probe: maximum peak without probe (blue), with probe (orange), and their difference (green). The difference is large at short distances and it decays at larger distances. (C) Neuropixels MEA probe: maximum peak without probe (blue), with probe (orange), and their difference (green). Also for this probe the difference is large at short distances and it reduces at further away from the neuron. (D) Ratio between peak with and without probe for the Neuronexus (red), the Neuropixels (blue) and the microwire probe (grey). The ratio is almost constant at different distances and the average value is 1.9 for the Neuronexus, 1.91 for the Neuropixels, and 1.05 for the microwire probe.

3.12. *The amplitude ratio is constant with probe distance.* In this section we analyze the trend of the probe-induced error depending on the vicinity of the probe. We swept the extracellular space from a closest distance between the probe and the somatic membrane of 7.5 µm to a maximum distance of 67.5 µm.

In figures 5(A)–(C) we plot the absolute peak values with (orange) and without probe (blue), as well as their difference (green) for the microwire (A), Neuronexus (B) and

Neuropixels (C) probes. For the microwire (A), as observed in the previous section, the *probe effect* is small and the maximum difference is 1.97 µV, which is 10.1% of the amplitude without probe, when the probe is closest. For the Neuronexus MEA probe (B), at short distances the difference between the peaks with and without probe is large—40.5 µV (88.8% of the amplitude without probe) at 7.5 µm probe-membrane distance—and it decreases as the probe distance increases. At the farthest distance, where the probe tip is at 75 µm from the

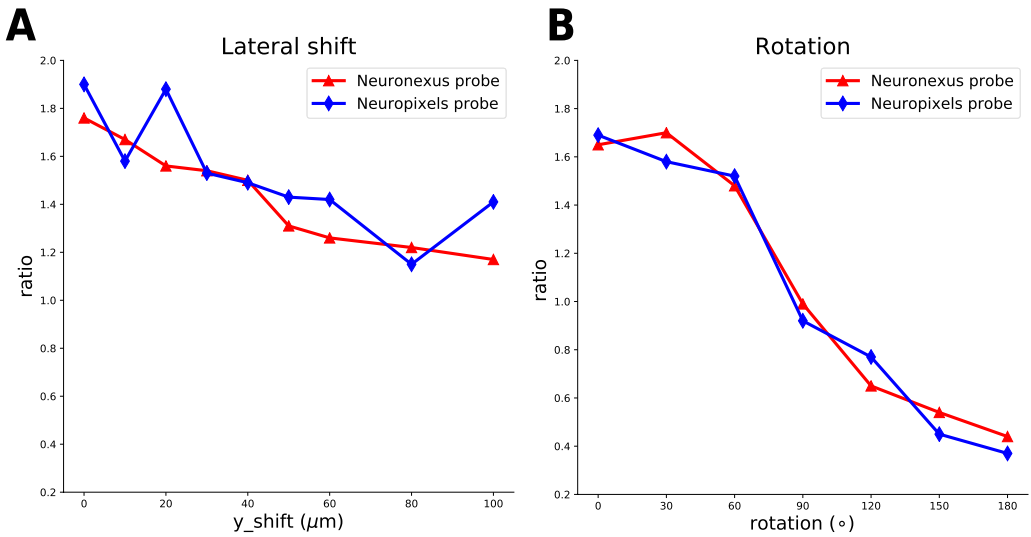


Figure 6. Effects of neuron probe alignment. (A) Amplitude ratio for different y lateral shifts for the Neuronexus (red) and Neuropixels (blue) probes. The ratio decreases almost linearly with the y shift. (B) Amplitude ratio for different probe rotations for the Neuronexus (red) and Neuropixels (blue) probes. At small rotations, the peak ratio is between 1.6 and 1.8, at 90° rotation (when the probe exposes its thinnest side to the neuron) it is around 1, and between 90° and 180° the shadowing effect of the probe makes the ratio lower than 1.

somatic membrane, the difference is $4.38 \mu\text{V}$, which is 90.2% of the amplitude without probe. For the Neuropixels MEA probe (C) the effect is in line with the Neuronexus probe, with a maximum difference of $41.07 \mu\text{V}$ (95.9% of the amplitude without probe) when the probe is closest and a minimum of $5.08 \mu\text{V}$, which is still 116.1% of the amplitude without probe, when the probe is located at the maximum distance. Note that the peak amplitudes on the microwire probe are smaller than the one measured on the MEAs at a similar distances. At the closest distance, for example, the Neuronexus MEA electrodes lie on the $y - z$ plane exactly at $7.5 \mu\text{m}$ from the somatic membrane. For the microwire, instead, $7.5 \mu\text{m}$ is the distance to the beginning of the cylindrical probe, whose tip extends in the x direction for $30 \mu\text{m}$. The simulated electric potential is the average of the electric potential computed on the microwire tip and it results in a much lower amplitude due to the fast decay of the extracellular potential with distance (see equation (13)).

In panel (D) of figure 5 we show the ratio between the peak with probe and without probe depending on the probe distance for the Neuronexus (red), Neuropixels (blue), and the microwire (grey) probes. The ratio for the microwire probe varies around 1 (average = 1.05), confirming that the *probe effect* can be neglected for microwire-like types of probe, due to their size and geometry. Instead, when a MEA probe is used, the average ratio is around 1.9 and its effect on the recordings cannot be neglected.

3.1.3. The probe effect is reduced when neuron and probe are not aligned. So far, we have shown results in which the neuron and the probe are perfectly aligned in the y direction, but the *probe effect* is likely to be affected by the neuron-probe

alignment, since the area of the MEA probe (we focus here on the Neuronexus and Neuropixels MEA probes as the effect using the microwire is negligible) *facing* the neuron changes depending on the lateral shift in the y direction and probe rotation.

To quantify the trend of the *probe effect* depending on the y shift, we ran simulations moving the probes at different y locations (10, 20, 30, 40, 50, 60, 80, and $100 \mu\text{m}$) and computed the ratios between the maximum peak with and without the MEA in the extracellular space. The simulations were run with *coarse 2* resolution and *boxsize 5* and the probe tip was at $40 \mu\text{m}$ from the center of the neuron. In figure 6(A) we show the peak ratios depending on lateral y shifts. The ratio appears to decrease almost linearly with the shifts, from a value of around 1.8–1.9 when the probe is centered (note that the peak ratio slightly varies depending on resolution and size, as covered in section 3.2) to a value of around 1.2 when the shift is $100 \mu\text{m}$ (the half width of the probe is $57 \mu\text{m}$ for Neuronexus and $35 \mu\text{m}$ for Neuropixels).

In order to evaluate the effect of rotating the probes, we ran simulations with the probe at $70 \mu\text{m}$ distance (to accommodate for different rotations), *coarse 2* resolution, *boxsize 4*, and rotations of 0, 30, 60, 90, 120, 150, and 180° . In figure 6(B) the peak ratios depending on the rotation angle are shown. For small or no rotations (0, 30°) the value is around 1.7 (note that we always selected the electrode with the largest amplitude, which might not be the same electrode for all rotations). For a rotation of 90° the peak ratio is around 1 (the probe exposes its thinnest side to the neuron) and for further rotations the probe’s shadowing effect makes the peak with the probe smaller (as observed in figure 4(C)), yielding peak ratio values below 1. These results demonstrate that the

Table 2. Solution variability depending on box (domain) size. The columns contain the maximum peak with the Neuronexus (MEA) probe, without the probe, the difference and ratio of the amplitudes with and without probe. The values are averaged over all resolutions.

Box size	V _{peak} with MEA (μV)	V _{peak} without MEA (μV)	Difference (μV)	Peak ratio
1	-40.12	-20.64	19.48	1.95
2	-41.46	-20.91	20.55	1.98
3	-41.91	-23.83	18.07	1.77
4	-43.10	-23.35	19.75	1.85
5	-43.09	-23.71	19.38	1.82

relative arrangement between the neuron and the probe play an important role in affecting the recorded signals.

3.2. EMI solution dependence on domain size and resolution

We generated meshes of four different resolutions and five different box sizes, as described in section 2.2, in order to investigate how the resolution and the domain size affect the finite element solutions. Since we are mainly interested in how the probe affects the extracellular potential and we showed that only for MEA probes this effect is large, we focus on the extracellular potential at the recording site with the maximum negative peak. We used the Neuronexus MEA probe for this analysis and the distance of the tip of the probe was $40\ \mu\text{m}$ (the recording sites plane is at $32.5\ \mu\text{m}$ from the somatic center). The recording site which experienced the largest potential deflection was at position $(32.5, 0, -13)\ \mu\text{m}$, i.e. the closest to the neuron soma in the axon direction. For a deeper examination of convergence of the EMI model refer to [6]. For resolutions *coarse 0* and *coarse 1* the box of size 4 and 5, and of size 5, respectively, were too large to be simulated.

In table 2 we show the values of the minimum EAP peak with and without the Neuronexus probe, their difference, and their ratio grouped by the domain (box) size and averaged over resolution. Despite some variability due to the numerical solution of the problem, there is a common trend in the peak values as the domain size increases: the minimum peaks tend to be larger in absolute values, both when the probe is in the extracellular space (from $-40.12\ \mu\text{V}$ for box size 1 to $-43.09\ \mu\text{V}$ for box size 5) and when it is not (from $-20.64\ \mu\text{V}$ for box size 1 to $-23.71\ \mu\text{V}$ for box size 5). This can be explained by the boundary conditions that we defined for the bounding box (equation (3)), which forces the electric potential at the boundaries to be 0. For this reason, a smaller domain size causes a steeper reduction of the extracellular potential from the neuron to the bounding box, making the peak amplitude, in absolute terms, smaller. The peak difference with and without the MEA probe appears to be relatively constant, but the peak ratio tends to slightly decrease with increasing domain size for the same reason expressed before (from 1.95 for box size 1 to 1.82 for box size 5). The solutions appear to be converging for box sizes 4 and 5, but the relative error (difference between box 1 and box 5 values divided by the value of box 5) is moderate (6.89% for the

Table 3. Solution variability depending on resolution (*Coarseness*). The columns contain the maximum peak with the Neuronexus (MEA) probe, without the probe, the difference and ratio of the amplitudes with and without probe. The values are computed with a box size 2.

Coarseness	V _{peak} with MEA (μV)	V _{peak} without MEA (μV)	Difference (μV)	Peak ratio
0	-41.74	-20.67	21.07	2.02
1	-40.74	-20.25	20.49	2.01
2	-41.26	-21.09	20.18	1.96
3	-42.11	-21.64	20.46	1.95

peak with probe, 12.95% for the peak without probe, and 4.14% for the peak ratio). Nevertheless, the 1.8–1.85 peak ratio values obtained with larger domain sizes should be a closer estimate of the true value.

Table 3 displays the same values of table 2, but with a fixed box size of 2 and varying resolution (*Coarseness*). The relative error (maximum difference across resolutions divided by the average values among resolutions) of the peak with the MEA is 3.3%, without the probe it is 6.65%, and for the peak ratio it is 3.53%.

Because the main purpose of this work was to qualitatively investigate the effect of various probe designs and the effect of distance, alignment, and rotation on the measurements, we used resolution *coarse 2* and box size 2, which represented an acceptable compromise between accuracy and simulation time. For investigating the effect of probe rotation and side shift we increased the box size to 4 and 5, respectively, to accommodate the position of the neural probe. Finally, in section 3.3 we increased the resolution to *coarse 0* and used box size 3 to obtain more accurate results for the comparison with the cable equation simulations.

3.3. Comparison with other approaches

After having investigated how an extracellular probe affects the amplitude of the recorded potentials and how this amplitude is modulated with distance, alignment, and rotation between the neuron and the probe, we now compare the EMI solution to other modeling approaches. We first analyze the differences between the EMI solution without the probe and the cable equation / current summation approach (CS) and between the EMI solution with the probe and the hybrid solution (HS). Then we focus on the HS, which combines a cable equation solution and an explicit model of the extracellular space, including the probe, in a FEM framework, and compare its solution to three correction strategies: the method of images (MoI), the scaled current summation (SCS), and the probe correction (PC).

In all the following simulations we used a mesh with *coarse 0* resolution and box size 3. The distance between the neuron soma center and the probe tip was $40\ \mu\text{m}$, resulting in recording sites on the $x = 32.5\ \mu\text{m}$ plane.

3.3.1. EMI, CS, and HS comparison. In order to compare the EMI simulations to conventional modeling, we built the

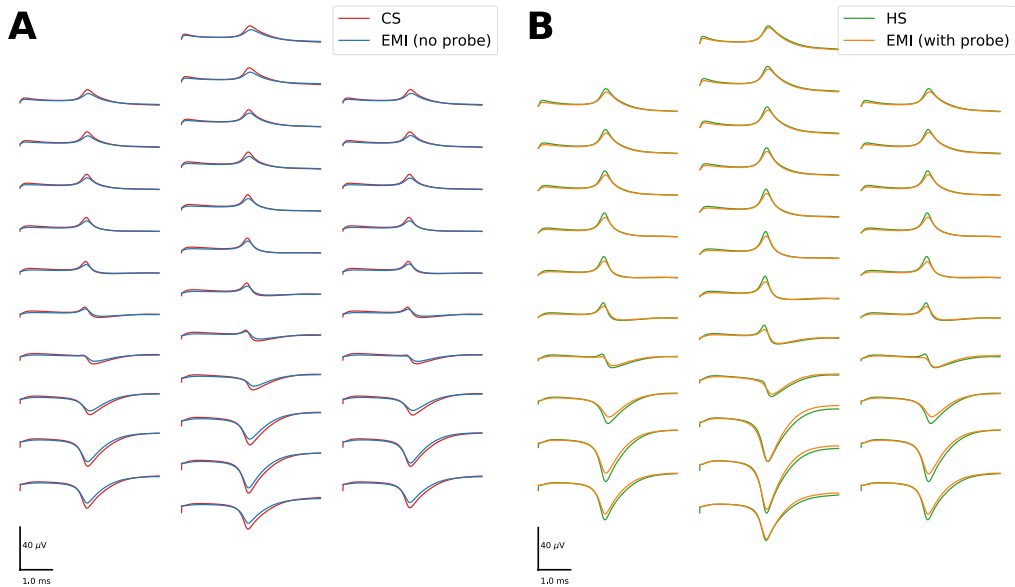


Figure 7. Comparison of the EAPs (A) between the current summation approach (CS, red) and the EMI model without probe (blue), displaying a peak amplitude difference of $4.91 \mu\text{V}$, and (B) between the hybrid solution (HS, green) and the EMI model with probe (orange), exhibiting a peak amplitude difference of $3.55 \mu\text{V}$.

same scenario shown in figure 2(B) (Neuronexus probe) using NEURON and LFPy, as described in section 2.4. As conventional modeling assumes an infinite and homogeneous medium, we compared the EAPs obtained by combining the cable equation solution (equation (12)) and the current summation formula (equation (13)) with the EMI simulations without the probe. The extracellular traces for the current summation approach (CS, red) and the EMI model (blue) are shown in figure 7(A). The EAPs almost overlap for every recording site, despite some differences in amplitude. On the electrode with the largest peak, the value for the EMI solution is $-23.03 \mu\text{V}$, while the value for the CS is $-27.95 \mu\text{V}$ (the difference is $4.91 \mu\text{V}$). This difference, which has been previously observed, is intrinsic to the EMI model [6], and can be due to self-ephaptic effects [6, 13–18]. Note also that the condition that forces the extracellular potential to zero at the boundary of the domain causes a steeper descent in the extracellular amplitudes, as discussed in section 3.2.

The hybrid solution (HS) uses currents computed with the cable equation and runs a FEM simulation of the extracellular space, including the probe. In figure 7(B) we show the extracellular potential of the EMI simulation with probe (orange) and the HS (green). Also in this case we observe that the EMI solution yields slightly smaller amplitudes with respect to the HS (EMI peak: $-42.6 \mu\text{V}$; HS peak $-46.15 \mu\text{V}$; difference: $3.55 \mu\text{V}$) and these differences can be once again traced back to underlying differences of the neural solver.

3.3.2. HS, MoI, SCS, and PC comparison. After having shown that there are intrinsic differences between the EMI model and solutions based on the cable equation (CS, HS), we

now compare two computationally less expensive strategies that could be used to account for the *probe effect* in modeling of extracellular potentials.

The MoI and SCS are attractive candidates due to their almost null computational cost, as they only multiply all values by a constant factor. The factor for infinite insulated planes, as described in section 2.4.1, is 2, but as shown in figures 5 and 6, for MEA probes it is somewhere between 0 and 2 depending on the neuron-probe lateral shift and rotation. In this scenario, the neuron is perfectly aligned with the probe and there is no rotation. The peak ratio for the SCS was computed by dividing the largest peaks of the HS and CS solutions and it was set to 1.65. In figure 8(A) the EAP from the HS (green), from the MoI (pink), and from the SCS with factor 1.65 (grey) are displayed. The MoI (pink) overshoots the estimation of the extracellular amplitudes (MoI peak: $-55.89 \mu\text{V}$; HS $-46.15 \mu\text{V}$; difference: $9.74 \mu\text{V}$). The SCS solution, expectedly, results in the same amplitude as the HS on the electrode with the largest peak, as the scaling factor was computed using the actual peak ratio between the HS and the CS solution. However, there are some discrepancies between HS and SCS. Figure 8(B) shows the distribution of peak ratios of all the 32 electrodes with respect to the HS peaks. The CS, MoI, and SCS solutions display a range of values in the peak ratios, showing that the amplitude modulation of the electrodes is not a constant value. This can be traced back to the fact that a lateral shift of the neuron reduces the peak ratio (figure 6(A)): electrodes on the side of the probe yield a lower effect than the ones at the center of the probe. Due to this variability, a correction strategy based on a constant scaling will not be able to accommodate for this effect.

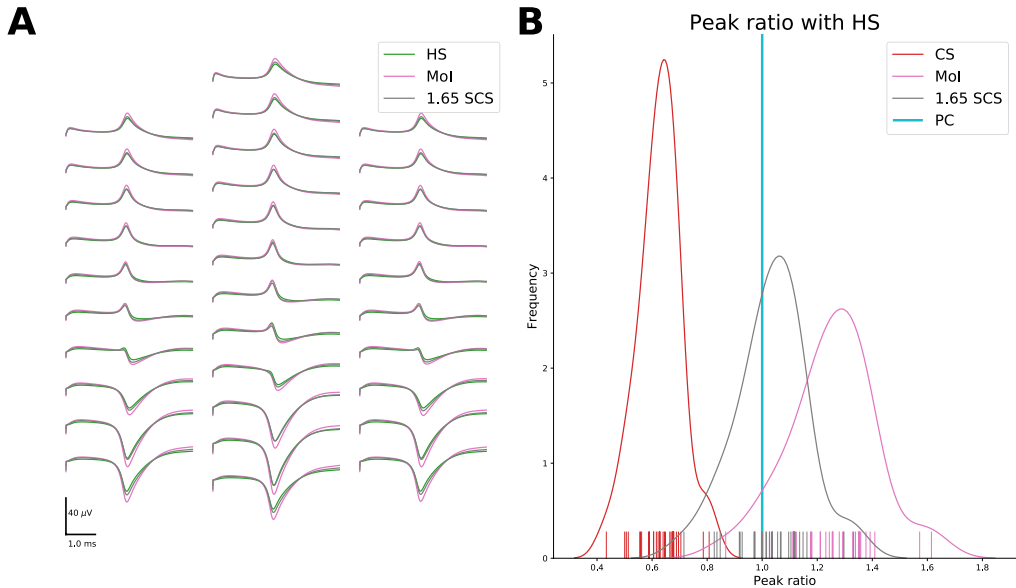


Figure 8. (A) EAPs of the Neuronex probe as computed using the hybrid solution (HS, green), the Method of Images (MoI, pink) and the scaled current summation with factor 1.65 (1.65 SCS, grey). (B) Peak ratio distribution of the electrodes of the Neuronex probe compared to the hybrid solution, from the current summation (CS, red), Method of Images (MoI, pink), the scaled current summation with factor 1.65 (1.65 SCS, grey), and the probe correction (PC, cyan) models. Note that the peak amplitudes computed from all the electrodes by the PC and HS approaches overlap perfectly, thus resulting in a single vertical line at peak ratio value 1.

The probe correction (PC) solution, based on the reciprocity principle (section 2.4.3), results in a solution perfectly coincident to the HS, at a much smaller computational cost (see table 5). In figure 8(B) the PC ratios are depicted as a vertical line at 1 because the peak amplitudes are exactly the same as the HS. The PC approach, in fact, pre-maps the effect of each electrode on the extracellular domain, effectively modeling in an efficient way the distribution of peak ratios observed when using the CS, MoI, and SCS methods.

In table 4 we summarize the comparison results, showing maximum, minimum, average peak ratios and the peak ratio distribution standard deviation for all the pairwise comparisons analyzed in this section.

3.4. CPU requirements

Whereas the EMI formulation represents a powerful and more detailed computational framework for neurophysiology simulations, it is associated with a much larger computational load. The simulations were performed on an Intel(R) Xeon(R) CPU E5-2623 v4 @ 2.60 GHz machine with 16 cores and 377 GB RAM running Ubuntu 16.04.3 LTS.

Table 5 contains the coarseness, domain size, number of tetrahedral cells, number of mesh vertices, total number of triangular cells (facets), facets on the surface of the neuron, the system size for the FEM problem, and the time in second (CPU time) to compute the solution for meshes without the probe in the extracellular domain. We show the results without probes in the extracellular domain, as they are they are computationally

Table 4. Summary of comparison results showing, for each comparison, the maximum, minimum, and average peak ratio, as well as the standard deviation of the peak ratio (PR) distribution. The peak ratios are the electrode-wise division between the peaks of the first and second models listed in the Comparison tab. *EMI (with)* and *EMI (no)* indicate the EMI solution with and without the probe in the extracellular space, respectively.

Comparison	Maximum PR	Minimum PR	Average PR	PR standard deviation
EMI (with)	2.16	1.4	1.81	0.19
EMI (no)				
CS—EMI (no)	1.6	1.16	1.39	0.1
HS—EMI (with)	1.49	1.01	1.25	0.11
CS—HS	0.81	0.43	0.63	0.08
MoI—HS	1.61	0.87	1.25	0.15
1.65 SCS—HS	1.33	0.72	1.03	0.13
PC—HS	1	1	1	0

more intense due to the fact that the volume inside the probe is not meshed (although the resolution on the probe surface is finer, the resulting system size without the probe is larger than with the probe). The CPU requirements and the time needed to run the simulation strongly depend on the resolution of the mesh: the problem with coarseness 3 and box size 3 takes around 1 h and 20 min (system size = 745 789), while for the same box size and coarseness 0, the time required is around

Table 5. Model type, FEM system size, resolution (*Coarseness*), box size, mesh parameters (number of cells, number of facets, number of neuron facets, and vertices), and CPU time to run the simulations. Note that for *coarse 2* and *coarse 3* the resolution of the neuron ($r_n = 4 \mu\text{m}$) is the same.

Model	System size	Coarse	Box size	Mesh Cells	Total facets	Neuron facets	Vertices	T (s)
EMI	337515	3	1	66171	135672	2552	12400	1414.24
EMI	516079	3	2	101443	207318	2552	18628	2813.22
EMI	562137	2	1	110363	225887	2480	20420	2589.83
EMI	745789	3	3	146905	299442	2552	26540	4569.11
EMI	835365	2	2	164331	335517	2480	29940	4753.39
EMI	1204001	2	3	237259	483371	2480	42666	10797.78
EMI	1225082	1	1	241402	491840	3888	43373	9593.98
EMI	1254096	3	4	247514	503291	2552	44013	10756.46
EMI	1881777	1	2	371471	755153	3888	65867	18880.78
EMI	1983058	3	5	391986	795536	2552	68875	23756.09
EMI	2110421	2	4	416949	846736	2480	73535	21582.90
EMI	2532813	0	1	501235	1015789	8376	87535	27676.91
EMI	2728288	1	3	539518	1094385	3888	94417	45430.64
EMI	3486058	2	5	689996	1398031	2480	119968	53132.75
EMI	3810512	0	2	755076	1527718	8376	130389	52495.76
EMI	4802239	1	4	951245	1925497	3888	164359	68474.42
EMI	5271370	0	3	1045440	2112965	8376	179195	82601.74
HS	403085	0	3	2299046	4665105	—	403085	3572.91
PC (map)	403085	0	3	2299046	4665105	—	403085	2015.02
PC (load)	403085	0	3	2299046	4665105	—	403085	409.91
PC (run)	403085	0	3	2299046	4665105	—	403085	3.51

22h (system size = 5 271 370). The domain size also strongly affects the mesh size and computation time. For example, for the *coarse 2* resolution, with respect to *box 1*, *box 2* is 1.83× slower, *box 3* 4.16×, *box 4* 8.33×, *box 5* 20.51×.

The last four rows show the CPU requirements for the HS and the different steps of the PC solution. These simulations, despite having the same resolution and box size as the most intense EMI simulation (*coarse 0* and box size 3), result in a much smaller system size, as they solve for the extracellular potential only (EMI also solves for intracellular potentials and currents in the entire domain). To perform a fair comparison with the EMI model, the computations were done in serial. Parallel solvers would likely speed up the HS and PC solutions and could be easily implemented. Simulating 5 ms using the HS takes about 1 h, compared to the 22h of the EMI solution. The PC performance is divided in three steps. *PC (map)* refers to the computation of the 32 FEM solutions (one for each Neuronexus electrode), and it takes slightly more than 30min. Once the pre-map is computed it can be used for any neural model. Loading the FEM solutions in memory (*PC (load)*) requires around 7 min and once loaded, it takes a few seconds (3.51 s) to compute the extracellular potential. While the HS and EMI solutions computation time increases with the duration of the simulation linearly, as they iteratively solve each timestep, the PC solution multiplies each transmembrane current timeseries for a pre-defined mapping. When we ran a 500 ms NEURON simulation and then computed the extracellular potentials with the PC method the *PC (run)* step took only 5.38 s.

4. Discussion

In this article, we have used a detailed modeling framework—the extracellular-membrane-intracellular (EMI) model [6, 20]—to evaluate the effect of placing an extracellular recording device (neural probe) on the measured signals. We used meshes representing a simplified neuron and two different kind of probes: a microwire (a cylindrical probe with diameter of 30 μm) and multi-electrode arrays (MEAs), modeling a Neuronexus commercially available silicon probe and the Neuropixels probe [36]. We quantified the *probe effect* by simulating the domain with and without the probe in the extracellular domain and we showed that the effect is substantial for the MEA probes (figures 3(B) and (C)), while it is negligible for microwires (figure 3(A)). The amplitude of the largest peak using the MEA probes is almost twice as large (~1.9 times) compared to the case with no probe, and this factor is relatively independent of the probe distance (figure 5(D)), but it is reduced when the neuron and the probe are shifted laterally (figure 6(A)) or when the probe is rotated (figure 6(B)). Moreover, we discussed the effect of varying the mesh resolution and of the size of the computational domain. We also compared our finite element solutions to solutions obtained by solving the conventional cable equation, and found that the latter gave result very similar to the finite element solution when the probe was removed from the extracellular space (figure 7(A)). Therefore, we suggest that the *probe effect* can be a key element in modeling experimental data obtained with MEA probes. However, clearly further analysis is needed to

clarify this matter. At present the computational cost of the EMI model prevents simulations of neurons represented using realistic geometries. Thus, in an effort to offer less computationally expensive solutions to include the *probe effect* in simulations, we investigated various correction methods resulting in more accurate predictions and we proposed the probe correction method, which allows to obtain accurate solutions with reasonable computational cost and resources.

4.1. Comparison with previous work

In this work we used a finite element approach [20] to simulate the dynamics of a simplified neuron and to compute extracellular potentials using the EMI model. The use of FEM modeling for neural simulations has been performed before [19, 29, 30, 47, 48], but mainly as an advanced tool to study neural dynamics and ephaptic effects. In Moffit *et al* [47], the authors simulated, using the cable equation approach, a neuron at 65 μm from a shank microelectrode with a single recording site, and then used the currents in a finite element implementation of the extracellular domain, including the shank microelectrode. They found that the amplitude of the recorded potential with the shank was 77-100% larger than the analytical solution, but the spike shape was similar to the analytical solution (equation (13)), in accordance with our results (figures 7(A) and (B)). The effects using MEA probes and varying distances, lateral shifts, and probe rotations were not investigated. In Ness *et al* [25], an analytical framework for *in vitro* planar MEA using the method of images [24] was developed. A detailed neural model was simulated using the cable equation and transmembrane currents were used as forcing functions for a finite element simulation to validate the analytical solutions. In the *in vitro* case, in which the MEA is assumed to be an infinite insulating plane, the authors showed that the insulating MEA layer affects the amplitudes of the recorded potentials, effectively increasing it by a maximum factor of 2, which can be analytically predicted by the method of images (MoI).

In this study, we investigated how large the effect of commonly used *in vivo* probes is using the advanced EMI modeling framework. Our results are in line with these previous findings and we also show that the geometry, in terms of size and alignment of the probe, plays a very important role. We show that large silicon probes can be almost regarded as insulated planes when the neuron is aligned to them (potential increased by factor ~ 1.9) for large ranges of distances (figure 5(D)). An interesting effect following the reduction of the amplitude factor with lateral shifts (figure 6(A)) is that neurons not aligned with the probe will be recorded with a lower signal-to-noise ratio (SNR) due to the smaller amplitude increase, assuming that other sources of noise are invariant with respect to the probe location (such as electronic noise and biological noise from far neurons). This might bias neural recordings towards identifying neurons that are closer to the center of the probe, rather than the ones lying at the probes' sides. However, this conclusion is speculative and might be affected by other factors, such as the distribution of neurons

around the probe and their morphology (which contributes to the EAP). Therefore, ground truth information about the position of the recorded neurons and their reconstructed morphologies are needed for a quantitative evaluation of this phenomenon.

4.2. Limitations and extensions

4.2.1. Mesh improvements. The EMI model is, in principle, able to accurately represent the neuron and the neural probe. However, the accuracy of the model comes at the cost of computational resources. In order to be able to run simulations in a reasonable amount of time, the geometry of the neuron needed to be simplified considerably. First, we used a simple neuron in terms of a ball-and-stick with axon. This model is able to describe certain aspects of the neuronal dynamic [35], but it clearly cannot reach a level of detail of some more realistic morphologies, such as the reconstructed models made available by various initiatives [1–5]. We quantified the amplitude shift due to the probe in the extracellular domain (~ 1.9 on average for the MEA probes when neuron and probe are aligned), but this factor most likely also depend on the specific cell morphology that we used, and not only on the probe design and geometry. Therefore, we aim at extending the framework [49] for generating finite element meshes from publicly available realistic morphologies [5], allowing us to explore the *probe effect* for more complex morphologies.

Furthermore, we assumed *ideal* recording sites with an infinite input impedance which does not allow any current to flow in. In reality, recording electrodes have a high, but not infinite impedance that could be modeled by considering electrodes as an additional domain with very low conductance, even if it has been shown that for normal electrodes' impedance the effect of conductive and equipotential recording sites is negligible [32].

4.2.2. Computational costs. In section 3.4 we showed that the EMI model is *much more* computationally demanding than conventional modeling using cable and volume conduction theory. For the simplest simulation performed in this study (*coarse 3* and *box size 1*), a system with 337515 unknowns was solved in about 40 min. The NEURON simulations described in section 2.4 took ~ 0.59 s to run, about 2400 times faster than the simplest EMI simulation performed here. However, because of our implementation and solution strategy for FEM, this factor should be considered as a rather pessimistic upper bound. In particular, the employed version of FEM-ICS (2017.2.0) does not allow for finite element spaces with components discretized on meshes with different topology. For example, the extra/intra-cellular potentials are defined on the entire Ω rather than Ω_e and Ω_i only, while the domain for the transmembrane potential v is Γ , but the space for v is setup on *all* facets of the mesh. For simplicity of implementation, the v unknowns on facets outside of Γ are forced to be zero by additional constraints and are not removed from the linear system. The LU solver thus solves also for the unphysical/extra unknowns and the memory footprint and solution times

are naturally higher. The number of unphysical unknowns can be seen in table 5 as a difference between total number of facets in the mesh and the number of facets on the surface of the neuron. For example, in the largest system considered here, avoiding the unphysical unknowns would reduce the system size by about 2 million.

In addition to assembling the linear system with only the physical unknowns, a potential speed up could be achieved by employing iterative solvers with suitable preconditioners. That is, fast PDE solvers for diffusion equations typically use around 1s per million degrees of freedom. As we here employ a H(div) formulation, we expect the solution to be computed in around 5 s per million degrees with multilevel methods. As shown in table 5, 500 timesteps of solving systems with around one million degrees of freedom takes 82 600 s, which means 165 s per time steps. Hence, we may expect to speed up the solving procedure by around a factor 30 with better solvers. If further speed-up is required then finite element based reduced basis function method provides an attractive approach that should be addressed in future research.

4.2.3. Finite element methods are not alternatives to the conventional cable equation. The EMI framework, due to its computational requirements, is presently not an alternative to conventional modeling involving the cable equation (equation (12)) and the current summation formula (equation (13)). However, for specific applications, it can provide interesting insights. The hybrid solution combines the cable equation solution to finite element modeling, in practice solving the FEM problem only for the extracellular space and using the transmembrane currents computed by the cable equation as forcing functions [21–23, 25, 47]. However, the HS is also computationally expensive and it increases in complexity with longer simulation durations. Similar considerations can be made if Boundary Element Methods (BEM) [50] are employed instead of FEM ones, even though they are less computationally intense than the current FEM formulation. One possible drawback of BEM solvers is that they could not accommodate for anisotropic conductivity, while FEM solvers could in principle solve meshes with non-homogeneous conductivity between surfaces [51].

Another much faster option could be using approaches based constant scaling, such as MoI and SCS. However, even correcting with a right factor smaller than 2, these methods cannot account for the variability of peak ratios among the electrodes (figure 8(B)). Therefore, we suggested here the probe correction (PC) method, which combines a one-time finite element simulation to model how each electrode of a specific probe affects the extracellular domain, and then uses the reciprocity principle to compute the potential on the recording sites arising from transmembrane currents. We showed that this method is able to reach the HS accuracy at a much smaller computational time (table 5), which is also not strongly dependent on the simulation duration. Moreover, the time required to compute the probe specific mapping (PC (*map*)) and loading the FEM solutions in memory (PC (*load*)) could be further reduced by decreasing the mesh resolution.

This possibility should be further investigated with a convergence analysis, similar to section 3.2 for the EMI model.

4.3. Significance of the probe effect

The effect of the recording device has not been fully taken into consideration in mathematical models of the extracellular field surrounding neurons. The *probe effect* needs to be considered when modeling silicon MEA, whose sizes are significantly larger than the recorded neurons. The assumption of an infinite and homogeneous medium is in fact largely violated when such *bulky* probes are in the extracellular space in the proximity of the cells. Although the tissue can be regarded as purely conductive and with a constant conductivity [52], these probes represent clear discontinuities in the extracellular conductivity, which strongly affect the measured potential due to their insulating properties. While the *probe effect* is large for MEAs, we found that it was negligible for microwire-type of probes, mainly for two reasons: first, the microwire is thinner and overall smaller than the MEA; second, the electric potential is sampled at the tip of the probe and in the entire semi-space below the microwire currents are free to flow without any obstacle.

When dealing with silicon MEAs, though, this effect could be crucial for certain applications that require to realistically describe recordings. For example, Gold *et al* [26] used, in simulation, extracellular action potentials (EAP) to constrain conductances of neuronal models. Clearly, neglecting the *probe effect* would result in an incorrect parameterization of the models in this case.

Another example in which including this effect could be beneficial is when EAP are used to localize the somata position with respect to the probe. This is traditionally done by solving the *inverse problem*: a simple model, such as a monopolar current source [53–55], a dipolar-current source [53, 56, 57], line-source models [58, 59], or a ball-and-stick model [60], is moved around the extracellular space to minimize the error between the recorded potential and the one generated by the model. Ignoring the probe might result in larger localization errors.

Recently, we used simulated EAP on MEA as *ground truth* data, from which features were extracted to train machine-learning methods to localize neurons [27, 28] and recognize their cell type from EAPs [28]. When training such machine-learning models on simulated data and applying them to experimental data, neglecting the *probe effect* could confound the trained model and yield prediction errors.

Moreover, explaining experimental recordings on MEA without considering the probe might cause discrepancies between the modeling and experimental results hard to reconcile. On the other hand, in order to fully explain and validate our findings, an experiment with accurate co-location of extracellular recordings and cell position (and ideally morphology) is required. For example, an experimental setup in which a planar MEA is combined with two-photon calcium imaging [61] could provide an accurate estimate of the relative position between the neurons and the MEA.

In conclusion, we presented numerical evidence that suggests that the *probe effect*, especially when using multi-electrode silicon probes, affects the way we model extracellular neural activity and interpret experimental data and cannot be neglected for specific applications.

Acknowledgments

APB and KHJ are part of the Simula-UCSD-University of Oslo Research and PhD training (SUURPh) program, an international collaboration in computational biology and medicine funded by the Norwegian Ministry of Education and Research. TVN would like to acknowledge the European Union Horizon 2020 Research and Innovation Programme under Grant Agreement No. 720270 and 785907 (Human Brain Project (HBP) SGA1 and SGA2). PB would like to acknowledge the Defense Advanced Research Project Agency (DARPA), Contract No. N66001-17-C-4002. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policy of the Department of Defense of the U S Government.

ORCID iDs

Alessio Paolo Buccino  <https://orcid.org/0000-0003-3661-527X>

Miroslav Kuchta  <https://orcid.org/0000-0002-3832-0988>

Karoline Horgmo Jæger  <https://orcid.org/0000-0003-4234-9094>

Torbjørn Veffferstad Ness  <https://orcid.org/0000-0001-9080-8502>

Pierre Berthet  <https://orcid.org/0000-0002-6878-6842>

Kent-Andre Mardal  <https://orcid.org/0000-0002-4946-1110>

Gert Cauwenberghs  <https://orcid.org/0000-0002-3166-5529>

Aslak Tveito  <https://orcid.org/0000-0003-2362-146X>

References

- [1] Ramaswamy S et al 2015 The neocortical microcircuit collaboration portal: a resource for rat somatosensory cortex *Front. Neural Circuits* **9** 44
- [2] Markram H et al 2015 Reconstruction and simulation of neocortical microcircuitry *Cell* **163** 456–92
- [3] Gouwens N W, Berg J, Feng D, Sorensen S A, Zeng H, Hawrylycz M J, Koch C and Arkhipov A 2018 Systematic generation of biophysically detailed models for diverse cortical neuron types *Nat. Commun.* **9** 710
- [4] Ascoli G A 2006 Mobilizing the base of neuroscience data: the case of neuronal morphologies *Nat. Rev. Neurosci.* **7** 318
- [5] Ascoli G A, Donohue D E and Halavi M 2007 Neuromorpho.org: a central resource for neuronal morphologies *J. Neurosci.* **27** 9247–51
- [6] Tveito A, Jæger K H, Lines G T, Paszkowski Ł, Sundnes J, Edwards A G, Mäki-Marttunen T, Haldnes G and Einevoll G T 2017 An evaluation of the accuracy of classical models for computing the membrane potential and extracellular potential for neurons *Front. Comput. Neurosci.* **11** 27
- [7] Holt G R and Koch C 1999 Electrical interactions via the extracellular potential near cell bodies *J. Comput. Neurosci.* **6** 169–84
- [8] Gold C, Henze D A, Koch C and Buzsáki G 2006 On the origin of the extracellular action potential waveform: a modeling study *J. Neurophysiol.* **95** 3113–28
- [9] Carnevale N T and Hines M L 2006 *The NEURON Book* (Cambridge: Cambridge University Press) (<https://doi.org/10.1017/CBO9780511541612>)
- [10] Hines M L, Davison A P and Muller E 2009 NEURON and Python *Front. Neuroinf.* **3** 1
- [11] Lindén H, Hagen E, Leski S, Norheim E S, Pettersen K H and Einevoll G T 2014 LFPy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons *Front. Neuroinf.* **7** 41
- [12] Hagen E, Næss S, Ness T V and Einevoll G T 2018 Multimodal modeling of neural network activity: computing LFP, ECoG, EEG and MEG signals with LFPy 2.0 *Front. Neuroinform.* **12** 92
- [13] Buzsáki G, Anastassiou C A and Koch C 2012 The origin of extracellular fields and currents—EEG, ECoG, LFP and spikes *Nat. Rev. Neurosci.* **13** 407–20
- [14] Bhalla U S 2012 *Multi-compartmental models of neurons Computational Systems Neurobiology* (Berlin: Springer) pp 193–225
- [15] Goldwyn J H and Rinzel J 2016 Neuronal coupling by endogenous electric fields: cable theory and applications to coincidence detector neurons in the auditory brain stem *J. Neurophysiol.* **115** 2033–51
- [16] Anastassiou C A, Perin R, Markram H and Koch C 2011 Ephaptic coupling of cortical neurons *Nat. Neurosci.* **14** 217
- [17] Anastassiou C A and Koch C 2015 Ephaptic coupling to endogenous electric field activity: why bother? *Curr. Opin. Neurobiol.* **31** 95–103
- [18] Bokil H, Laaris N, Binder K, Ennis M and Keller A 2001 Ephaptic interactions in the mammalian olfactory system *J. Neurosci.* **21** 1–5
- [19] Agudelo-Toro A and Neef A 2013 Computationally efficient simulation of electrical activity at cell membranes interacting with self-generated and externally imposed electric fields *J. Neural Eng.* **10** 026019
- [20] Tveito A, Jæger K H, Kuchta M, Mardal K-A and Rognes M E 2017 A cell-based framework for numerical modeling of electrical conduction in cardiac tissue *Front. Phys.* **5** 48
- [21] Capogrosso M, Wenger N, Raspopovic S, Musienko P, Beauparlant J, Luciani L B, Courtine G and Micera S 2013 A computational model for epidural electrical stimulation of spinal sensorimotor circuits *J. Neurosci.* **33** 19326–40
- [22] Lubba C H, Le Guen Y, Jarvis S, Jones N S, Cork S C, Eftekhari A and Schultz S R 2018 PyPNS: Multiscale simulation of a peripheral nerve in python *Neuroinformatics* **1**–19
- [23] Lertmanorat Z and Durand D M 2004 A novel electrode array for diameter-dependent control of axonal excitability: a simulation study *IEEE Trans. Biomed. Eng.* **51** 1242–50
- [24] Jackson J D 1975 *Electrodynamics* (New York: Wiley)
- [25] Ness T V, Chintaluri C, Potworowski J, Łęski S, Głąbska H, Wójcik D K and Einevoll G T 2015 Modelling and analysis of electrical potentials recorded in microelectrode arrays (MEAs) *Neuroinformatics* **13** 403–26
- [26] Gold C, Henze D A and Koch C 2007 Using extracellular action potential recordings to constrain compartmental models *J. Comput. Neurosci.* **23** 39–58
- [27] Buccino A P, Ness T V, Einevoll G T, Cauwenberghs G and Häflliger P D 2017 Localizing neuronal somata from multi-electrode array *in vivo* recordings using deep learning *39th Annual Int. Conf. IEEE Eng. Med. Biol. Soc. (IEEE)* pp 974–7

- [28] Buccino A P, Kordovan M, Ness T V, Merkt B, Häflicher P D, Fyhn M, Cauwenberghs G, Rotter S and Einevoll G T 2018 Combining biophysical modeling and deep learning for multi-electrode array neuron localization and classification *J. Neurophysiol.* **120** 1212–32
- [29] Krassowska W and Neu J C 1994 Response of a single cell to an external electric field *Biophys. J.* **66** 1768–76
- [30] Ying W and Henriquez C S 2007 Hybrid finite element method for describing the electrical response of biological cells to applied fields *IEEE Trans. Biomed. Eng.* **54** 611–20
- [31] Henriquez F, Jerez-Hanckes C and Altermatt M F R 2013 Dynamic finite-element model of axon extracellular stimulation *6th Int. IEEE/EMBS Conf. on Neural Engineering (IEEE)* pp 589–92
- [32] Moulin C, Glière A, Barbier D, Joucla S, Yvert B, Mailley P and Guillemaud R 2008 A new 3d finite-element model based on thin-film approximation for microelectrode array recording of extracellular action potential *IEEE Trans. Biomed. Eng.* **55** 683–92
- [33] Geuzaine C and Remacle J-F 2009 Gmsh: a 3d finite element mesh generator with built-in pre-and post-processing facilities *Int. J. Numer. Methods Eng.* **79** 1309–31
- [34] Mainen Z F, Joerges J, Huguenard J R and Sejnowski T J 1995 A model of spike initiation in neocortical pyramidal neurons *Neuron* **15** 1427–39
- [35] Pettersen K H and Einevoll G T 2008 Amplitude variability and extracellular low-pass filtering of neuronal spikes *Biophys. J.* **94** 784–802
- [36] Jun J J et al 2017 Fully integrated silicon probes for high-density recording of neural activity *Nature* **551** 232
- [37] Hodgkin A L and Huxley A F 1952 A quantitative description of membrane current and its application to conduction and excitation in nerve *J. Physiol.* **117** 500–44
- [38] Rognes M E, Farrell P E, Funke S W, Hake J E and Maleckar M 2017 Cbcbeat: an adjoint-enabled framework for computational cardiac electrophysiology *J. Open Source Softw.* **2** 224
- [39] Logg A, Mardal K-A and Wells G 2012 *Automated Solution of Differential Equations by the Finite Element Method: the FEniCS Book* vol 84 (Berlin: Springer) (<https://doi.org/10.1007/978-3-642-23099-8>)
- [40] Raviart P-A and Thomas J 1977 Primal hybrid finite element methods for 2nd order elliptic equations *Math. Comput.* **31** 391–413
- [41] Amestoy P R, Duff I S, Koster J and L'Excellent J-Y 2001 A fully asynchronous multifrontal solver using distributed dynamic scheduling *SIAM J. Matrix Anal. Appl.* **23** 15–41
- [42] Balay S et al 2018 PETSc Web page (www.mcs.anl.gov/petsc)
- [43] Sterratt D, Graham B, Gillies A and Willshaw D 2011 *Principles of Computational Modelling in Neuroscience* (Cambridge: Cambridge University Press) (<https://doi.org/10.1017/CBO9780511975899>)
- [44] Ermentrout G B and Terman D H 2010 *Mathematical Foundations of Neuroscience* vol 35 (Berlin: Springer) (<https://doi.org/10.1007/978-0-387-87708-2>)
- [45] Scott A 2002 *Neuroscience: a Mathematical Primer* (Berlin: Springer) (<https://doi.org/10.1007/b98897>)
- [46] Panofsky W K and Phillips M 2005 *Classical Electricity and Magnetism* (Mineola, NY: Dover)
- [47] Moffitt M A and McIntyre C C 2005 Model-based analysis of cortical recording with silicon microelectrodes *Clin. Neurophysiol.* **116** 2240–50
- [48] Lempka S F and McIntyre C C 2013 Theoretical analysis of the local field potential in deep brain stimulation applications *PLoS One* **8** e59839
- [49] Mörschel K, Breit M and Queisser G 2017 Generating neuron geometries for detailed three-dimensional simulations using anamorph *Neuroinformatics* **15** 247–69
- [50] Gramfort A, Papadopoulos T, Olivi E and Clerc M 2010 OpenMEEG: opensource software for quasistatic bioelectromagnetics *Biomed. Eng. online* **9** 45
- [51] Clerc M, Dervieux A, Faugeras O, Keriven R, Kybic J and Papadopoulos T 2002 Comparison of BEM and FEM methods for the E/MEG problem *Proc. of BIOMAG Conf.* (Citeseer)
- [52] Goto T, Hatanaka R, Ogawa T, Sumiyoshi A, Riera J and Kawashima R 2010 An evaluation of the conductivity profile in the somatosensory barrel cortex of wistar rats *J. Neurophysiol.* **104** 3388–412
- [53] Blanche T J, Spacek M A, Hetke J F and Swindale N V 2005 Polytrodes: high-density silicon electrode arrays for large-scale multiunit recording *J. Neurophysiol.* **93** 2987–3000
- [54] Chelaru M I and Jog M S 2005 Spike source localization with tetrodes *J. Neurosci. Methods* **142** 305–15
- [55] Kubo T, Katayama N, Karashima A and Nakao M 2008 The 3d position estimation of neurons in the hippocampus based on the multi-site multi-unit recordings with silicon tetrodes *30th Annual Int. Conf. IEEE Eng. Med. Biol. Soc. (IEEE)* pp 5021–4
- [56] Mechler F, Victor J D, Ohiorhenuan I, Schmid A M and Hu Q 2011 Three-dimensional localization of neurons in cortical tetrode recordings *J. Neurophysiol.* **106** 828–48
- [57] Mechler F and Victor J D 2012 Dipole characterization of single neurons from their extracellular action potentials *J. Comput. Neurosci.* **32** 73–100
- [58] Somogyvári Z, Zalányi L, Ulbert I and Érdi P 2005 Model-based source localization of extracellular action potentials *J. Neurosci. Methods* **147** 126–37
- [59] Somogyvári Z, Cserpán D, Ulbert I and Érdi P 2012 Localization of single-cell current sources based on extracellular potential patterns: the spike CSD method *Eur. J. Neurosci.* **36** 3299–313
- [60] Ruz I D and Schultz S R 2014 Localising and classifying neurons from high density MEA recordings *J. Neurosci. Methods* **233** 115–28
- [61] Shew W L, Bellay T and Pleniz D 2010 Simultaneous multi-electrode array recording and two-photon calcium imaging of neural activity *J. Neurosci. Methods* **192** 75–82

Appendices

Appendix A

Other projects

Open source modules for tracking animal behavior and closed-loop stimulation based on Open Ephys and Bonsai

In this journal contribution to the *Journal of Neural Engineering*²³¹, we developed a system for tracking and closed-loop stimulation of rodents in open-field experiments. The system uses open-source systems for image analysis, Bonsai²³², and electrophysiology, Open Ephys²³³. We contributed featured plugins for tracking the animal position in real-time and setting up position-dependent closed-loop stimulations. We showed the reliability of the system, its capability of recording place and grid cells, and a sample use case to optogenetically stimulate a grid-cell within its grid field. More recently, we have been using the system also to selectively rewire place cells by stimulating them electrically outside of their place field.

I am the first and corresponding author for this paper, contributing in designing and conceiving the project, implementing the software, running tests and experiments, and writing most of the paper.

Scalable Spike Source Localization in Extracellular Recordings using Amortized Variational Inference

In this conference paper, recently accepted to *NeurIPS 2019*²³⁴, we presented a spike localization method for multi-electrode array recordings based on a variational autoencoder (VAE). The main idea is to use a deep network encoder and a model-based decoder to predict the 3D location of spike waveforms. We showed that the use of a VAE improves the performance of spike localization both in terms of accuracy with respect to center-of-mass estimates and it is much more computationally efficient than Markov Chain Monte Carlo (MCMC) approaches. The results were obtained both for a dense square MEA simulated with MEArec (Paper I) and for an experimental Neuropixels dataset. A precise localization of individual spike waveforms could be beneficial for spike sorting

A. Other projects

methods that use this feature for clustering^{52,53}.

My contribution to this article has been mainly in simulating the extracellular recordings, participating in discussions about the methods and results, and revising and approving the final manuscript.