



# Development of an STDP neural network for unsupervised online spike-sorting

Marie Bernert

## ► To cite this version:

Marie Bernert. Development of an STDP neural network for unsupervised online spike-sorting. Neuroscience. Université Grenoble Alpes, 2019. English. NNT : 2019GREAS001 . tel-03035855

**HAL Id: tel-03035855**

**<https://theses.hal.science/tel-03035855>**

Submitted on 2 Dec 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

## **THÈSE**

Pour obtenir le grade de

**DOCTEUR DE LA COMMUNAUTE UNIVERSITE  
GRENOBLE ALPES**

Spécialité : **Modèles, méthodes et algorithmes en biologie**

Arrêté ministériel : 25 mai 2016

Présentée par

**Marie BERNERT**

Thèse dirigée par **Blaise YVERT**, Directeur de recherche,  
**INSERM**,

préparée au sein du **Laboratoire BrainTech**  
dans l'**École Doctorale Ingénierie pour la Santé, la Cognition et  
l'Environnement**

## **Développement d'un réseau de neurones STDP pour le tri en ligne et non-supervisé de potentiels d'action.**

Thèse soutenue publiquement le **24 janvier 2019**,  
devant le jury composé de :

**M. Jean-François BECHE**

Ingénieur de recherche (CEA), Membre

**M. Luca BERDONDINI**

Professeur (IIT Genova), Rapporteur

**M. Christophe POUZAT**

Chargé de recherche (CNRS), Membre

**M. Simon THORPE**

Directeur de recherche (CNRS), Rapporteur, Président du jury

**Mme. Elisa VIANELLO**


Ingénieur de recherche (CEA), Membre

**M. Blaise YVERT**

Directeur de recherche (INSERM), Membre, Directeur de thèse







# DEVELOPMENT OF AN STDP NEURAL NETWORK FOR UNSUPERVISED ONLINE SPIKE-SORTING

PHD THESIS BY MARIE BERNERT, SUPERVISED BY BLAISE YVERT,  
DEFENDED ON JANUARY 24<sup>TH</sup> 2019





## REMERCIEMENTS

Après plus de trois ans de travail, ma thèse s'achève sur ces quelques pages. Celle-ci n'aurait pas pu avoir lieu sans la participation de différentes personnes que j'aimerais citer ici.

Tout d'abord, je voudrais remercier mon directeur de thèse, Blaise Yvert, qui a été à mes côtés tout au long de ce travail laborieux mais riche d'enseignements. J'ai pu apprécier son optimisme et ses encouragements. Je remercie également mes co-encadrants, Elisa Vianello et Jean-François Bêche, avec qui j'ai pu avoir des discussions intéressantes à différents stades de ma thèse.

Je remercie Luca Berdondini et Simon Thorpe, qui ont accepté d'être rapporteurs pour cette thèse, pour leur relecture attentive de ce manuscrit et leurs commentaires constructifs. Je remercie de même les autres membres du jury, Elisa Vianello, Jean-François Bêche et Christophe Pouzat, pour avoir accepté d'écouter et d'évaluer mon travail.

J'aimerais également remercier toutes les personnes avec qui j'ai pu collaborer durant mon travail. Je remercie Thilo Werner, encadré par Elisa Vianello, avec qui j'ai pu avoir de quelques échanges au début de ma thèse. Je remercie Takashi Kohno et Timothée Levi, qui ont eu la gentillesse de m'accueillir dans leur laboratoire au Japon pendant deux semaines, pour travailler sur une implémentation FPGA. Je remercie également Joël Fournier que j'ai eu le plaisir d'encadrer pour un stage sur ce même sujet. Enfin je remercie les personnes de l'équipe avec qui j'ai eu l'occasion de partager des données et des résultats, notamment Philémon Roussel et Florent Bocquelet.

Je voudrais aussi remercier notre gestionnaire de laboratoire Stéphanie Mollard, ainsi que l'assistante de Clnatec, Bénédicte Guehl, pour leur travail efficace et indispensable pour assurer le bon fonctionnement d'un laboratoire.

Je remercie aussi, pour leur sympathie et leur bonne humeur, tous les membres de l'équipe, Marc, Philémon, Florent, Cyril, Marie, Anne, Gaëlle, Gaël, Éric, Paul, Jean-Marie, Fanny, Dodji, Mehrdad, et tous mes collègues du laboratoire BrainTech que j'ai côtoyés pendant ces quelques années.

Enfin je voudrais remercier mes proches et ma famille qui ont toujours été là pour moi. Je remercie particulièrement mon compagnon Guillaume Pagès, qui a su m'apporter son aide et son soutien tout au long de cette thèse. Pour finir, j'ai une pensée toute particulière pour un petit bout de chou, qui m'a accompagné bien malgré lui pendant ces derniers mois de thèse.

## ABSTRACT

Pattern recognition is a fundamental task for living beings and is performed very efficiently by the brain. Artificial deep neural networks are making quick progress in reproducing these performances and have many applications such as image recognition or natural language processing. However, they require extensive training on large datasets and heavy computations. A promising alternative are spiking neural networks, which closely mimic what happens in the brain, with spiking neurons and spike-timing-dependent plasticity (STDP). They are able to perform unsupervised learning and have been used for visual or auditory pattern recognition. However, for now applications using STDP networks lag far behind classical deep learning. Developing new applications for this kind of networks is all the more at stake that they could be implemented in low power neuromorphic hardware that currently undergoes important developments, in particular with analog miniaturized memristive devices able to mimic synaptic plasticity. In this work, we chose to develop an STDP neural network to perform a specific task: spike-sorting, which is a crucial problem in neuroscience. Brain implants based on microelectrode arrays are able to record the activity of individual neurons, appearing in the recorded signal as peak potential variations called action potentials. However, several neurons can be recorded by the same electrode. The goal of spike-sorting is to extract and separate the activities of different neural cells from a common extracellular recording taking advantage of the fact that the shape of an action potential on an electrode depends on the neuron it stems from. Thus spike-sorting can be seen as an unsupervised pattern recognition task where the goal is to detect and classify different waveforms. Most classical spike-sorting approaches use three separated steps: detecting all action potentials in the signal, extracting features characterizing their shapes, and separating these features into clusters that should correspond to different neural cells. Though online methods exist, most widespread spike-sorting methods are offline or require an offline preprocessing step, which is not compatible with online application such as Brain-computer interfaces (BCI). Moreover, the development of ever larger microelectrode arrays creates a need for fully automatic and computationally efficient algorithms. Using an STDP network brings a new approach to meet these requirements. We designed a network taking the electrode signal as an input and giving out spikes that correspond to the spiking activity of the recorded neural cells. It is organized into several layers, designed to achieve different processing steps, connected in a feedforward way. The first layer, composed of neurons acting as sensory neurons, convert the input signal into spike train. The following layers are able to learn patterns from the previous layer thanks to STDP rules. Each layer implements different mechanisms that improve their performances, such as resource-dependent STDP, intrinsic plasticity, plasticity triggered by inhibition, or neuron models having rebound spiking properties. An attention mechanism has been implemented to make the network sensitive only to parts of the signal containing action potentials. This network was first designed to process data from a single electrode, and then adapted to process data from multiple electrodes. It has been tested both on simulated data, which allowed to compare the network output to the known ground truth, and on real extracellular recordings associated with intracellular recordings that give an incomplete ground truth. Different versions of the network were evaluated and compared to other spike-sorting algorithms, and found to give very satisfying results. Following these software simulations, we initiated an FPGA implementation of the method, which constitutes a first step towards embedded neuromorphic implementation.

## RESUME

La reconnaissance de motifs est une tâche cruciale pour les êtres vivants, exécutée avec efficacité par le cerveau. Les réseaux de neurones profonds artificiels reproduisent de mieux en mieux ces performances, avec des applications telles que la reconnaissance d'images ou le traitement du langage. Ils nécessitent cependant un apprentissage intensif sur de grands jeux de données et coûteux en calculs. Les réseaux de neurones à impulsions, plus proches du fonctionnement du cerveau avec des neurones émettant des impulsions et des lois d'apprentissage dites STDP dépendant du temps entre deux impulsions, constituent une alternative intéressante. Ils permettent un apprentissage non supervisé et ont déjà été utilisés pour la reconnaissance visuelle ou auditive, mais les applications restent limitées par rapport à l'apprentissage profond classique. Il est d'autant plus intéressant de développer de nouvelles applications pour ces réseaux qu'ils peuvent être implémentés sur des circuits neuromorphiques connaissant aujourd'hui des développements importants, notamment avec les composants analogiques « memristifs » qui miment la plasticité synaptique. Ici, nous avons choisi de développer un réseau STDP pour un problème crucial en neuroscience: le spike-sorting. Les implants cérébraux composés de matrices de microélectrode permettent d'enregistrer l'activité individuelle de multiples neurones, prenant la forme de pics de potentiel dans le signal, appelés potentiels d'action. Une même électrode enregistre l'activité de plusieurs neurones. Le spike-sorting a pour but de détecter et trier cette activité, en utilisant le fait que la forme d'un potentiel d'action dépend du neurone qui l'a émis. Il s'agit donc d'un problème de reconnaissance de motifs non supervisée. Les méthodes classiques de spike-sorting consistent en trois étapes : la détection des potentiels d'action, l'extraction de traits caractéristiques de leurs formes, et le tri de ces caractéristiques en groupes correspondant alors aux différentes cellules neurales. Bien que les méthodes online existent, les méthodes les plus répandues nécessitent un traitement offline, qui n'est pas compatible avec les applications temps réelles telles que les interfaces cerveau-machine (BCI). De plus, le développement de matrices de microélectrodes toujours plus denses nécessite des méthodes automatiques et efficaces. Utiliser un réseau STDP apporte une nouvelle méthode pour répondre à ces besoins. Le réseau que nous avons conçu prend en entrée le signal de l'électrode et produit en sortie un train d'impulsions qui correspond à l'activité des cellules enregistrées. Il est organisé en différentes couches, connectées en série, chacune effectuant une étape du traitement. La première couche, constituée de neurones senseurs, convertit le signal d'entrée en train d'impulsions. Les couches suivantes apprennent les motifs générés par la couche précédente grâce aux lois STDP. Chaque couche est améliorée par l'implémentation de différents mécanismes, tels que le STDP avec ressources, l'adaptation de seuil, la plasticité déclenchée par l'inhibition, ou un modèle de neurone déchargeant par rebond. Un mécanisme d'attention permet au réseau de ne traiter que les parties du signal contenant des potentiels d'action. Ce réseau a été conçu dans un premier temps pour traiter des données mono-électrode, puis adapté pour traiter des signaux provenant d'électrodes multiples. Il a été testé d'abord sur des données simulées qui permettent de comparer la sortie du réseau à la vérité, puis sur des enregistrements réels de microélectrodes associés à des enregistrements intracellulaires donnant une vérité partielle. Les différentes versions du réseau ont été ainsi évaluées et comparées à d'autres algorithmes, donnant des résultats très satisfaisants. Suite à ces résultats simulés sur ordinateur, nous avons travaillé à une implémentation FPGA, constituant une première étape vers une implémentation embarquée neuromorphique.

## CONTENT

|  |    |
|--|----|
| Remerciements .....  | 2  |
| Abstract .....   | 3  |
| Resumé .....   | 4  |
| List of figures .....  | 8  |
| List of tables .....   | 11 |
| List of abbreviations .....  | 12 |
| I. Introduction: context and goal of the thesis .....                  | 14 |
| A. Stakes of recording the brain .....                                 | 14 |
| B. Microelectrode arrays recordings and spike-sorting. ....            | 15 |
| C. Pattern recognition and artificial neural networks .....            | 16 |
| D. The advent of neuromorphic hardware for low-power computing.....    | 18 |
| E. Goal of the thesis: online spike-sorting with an STDP network ..... | 19 |
| II. Spike-sorting State of the art .....                               | 24 |
| A. Spike-sorting principle.....  | 24 |
| B. Classical methods .....   | 24 |
| 1. Detection.....  | 25 |
| 2. Feature extraction.....   | 26 |
| 3. Clustering .....  | 26 |
| 4. Template matching and other global approaches .....                 | 28 |
| C. Online vs. offline methods .....                                    | 29 |
| D. Using multiple electrodes.....                                      | 29 |
| E. Common difficulties .....   | 31 |
| 1. Bursts of action potentials .....                                   | 31 |
| 2. Non stationary data .....   | 32 |
| 3. Temporal waveform overlap.....                                      | 32 |
| F. Spike-sorting software implementations .....                        | 33 |
| III. Artificial STDP Spiking neural networks state of the art .....    | 40 |
| A. Neural code: how to encode information in a neural network .....    | 40 |
| 1. Rate coding and its limitations.....                                | 40 |
| 2. Different forms of pulse coding .....                               | 41 |
| B. Neuron models and their properties .....                            | 43 |
| C. Synaptic plasticity.....  | 45 |
| 1. Long-term STDP .....  | 45 |

|     |   |     |
|-----|---|-----|
| 2.  | Short-term plasticity .....                                   | 47  |
| 3.  | Homeostasis and metaplasticity .....                          | 48  |
| D.  | Network properties .....                                      | 49  |
| E.  | Applications to pattern recognition tasks .....               | 50  |
| IV. | Network model.....  | 58  |
| A.  | Encoding the input signal .....                               | 58  |
| B.  | Attention mechanism .....                                     | 61  |
| 1.  | Short-term plasticity .....                                   | 61  |
| 2.  | Attention neuron .....  | 63  |
| C.  | Learning of waveform elements by the intermediate layer ..... | 65  |
| 1.  | LIF neuron .....  | 65  |
| 2.  | The STDP rule used .....                                      | 66  |
| 3.  | Receptive field size.....                                     | 68  |
| 4.  | Winner-Take-All property mechanism.....                       | 70  |
| D.  | Output layer .....  | 72  |
| 1.  | Intrinsic plasticity .....                                    | 73  |
| 2.  | Lateral STDP .....  | 74  |
| 3.  | Delays and inhibition by the attention neuron.....            | 76  |
| 4.  | LTS neurons.....  | 77  |
| E.  | Adaptation to polytrodes .....                                | 81  |
| 1.  | Test on attention neuron .....                                | 82  |
| 2.  | Output layer structure .....                                  | 83  |
| V.  | Performance assessment .....                                  | 88  |
| A.  | Testing datasets.....   | 88  |
| 1.  | Simulated data .....  | 88  |
| 2.  | Real recordings .....   | 92  |
| B.  | Performance indices.....                                      | 93  |
| 1.  | Spike-sorting performance .....                               | 93  |
| 2.  | ROC curve for the attention neuron .....                      | 95  |
| 3.  | Intermediate layer quality .....                              | 95  |
| C.  | Comparison with other spike-sorting methods .....             | 96  |
| 1.  | Tests with Osort and Wave_clus.....                           | 96  |
| 2.  | Statistical tests .....                                       | 97  |
| VI. | Implementations and results .....                             | 100 |

|       |   |     |
|-------|---|-----|
| A.    | MiniNet.....                                      | 100 |
| 1.    | Implementation .....                              | 100 |
| 2.    | Results.....                                      | 102 |
| B.    | ANNet .....                                       | 104 |
| 1.    | Implementation .....                              | 104 |
| 2.    | Results.....                                      | 106 |
| 3.    | Preliminary tests on tetrode data .....           | 109 |
| C.    | LTSNet .....                                      | 110 |
| 1.    | Implementation .....                              | 110 |
| 2.    | Results.....                                      | 113 |
| D.    | PolyNet .....                                     | 117 |
| 1.    | Implementation .....                              | 117 |
| 2.    | Results.....                                      | 119 |
| E.    | Estimation of the resources used.....             | 122 |
| F.    | Preliminary FPGA Implementation.....              | 123 |
| VII.  | Conclusion and perspectives.....                  | 128 |
| A.    | Main contributions .....                          | 128 |
| B.    | Discussion and future work.....                   | 130 |
| C.    | Impact and perspectives .....                     | 131 |
| VIII. | Annexes .....                                     | 134 |
| A.    | Intermediate layer receptive field shape .....    | 134 |
| B.    | Article about ANNet .....                         | 137 |
| IX.   | Résumé en français .....                          | 156 |
| A.    | Introduction.....                                 | 156 |
| B.    | Etat de l'art des méthodes de spike-sorting ..... | 156 |
| C.    | Etat de l'art des réseaux de neurones STDP .....  | 157 |
| D.    | Modèle de réseau développé dans la thèse .....    | 159 |
| E.    | Evaluation des performances.....                  | 160 |
| F.    | Implémentations et résultats.....                 | 161 |
| G.    | Conclusion et perspectives.....                   | 162 |

## LIST OF FIGURES

|  |    |
|--|----|
| Figure I-1: Emission and propagation of an action potential in neurons.....  | 14 |
| Figure I-2: Examples of microelectrode arrays (MEA).....   | 16 |
| Figure I-3: Difference between formal neurons and spiking neurons. ....  | 17 |
| Figure I-4: Example of spike-timing-dependent plasticity, observed experimentally. ....  | 18 |
| Figure I-5: Example of neuromorphic device, including memristive synapses and Leaky-Integrate-and-Fire neurons. ....   | 19 |
| Figure II-1: Spike-sorting principle.....  | 24 |
| Figure II-2: Decomposition of spike-sorting into three main steps.....   | 25 |
| Figure II-3: Illustration of the expectation-maximization algorithm.....   | 27 |
| Figure II-4: Mean-shift algorithm illustration.....  | 28 |
| Figure II-5: Example of multiple electrodes recording.....   | 30 |
| Figure II-6: Two examples of bursts, during which the amplitude of the action potential decreases. ....  | 31 |
| Figure II-7: Example of non-stationary data during a long recording. ....  | 32 |
| Figure II-8: Example of action potential overlap. ....   | 33 |
| Figure III-1: Latency coding principle. ....   | 42 |
| Figure III-2: Illustration of polychrony.....  | 42 |
| Figure III-3: Different neuronal response to different stimuli.....  | 44 |
| Figure III-4: Experimentally observed STDP.. ....  | 46 |
| Figure III-5: Different forms of STDP rules. ....  | 46 |
| Figure III-6: STP observed experimentally.....   | 47 |
| Figure III-7: Example of spike pattern recognition by (Masquelier et al. 2008).....  | 51 |
| Figure III-8 : Example of visual pattern recognition by (Srinivasa et al. 2014).....   | 51 |
| Figure IV-1: Global structure of the STDP network.....   | 58 |
| Figure IV-2: Input layer implementation.....   | 60 |
| Figure IV-3: Modeled and simulated EPSP for each input value.....  | 62 |
| Figure IV-4: Effect of $DV_m$ and $W_{min}$ on the input value-EPSP relation, for a binary encoding. ....  | 63 |
| Figure IV-5: Detection of an action potential by the attention neuron. ....  | 64 |
| Figure IV-6: The attention neuron potential increases when an action potential is present in the input signal.....   | 65 |
| Figure IV-7: STDP rule applied on the synapse connecting the input layer to the intermediate layer. ....   | 67 |
| Figure IV-8: Receptive field shape.....  | 69 |
| Figure IV-9: Effect of the WTA mechanism. ....   | 71 |
| Figure IV-10: Effect of the k-WTA mechanism on the receptive fields. ....  | 71 |
| Figure IV-11: Comparison of the intermediate layer output quality for different k-WTA and different $DV_m$ , on the simulated single electrode dataset. ....                                   | 72 |
| Figure IV-12: Two different implementation of the intrinsic plasticity rule. ....  | 74 |
| Figure IV-13: Lateral STDP principle.....  | 75 |
| Figure IV-14: Improvement of the performance with lateral STDP implemented on the output layer of the LTSNet network.....  | 76 |
| Figure IV-15: Spike pattern received by the output layer when using a structure with transmission delays on synapses from the intermediate layer and inhibition from the attention neuron..... | 77 |



|  |     |
|--|-----|
| Figure IV-16: Example of potential rebound of the DSSN neuron model, generating a spike after the end of the stimulus.....   | 78  |
| Figure IV-17: Spiking properties of the LTS neuron after receiving a negative stimulus, with different models. ....  | 80  |
| Figure IV-18: Discrimination performance, assessed through an F-score, obtained with the LTS neuron on different conditions.....   | 81  |
| Figure IV-19: General principle of our adaptation of the network to the case of multiple electrodes  | 82  |
| Figure IV-20: Two different polytrode structures for the attention neuron.....   | 83  |
| Figure IV-21: Comparison of the ROC scores obtained with the two different structures for the attention neuron. ....   | 83  |
| Figure IV-22: Different polytrode structures tested for the output layer. ....   | 84  |
| Figure IV-23: Recall of action potential from each different neural cell, on each output sublayer, on a simulated polytrode recording with four different neural cell and an SNR of 6..... | 85  |
| Figure V-1: The two waveforms used in the preliminary dataset .....  | 89  |
| Figure V-2: The three waveforms used in the simulated single electrode dataset .....   | 89  |
| Figure V-3: Model of the transmembrane current through the neuron.....   | 90  |
| Figure V-4: Example of cells position relatively to the electrode line. The cells are placed along a line 20 $\mu\text{m}$ away from the electrode line.. ....                             | 91  |
| Figure V-5: Example of action potential waveforms obtained for each simulated electrode and each simulated neural cell .....   | 92  |
| Figure V-6: Sample of the tetrode recording d533101, with the four channels of the tetrode after filtering and the ground truth extracted from the intracellular recording. ....           | 93  |
| Figure V-7: Principle of the ROC curve .....   | 95  |
| Figure VI-1: MiniNet structure .....   | 100 |
| Figure VI-2: Qualitative results of MiniNet on a recording from the preliminary dataset.....   | 103 |
| Figure VI-3: Example of performance of MiniNet on one recording from the preliminary dataset. ...  | 104 |
| Figure VI-4: ANNet structure .....   | 105 |
| Figure VI-5: Behavior of the intermediate layer of ANNet.....  | 106 |
| Figure VI-6: Examples of ANNet output and performance .....  | 107 |
| Figure VI-7: ANNet performances compared to Osort and Wave_clus. ....  | 108 |
| Figure VI-8: ANNet performance on simulated recordings with different firing rate scenarios.....   | 109 |
| Figure VI-9 : Two different structures adapted for tetrode recordings, for the ANNet implementation. ....  | 110 |
| Figure VI-10 : F-scores obtained with the two tested structures, compared to the best single electrode performance.....  | 110 |
| Figure VI-11: LTSNet structure .....   | 111 |
| Figure VI-12: LTSNet intermediate layer behavior .....   | 114 |
| Figure VI-13: Example of potential rebound of LTS neurons in the output layer. ....  | 115 |
| Figure VI-14: Example of LTSNet output on a 5-s segment of simulated signal. ....  | 116 |
| Figure VI-15: LTSNet results on the single electrode simulated dataset, compared to ANNet, Osort and Wave_clus. ....   | 117 |
| Figure VI-16: PolyNet structure for polytrode recordings (illustration for a line of 6 electrodes). ....   | 118 |
| Figure VI-17: Example of output on a simulated polytrode recording with an SNR of 6. ....  | 120 |
| Figure VI-18: Action potentials stemming from different cells are recognized by a different output neurons.....  | 121 |

|   |     |
|---|-----|
| Figure VI-19: Mean scores on the polytrode simulated recordings with different SNR, for each type of error..... | 121 |
| Figure VI-20: Parts of the network implemented on FPGA.....   | 124 |
| Figure VI-21: Genesys 2 board used for FPGA implementation.....   | 124 |
| Figure VI-22: Attention neuron simulation implemented on an FPGA. ....  | 125 |

## LIST OF TABLES

|   |     |
|---|-----|
| Table II-1: List of existing spike-sorting software.....  | 35  |
| Table IV-1: Parameters used for the simplified LTS model.....   | 80  |
| Table IV-2: Mean F-scores obtained with the different structures tested on the simulated polytrode dataset..... | 84  |
| Table V-1: Parameters used to generate the three waveforms of the simulated single electrode dataset.....       | 89  |
| Table V-2: Noise level used and corresponding SNR for the simulated single electrode dataset .....              | 90  |
| Table V-3: Amplitude of the each neural cell's action potential for the different SNR .....                     | 91  |
| Table VI-1: Main features of MiniNet.....   | 100 |
| Table VI-2: Detailed parameters of MiniNet .....  | 101 |
| Table VI-3: Main features of ANNet .....  | 105 |
| Table VI-4: Main features of LTSNet .....   | 111 |
| Table VI-5: Detailed parameters of LTSNet .....   | 111 |
| Table VI-6: Resources used for MiniNet.....   | 122 |
| Table VI-7: Resources used for ANNet .....  | 122 |
| Table VI-8: Resources used for LTSNet.....  | 123 |

## LIST OF ABBREVIATIONS

|             |                                   |
|-------------|-----------------------------------|
| <b>ANN</b>  | <b>Artificial Neural Network</b>  |
| <b>BCI</b>  | Brain Computer Interface          |
| <b>FPGA</b> | Field-Programmable Gate-Array     |
| <b>IP</b>   | Intrinsic Plasticity              |
| <b>LIF</b>  | Leaky-Integrate-and-Fire          |
| <b>LTS</b>  | Low Threshold Spiking             |
| <b>MEA</b>  | Micro-Electrode Array             |
| <b>ROC</b>  | Receiver Operating Characteristic |
| <b>SNN</b>  | Spiking Neural Network            |
| <b>SNR</b>  | Signal-to-Noise Ratio             |
| <b>STDP</b> | Spike-Timing-Dependent Plasticity |
| <b>STP</b>  | Short-Term Plasticity             |



## I. INTRODUCTION: CONTEXT AND GOAL OF THE THESIS

### A. Stakes of recording the brain

Understanding the brain is maybe one of the most interesting challenges for scientists nowadays. Will our intelligence, some day, be able to understand its own mechanics? The long path towards this goal begins by observing what is going on in our head. The human brain is constituted of tens of billions of neurons, and thousands of times more synapses connecting them. These two elements constitute the computational bricks of the brain, connected together in an incredibly complex network that neuroscientists are trying to unravel. Anatomical observations allow us to understand how the neurons and their connections are organized in the brain. This makes us progress towards understanding how it works, but needs to be completed by functional observations, in other words measuring the neuronal activity, to understand how the information is encoded, transmitted and processed in the brain.

Neuronal activity is both electric and chemical. At its resting state, a neuron has its membrane potential at an equilibrium (around -70 mV). This potential then varies depending on the activity received from other neurons through synapses. When it increases up to a threshold value, the neuron fires. Different voltage-gated ion channels become activated, which modifies the transmembrane currents and triggers an abrupt increase of the neuron's potential followed by an abrupt decrease before returning to its resting state. This phenomenon is called action potential or spike. This action potential propagates along the neuron's axon, through similar ion channels' mechanisms. The axon is terminated by synapses through which the action potential is chemically transmitted to other neurons triggering the opening of channels that in turn modify the receptor neurons' potentials (Figure I-1). Action potentials are thus the nervous system's information carrier. Recording the spiking activity of different neurons is thus crucial to understand how the information is encoded and processed in the brain.

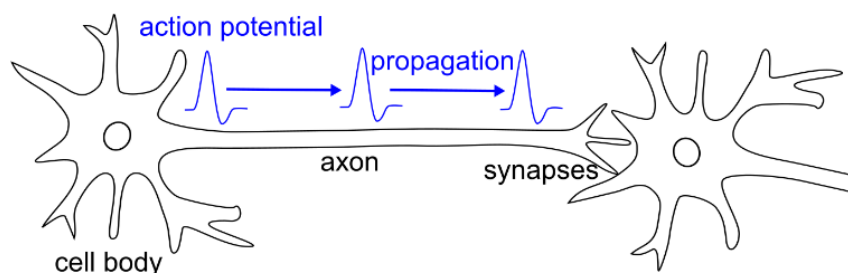


Figure I-1: Emission and propagation of an action potential in neurons

It is known that specific neural cells participate in the encoding of specific behaviors, stimuli or concepts. For example, in the visual cortex, some neural cells are sensitive to visual stimuli oriented in a specific direction, and different cells have different preferred directions. Similarly, it has been found that, in the motor cortex, some neurons code for different directions of the arm movement and that a complex movement can be decoded from the collective firing of a group of cells (Georgopoulos et al. 1986; Schwartz 1994). Spatial representation and navigation is encoded by neural cells in the

hippocampus, called place cells, that code for specific locations in an environment (Moser et al. 2008). Another study also shows that single facial characteristics are encoded by single cells (Chang & Tsao 2017). Recording individual neural spiking activity is thus crucial to analyze how neural information is encoded. Recording several neurons simultaneously also allows to understand how the spiking activity of different neurons interact to encode and to process more complex concepts (Buzsáki 2010).

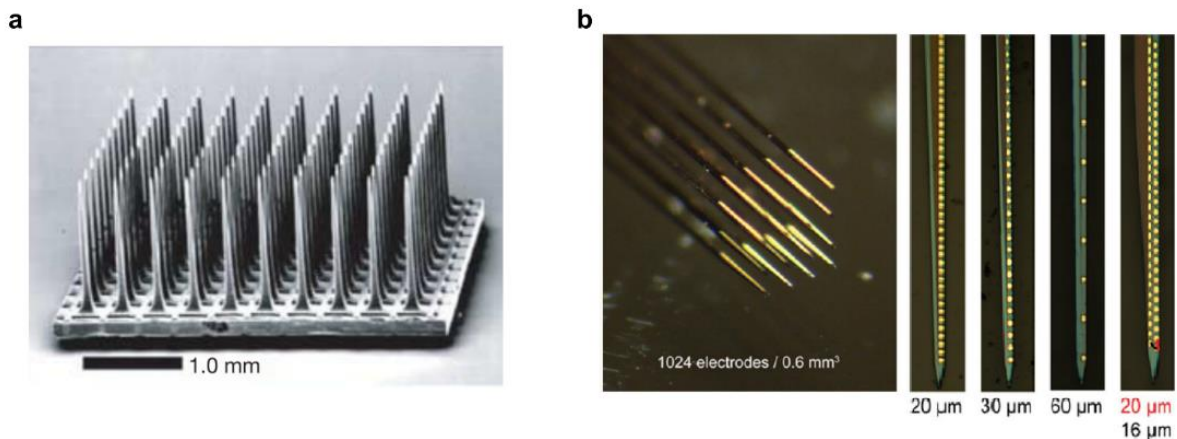
Besides improving our understanding of brain functions, decoding the information from neural recordings has practical applications such as neuro-prostheses and neuro-rehabilitation with Brain Computer Interfaces (BCIs). BCIs consist in using the decoded activity from a neural recording to control an artificial actuator. Hopes are to be able to replace defective functions with neuro-prostheses, for example with a BCI controlling a robotic arm for tetraplegic patients (Hochberg et al. 2012) or a BCI controlling a speech synthesizer for aphasic patients (Bocquelet et al. 2017). Decoding brain activity can also be used for neuro-rehabilitation, which allows to restore motor function by stimulating muscles according to the decoded activity using either electrodes on the surface of the skin (Bouton et al. 2016) or implanted ones (Ajiboye et al. 2017). In these systems, the processing of neural activity recording should be done in real-time. Moreover, the recording's quality need to be high enough to ensure the possibility to decode complex behaviors. Ideally, this implies being able to record individual spiking activity from a large number of neurons. Experimentally, it has been shown that the number of cells recorded actually improves the overall decoding quality (Wessberg et al. 2000; Ifft et al. 2013).

Recording technologies have evolved to meet these requirements. Nowadays, microelectrode arrays (MEA) are able to record the individual activity of hundreds of neurons and the technology is still improving. To fully take advantage of these advances, new algorithms have to be developed for efficiently processing the corresponding data online and in real time.

## B. Microelectrode arrays recordings and spike-sorting.

Neural activity recording technologies are diverse, each with their own advantages and drawbacks. Among the most known non-invasive methods, we can cite EEG, MEG, fMRI or PET scan. EEG measures the electrical fields generated on the scalp by the coordinated activity of neurons. MEG measures the magnetic fields outside the head. fMRI and PET scan measure metabolic changes such as the blood flow or oxygenation variations through the brain, correlated with neural activity. These non-invasive methods are useful for diagnostics and functional experiments but do not allow the recording of individual cells. In contrast, the patch clamp method consists in placing a micropipette in contact with a neural cell's membrane to record its intracellular potential. This method allows to record individual cells, but only a few neurons can be recorded simultaneously and the method is not suitable for in vivo experiments. Microelectrode arrays (MEA) offer a good solution to record the individual activity of numerous cells in vivo. They consist in micrometer-scale electrodes, often placed on sharp needles, arranged in an array, and implanted a few millimeters under the surface of the brain in the extracellular medium. When a neuron fires, the action potential induces transmembrane currents which modifies the potential of the extracellular medium. Thus an action potential can be recorded by an extracellular electrode placed close enough to the neuron (under about 100 $\mu$ m). Hence MEAs are able to record individual neural spikes.

An example of widely used MEA for in vivo experiments is the Utah array (Figure I-2.a), which contains one hundred microelectrodes. MEAs are always improving, with arrays containing up to thousands of electrodes, spaced by tens of micrometers (Alivisatos et al., 2013; Angotzi, Malerba, Zucca, & Berdondini, 2015; Lopez et al., 2016; Pothof et al., 2016; Rios, Lubenov, Chi, Roukes, & Siapas, 2016; Seidl et al., 2012) (Figure I-2.b). The increasing number electrodes on the same device allows on one hand to record more and more neurons simultaneously, opening opportunities for complex decoding. On the other hand, dense MEAs generate a huge flow of data, which require suitable processing algorithms, especially for real-time applications. Ideally, most processing should be done at the level of the electrode, to reduce the flow of data to transmit to the rest of the experimental chain.



**Figure I-2: Examples of microelectrode arrays (MEA).** (a) Utah array, adapted from (Hochberg et al. 2006). (b) Dense MEA, adapted from (Rios et al. 2016).

In particular, one type of processing that is usually done on MEA recordings is spike-sorting. As stated previously, an extracellular electrode is able to record the spiking activity of nearby neurons. Thus, the activity of the few neural cells surrounding the electrode is recorded on the same electrode signal by superposition. As the shape of the action potential recorded by the electrode depends on how the cell is positioned relatively to the electrode, it is possible to separate the spiking activity of the few recorded cells thanks to the different action potential shapes. The process is called spike-sorting. Many algorithms exist for spike-sorting, reviewed in Section II. Most of them are satisfying for offline processing and for a limited number of electrodes. However close-loop experiments require an immediate feedback from the neural data decoding, which implies real-time spike-sorting. This constraint is even more difficult to meet when using a high number of electrodes, as it increases the amount of data to process. Beside online applications, a high number of electrodes also make human supervision on spike-sorting impossible, thus requiring fully automatic methods. Therefore there is a need for new efficient online and automatic spike-sorting methods.

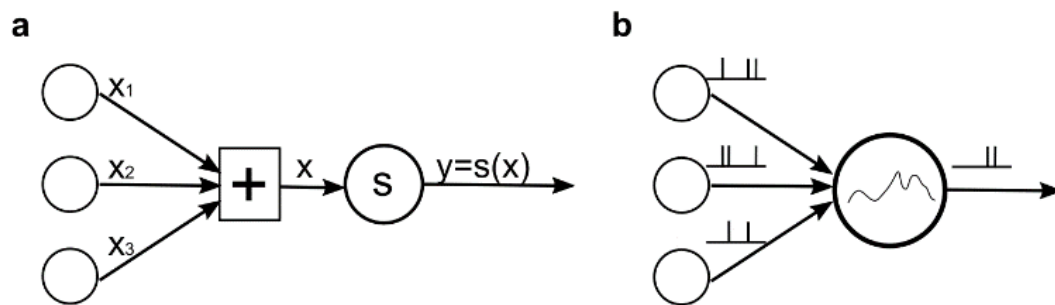
### C. Pattern recognition and artificial neural networks

The principle of spike-sorting is to detect action potentials in a signal, and to classify them according to their shapes. Spike-sorting is thus a pattern recognition problem, a field that is widely explored in artificial intelligence. The most known forms of pattern recognition are image processing or natural



language processing. Many solutions exist to solve this kind of problem, sometimes inspired by the natural ability of our brain to solve these tasks.

In particular, artificial neural networks are very popular in artificial intelligence and widely used for pattern recognition tasks. They are usually constituted of formal neurons, modeled by a mathematical activation function transforming a real input value, which is the sum of other neurons' outputs, into a real output value (Figure I-3.a). The neurons are connected together through synapses characterized by a weight, which defines how much a presynaptic neuron influences the postsynaptic neuron. The mathematical properties of these networks make the synapses' weights optimization possible through a gradient descent method, called backpropagation (Rumelhart et al. 1986), so that the network 'learns' to achieve a specific task. This neural network model is very convenient but not biologically realistic. In contrast, spiking neural networks (SNNs) use spikes to convey information. The spikes are modeled as discrete binary events transmitted between neurons through synapses. A spiking neuron receives spikes from input neurons, integrates them through a temporal dynamic and in turn emits other spikes (Figure I-3.b). Though SNNs can also be trained using a backpropagation algorithm (Bohte et al. 2002), it is possible to use biologically realistic learning rules. In particular, it has been shown experimentally, by stimulating pairs of real neurons, that the weight of a synapse changes depending on the time difference between a postsynaptic spike and a presynaptic spike (Bi & Poo 1998). This phenomenon is called spike-timing-dependent plasticity (STDP) (Figure I-4) and can be implemented in SNNs, which are then called STDP networks. STDP rules allow spiking networks to learn patterns by positively reinforcing synapses contributing to the postsynaptic activity. A review of STDP network models is given in Section III.



**Figure I-3: Difference between formal neurons and spiking neurons. (a) Formal neurons process real values through an activation function. (b) Spiking neurons process spikes through a dynamic integration.**

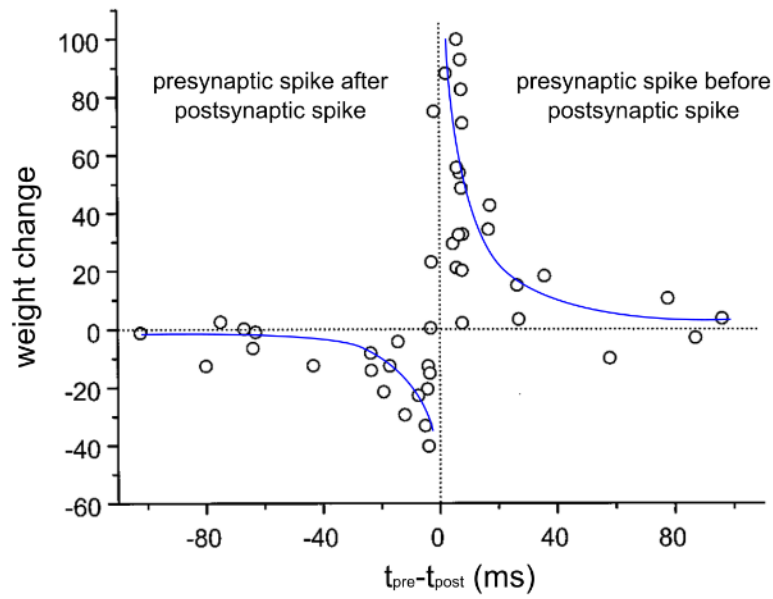


Figure I-4: Example of spike-timing-dependent plasticity, observed experimentally. Adapted from (Bi & Poo 1998)

STPD networks have been used for a few pattern recognition tasks, but applications to real world problems are still sparse. Indeed, there is at the moment no standard way to train an STDP network to achieve a specific task. However, STDP networks have other advantages compared to formal neural networks. First, STDP networks naturally allow unsupervised learning, which is essential for spike-sorting as the ground truth is unknown. Second, formal neural networks usually require training on a huge amount of data, whereas STDP networks can learn on few examples.

#### D. The advent of neuromorphic hardware for low-power computing

Though artificial neural networks are widely used for pattern recognition tasks, their main inconvenient is that they involve many neurons and many synapses working in parallel, which is computationally very demanding, especially when they are executed on traditional computers with sequential architectures. This is why neuromorphic hardware, whose purpose is to natively implement neural networks in integrated chips, currently undergoes important developments. In particular, memristive components are being developed, which are able to mimic a synapse with STDP at a miniaturized scale (Figure I-5) (Jo et al. 2010; Indiveri et al. 2013; Indiveri et al. 2015; Park et al. 2015; Saïghi et al. 2015; Rajendran & Alibart 2016; La Barbera et al. 2016; Sourikopoulos et al. 2017). This kind of devices thus opens an opportunity for STPD networks to be implemented in low-power miniaturized devices. Therefore both the unsupervised learning properties of STDP networks and their possibility to be implemented in such devices offer important perspectives for pattern recognition in the future.

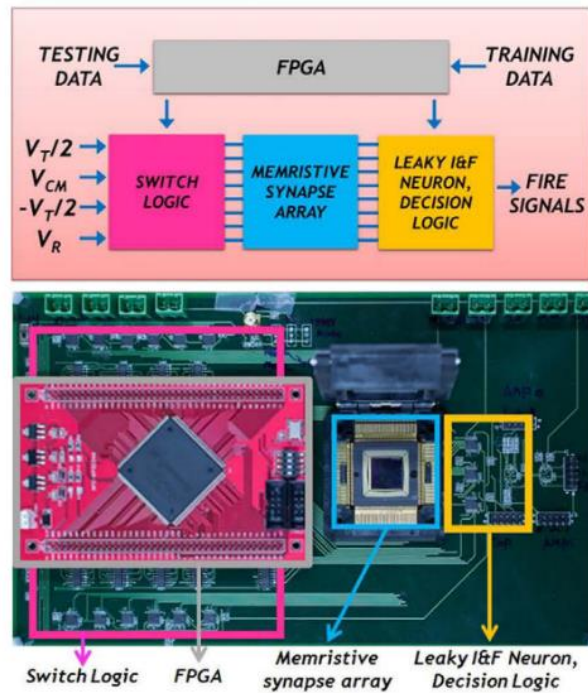


Figure I-5: Example of neuromorphic device, including memristive synapses and Leaky-Integrate-and-Fire neurons. Adapted from (Park et al. 2015)

## E. Goal of the thesis: online spike-sorting with an STDP network

In this context, the goal of this thesis work was to design a new unsupervised and online method for spike-sorting, using an STDP network. In contrast to (Zhang et al. 2015), where an STDP network was used as part of the spike-sorting method, here we aim at performing the entire spike-sorting process with an STDP network, without any pre-processing or post-processing. We also want the network to process the electrode signal online. This means that the electrode signal is directly streamed as an input to the network, and that the output spike train should correspond to the sorted spiking activity, each output spike corresponding to an action potential in the electrode signal. A preliminary study was done towards this goal (Werner et al. 2016). This network was tested on data with very high signal-to-noise ratio (SNR) but was not suitable at typical SNR found in cortical neural recordings. Moreover, it required a bank of filters to decompose the input signal, which added computation cost to the method. Here, no further processing is applied beyond the initial band-pass filter eliminating the slowly evolving local field potentials (LFP).

Designing such a network implied finding a suitable network structure, choosing neurons and synapses models for each functional parts of the network, and parameterizing correctly each element to achieve the desired goal. My work was mainly focused on processing single electrode signals, which constitutes the first step towards processing richer recordings. However, a design adapted for multiple electrodes was also tested. The problem was treated from an algorithmic point of view, leaving aside the hardware implementation but keeping in mind that the designed network should be adaptable to a neuromorphic implementation. Nevertheless, a preliminary work was done to implement the network

on a field-programmable gate-array (FPGA), which is much more adapted than a computer for parallel processing.

## References

- Ajiboye, A.B. et al., 2017. Restoration of reaching and grasping movements through brain-controlled muscle stimulation in a person with tetraplegia: a proof-of-concept demonstration. *The Lancet*, 389(10081), pp.1821–1830. Available at: <https://linkinghub.elsevier.com/retrieve/pii/S0140673617306013>.
- La Barbera, S. et al., 2016. Interplay of multiple synaptic plasticity features in filamentary memristive devices for neuromorphic computing. *Scientific Reports*, 6(1), p.39216.
- Bi, G.Q. & Poo, M.M., 1998. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(24), pp.10464–10472.
- Bocquelet, F. et al., 2017. Key considerations in designing a speech brain-computer interface. *Journal of Physiology - Paris*, 110(4), pp.392–401. Available at: <http://dx.doi.org/10.1016/j.jphysparis.2017.07.002>.
- Bohte, S.M., Kok, J.N. & La Poutré, H., 2002. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1–4), pp.17–37.
- Bouton, C.E. et al., 2016. Restoring cortical control of functional movement in a human with quadriplegia. *Nature*, pp.1–13. Available at: <http://www.nature.com/doi/10.1038/nature17435>.
- Buzsáki, G., 2010. Neural Syntax: Cell Assemblies, Synapsembles, and Readers. *Neuron*, 68(3), pp.362–385.
- Chang, L. & Tsao, D.Y., 2017. The Code for Facial Identity in the Primate Brain. *Cell*, 169(6), p.1013–1028.e14. Available at: <http://dx.doi.org/10.1016/j.cell.2017.05.011>.
- Georgopoulos, a P., Schwartz, a B. & Kettner, R.E., 1986. Neuronal population coding of movement direction. *Science (New York, N.Y.)*, 233(4771), pp.1416–1419.
- Hochberg, L.R. et al., 2006. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099), pp.164–171.
- Hochberg, L.R. et al., 2012. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398), pp.372–375. Available at: <http://dx.doi.org/10.1038/nature11076>.
- Ifft, P.J. et al., 2013. A Brain-Machine Interface Enables Bimanual Arm Movements in Monkeys. *Science Translational Medicine*, 5(210), p.210ra154-210ra154. Available at: <http://stm.sciencemag.org/cgi/doi/10.1126/scitranslmed.3006159>.
- Indiveri, G. et al., 2013. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology*, 24(38).
- Indiveri, G., Corradi, F. & Qiao, N., 2015. Neuromorphic architectures for spiking deep neural networks. In *Technical Digest - International Electron Devices Meeting, IEDM*. p. 4.2.1-4.2.4.
- Jo, S.H. et al., 2010. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Letters*, 10(4), pp.1297–1301.
- Moser, E.I., Kropff, E. & Moser, M.-B., 2008. Place Cells, Grid Cells, and the Brain's Spatial Representation System. *Annual Review of Neuroscience*, 31(1), pp.69–89. Available at: <http://www.annualreviews.org/doi/10.1146/annurev.neuro.31.061307.090723>.

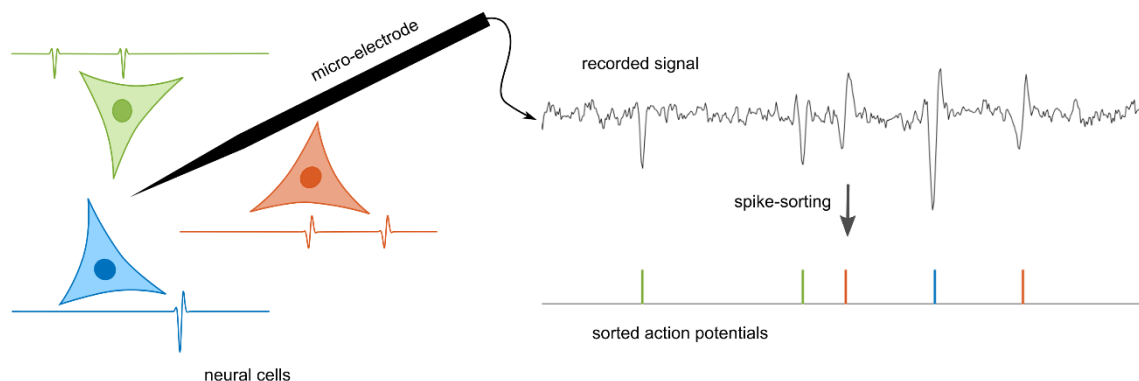
- Park, S. et al., 2015. Electronic system with memristive synapses for pattern recognition. *Scientific reports*, 5, p.10123. Available at: <http://www.nature.com/srep/2015/150505/srep10123/full/srep10123.html>.
- Rajendran, B. & Alibart, F., 2016. Neuromorphic Computing Based on Emerging Memory Technologies. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6(2), pp.198–211.
- Rios, G. et al., 2016. Nanofabricated Neural Probes for Dense 3-D Recordings of Brain Activity. *Nano Letters*, 16(11), pp.6857–6862.
- Rumelhart, D.E., Hinton, G.E. & Williams, R.J., 1986. Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland, eds. *Parallel distributed processing*. pp. 318–362. Available at: [https://web.stanford.edu/class/psych209a/ReadingsByDate/02\\_06/PDPVolIIChapter8.pdf](https://web.stanford.edu/class/psych209a/ReadingsByDate/02_06/PDPVolIIChapter8.pdf).
- Saighi, S. et al., 2015. Plasticity in memristive devices for spiking neural networks. *Frontiers in Neuroscience*, 9(MAR), pp.1–16.
- Schwartz, a B., 1994. Direct cortical representation of drawing. *Science (New York, N.Y.)*, 265(5171), pp.540–542.
- Sourikopoulos, I. et al., 2017. A 4-fJ/spike artificial neuron in 65 nm CMOS technology. *Frontiers in Neuroscience*, 11(MAR), pp.1–14.
- Werner, T. et al., 2016. Spiking Neural Networks Based on OxRAM Synapses for Real-Time Unsupervised Spike Sorting. *Frontiers in neuroscience*, 10(November), p.474. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/27857680>.
- Wessberg, J. et al., 2000. Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature*, 408(6810), pp.361–365.
- Zhang, B. et al., 2015. A neuromorphic neural spike clustering processor for deep-brain sensing and stimulation systems. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, pp. 91–97. Available at: <http://ieeexplore.ieee.org/document/7273496/>.



## II. SPIKE-SORTING STATE OF THE ART

### A. Spike-sorting principle

The signal recorded by an extracellular microelectrode contains a local field potential, action potentials, and noise. The local field potential is believed to correspond to the average current generated by cellular elements in a local area, mostly reflecting synaptic activity (Buzsáki et al. 2012). It varies slowly and can be removed with a high-pass filter (above about 200-300 Hz). After filtering, we obtain a noisy signal that contains action potentials emitted by different neurons close to the electrode. The role of spike-sorting is to detect and sort these action potentials, to obtain the individual spiking activity of each neural cell. The principle of spike-sorting mainly relies on the fact that the shape of an action potential recorded by an electrode depends on the geometries of both the neuron and the electrode, and their relative positions. Thus, on an electrode signal, action potentials with the same shape most probably come from the same neuron whereas action potentials with different shapes come from different neurons (Figure II-1). Spike-sorting methods thus use the action potential waveforms, or features extracted from these waveforms to sort the activities of different neurons. Possibly, additional information such as timing information like inter-spike interval (Delescluse & Pouzat 2006) or spatial information (Rossant et al. 2016; Hilgen et al. 2017) can be used.

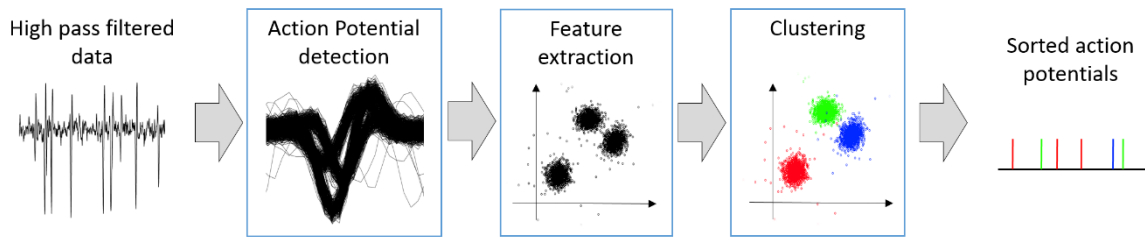


**Figure II-1: Spike-sorting principle.** An extracellular microelectrode records action potentials from several neural cells. These action potentials can be sorted as they have different shapes depending on which cell emits them.

### B. Classical methods

Once the signal has been correctly filtered, it is constituted of the superposition of noise and action potentials emitted by different neurons. Most spike-sorting methods are decomposed into three distinct steps: a detection step to find the action potentials in the signal, a feature extraction step to extract features characterizing the detected action potential shapes, and a clustering step to group the feature vectors into different clusters corresponding to the different neural cells recorded in the signal (Figure II-2). This section describes the most used algorithms for each of these steps, as well as some more global approaches to spike-sorting. A good review can also be found in (Rey et al. 2015).





**Figure II-2: Decomposition of spike-sorting into three main steps.**

### 1. Detection

The simplest way to detect action potentials in a microelectrode signal is to use a simple threshold. Each time the recorded potential crosses this threshold, it is considered that an action potential is present. This threshold can be positive or negative or both depending on the expected sign of the action potential peak. The threshold is most of the time chosen between 3 and 4 times the standard deviation of the noise, which gives a good compromise between false negatives and false positives. A first step is thus to estimate this standard deviation. In case of a Gaussian noise, the standard deviation can be estimated based on the median, with  $\sigma = \text{median}(|s|)/0.6745$ , where  $s$  is the signal (Quiroga et al. 2004; Rossant et al. 2016). This method using median is robust to the presence of action potentials in the signal.

More sophisticated methods apply a preprocessing to the signal before thresholding. The idea is to better take into account the characteristics of an action potential. Indeed even if their shape differ from one neuron to another, they share common properties such as a similar frequency content, or similar features in their shapes. Some examples of such methods are the use of an energy operator (Rutishauser et al. 2006), or the use of wavelets (Nenadic & Burdick 2005; Escola et al. 2007). In case of a template matching method (see Section II.B.4), it is also possible to use the templates for detection (Bankman et al. 1993). Template matching methods are often used in combination with a preliminary simpler detection method to establish the templates. (Delescluse & Pouzat 2006) used a detection method where a single template was computed for detection.

In case of multiple electrodes, the detection has to take into account spatial information. Indeed an action potential is recorded on several neighboring electrodes but not all electrodes. Thus the signal shape is relevant only on the electrodes where the action potential appears. The number of electrodes detecting an action potential from a neural cell can vary depending on the position and nature of the cell. As an example, (Rossant et al. 2016) used a detection algorithm where a high threshold is applied first to detect the presence of action potential. Then a lower threshold is applied and spatially connected components containing at least one high-threshold crossing potential are considered to correspond to one action potential. This dual-threshold method allows to limit false positives with the high threshold while merging correctly all electrodes involved in a same action potential thanks to the low threshold.

Once an action potential has been detected, its shape is stored for further processing. To be coherent between two occurrences of a similar action potential, it is necessary to always store the same number of samples around a reference time point. This reference point is often chosen as the maximum or minimum of the action potential, which is more robust than taking the point at which the waveform crosses the threshold as it can vary because of noise. Additionally, the signal is often up-sampled before alignment to be robust to sampling jitter.

## 2. Feature extraction

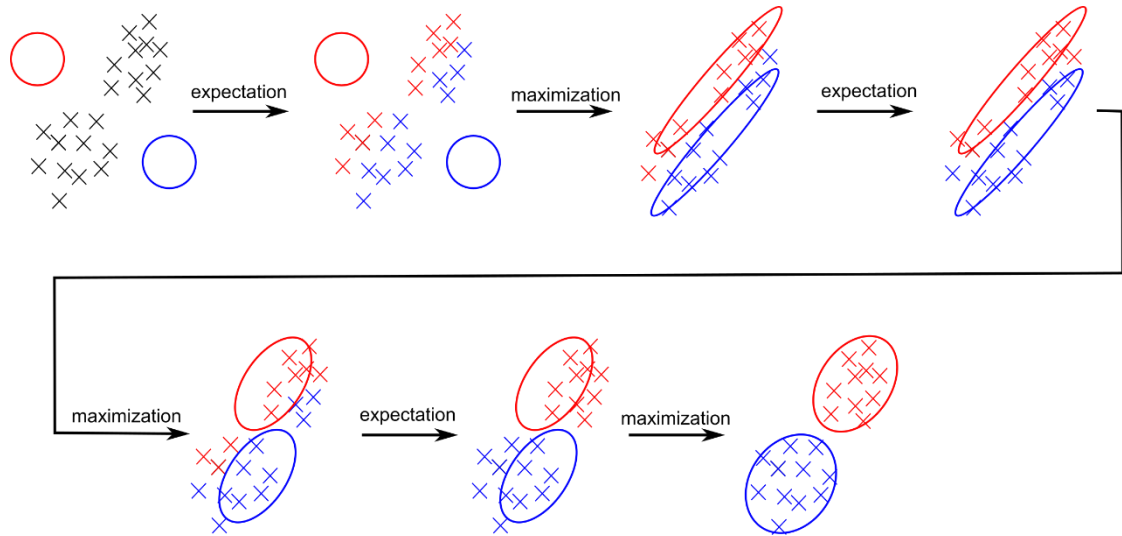
Some spike-sorting algorithms such as in (Rutishauser et al. 2006) or (Pouzat et al. 2002) or more generally template matching algorithms use directly the whole action potential waveform for clustering. However this requires to manipulate high-dimensional vectors, especially in the case of multiple electrodes, which can impair the execution time and even the clustering quality, a phenomenon known as the “curse of dimensionality”. For this reason most spike-sorting methods use a reduced number of features, extracted from the waveforms. This can be very simple features such as the amplitude or the width of the waveform. As an example, (Delescluse & Pouzat 2006) use the peak amplitudes on the four recording sites of a tetrode. Other methods use more complex features such as wavelets (Quiroga et al. 2004; Hulata et al. 2002). It is thus possible to use predefined features, however most methods use an automatic dimensionality reduction algorithm such as principal component analysis (PCA) (Shoham et al. 2003; Rossant et al. 2016; Hilgen et al. 2017), which projects the waveforms into a subspace that accounts for most of the variations between waveforms. In case of multiple electrodes, spatial information can also be used for clustering. In (Rossant et al. 2016), a vector indicating on which electrodes the action potential has been detected is used, in addition to PCA features. In (Hilgen et al. 2017), the spatial position of each action potential is estimated through a method based on barycenters and used as a feature.

## 3. Clustering

After the detection and the features extraction steps, each detected action potential is represented by a vector in an N-dimensional space. Two action potentials that are close to each other in this space are likely to have a similar waveform and thus to stem from the same neural cell. The goal is thus to sort the feature vectors into clusters, corresponding to action potential waveforms stemming from different neural cells. This last step is therefore a clustering problem, for which many solutions exist.

One classical method often used in spike-sorting is the expectation-maximization algorithm. The data to cluster is modelled as stemming from a mixture of a given stochastic distribution, most of the time a Gaussian distribution. This is an iterative algorithm, where the parameters of the data distribution are updated at each iteration. In the expectation step, each data point's label is estimated given the current model parameters. Then in the maximization step, the parameters are optimized to best fit the estimated labels. These two steps are repeated until convergence (Figure II-3) has been achieved. An improved version of this algorithm has been used for example in (Rossant et al. 2016), where each data point is associated to a mask giving information about which dimensions are the most relevant

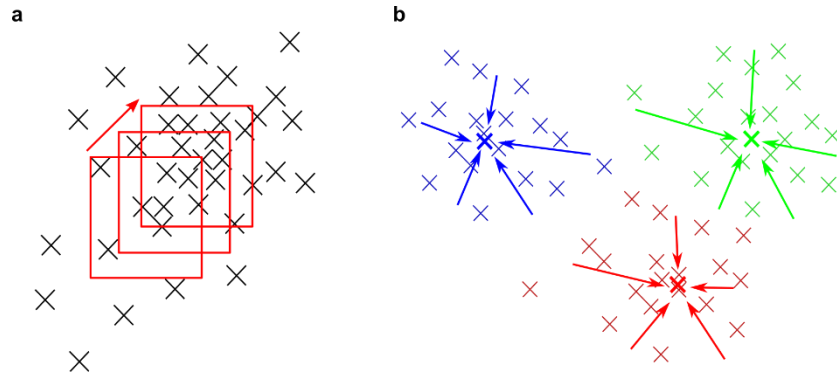
for clustering. (Shoham et al. 2003) also use the expectation-maximization algorithm on a mixture of t-distributions.



**Figure II-3: Illustration of the expectation-maximization algorithm. The expectation steps assign data points to a cluster. The maximization steps optimize the distribution's parameters to fit the data points' labels.**

Another widely used method is the K-mean algorithm. K-mean is also an iterative algorithm where the centers of the clusters are updated at each iteration: the data points are labeled according to the nearest center, and then the centers are updated as the barycenter of the corresponding points. This method has been used for example in (Chah et al. 2011). (Oliynyk et al. 2012) use a fuzzy C-means algorithm, which is a variation of the K-means algorithm where the belonging of a point to a cluster is weighted.

The expectation-maximization as well as the K-mean algorithms require the user to initially choose the number of clusters to find. This is not satisfying for spike-sorting, as the number of clusters is a priori unknown and should ideally be determined automatically by the algorithm. A way to overcome this problem is to automatically test different numbers of cluster and select the best one afterward. However other algorithms do not have this default. This is the case for example of the mean-shift algorithm, used for example in (Marre et al. 2012) and (Hilgen et al. 2017). Its principle is that for each point of the space, we define a window around this point. The barycenter of the data points within this window is computed and the window is shifted to be centered on this barycenter. The process is repeated until the window stabilizes on a local maxima (Figure II-4.a). This way, each point can be assigned to the nearest local maxima, which correspond to a cluster (Figure II-4.b). This method is almost non-parametric as the only parameter to choose is the size of the window.



**Figure II-4: Mean-shift algorithm illustration. (a) Computation of local density maxima thanks to a window, shifted to its barycenter at each step. (b) Each point a space is assigned to the nearest local density maxima, with the method presented in (a), to form clusters.**

Another example of non-parametric method is the super-paramagnetic clustering used in (Quiroga et al. 2004). The main idea of this algorithm is to randomly assign a cluster to a point, and this new cluster assignment is propagated to neighbor points with a probability depending on the distance between points. Thus points that are close to one another will tend to change their labels together. This process is repeated many times, to ensure a relevant classification.

Finally, (Zhang et al. 2015) use an STDP network as part of their clustering algorithm. This network is inspired from pattern recognition for static inputs. It has one input layer and one output layer. Each detected waveform is given as an input of the network after being converted in an input spike train through a binary encoding of the signal values. Each time a waveform is presented, an output neuron, corresponding to a cluster, is activated. During the training phase the synaptic weights are also updated for each presented waveform. The number of clusters is monitored externally by activating or deactivating neurons of the output layer. Clusters are added if too many points are not classified and clusters are removed if their corresponding neuron does not fire for a long time.

#### 4. Template matching and other global approaches

Separating the spike-sorting problem into three steps provides a simple methodology. However these steps are not independent and the performance of one step might depend on how the previous steps are solved. A global approach to the spike sorting problem might lead to better performances. As a simple example, it is much easier to detect an action potential in a noisy signal if we know its exact waveform. Most methods that use such a global approach are based on the use of templates corresponding to each different action potential waveforms, and are thus called template matching methods. These methods often rely on an initial clustering step to initialize the templates, usually using a classical three-step method as presented previously. However, it is not necessary in this preprocessing step neither to find all the action potentials, nor to assign all the found action potentials to a cluster, but simply to find the centroids of the clusters that will constitute the templates. (Yger et al. 2016) is an example of such a template matching method. After establishing a set of templates, the scalar product of the signal with each time-shifted version of each template is computed. The template and shift time with the highest scalar product is selected and considered as a found and classified action potential. This process is then iterated, as part of a greedy algorithm. In (Franke et al. 2010),

the signal is considered as the sum of the convolution of the ground truth spike trains with waveforms. The templates estimated by the preprocessing step are used to establish filters that approximate the deconvolution of the signal. (Ekanadham et al. 2014) presents an even more global approach, in the sense that they optimize simultaneously both the template waveforms and the action potential times and amplitudes, to maximize the probability to have such amplitude and spike times given the waveforms and observed signal. More precisely, they alternate a step optimizing the spike times and amplitudes and a step optimizing the action potential waveforms.

### C. Online vs. offline methods

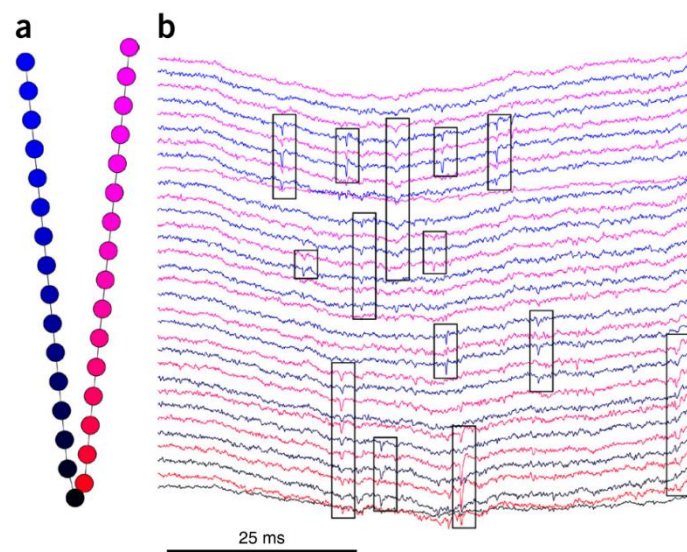
Most spike-sorting methods require some offline processing. Though offline spike-sorting can be used for post-experiment data analysis, it is necessary for closed-loop experiments to have an online real-time spike-sorting method. For a classical three step methods, if at least one of the steps requires an offline processing, then the spike-sorting method cannot be executed online without adaptation. The detection step can usually be executed online, as the thresholding and the possible preprocessing operator are local operations that can be applied online. Extracting predefined features from a waveform, such as its amplitude, can also be done online. On the other hand, automatic dimensionality reduction algorithms such as PCA require a consequent number of data points, and thus cannot be applied online. However, is it possible to compute the projection as an offline preprocessing step and then apply this projection online. Most clustering algorithms, such as expectation-maximization, K-means or mean-shift, also require an offline processing step for the same reason. A noticeable exception is given by (Rutishauser et al. 2006) whose clustering algorithm can be applied online. At each detection of an action potential, the corresponding waveform is assign to a cluster according to the distance with the cluster centroid, and the centroid of the assigned cluster is then updated. Template matching algorithm are often suitable for online execution, though the initialization of the template might require an initial short offline preprocessing step. This is the case of the method presented in (Franke et al. 2010) which can be applied online once the templates have been defined.

Another factor that can jeopardize real-time spike sorting is the computation time. In addition to intrinsically allow an online processing, a real-time spike-sorting method should also be able to detect and classify an action potential within a few milliseconds, which is the duration of an action potential. In particular, the clustering part can be very time-consuming if too many dimensions are used. This time constraint becomes particularly critical when the number of recording electrodes is important as in currently-developed cortical implants, generating large amounts of data to process.

### D. Using multiple electrodes

Using large-scale multi-electrode implants allows to record more neural cells, and brings more information for spike-sorting but requires a suitable spike-sorting method. In the case of an electrode array where electrodes are far enough from each other to record completely different neurons, such as Utah arrays, each electrode can be processed independently as a single electrode. In contrast, a

few electrodes very close to one another, such as tetrodes, may record the same neural cells, and the set of electrodes can thus be processed together as a single electrode by simply concatenating the features. In between, using dense microelectrode arrays brings an interesting spatial dimension to the spike-sorting problem. Indeed, in that case, each electrode detects the activity of a few neural cells, as for a single electrode, but each neural cell is also recorded by several electrodes but not all the electrodes of the array (Figure II-5). Hence action potentials are spatiotemporal events. They are limited both in time, as they last about a millisecond, and in space, as they are visible on a few neighboring electrodes. This spatial dimension is all the more important as, because of the high number of recorded neural cells, many action potentials will overlap in time. However they can be easily discriminated if they do not overlap in space. This spatial aspect thus affects all the steps of the spike-sorting algorithm. (Lefebvre et al. 2017) gives good review about existing multi-electrode spike-sorting algorithms.



**Figure II-5: Example of multiple electrodes recording, adapted from (Rossant et al. 2016). (a) Geometry of the electrode array. (b) Sample of recorded signal. Action potentials are highlighted with rectangles.**

Detection should not only give the timestamp of the action potential event but also information about its location. A common way to perform a non-redundant detection is to search for local minima (or maxima) both in time and space, which is done for example in (Yger et al. 2016). Each event thus correspond to one timestamp and one electrode. (Rossant et al. 2016) has a different approach, where events are described as spatially connected components over a neighborhood graph. The events are detected as crossing a high threshold and then extended to neighboring electrodes crossing a lower threshold. This spatial information should then be used for clustering. In (Rossant et al. 2016), the electrodes detected as belonging to the action potential event are represented in a mask vector, whose components are equal to one for these electrodes, and to zero for the others. This mask is then used in a masked expectation-maximization algorithm to indicate where the relevant values are, the irrelevant ones being replaced by a random distribution corresponding to noise. (Yger et al. 2016) used a very practical approach. As each event is associated to one electrode, the clustering is done independently on each electrode. Some clusters can be merged afterward if similar action potentials sometimes have their peaks on one electrode and some other times on another neighboring one. An interesting approach is used in (Hilgen et al. 2017), as the location of an action potential is processed

like another feature. Indeed for each action potential event, a barycenter is computed and the coordinates are used as features in a mean-shift algorithm. Template matching methods are not much affected by the use of multiple electrodes. Indeed, once the templates have been established, they implicitly contain the information about the action potential location, as the template is null on electrodes where the action potential is not visible. The spatial aspect is thus mainly used in the initialization of the templates.

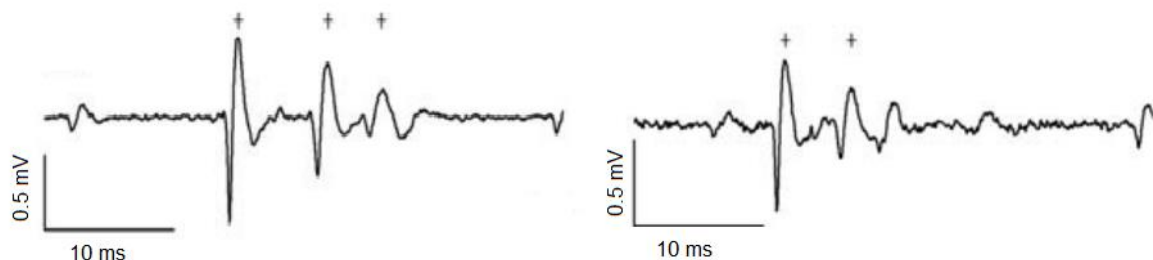
Another particularity of spike-sorting on multiple electrodes is that the noise can be correlated across different electrodes. Knowing the correlation matrix between the electrodes, it is possible to apply a linear transformation to the multiple-electrode signal to obtain a new signal with the same number of dimensions but where the dimensions are not correlated. This process, known as noise whitening, is used in many spike-sorting methods (Delescluse & Pouzat 2006; Marre et al. 2012; Ekanadham et al. 2014; Yger et al. 2016) and can also be used to remove temporal correlations.

## E. Common difficulties

In the previous section we presented the general principles of spike-sorting and some classical algorithms to solve it. However, while designing a spike-sorting method, one can be confronted to several difficulties specific to neural recordings. In particular the fact that the action potentials emitted by one neural cell have each time the same shape is not always true. Here we present some classical known difficulties encountered in spike-sorting and methods to overcome them.

### 1. Bursts of action potentials

Neural cells sometimes fire in burst, which means they emit several spikes within a short period of time. The problem is that the amplitude of the action potential tends to decrease at each occurrence within a burst (Figure II-6). The assumption that the shape of the action potential does not vary is thus not true in this case.



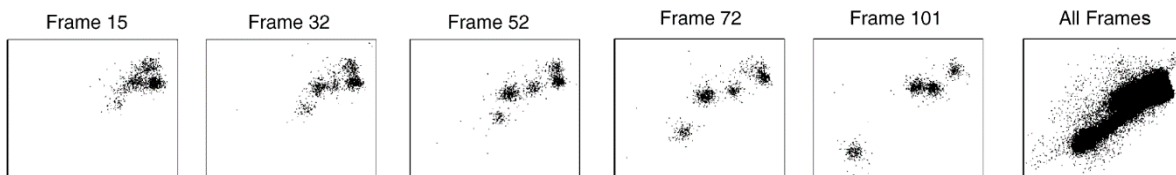
**Figure II-6: Two examples of bursts, during which the amplitude of the action potential decreases. Adapted from (Delescluse & Pouzat 2006)**

To tackle this problem, some spike-sorting algorithms use the weakest assumption that the action potential from a given neural cell always has the same base waveform, but modulated in amplitude. The method presented in (Yger et al. 2016) is a template matching algorithm that uses this assumption.

An action potential with a different amplitude can still be detected with the corresponding template, and its amplitude is computed after detection and classification. (Franke et al. 2010) and (Ekanadham et al. 2014) also use templates modulated by an amplitude in their algorithm. In both cases the amplitude estimation is also used to estimate correctly the templates. (Delescluse & Pouzat 2006) used a different strategy, as the only features used in their algorithm are the action potentials' amplitudes and not their precise shapes. Their algorithm relies on an explicit model of the action potential amplitude depending on the inter-spike interval, an information that is not used in the previously presented algorithms.

## 2. Non stationary data

Another situation where the assumption that waveforms do not change is violated is the case of long recordings, during which electrodes can slowly drift, causing slow changes in the action potential waveforms (Figure II-7). A common way to solve the problem is to slice the recording into chunks that are short enough to assume that there are no significant waveform changes within one chunk. This is done for example in (Franke et al. 2010). Their method is an online template matching method, for which the templates are updated at the end of every chunk. (Bar-hillel et al. 2006) also divides the recording into chunks. The action potentials are modeled as a chain of Gaussian mixtures, and the method combines an expectation-maximization algorithm with a Bayesian method to link the mixtures of each chunk. In (Calabrese & Paninski 2011), another variation of the expectation maximization is used, by combining it with a Kalman filter applied on the Gaussian mixture parameters.

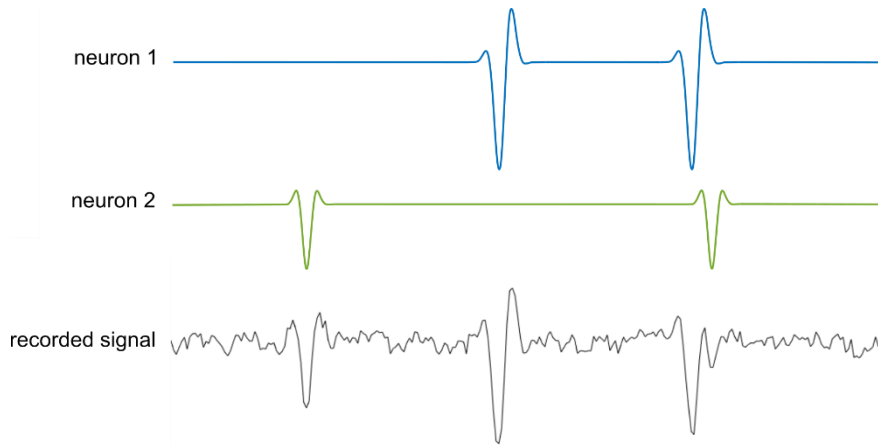


**Figure II-7: Example of non-stationary data during a long recording, adapted from (Bar-hillel et al. 2006). The projection of detected action potentials' waveforms on two dimensions are shown at different times of the recording. The waveforms slowly evolve with time. When all action potentials are observed together (last picture), it is not possible to distinguish clusters.**

## 3. Temporal waveform overlap

When action potentials from different neural cells occur in a short time interval, their waveforms may overlap in the signal. The resulting waveform is the sum of the different waveforms, which often makes it impossible to recognize for traditional algorithms (Figure II-8).





**Figure II-8: Example of action potential overlap. The first two action potentials are not overlapping and are clearly recognizable. The last two action potentials are overlapping and sum up into a new waveform that cannot be classified easily.**

A first type of approach to solve this problem is to proceed by iterations, as this is done in (Yger et al. 2016). Their method is a greedy template matching algorithm. At each iteration the template and time position obtaining the highest template matching score is selected and the corresponding waveform is then removed from the signal. Thus if at least one of two overlapping action potentials is detected, the second one can also be detected once the first one has been removed from the signal. The second type of approach is to use an algorithm based on a model that directly takes into account that action potential waveforms are summed. This is the case of (Franke et al. 2010). The first step of the algorithm is to filter the signal with filters based on templates, which is a linear operation. The second step is an independent component analysis (ICA), which by nature takes into account the fact that signals are summed. Their study shows that the method is effectively robust to overlapping action potentials. In (Ekanadham et al. 2014), the signal is modeled as the sum of templates multiplied by an amplitude coefficient, the amplitude being null when there is no action potential. The goal is thus to find the templates and their amplitudes for each sampling time. The amplitudes and the templates are alternatively optimized, using respectively a gradient descent and a least-squares method. The common point between these two methods, that makes them robust to overlapping action potentials, is that they do not consider action potentials independently but all together. Noticeably, methods able to disentangle overlapping waveforms are methods that have a global approach in contrast to methods decomposed in the three classical steps. Indeed once a detection step has been applied, the detected waveform is considered to correspond to one action potential. Therefore an action potential corrupted by another action potential cannot be classified correctly.

## F. Spike-sorting software implementations

Spike-sorting algorithms are numerous and diverse, and software implementations of spike-sorting methods are even more diverse. A list of existing spike-sorting software can be found on the following web page, maintained by Simon Kornblith and reproduced in Table II-1: <http://simonster.github.io/SpikeSortingSoftware/>. Some software applications implement a specific

spike-sorting algorithm, some others offer several combinations of the different processing steps. For similar algorithms, the choice of the language and implementation optimizations can have a strong impact on the execution time. The user interface design also plays an important role in the software usage. Despite the diversity of existing software applications, only a few allow a fully unsupervised online spike-sorting, and the execution time often becomes important for large MEA.

Table II-1: List of existing spike-sorting software. Adapted from <http://simonster.github.io/SpikeSortingSoftware/>

| Software                  | Language       | Detection  | Feature Extraction   | Clustering   | Drift | Overlap | Large-scale MEA | Publications                                 | Comments   |
|---------------------------|----------------|--|--|--|-------|---------|-----------------|--|--|
| BinaryPursuitSpikeSorting | MATLAB         | binary pursuit   | N/A  | binary pursuit   | No    | Yes     | No              | (Pillow et al. 2013)                         |  |
| bpsort                    | MATLAB         | binary pursuit, raw signal threshold with alignment (initialization) | PCA (initialization)   | t-distribution MM (initialization)                                 | Yes   | Yes     | Yes             |  |  |
| CBPSpikesortDemo          | MATLAB         | continuous basis pursuit   | continuous basis pursuit   | continuous basis pursuit   | No    | Yes     | Yes?            | (Ekanadham et al. 2014)                      |  |
| ClusterLizard             | C++            | raw signal threshold   | wavelets + Lillifors test  | Euclidean distance   | Yes   | No      | No              | (Knieling et al. 2016)                       |  |
| Combinato                 | Python         | raw signal threshold with alignment                                  | Wavelets   | superparamagnetic clustering + template matching                   | Yes   | No      | No              | (Niediek et al. 2016)                        | GUI for inspecting clusters                      |
| EToS                      | C++            | raw signal threshold   | multimodality-weighted PCA, multimodality pick-up algorithm, Graph Laplacian features, PCA | Variational Bayes and EM t-distribution and Gaussian mixture model | No    | No      | No              | (Takekawa et al. 2010; Takekawa et al. 2012) |  |
| FMMSpikeSorter            | MATLAB         | raw signal threshold   | focused mixture model  | focused mixture model  | Yes?  | No      | Yes?            | (Carlson et al. 2014)                        |  |
| gpu_python                | Python         | N/A  | N/A  | Generalized Polya urn dependent Dirichlet process MM               | Yes   | No      | No              | (Gasthaus et al. 2009)                       |  |
| KFMM                      | MATLAB         | N/A  | N/A  | Kalman filter EM GMM   | Yes   | No      | No              | (Calabrese & Paninski 2011)                  |  |
| KiloSort                  | MATLAB, CUDA C |  | Spatiotemporal SVD   | Template matching via stochastic batch optimization                | No    | Yes     | Yes             | (Pachitariu et al. 2016)                     |  |
| MoDT                      | MATLAB, CUDA C | N/A  | PCA  | Mixture of drifting t-distributions                                | Yes   | No      | No              | (Shan et al. 2017)                           |  |
| moksm                     | MATLAB         | N/A  | N/A  | Mixture of drifting t-distributions                                | Yes   | No      | No              | (Shan 2014)                                  |  |
| opass                     | MATLAB         | gamma process model  | PCA  | gamma process model  | Yes   | Yes     | Yes?            | (Carlson et al. 2013)                        |  |
| OpenElectrophy            | Python         | raw signal threshold, MTEO   | PCA, ICA, wavelets   | EM GMM, K-means, mean-shift  | No    | No      | No              |  | GUI for manual clustering and inspecting results |

|                             |             |  |   |   |      |     |     |  |   |
|-----------------------------|-------------|--|---|---|------|-----|-----|--|---|
| OSort                       | MATLAB      | local energy threshold with alignment                    | N/A                                     | template matching   | Yes  | No  | No  | (Rutishauser et al. 2006)                    |   |
| pebble                      | C++, MATLAB | ?  | PCA                                     | ISO-SPLIT   | No   | No  | No  | (Magland & Barnett 2016)                     |   |
| phy (previously klustakwik) | Python      | raw signal threshold with alignment                      | PCA                                     | EM GMM, masked EM GMM   | No   | No  | Yes | (Kadir et al. 2014; Rossant et al. 2016)     | GUI for inspecting sorting results. Large, active community       |
| spikesort                   | MATLAB      | raw signal threshold (height + width)                    | PCA, factor analysis, sparse PCA, t-SNE | 1D GMM, 1D k-means  | No   | No  | No  |  | Claims "99.5% accuracy." Only supports clustering into two units. |
| SpikeSorter.jl              | Julia       | hidden markov model                                      | spike width, trough to valley ratio     | hidden markov model   | No   | No  | No  |  | Algorithm described in Spike sorting with hidden Markov models    |
| SpikeSorting.jl             | Julia       | raw signal threshold, power, nonlinear energy, alignment | PCA                                     | OSort-style template matching                                   | Yes? | No  | No  |  | Many options planned; see documentation                           |
| spyke                       | Python      | raw signal threshold with alignment                      | PCA, ICA                                | gradient ascent (mean-shift variant)                            | No   | No  | Yes | (Spacek et al. 2009; Swindale & Spacek 2014) | GUI   |
| SpyKING Circus              | Python      | raw signal threshold + iterative template matching       | PCA                                     | local density clustering (Rodriguez & Laio) + template matching | No   | Yes | Yes | (Yger et al. 2016)                           |   |
| tridesclous                 | Python      | raw signal threshold with alignment                      | PCA                                     | EM GMM, k-means   | No   | No  | No  |  |   |
| Wave_clus                   | MATLAB      | raw signal threshold with alignment                      | wavelets + Lillifors test               | superparamagnetic   | No   | No  | No  | (Quiroga et al. 2004)                        |   |

## References

- Bankman, I.N., Johnson, K.O. & Schneider, W., 1993. Optimal Detection, Classification, and Superposition Resolution in Neural Waveform Recordings. *IEEE Transactions on Biomedical Engineering*, 40(8), pp.836–841.
- Bar-hillel, A., Spiro, A. & Stark, E., 2006. Spike sorting : Bayesian clustering of non-stationary data. , 157, pp.303–316.
- Buzsáki, G., Anastassiou, C. a & Koch, C., 2012. The origin of extracellular fields and currents--EEG, ECoG, LFP and spikes. *Nature reviews. Neuroscience*, 13(6), pp.407–20. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/22595786>.
- Calabrese, A. & Paninski, L., 2011. Kalman filter mixture model for spike sorting of non-stationary data. *Journal of Neuroscience Methods*, 196(1), pp.159–169. Available at: <http://dx.doi.org/10.1016/j.jneumeth.2010.12.002>.
- Carlson, D.E. et al., 2014. Multichannel electrophysiological spike sorting via joint dictionary learning and mixture modeling. *IEEE Transactions on Biomedical Engineering*, 61(1), pp.41–54.
- Carlson, D.E. et al., 2013. Real-Time Inference for a Gamma Process Model of Neural Spiking. *Advances in Neural Information Processing Systems* 26, pp.1–9. Available at: <http://papers.nips.cc/paper/5061-real-time-inference-for-a-gamma-process-model-of-neural-spiking>.
- Chah, E. et al., 2011. Automated spike sorting algorithm based on Laplacian eigenmaps and k-means clustering. *Journal of neural engineering*, 8, p.016006.
- Delescluse, M. & Pouzat, C., 2006. Efficient spike-sorting of multi-state neurons using inter-spike intervals information. , 150, pp.16–29.
- Ekanadham, C., Tranchina, D. & Simoncelli, E.P., 2014. A unified framework and method for automatic neural spike identification. *Journal of Neuroscience Methods*, 222, pp.47–55. Available at: <http://dx.doi.org/10.1016/j.jneumeth.2013.10.001>.
- Escola, R. et al., 2007. Wavelet-based scale-dependent detection of neurological action potentials. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, pp. 1888–1891. Available at: <http://ieeexplore.ieee.org/document/4352684/>.
- Franke, F. et al., 2010. An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes. *Journal of Computational Neuroscience*, 29(1–2), pp.127–148. Available at: <http://link.springer.com/10.1007/s10827-009-0163-5>.
- Gasthaus, J. et al., 2009. Dependent Dirichlet Process Spike Sorting. *Advances in Neural Information Processing Systems (NIPS)*, pp.497–504.
- Hilgen, G. et al., 2017. Unsupervised Spike Sorting for Large-Scale, High- Density Multielectrode Arrays. *Cell Reports*, 18(10), pp.2521–2532. Available at: <http://dx.doi.org/10.1016/j.celrep.2017.02.038>.
- Hulata, E., Segev, R. & Ben-Jacob, E., 2002. A method for spike sorting and detection based on wavelet packets and Shannon's mutual information. *Journal of Neuroscience Methods*, 117(1), pp.1–12.
- Kadir, S., Goodman, D. & Harris, K., 2014. High-dimensional cluster analysis with the masked EM algorithm. *Neural computation*, 1872, pp.1840–1872.
- Knieling, S. et al., 2016. An Unsupervised Online Spike-Sorting Framework. *International Journal of Neural Systems*, 26(05), p.1550042. Available at:

- <http://www.worldscientific.com/doi/abs/10.1142/S0129065715500422>.
- Lefebvre, B., Yger, P. & Marre, O., 2017. Recent progress in multi-electrode spike sorting methods. *Journal of Physiology Paris*. Available at: <http://dx.doi.org/10.1016/j.jphysparis.2017.02.005>.
- Magland, J.F. & Barnett, A.H., 2016. Unimodal clustering using isotonic regression : IOS-SPLIT. *Arxiv*.
- Marre, O. et al., 2012. Mapping a Complete Neural Population in the Retina. *Journal of Neuroscience*, 32(43), pp.14859–14873. Available at: <http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.0723-12.2012>.
- Nenadic, Z. & Burdick, J.W., 2005. Spike detection using the continuous wavelet transform. *Bio-Medical Engineering*, 52(1), pp.74–87.
- Niediek, J. et al., 2016. Reliable analysis of single-unit recordings from the human brain under noisy conditions: Tracking neurons over hours. *PLoS ONE*, 11(12), pp.1–26.
- Oliynyk, A. et al., 2012. Automatic online spike sorting with singular value decomposition and fuzzy C-mean clustering. *BMC Neuroscience*, 13(1), p.96.
- Pachitariu, M. et al., 2016. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *bioRxiv*, p.061481. Available at: <http://biorxiv.org/lookup/doi/10.1101/061481>.
- Pillow, J.W. et al., 2013. A Model-Based Spike Sorting Algorithm for Removing Correlation Artifacts in Multi-Neuron Recordings. *PLoS ONE*, 8(5), p.e62123. Available at: <http://dx.plos.org/10.1371/journal.pone.0062123>.
- Pouzat, C., Mazor, O. & Laurent, G., 2002. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. *Journal of Neuroscience Methods*, 122(1), pp.43–57.
- Quiroga, R.Q., Nadasdy, Z. & Ben-Shaul, Y., 2004. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8), pp.1661–1687.
- Rey, H.G., Pedreira, C. & Quiroga, R.Q., 2015. Past, present and future of spike sorting techniques. *Brain Research Bulletin*, pp.1–12. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0361923015000684>.
- Rossant, C. et al., 2016. Spike sorting for large, dense electrode arrays. *Nature neuroscience*, 19(4), pp.634–641. Available at: <http://biorxiv.org/content/early/2015/02/16/015198%5Cnhttp://biorxiv.org/content/early/2015/02/16/015198.full.pdf>.
- Rutishauser, U., Schuman, E.M. & Mamelak, A.N., 2006. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of Neuroscience Methods*, 154(1–2), pp.204–224.
- Shan, K., 2014. EM for a mixture of drifting t-distributions. , pp.1–7.
- Shan, K.Q., Lubenov, E. V. & Siapas, A.G., 2017. Model-based spike sorting with a mixture of drifting t-distributions. *Journal of Neuroscience Methods*, 288, pp.82–98.
- Shoham, S., Fellows, M.R. & Normann, R. a., 2003. Robust, automatic spike sorting using mixtures of multivariate t-distributions. *Journal of Neuroscience Methods*, 127(2), pp.111–122.
- Spacek, M., Blanche, T. & Swindale, N., 2009. Python for large-scale electrophysiology. , 2(January), pp.1–10.
- Swindale, N. V & Spacek, M.A., 2014. Spike sorting for polytrodes : a divide and conquer approach. ,

8(February), pp.1–21.

Takekawa, T., Isomura, Y. & Fukai, T., 2010. Accurate spike sorting for multi-unit recordings. *European Journal of Neuroscience*, 31(2), pp.263–272.

Takekawa, T., Isomura, Y. & Fukai, T., 2012. Spike sorting of heterogeneous neuron types by multimodality-weighted PCA and explicit robust variational Bayes. *Frontiers in Neuroinformatics*, 6(March), pp.1–13. Available at: <http://journal.frontiersin.org/article/10.3389/fninf.2012.00005/abstract>.

Yger, P., Spampinato, G.L.B. & Esposito, E., 2016. Fast and accurate spike sorting in vitro and in vivo for up to thousands of electrodes. *bioRxiv*, pp.1–21.

Zhang, B. et al., 2015. A neuromorphic neural spike clustering processor for deep-brain sensing and stimulation systems. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, pp. 91–97. Available at: <http://ieeexplore.ieee.org/document/7273496/>.

### III. ARTIFICIAL STDP SPIKING NEURAL NETWORKS STATE OF THE ART

The most known type of artificial neural networks are formal neural networks, which are very popular in artificial intelligence. STDP networks differ from this type of neural networks on two main points. First, STDP networks are spiking neural networks (SNN). They are constituted of neurons emitting spikes, which are discrete events characterized by their time of emission, whereas formal neurons output a real value with no temporal information. In SNNs, information is thus conveyed through sequences of spikes, also called spike trains. Second, formal neural networks are usually trained to perform a task through a global supervised or reinforcement algorithm, optimizing the weights of the synapses connecting the neurons. STDP networks implement a spike-timing-dependent plasticity (STDP), which is a local learning rule, inspired from biological synapses (Bi & Poo 1998), which modifies a synapse weight depending on the time difference between spikes emitted by the presynaptic and the postsynaptic neurons. From these two points of view, STDP networks are thus closer to biological reality than formal neural networks. For this reason, they are often used to investigate how information is processed in the brain either in an effort to reproduce a specific neural function, or to study their computational properties. They also have been used to solve concrete pattern recognition problems in different application fields. In this section we will review different ways of encoding information in spiking neural network, neuron models and synaptic plasticity models and their properties, some important network properties and some existing applications of STDP networks to pattern recognition tasks.

#### A. Neural code: how to encode information in a neural network

Understanding how the information is encoded in the brain is a fundamental question in neuroscience, for which there is, at the moment, no clear answer. Through time, many possible coding schemes have been proposed. When designing an artificial neural network, whether to test a model or for a computational application, the choice of how the information is encoded is crucial as it deeply impacts the network mechanics.

##### 1. Rate coding and its limitations

Historically, the most widely used neural code is rate coding. The idea of rate coding is that the information is contained in the firing rate of each neuron and not in the precise spike times. When dealing with spiking neurons, their firing rates can be computed by averaging the number of spikes emitted within a short time period. Other possibilities are to average the firing rate over a population of neurons, which requires an entire population to code for the same variable, or to average over several runs of an experiment. The advantage of rate coding is that the firing rate takes continuous values, varying continuously in time and therefore is easy to manipulate. An illustrative example is the population vector, which can be used to represent the direction of movement encoded by an ensemble of cortical motor neurons (Georgopoulos et al. 1986; Schwartz 1994). This gave birth to the second generation of artificial neural networks. In contrast with the first generation where the neurons' outputs are binary, here the neurons' outputs are real values, obtained through an activation function



of the weighted sum of their inputs. This kind of neural network has the strong advantage that they can be trained for a specific task using a backpropagation algorithm (Rumelhart et al. 1986).

The efficiency of the backpropagation algorithm explains its popularity in artificial intelligence applications, however it is not biologically realistic and cannot explain learning in the brain. A strong argument supporting the fact that the firing rate cannot be the only way the information is encoded in the nervous system is that the speed at which some tasks are executed is incompatible with the time necessary to estimate a firing rate (Gerstner et al. 1993; Thorpe & Gautrais 1997; Johansson & Birznieks 2004; VanRullen et al. 2005). For example a visual task takes about 150ms to complete whereas, it would take about 100ms at each layer to integrate a reliable firing rate (Thorpe & Gautrais 1997). Another argument is the fact that rate coding constitutes a loss of information relatively to pulse coding, where individual spike times are taken into account. As individual spike times carry more information than an average firing rate, spiking networks are thus theoretically more powerful computationally (Maass 1997a; Maass 1997b; Gerstner et al. 1993). Although the rate-coding paradigm has shown its efficiency with formal neural networks, a pulse-coding paradigm is naturally more relevant for spiking neural networks, and probably closer to biological reality.

## 2. Different forms of pulse coding

Pulse coding can be declined into different forms, depending on how the information contained in the spike times is interpreted. A first important aspect is how an analog stimulus can be encoded into a spike train. The second aspect of pulse coding is how information carried in the spike train is used by the network.

A first method to encode an analog stimulus into spikes is the latency code (Vanrullen & Thorpe 2002; Masquelier & Thorpe 2007; Thorpe & Gautrais 1997). The principle of latency coding is an intensity-latency conversion, which means that the latency of the first spike emitted by a neuron relatively to the onset of the stimulus depends on the intensity of the stimulus (see Figure III-1). This is quite natural when using integrate-and-fire neurons as neurons integrating stronger stimulus fire first. (Johansson & Birznieks 2004) show experimental evidence that this kind of code is actually found in sensory neurons. Latency coding requires a reference time, from which the latency is computed. It has been proposed that one possible reference signal could be the oscillatory brain activity, thus constituting a variation of latency coding called phase coding (Hopfield 1995; McLelland & Paulsen 2009; Masquelier 2014). It is indeed possible, by using a LIF neuron and an oscillatory signal to convert an analog stimulus or a rate code into a phase code. Again, phase coding has also been observed experimentally (McLelland & Paulsen 2009).

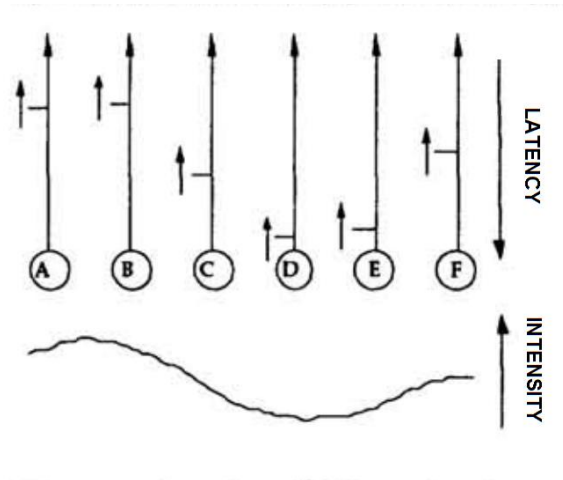


Figure III-1: Latency coding principle. Adapted from (Thorpe & Gautrais 1997)

The second important point in neural code is how spike time information is used in the network. In some case, the relevant information is carried by the strongest stimuli and thus by the early-arriving spikes. This paradigm is used in (Masquelier & Thorpe 2007), where the information of the first spike time among sets of neurons is used for image processing. More generally, network where the relevant information is carried by the spikes' firing order are said to use a rank-order code (Vanrullen & Thorpe 2002; Thorpe & Gautrais 1997) and are used when the earliest spikes are the most important ones. This is emphasized in (Vanrullen & Thorpe 2002) where an image processing algorithm is modeled using rank-order coding and where most of the original image can be reconstructed using only a small fraction of the spikes, i.e. those emitted early after the stimulus onset. In some other cases, the exact spike times are crucial for the network mechanics. For example, a network can use spike synchrony as a relevant pattern, which means that several neurons coding for different variables need to fire at the same time to trigger a spike in another neuron (Singer 1993). This is natural for Leaky-Integrate-and-Fire (LIF) neurons that need to receive enough spikes simultaneously to fire. More generally, the relevant information can be contained in the spike time differences between several neurons, meaning that input neurons need to fire with precise spike time differences for the spike sequence to be recognized. This property was termed as "polychrony" by (Izhikevich 2006). To achieve this polychrony, a crucial element is the transmission delay between neurons, which makes it possible for spikes emitted by different neurons at different times to arrive simultaneously on the same readout neuron, as illustrated in Figure III-2 (Gerstner et al. 1993; Hopfield 1995; Izhikevich 2006).

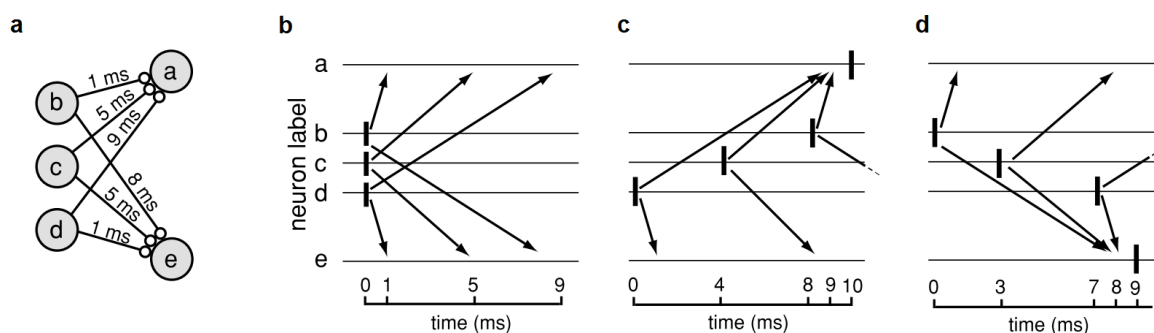


Figure III-2: Illustration of polychrony. Adapted from (Izhikevich 2006). (a) Synaptic connections with different transmission delays. (b) Synchronous firing do not elicit postsynaptic spikes. (c) Example of spike pattern triggering a postsynaptic spike in neuron a. (d) Example of spike pattern triggering a postsynaptic spike in neuron.

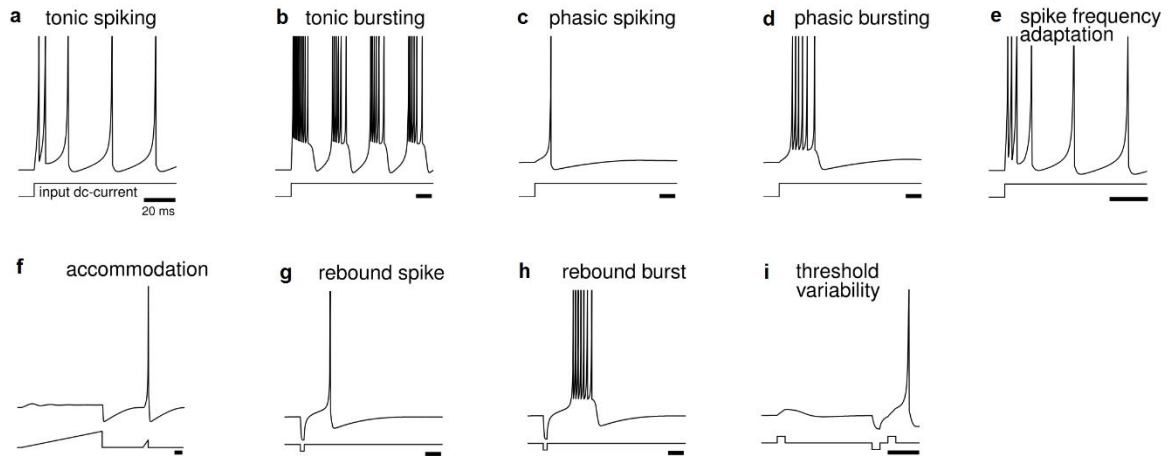
Overall, whereas the rate code is simple to use due to its analog nature, pulse codes are more difficult to manipulate and require to make choices on how the time information is used but can lead to possibly more powerful computational properties.

## B. Neuron models and their properties

Neurons are one of the two key elements, with synapses, of an artificial neural network. In spiking neural network, they obey to a temporal dynamic which can be inspired from what happens with real neural cells in the brain.

Several types of neurons exist in the brain and they exhibit different behaviors. (Hodgkin 1948) studied the response of axons to stimuli and established a behavioral classification: class 1 axons, which have a wide range of spiking frequency response depending on the stimulus strength, and class 2 axons, whose frequency response is relatively insensitive to the stimulus strength. Since then, the behavioral classification of neurons has been refined, the most known classes of neurons being the regular spiking neurons (RS), the intrinsically bursting neurons (IB), the fast spiking neurons (FS) and the low-threshold spiking neurons (LTS) (Connors & Gutnick 1990; Gibson et al. 1999; Izhikevich 2003; Pospischil et al. 2008). The behavior of a neuron is determined by its response to different kinds of stimuli. (Izhikevich 2004) gives a good review of the different possible responses to different stimuli. In particular, the response can be:

- tonic spiking or bursting when the neuron fires or bursts continuously when stimulated (Figure III-3.a and b);
- phasic spiking or bursting when the neuron fires or bursts only once at the beginning of the stimulation (Figure III-3.c and d);
- frequency adaptation when the neuron adapts to the stimulus by decreasing its firing frequency during a maintained stimulus (Figure III-3.e);
- accommodation when the neuron is not sensitive to slowly increasing stimulus but only sharp ramps (Figure III-3.f);
- rebound spike or burst when the neuron can fire or burst after an inhibitory stimulus (Figure III-3.g and h);
- threshold variability when the neuron's threshold can vary depending on its previous activity (Figure III-3.i).



**Figure III-3: Different neuronal response to different stimuli. Adapted from (Izhikevich 2004)**

When designing an artificial spiking neural network, one needs to find a neuron model able to reproduce the desired behavior. A very biologically realistic model was established by (Hodgkin & Huxley 1952), which modeled precisely all the ionic currents within an axon during the generation of an action potential. This model is highly configurable and can reproduce any kind of observed behavior. However its complexity makes it difficult to use and computationally demanding. Depending on the goal to achieve, a neuron model does not necessarily need to be highly biologically meaningful, but to reproduce efficiently some neural behavior. Neuron models can be divided into several main types. First, models that, like the Hodgkin-Huxley model, reproduce precisely the neuron's membrane potential dynamics, including the action potential dynamic. Some of these models are biologically meaningful, such as (Morris & Lecar 1981), while others focus on reproducing efficiently the neuron dynamic (FitzHugh 1961; Rose & Hindmarsh 1989; Nanami & Kohno 2016). Second, models that describe the subthreshold potential dynamics but handle spikes as discrete events occurring when the neuron potential reaches its threshold. These models, sometimes called integrate-and-fire models, have the advantage to greatly reduce the computational cost, as the rapid dynamic of an action potential does not need to be computed (Izhikevich 2004). The most widely used neuron model of this type is the Leaky-Integrate-and-Fire (LIF) model, whose potential evolution is simply described by the following equation:  $\tau \frac{du}{dt} = -u(t) + RI(t)$ . The evolution of the potential  $u$  combines a leak with a time constant  $\tau$  and an integration of the input current  $I$ . When the potential  $u$  reaches a fixed threshold level, a spike is emitted and the potential is reset to a predefined value. Many variations of this model have been developed, with more variables to induce interesting firing properties (Smith et al. 2000; Izhikevich 2001; Brette & Gerstner 2005). An noticeable model of this type was developed by Izhikevich (Izhikevich 2003), which, in spite of its simplicity, is able to mimic many different neuronal behaviors. Another type of model that process spikes as discrete events is the Spike Response Model (SRM). The SRM expresses the potential of the neuron as a function of time that depends on the previously received and emitted spikes. The spike times are still defined as the times when the potential reaches a threshold.

Many different neural behaviors and many different neuron models exist. Choosing a model depends on which behavior needs to be reproduced with which degree of accuracy and with which efficiency.

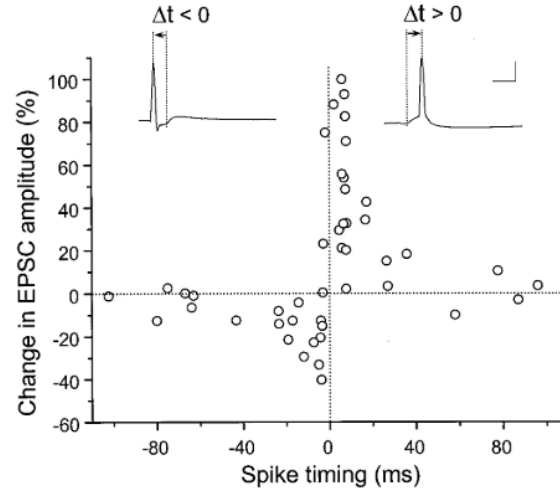
## C. Synaptic plasticity

The ability of neural networks to learn is due to their synaptic plasticity, in other word, the ability of synapses to change their weight. Indeed, in a spiking neural network, synapses transmit spikes from a presynaptic neuron to a postsynaptic neuron, and the weight of a synapse affects its ability to influence its postsynaptic neuron. At the network's level, the synaptic plasticity can thus modify the network's response to different stimuli. According to Hebb's postulate, the weight of a synapse evolves following local plasticity rules depending on the activities of the post and presynaptic neurons. In most models, these weight changes are induced by presynaptic and postsynaptic spikes, and depend on the time difference between them. This kind of plasticity is thus called spike-timing-dependent plasticity (STDP). The most common STDP rules induce changes triggered by pairs of pre- and postsynaptic spikes, but changes can also happen on single spike or depending on more than two spikes. The changes can be persistent, for long-term plasticity, or non-persistent for short term plasticity. The following sections summarize the most common models of plasticity.

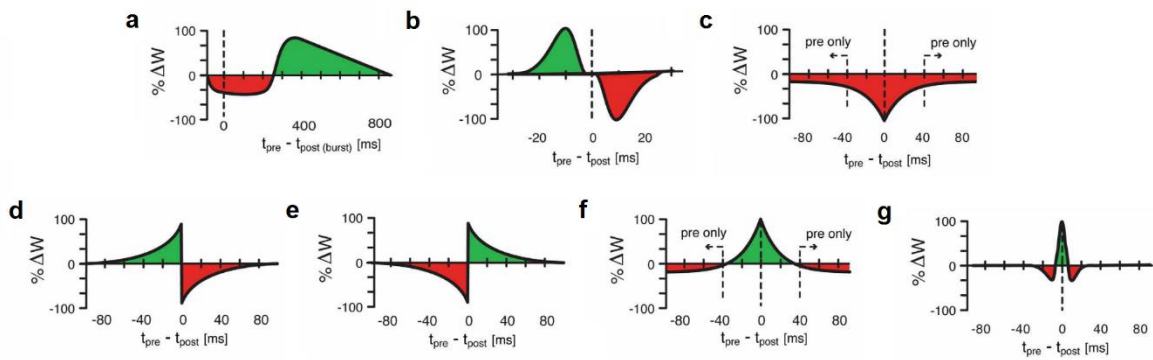
### 1. Long-term STDP

The most known spike-timing-dependent plasticity rule is the long-term plasticity induced by pairs of presynaptic and postsynaptic spikes. This form of plasticity was first observed experimentally by (Bi & Poo 1998). When presynaptic and postsynaptic spikes separated by a given time interval are repeatedly induced, a persistent change in the strength of the synapse is observed. The amount of change depends on the time interval separating the presynaptic spike and the postsynaptic spike, which constitutes the core principle of pair-based STDP rules.

In the experiment from (Bi & Poo 1998), the weight change is positive when the presynaptic spike precedes the postsynaptic spike, corresponding to a long term potentiation (LTP), and negative in the other case, corresponding to a long term depression (LTD) (see Figure III-4). The weight change is large when this time interval is short and tends to zero when the time interval is long. This is coherent with Hebb's postulate (Hebb 1949) that the connection between a cell repeatedly participating is the activation of another cell is reinforced. In STDP cases, presynaptic spikes shortly preceding postsynaptic spikes are likely participating in triggering the postsynaptic spike, and thus reinforce the synapse's weight. The dependence between the spike time interval and the weight's change can be, in this case, modeled as a double exponential function. Since then, other shapes for this time dependence have been observed and used in models such as symmetric dependences or anti-hebbian plasticity where the time dependence is reversed compared to the classical time dependence (Woodin et al. 2003; Luz & Shamir 2012; Vogels et al. 2013; D'amour & Froemke 2015; Abbott & Nelson 2000; Srinivasa et al. 2014) (see Figure III-5).



**Figure III-4: Experimentally observed STDP.** Adapted from (Bi & Poo 1998). The weight's change is positive for a presynaptic spike preceding a postsynaptic spike and negative otherwise.



**Figure III-5: Different forms of STDP rules, observed experimentally (a,b,c), or used in models (d,e,f,g).** Adapted from (Vogels et al. 2013)

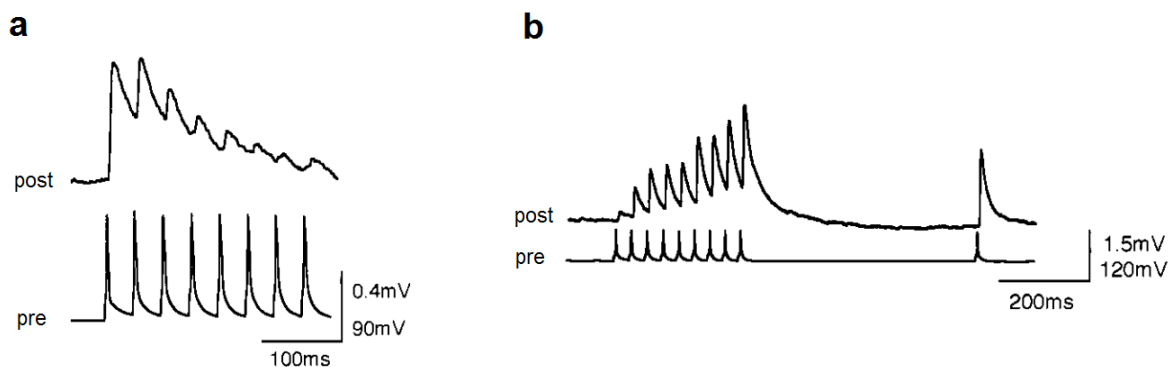
The weight's change applied on a synapse can either depend or not, on the current weight of the synapse. A common way to model a weight dependence is to make the weight depression proportional to the current weight  $w$ , and the weight potentiation proportional to  $(1-w)$ . This kind of weight dependence is often called multiplicative STDP, in contrast to additive STDP for a synaptic plasticity rule that does not depend on the current weight. When used in a spiking neural network, multiplicative and additive STDP do not have the same properties (Rubin et al. 2001; Gütiç et al. 2003). Indeed, when using additive STDP, synapses converge towards either their maximum or their minimum weights, whereas multiplicative STDP leads to continuously distributed equilibrium weights.

When dealing with realistic spike trains, more than one postsynaptic and presynaptic spikes contribute to the synapse plasticity. A pair-based STDP model thus has to specify which spikes' pairs have to be taken into account. There are many possibilities such as the all-to-all pairing scheme, where all possible combinations are taken into account, or a nearest-neighbor scheme where only neighboring spikes are taken into account. These different pairing schemes can be implemented using local variables (Morrison et al. 2008), which avoids having to store past spikes.

Experiments such as triplet and quadruplet experiments (Froemke & Dan 2002; Wang et al. 2005) or frequency dependence experiments (Sjöström et al. 2001) show the limitation of the pair-based model to explain synaptic plasticity when multiple spikes are involved. For example, in a triplet protocol, three spikes are presented to the synapse, either in a pre-post-pre scheme or a post-pre-post scheme. According to a pair-based model, the resulting weight change should be the same with both schemes, which is not what is observed. New models have been developed to explain these experiments, such as the suppression model (Froemke & Dan 2002), or the triplet model (Pfister & Gerstner 2006). Both take into account previous postsynaptic and presynaptic spikes to modulate the impact of a pre-post or post-pre pairs on the synapse weight.

## 2. Short-term plasticity

Long-term STDP induces persistent weight changes depending on the correlation between postsynaptic and presynaptic activities. These change are persistent in the sense that in the absence of activity, the synapses' weights remain unchanged. On the contrary, short-term plasticity (STP) is a plasticity triggered by transmission of presynaptic spikes that induces non-persistent changes, as the synapse's weight return to a baseline value in the absence of activity. As for long-term plasticity, short-term plasticity has been observed on real synapses through numerous experiments (Markram et al. 1998; Zucker & Regehr 2002). Short-term plasticity may result in either a depression, when each presynaptic spike decreases the synapse weight (Figure III-6.a), or a facilitation when each presynaptic spike temporarily increases the synapse's weight (Figure III-6.b). These phenomena can be explained by the fact that the transmission of spikes through a synapse depends on some resources, whose quantity is modified at each spike transmission. This idea of resource dependence has been used to developed models of short-term plasticity, which are able to reproduce both facilitation and depression (Abbott et al. 1997; Tsodyks et al. 2000). As changes induced by short-term plasticity only take effect during a short period and rapidly fade with time, it cannot be considered as directly participating in learning a task. However it has interesting properties that contribute to the network's efficiency. Indeed, short-term plasticity is able to regulate the postsynaptic potential induced by input spikes, which leads to a better discrimination of changes in the input spike train (Abbott et al. 1997).



**Figure III-6: STP observed experimentally. Adapted from (Markram et al. 1998). (a) Example of short term depression. (b) Example of short term facilitation. Bottom traces show the presynaptic spikes, top traces show the postsynaptic potential.**

### 3. Homeostasis and metaplasticity

To be computationally relevant, the learning process in a neural network should both be stable, which means synapses' weights and neurons' activity should not diverge nor become totally null, and induce some selectivity, which means that depending on the input activity and the network structure, some synapses are potentiated whereas others are depreciated in order to have specific responses to different stimuli. Most STDP rules constitute a positive reinforcement, as synapses that contribute to make a postsynaptic neuron fire are potentiated. This process can thus be unstable if not correctly tuned. A standard way to obtain a stable network is to choose an STDP rule that induce more LTD than LTP in the absence of correlation (Masquelier et al. 2008; Humble et al. 2012; Srinivasa et al. 2014). Some other STDP rules can inherently be stable yet lack the competitive property. This is the case for example of multiplicative STDP for which the synapses' weights naturally converge but in a unimodal distribution (Rubin et al. 2001; Gütiğ et al. 2003). More generally, obtaining a compromise between the stability and the competition requires a fine tuning of the learning rules (Yger & Gilson 2015; Babadi & Abbott 2016; Gütiğ et al. 2003).

Therefore, it can be interesting, both for obtaining realistic models or for computational properties, to introduce complementary plasticity rules regulating the STDP rule. These kinds of rules are known as homeostatic mechanisms, as they allow to obtain homeostasis, i.e. stability of the network firing rates through learning. Whereas STDP rules are homosynaptic, depending on the pre- and postsynaptic activities on a given synapse, homeostatic mechanisms can be heterosynaptic, as changes on one synapse can influence neighboring synapses. They also act at a longer time scale than the STDP rule they regulate (Zenke et al. 2013; Yger & Gilson 2015). There is some evidence that plasticity acting at different timescales and on more than single synapses actually takes place in the brain, though such plasticity can take many different forms, whether modifying directly the synapse weight, or modifying the neuron excitability or modulating STDP rules (Abbott & Nelson 2000). They can also have different purposes such as stabilizing the neuronal activity, but also improving the selectivity or allowing reinforcement learning.

Homeostatic mechanisms acting directly on the synapse weight are also known as synaptic scaling. The principle of synaptic scaling is to limit or to impose the total weight of synapses arriving on the same postsynaptic neuron, thus stabilizing learning by preventing the total weight to grow up indefinitely and encouraging selectivity by preventing all synapses from fading. Models implementing a synaptic scaling often simply renormalize the synaptic weight at each weight change (Malsburg 1973; Lazar 2009; Aswolinskiy & Pipa 2015; Humble et al. 2012). Homeostatic mechanisms can also modify the neuron's excitability, a process sometimes called intrinsic plasticity. (Zhang & Linden 2003) suggest that intrinsic plasticity plays an important role in the brain. Its main function is to regulate the activity of individual neurons. In models such as (Lazar 2009; Aswolinskiy & Pipa 2015), the intrinsic plasticity adjusts the neuron's threshold depending on its activity so it reaches a mean firing rate target. Finally, some regulation mechanism modulates the STPD rule itself. This kind of plasticity is called metaplasticity. One of the first model of metaplasticity was developed in (Bienenstock et al. 1982) to explain the development of neuronal selectivity. It involves in this case a Hebbian plasticity that depends on the instantaneous pre- and postsynaptic activities. This plasticity is further modulated by a moving average of the postsynaptic activity, which thus constitutes a form of metaplasticity. (Zenke et al. 2013) used a similar mechanism on an STDP rule: the amount of depression induced by the STDP



rule depends on the average output activity. Another example of metaplasticity is presented in (Hunzinger et al. 2012). In this study the amount of synaptic change depends on a resource variable that is decreased at each synaptic change and then slowly recovers. The resource variable being shared across synapses, this typically constitutes a heterosynaptic metaplasticity. An interesting application of metaplasticity is to implement a reinforcement learning in an STDP network. (Izhikevich 2007) and (Legenstein et al. 2008) both presented a model where the STDP weight change is not immediately applied but converted into a fading eligibility trace. The model also includes a reward signal that is increased each time a reward is received and also fades with time. The weight change is then computed as the product of this eligibility trace with the reward signal. Following the same idea, (Aswolinskiy & Pipa 2015) also modulate the STDP rule with a reward variable.

#### D. Network properties

Neurons and synapses are the two elementary components of an artificial neural networks. Each of these elements have their properties and their temporal dynamics. When combined together to form a neural network, a new dynamic emerges, which can be complex to study. When studying networks, two complementary approaches exist. On one hand, some studies focus on studying the network's neural activity with fixed synaptic connections, often chosen following a random distribution. On the other hand, other studies focus on the evolution of synaptic weights given assumptions on the neurons' activity.

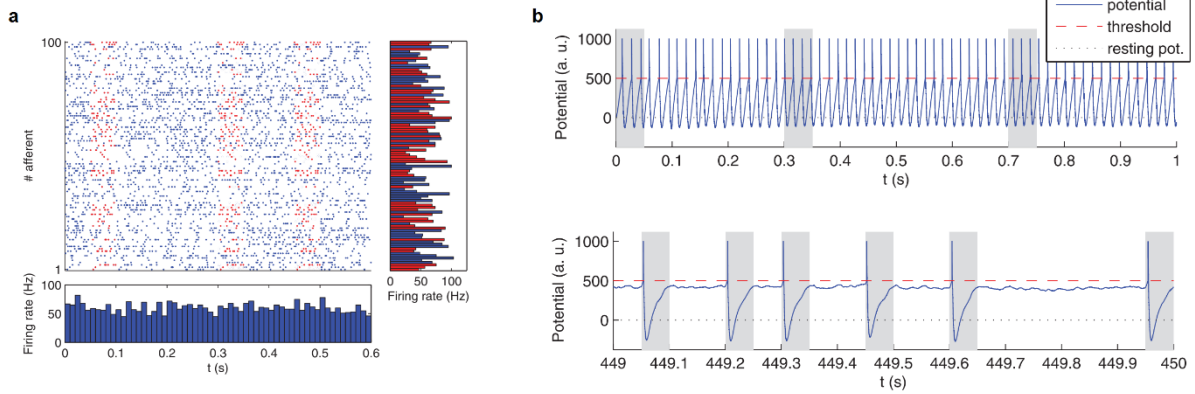
One type of neural networks that has been largely studied are sparsely connected recurrent networks, possibly taking an external stochastic input, as a model of what happens in the cortex (van Vreeswijk & Sompolinsky 1996; Amit & Brunel 1997; Brunel 2000; Ostojic 2014). It has been shown that these networks can reach, under certain conditions, a stable state called the asynchronous state, where neurons can be described as independent Poisson neurons, firing at a constant and homogeneous firing rate. The sparsity of the network allows to assume that the neurons activities are independent, as their inputs are independent from one another. Assuming that the inputs of each neuron come from independent Poisson neurons, each emitting spikes according to a Poisson process, the total input current follow a white Gaussian distribution. As a consequence, the neuron behaves as a Poisson neuron whose firing rate depends on the mean and the variance of its input (Amit & Brunel 1997). This mean-field approach allows to deduce the firing rate of the asynchronous state. An important condition from the asynchronous state to be stable is that the excitation and the inhibition should be correctly balanced in the network (van Vreeswijk & Sompolinsky 1996; Amit & Brunel 1997; Brunel 2000; Ostojic 2014). Indeed, at this condition, the average input of a neuron is subthreshold and its action potentials are mainly due to fluctuations, ensuring an irregular firing. In other conditions, states where with oscillations and synchronization between neurons can emerge (Brunel 2000). Other studies also showed that with a strong synaptic coupling the neural activity is chaotic, with high fluctuations of the firing rates both in time and across neurons, though the population firing rate is still constant on average (Ostojic 2014). Overall, random networks exhibit various possible behaviors and constitute a good basis to understand the dynamic of neural networks.

The second important aspect in neural network is synaptic learning. Learning in neural networks modifies the structure of the network depending on sensory inputs. The most widely used method to

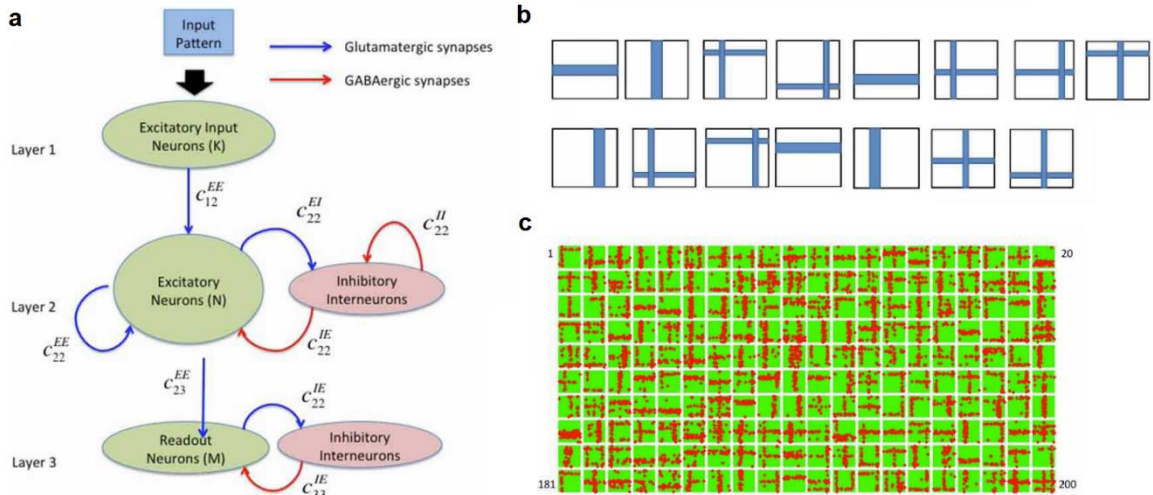
study synaptic weights evolution is to make assumptions on the firing rates and the time correlations between and within groups of neurons. This then allows to compute the average weight change of a particular synapse, or of groups of synapses, depending on the STDP rule used. This method can be used to study various STDP rules' properties, such as the weight distribution depending on the weight dependence of the STPD rule (Rubin et al. 2001; Gütig et al. 2003), the stability and the selectivity of different triplet rules (Babadi & Abbott 2016) or the weight evolution in a recurrent neural network (Burkitt et al. 2007). As stated previously, balance between inhibition and excitation is important for the stability of a network, which seems to be confirmed by experiments (Kirkwood 2015). Though this balance can be self-maintained in a random recurrent network (van Vreeswijk & Sompolinsky 1996), networks with a feedforward structure and a varying sensory input require a precise balance. Studies on inhibitory STDP rules show that it is possible to obtain a precise excitatory-inhibitory balance through learning on inhibitory synapses (Vogels et al. 2012; Luz & Shamir 2012). Beyond theoretical results, (Srinivasa et al. 2014) showed that the use of inhibitory STDP to balance inhibition and excitation leads to better results in a pattern recognition task.

## E. Applications to pattern recognition tasks

One difficulty with STDP networks is that there is no standard method to train a network to achieve a specific task through STDP. Some studies have thus focused on learning random spike train patterns to demonstrate STDP network learning capabilities. (Masquelier et al. 2008) demonstrated that a very simple monolayer STDP network is able to learn, in an unsupervised manner, to recognize the beginning of a pattern (Figure III-7). This work was extended to demonstrate its ability to discriminate different patterns or different parts of a pattern (Masquelier et al. 2009). Following the same idea, (Hunzinger et al. 2012) showed that a resource-dependent plasticity improves the performance of STDP learning on a pattern recognition task. Application to more concrete pattern recognition tasks, such as visual recognition, are also emerging. For example, (Masquelier & Thorpe 2007) used an STDP network to recognize faces and vehicles in pictures. Their network is well structured, alternating learning layers with shared weights and pooling layers, and used a classical classifier as a readout function. (Brader et al. 2007) used a one-layer STDP network with a teaching signal for a digit recognition task. (Bichler et al. 2012) also trained a network to recognize patterns in a video stream, though the recognized patterns do not include time aspects such as movements. In this case the network was designed to be implemented in neuromorphic devices mimicking STDP at a miniaturized scale. Most of these applications use a feedforward structure, sometimes with only one layer. (Srinivasa et al. 2014) used an interesting network structure, based on the idea of reservoir computing (Figure III-8). This network is constituted of pools of recurrently connected excitatory and inhibitory neurons, the different pools being themselves connected in a feedforward manner. This STDP network was applied to a visual pattern recognition task, and shows that the introduction of plasticity on inhibitory synapse improves the discrimination performances. Beyond visual recognition, (Suri et al. 2013) also show an example of application to auditory pattern recognition. At the moment, STDP applications to real-word pattern recognition tasks are still limited, but the existing applications show interesting learning properties such as unsupervised learning. More work needs to be done to understand how to efficiently design an STDP network for a specific task.



**Figure III-7: Example of spike pattern recognition by (Masquelier et al. 2008).** One LIF neuron is trained to learn an arbitrary spike pattern through STDP. (a) Illustration of the pattern to learn (in red). Between patterns (in blue), the input activity consist in random Poisson spike trains. (b) Illustration of the neuron learning. Top: before learning the neuron fire independently of the pattern occurrences. Bottom: After learning, the neuron fire at the beginning of the pattern (pattern occurrence are shown in grey).



**Figure III-8 : Example of visual pattern recognition by (Srinivasa et al. 2014).** (a) Structure of the network. (b) Patterns to learn. (c) Weights learnt by the reservoir excitatory neurons (Layer 2).

## References

- Abbott, L.F. et al., 1997. Synaptic Depression and Cortical Gain Control. *Science*, 275, pp.220–223.
- Abbott, L.F. & Nelson, S.B., 2000. Synaptic plasticity: taming the beast.
- Amit, D. & Brunel, N., 1997. Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cerebral Cortex*, 7(3), pp.237–252. Available at: <https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/7.3.237>.
- Aswolinskiy, W. & Pipa, G., 2015. RM-SORN: a reward-modulated self-organizing recurrent neural network. *Frontiers in Computational Neuroscience*, 9(March), pp.1–15. Available at: <http://journal.frontiersin.org/article/10.3389/fncom.2015.00036>.
- Babadi, B. & Abbott, L.F., 2016. Stability and Competition in Multi-spike Models of Spike-Timing Dependent Plasticity. *PLOS Computational Biology*, 12(3), p.e1004750. Available at: <http://dx.plos.org/10.1371/journal.pcbi.1004750>.
- Bi, G.Q. & Poo, M.M., 1998. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(24), pp.10464–10472.
- Bichler, O. et al., 2012. Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Networks*, 32, pp.339–348. Available at: <http://dx.doi.org/10.1016/j.neunet.2012.02.022>.
- Bienenstock, E.L., Cooper, L.N. & Munro, P.W., 1982. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 2(1), pp.32–48.
- Brader, J.M., Senn, W. & Fusi, S., 2007. Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural computation*, 19(11), pp.2881–2912.
- Brette, R. & Gerstner, W., 2005. Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity. *Journal of Neurophysiology*, 94, pp.3637–3642. Available at: <https://www.ncbi.nlm.nih.gov/pubmed/16014787>.
- Brunel, N., 2000. Dynamics of Sparsely Connected Networks of Excitatory and Inhibitory Spiking Neurons. *Journal of Computational Neuroscience*, 8, pp.183–208.
- Burkitt, a. N., Gilson, M. & Van Hemmen, J.L., 2007. Spike-timing-dependent plasticity for neurons with recurrent connections. *Biological Cybernetics*, 96(5), pp.533–546.
- Connors, B.W. & Gutnick, M.J., 1990. Intrinsic firing patterns of diverse neocortical neurons. *Trends in Neurosciences*, 13(3), pp.99–104.
- D'amour, J.A. & Froemke, R.C., 2015. Inhibitory and excitatory spike-timing-dependent plasticity in the auditory cortex. *Neuron*, 86(2), pp.514–528. Available at: <http://dx.doi.org/10.1016/j.neuron.2015.03.014>.
- FitzHugh, R., 1961. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophysical Journal*, 1(6), pp.445–466.
- Froemke, R.C. & Dan, Y., 2002. Spike-timing-dependent synaptic modification induced by natural spike trains. *Nature*, 416(6879), pp.433–438.
- Georgopoulos, a P., Schwartz, a B. & Kettner, R.E., 1986. Neuronal population coding of movement direction. *Science (New York, N.Y.)*, 233(4771), pp.1416–1419.

- Gerstner, W., Ritz, R. & van Hemmen, J.L., 1993. Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns. *Biological Cybernetics*, 69(5–6), pp.503–515.
- Gibson, J.R., Beierlein, M. & Connors, B.W., 1999. Two networks of electrically coupled inhibitory neurons in neocortex. *Nature*, 402(6757), pp.75–79. Available at: <http://www.nature.com/articles/47035>.
- Gütig, R. et al., 2003. Learning input correlations through nonlinear temporally asymmetric Hebbian plasticity. *The Journal of Neuroscience: the official journal of the Society for Neuroscience*, 23(9), pp.3697–3714. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/12736341>.
- Hebb, D.O., 1949. *The Organization of Behavior*, New York: John Wiley & Sons. Available at: <http://www.jstor.org/stable/1418888?origin=crossref>.
- Hodgkin, A.L., 1948. The local electric changes associated with repetitive action in a non-medullated axon. *Journal of Physiology*, 107(2), pp.165–181.
- Hodgkin, A.L. & Huxley, A.F., 1952. A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve. *Journal of Physiology*, 117, pp.500–544.
- Hopfield, J.J., 1995. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376(6535), pp.33–36.
- Humble, J., Denham, S. & Wennekers, T., 2012. Spatio-temporal pattern recognizers using spiking neurons and spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 6(October), pp.1–12.
- Hunzinger, J.F., Chan, V.H. & Froemke, R.C., 2012. Learning complex temporal patterns with resource-dependent spike timing-dependent plasticity. *Journal of Neurophysiology*, 108(2), pp.551–566.
- Izhikevich, E.M., 2006. Polychronization: computation with spikes. *Neural computation*, 18(2), pp.245–82. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/16378515>.
- Izhikevich, E.M., 2001. Resonate-and-fire neurons. *Neural Networks*, 14(6–7), pp.883–894.
- Izhikevich, E.M., 2003. Simple model of spiking neurons. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 14(6), pp.1569–72. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/18244602>.
- Izhikevich, E.M., 2007. Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, 17(10), pp.2443–2452.
- Izhikevich, E.M., 2004. Which Model to Use for Cortical Spiking Neurons. *IEEE Transactions on Neural Networks*, 15(5), pp.1063–1070.
- Johansson, R.S. & Birznieks, I., 2004. First spikes in ensembles of human tactile afferents code complex spatial fingertip events. *Nature Neuroscience*, 7(2), pp.170–177.
- Kirkwood, A., 2015. Balancing Excitation and Inhibition. *Neuron*, 86(2), pp.348–350. Available at: <http://dx.doi.org/10.1016/j.neuron.2015.04.009>.
- Lazar, A., 2009. SORN: a Self-organizing Recurrent Neural Network. *Frontiers in Computational Neuroscience*, 3(October), pp.1–9. Available at: <http://journal.frontiersin.org/article/10.3389/neuro.10.023.2009/abstract>.
- Legenstein, R., Pecevski, D. & Maass, W., 2008. A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Computational Biology*, 4(10).
- Luz, Y. & Shamir, M., 2012. Balancing feed-forward excitation and inhibition via hebbian inhibitory

- synaptic plasticity. *PLoS Computational Biology*, 8(1), pp.1–12.
- Maass, W., 1997a. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9), pp.1659–1671. Available at: <http://www.sciencedirect.com/science/article/pii/S0893608097000117>.
- Maass, W., 1997b. Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. *Advances in Neural Information Processing Systems*, 9, pp.211–217.
- Malsburg, C., 1973. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14(2), pp.85–100. Available at: <http://link.springer.com/10.1007/BF00288907>.
- Markram, H., Wang, Y. & Tsodyks, M., 1998. Differential signaling via the same axon of neocortical pyramidal neurons. *Proc Natl Acad Sci U S A*, 95(9), p.5323–8.
- Masquelier, T., 2014. Oscillations can reconcile slowly changing stimuli with short neuronal integration and STDP timescales. *Network (Bristol, England)*, 25(1–2), pp.85–96. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/24571100>.
- Masquelier, T., Guyonneau, R. & Thorpe, S.J., 2009. Competitive STDP-based spike pattern learning. *Neural computation*, 21(5), pp.1259–1276.
- Masquelier, T., Guyonneau, R. & Thorpe, S.J., 2008. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS ONE*, 3(1).
- Masquelier, T. & Thorpe, S.J., 2007. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Computational Biology*, 3(2), pp.0247–0257.
- McLelland, D. & Paulsen, O., 2009. Neuronal oscillations and the rate-to-phase transform: mechanism, model and mutual information. *The Journal of physiology*, 587(Pt 4), pp.769–785.
- Morris, C. & Lecar, H., 1981. Voltage Oscillations in the Barnacle Giant Muscle Fiber. *Biophysical Journal*, 35(1), pp.193–213. Available at: [http://dx.doi.org/10.1016/S0006-3495\(81\)84782-0](http://dx.doi.org/10.1016/S0006-3495(81)84782-0).
- Morrison, A., Diesmann, M. & Gerstner, W., 2008. Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, 98(6), pp.459–478.
- Nanami, T. & Kohno, T., 2016. An FPGA-based cortical and thalamic silicon neuronal network. , 2(4), pp.238–242.
- Ostojic, S., 2014. Two types of asynchronous activity in networks of excitatory and inhibitory spiking neurons. *Nat Neurosci*, 17(4), pp.594–600. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/24561997>.
- Pfister, J.-P. & Gerstner, W., 2006. Triplets of Spikes in a Model of Spike Timing-Dependent Plasticity. *Journal of Neuroscience*, 26(38), pp.9673–9682. Available at: <http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.1425-06.2006>.
- Pospischil, M. et al., 2008. Minimal Hodgkin-Huxley type models for different classes of cortical and thalamic neurons. *Biological Cybernetics*, 99(4–5), pp.427–441.
- Rose, R.M. & Hindmarsh, J.L., 1989. The assembly of ionic currents in a thalamic neurons: I. The three-dimensional model. *Proc. R. Soc. Lond. B*, 237, pp.267–288.
- Rubin, J., Lee, D.D. & Sompolinsky, H., 2001. Equilibrium properties of temporally asymmetric Hebbian plasticity. *Physical Review Letters*, 86(2), pp.364–367.
- Rumelhart, D.E., Hinton, G.E. & Williams, R.J., 1986. Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland, eds. *Parallel distributed processing*. pp.

- 318–362. Available at:  
[https://web.stanford.edu/class/psych209a/ReadingsByDate/02\\_06/PDPVolIIChapter8.pdf](https://web.stanford.edu/class/psych209a/ReadingsByDate/02_06/PDPVolIIChapter8.pdf).
- Schwartz, a B., 1994. Direct cortical representation of drawing. *Science (New York, N.Y.)*, 265(5171), pp.540–542.
- Singer, W., 1993. Synchronization of cortical activity and its putative role in information processing and learning. *Annual Review of Physiology*, 55, pp.349–374.
- Sjöström, P.J., Turrigiano, G.G. & Nelson, S.B., 2001. Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron*, 32(6), pp.1149–1164.
- Smith, G.D. et al., 2000. Fourier analysis of sinusoidally driven thalamocortical relay neurons and a minimal integrate-and-fire-or-burst model. *Journal of neurophysiology*, 83(1), pp.588–610. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/10634897>.
- Srinivasa, N., Cho, Y. & Hennequin, G., 2014. Unsupervised discrimination of patterns in spiking neural networks with excitatory and inhibitory synaptic plasticity. , 8(December), pp.1–23.
- Suri, M. et al., 2013. Bio-inspired stochastic computing using binary CBRAM synapses. *IEEE Transactions on Electron Devices*, 60(7), pp.2402–2409.
- Thorpe, S.J. & Gautrais, J., 1997. Rapid Visual Processing using Spike Asynchrony. *Neural Information Processing Systems*, pp.901–907.
- Tsodyks, M., Uziel, A. & Markram, H., 2000. Synchrony generation in recurrent networks with frequency-dependent synapses. *The Journal of neuroscience*, 20(1), p.RC50.
- VanRullen, R., Guyonneau, R. & Thorpe, S.J., 2005. Spike times make sense. *Trends in Neurosciences*, 28(1), pp.1–4.
- Vanrullen, R. & Thorpe, S.J., 2002. Surfing a spike wave down the ventral stream. , 42, p.15. Available at: [papers2://publication/uuid/7EF9F2D6-AC27-4B0E-9D87-80DC626F55E5](http://papers2://publication/uuid/7EF9F2D6-AC27-4B0E-9D87-80DC626F55E5).
- Vogels, T.P. et al., 2012. Inhibitory Plasticity Balances Excitation and Inhibition in Sensory Pathways and Memory Networks. , (December 2011).
- Vogels, T.P. et al., 2013. Inhibitory synaptic plasticity: spike timing-dependence and putative network function. *Frontiers in neural circuits*, 7(July), p.119. Available at: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3714539&tool=pmcentrez&render type=abstract>.
- van Vreeswijk, C. & Sompolinsky, H., 1996. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science (New York, N.Y.)*, 274(5293), pp.1724–6. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/8939866>.
- Wang, H.X. et al., 2005. Coactivation and timing-dependent integration of synaptic potentiation and depression. *Nature Neuroscience*, 8(2), pp.187–193.
- Woodin, M. a., Ganguly, K. & Poo, M.M., 2003. Coincident pre- and postsynaptic activity modifies GABAergic synapses by postsynaptic changes in Cl<sup>-</sup> transporter activity. *Neuron*, 39(5), pp.807–820.
- Yger, P. & Gilson, M., 2015. Models of Metaplasticity: A Review of Concepts. *Frontiers in Computational Neuroscience*, 9(November), pp.1–14. Available at: <http://journal.frontiersin.org/article/10.3389/fncom.2015.00138>.
- Zenke, F., Hennequin, G. & Gerstner, W., 2013. Synaptic Plasticity in Neural Networks Needs Homeostasis with a Fast Rate Detector. *PLoS Computational Biology*, 9(11).

- Zhang, W. & Linden, D.J., 2003. The other side of the engram: experience-driven changes in neuronal intrinsic excitability. *Nature reviews. Neuroscience*, 4(11), pp.885–900.
- Zucker, R.S. & Regehr, W.G., 2002. Short-term synaptic plasticity. *Annual Review of Physiology*, 64(1), pp.355–405. Available at:  
<http://www.annualreviews.org/doi/abs/10.1146/annurev.physiol.64.092501.114547>.





## IV. NETWORK MODEL

We built an STDP network for spike sorting which takes a single electrode signal as input, and outputs a spike-train that corresponds to the sorted action potential recorded in the signal. It is organized into layers connected in a feed-forward manner, with all-to-all connections (Figure IV-1). Each layer is designed to achieve a specific sub-processing task. The neuron model and the synaptic plasticity are both adjusted for each layer to obtain the desired behavior. The final version of the network consists in three main layers: an encoding layer that encodes the input signal into spikes, an intermediate layer that learns intermediate patterns, and an output layer that finalizes the process and fires once for each action potential in the signal. The network also features an attention mechanism. Each part of the network can be implemented with different solutions, and some mechanisms can be added to improve its performance. Through this PhD work three different versions of a complete network were implemented for processing single electrodes: MiniNet, ANNet and LTSNet, described in Section VI. The last version, LTSNet was also adapted to multi-electrode processing, in a version called PolyNet. Each of these versions uses different implementation solutions for each element of the network, some because they bring an obvious improvement, others because they combine well with other choices. Here we focus on the different possible solutions to implement each different functional elements of the network for a single electrode case, and how the network structure can be adapted to process multi-electrodes' signals.

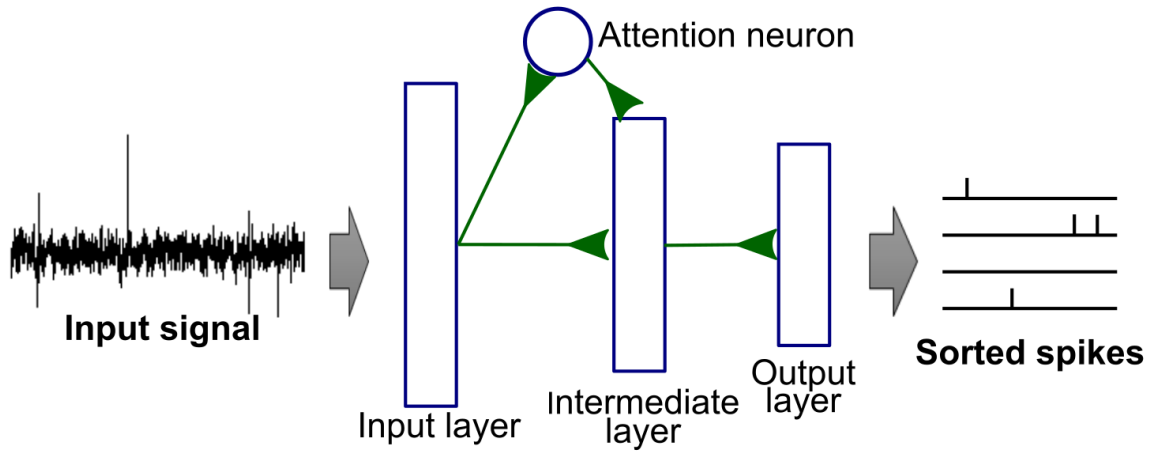


Figure IV-1: Global structure of the STDP network. The network is constituted of several layers, each having a different functional purpose, connected in a feedforward manner. An attention mechanism is implemented through an attention neuron. The network takes as input an electrode signal and outputs a spike train corresponding to the sorted action potentials.

### A. Encoding the input signal

The STDP network takes an analog signal representing a potential varying in time as an input. It is thus necessary to encode this input signal into a spike train for further processing by the network. This can be interpreted as a sensory functionality, by analogy with sensory neural cells that convert light

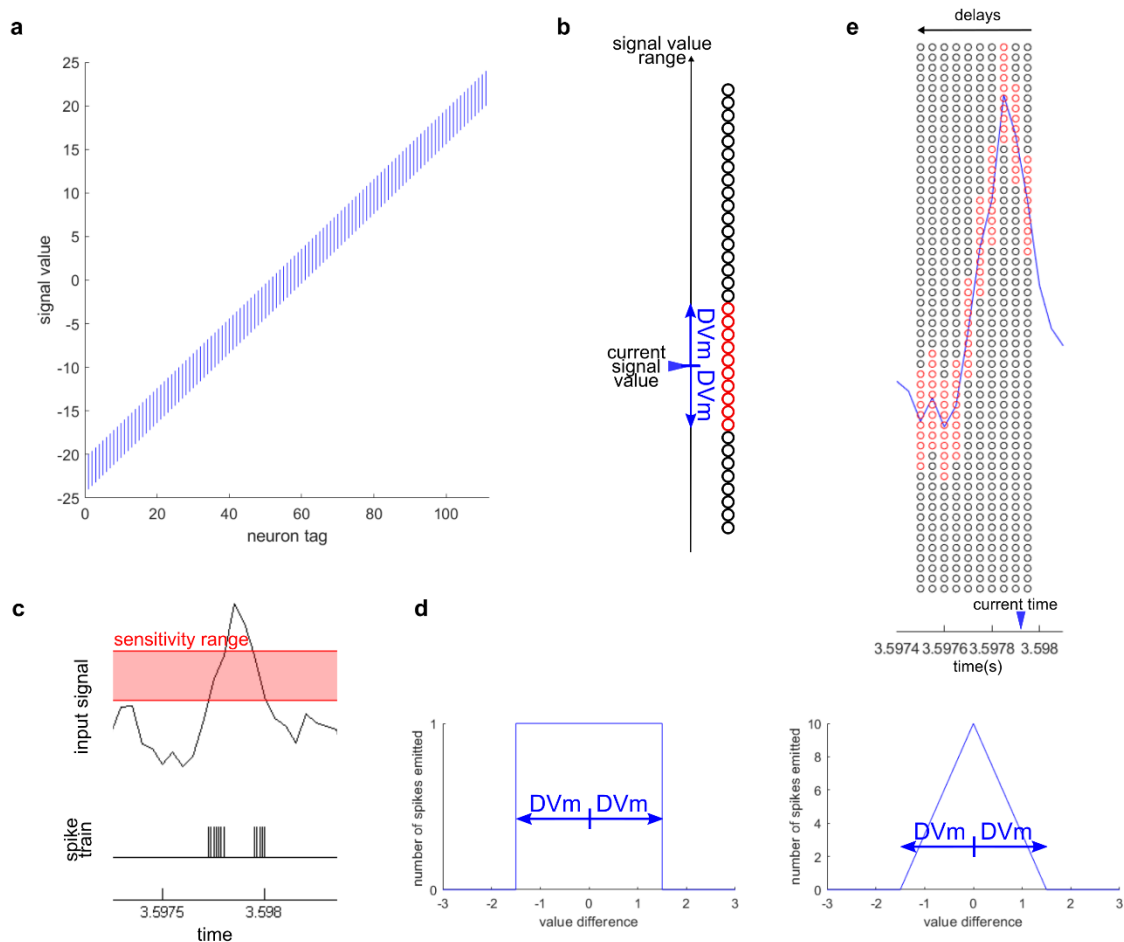
intensity (in the retina) or pressure (in the skin) into spike trains for further processing by the central nervous system. Following this analogy, a first approach to encode the signal would be to use a neuron whose firing rate would increase with the value of the input signal, thus implementing a firing-rate code. This would give information whether the amplitude of the input signal is high or low, however, it would be difficult to discriminate precisely different amplitudes, as two different amplitudes would be encoded by the same neuron, with only a quantitative difference in its firing rate. Instead, our input layer is composed of a set of sensory neurons, each neuron being sensitive to a different range of signal amplitudes, defined by an individually selected central value  $c$ , and a margin  $DV_m$ , common to all input neurons. When the input signal value is within a neuron's sensitivity range, which is to say between  $c - DV_m$  and  $c + DV_m$ , the neuron fires, otherwise it remains silent. Those sensitivity ranges are regularly spaced, partially overlapping, and their union covers the entire possible range of signal values (Figure IV-2.a). By analogy with the visual system, this works as if the signal amplitude was an object moving along one dimension and activating neurons depending on its position (Figure IV-2.b). An important property of this way to encode the input signal is that the number of spikes emitted by the input layer does not depend on the exact signal value, thus all signal values generate the same excitation level on the next layer.

In our case, the input signal is a sampled signal. Thus, a natural way to implement the sensory neurons is to compute their activation at regular time step, each time a new sample arrives (Figure IV-2.c). At each sampling step, each sensory neuron either fires once if the signal sample is within its sensitivity range or remains silent (Figure IV-2.d, left). Another possibility is that each neuron emits several spikes at each sampling step, depending on how close the signal value is from the sensitivity range center. For example, the number of spikes emitted can be maximum when the signal value equals the sensitivity range center, and then decreases linearly until the signal value reaches the limits of the sensitivity range, for which no spike is emitted (Figure IV-2.d, right). These two ways of encoding the signal, which we call later binary encoding and triangular encoding, have different properties that are studied in Section IV.B. In our different network implementations, only the binary encoding method has been tested, for its simplicity.

The sampling frequency used for encoding is an important parameter for the spike-sorting performance of the network. Indeed, to ensure that the next layers will properly learn the spike pattern generated by the input layer, it is necessary that for two different occurrences of the same waveform, some common sensory neurons get activated, regardless of the noise and sampling time jitter. A good criterion is to choose the sampling step short enough so that, even for the fastest variations of the signal, two consecutive samples always activate some common neurons. In practice and on the different recordings we tested, we found that an 80-kHz sampling frequency was a good compromise, between sufficient information and too much data generated. This was thus the value used in all our implementations, except for the first one (MiniNet) where it was only 20 kHz.

To discriminate between different action potentials in the signal, we need to take into account not only the amplitudes of an action potential but its entire waveform, in other words the signal amplitude variations in time. Thus the input layer should give information about the value of the signal at different time points. To do so, we introduce synapses with different transmission delays stemming from each sensory neuron, as previously done for example in (Gerstner et al. 1993), (Hopfield 1995) or (Ghosh-Dastidar & Adeli 2007). From the point of view of the layers that receive the input layer spike train through these synapses, this looks like the original set of sensory neurons is duplicated into several

sets of neurons acting alike yet with different delays. Each of the delayed set of sensory neurons encodes the signal value with a given delay from current time (Figure IV-2.e). The idea is that the plasticity rules will then potentiate some of these delayed synapses and depreciate others, as this is believed to happen during the development of the barn owl auditory system (Gerstner et al. 1996). The difference between two delayed synapses and the number of delays determine the time resolution of the encoded signal and the size of the encoded signal time window at each sampling step. The number of delays is a compromise between the number of synapses and the discriminative power of the input spike patterns. Noticeably, if the encoded time window becomes too large (typically larger than an action potential), it loses its discriminative power as the signal value outside the action potential is within the noise range and thus close to zero and non-discriminative. In practice we found that a 0.05-ms time resolution was enough, and we chose a time window of 0.5 ms (thus 10 different delays), which is usually shorter than one action potential.



**Figure IV-2: Input layer implementation.** (a) Sensitivity range for each input neuron. The sensitivity ranges overlap and cover all possible signal values. (b) Activation of each encoding neuron for a given signal value. Activated neurons are represented in red. (c) Activation of one encoding neuron thorough time. The neuron fires at regular time steps when the signal is within its sensitivity range. (d) Different activation functions: binary (left) and triangular (right). With the binary activation, encoding neurons fire one spike at each time step if the signal value is within their sensitivity range. With the triangular activation, encoding neurons can fire several spikes depending on the position of the signal value within their sensitivity range. (e) Several signal values, corresponding to different delays, are encoded at each time step.

## B. Attention mechanism

The input signal the network has to process fluctuates around zero most of the time, as the action potentials occur sparsely. At some point in the process it is thus necessary to discriminate the parts of the signal corresponding to an action potential from the rest of the signal. Indeed, by the way the signal is encoded, there is no intrinsic difference between parts of the signal containing an action potential and parts containing only noise, and both can be recognized as patterns. As this is well described in (Masquelier et al. 2008), when a neuron learns a pattern through an STDP rule, the neuron tends to fire earlier each time the pattern is presented. In this study, this process naturally stops when the neuron reaches the beginning of the pattern, as the spike train before the pattern is random and cannot be learnt. In our case, a neuron learning an action potential waveform would each time recognize earlier parts of the waveform and would end up recognizing the null pattern between action potentials. Even though once some neurons have learned the null part of the signal, they could prevent other neuron from firing when the signal is null, we found better for a more stable learning that an attention mechanism detecting the action potentials intervene at an early stage of the process, just after the input layer, preventing the null parts of the signal to be learnt. This attention mechanism was implemented using a short-term plasticity. In the first versions of the network this short-term plasticity was applied directly on the synapses between the input and the intermediate layer, but we later introduced an attention neuron whose role was to specifically detect where action potentials occurred in the signal.

### 1. Short-term plasticity

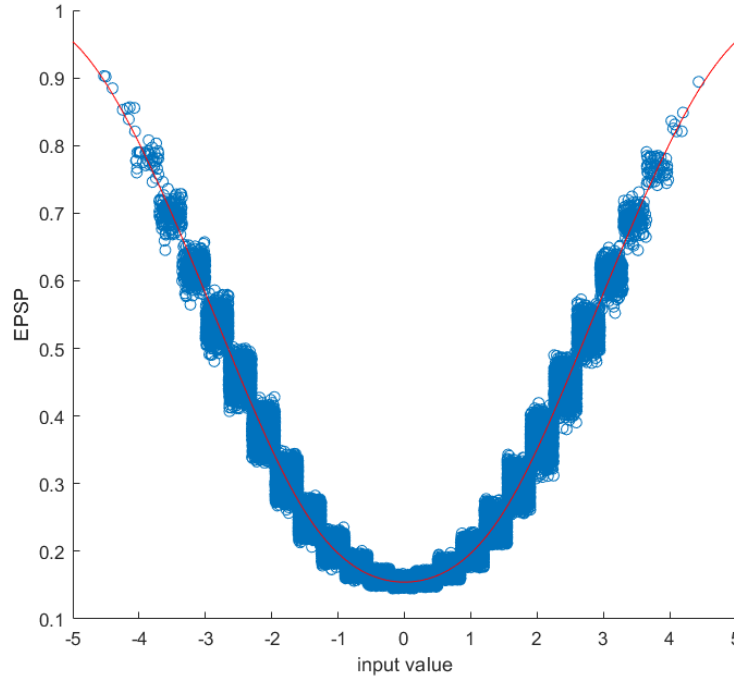
We implemented an attention mechanism thanks to a short-term plasticity (STP) rule. Indeed STP, and more precisely short-term depression, has the property to weaken the weights of the synapses for which a presynaptic spike occurs at high frequency (Abbott et al. 1997), as this is the case for sensory neuron coding for near-zero values. As a result, the postsynaptic neuron will be less excited when the input signal is within the noise range. In our implementation the short-term plasticity rule is governed by the following equation:

$$\frac{dw}{dt} = \frac{1}{\tau_{stp}} (1 - w) - \sum_s w * f_d * \delta(t - t_s),$$

where  $w$  is the synaptic weight,  $\tau_{stp}$  the STP recovery time constant,  $f_d$  the STP depression factor and  $t_s$  the presynaptic spike times. We chose  $\tau_{stp}$  one order higher than the typical duration of an action potential so that the synaptic weight did not change significantly during an action potential. Given this equation we can compute the equilibrium weight for a presynaptic neuron firing at regular pace, with a frequency  $f$ . When such a synapse, with a weight  $w_{eq}$ , fires, its weight becomes  $(1-f_d)w_{eq}$ . After a time interval  $1/f$ , its weight should return to  $w_{eq}$  through the exponential recovery before firing again. This can be expressed as  $(1 - w_{eq}) = \exp(-1/f * \tau_{stp})(1 - (1 - f_d)w_{eq})$ . Thus, the equilibrium weight is given by:

$$w_{eq} = \frac{1 - \exp(-\frac{1}{f * \tau_{stp}})}{1 - (1 - f_d)\exp(-\frac{1}{f * \tau_{stp}})}$$

As expected,  $w_{eq}$  is close to 1 when  $f$  is close to zero, and decreases when  $f$  increases. With the hypothesis that the input signal follows a Gaussian noise distribution, we can deduce the mean firing rate of each sensory neuron and thus approximate its weight. Thanks to this relation, we can choose a minimum weight  $W_{min}$ , obtained for a neuron firing at each sampling step and adjust  $f_d$  in consequence. Knowing these equilibrium weights, we can then compute the excitatory postsynaptic potential (EPSP) generated by each possible input signal value, which is the sum of the weighted spikes received by the postsynaptic neuron. We computed this approximation numerically and compared it to simulation results, as shown in Figure IV-3. We also observed the effects on the EPSP of the choices of both  $DV_m$  and  $W_{min}$  values (Figure IV-4). Input values close to zero generate a weak excitation whereas when the input value moves away from zero the postsynaptic excitation increases, which is the desired effect.



**Figure IV-3: Modeled and simulated EPSP for each input value. Model is shown in red, values obtained in simulation, with a normal Gaussian noise as input signal, are shown in blue. This was obtained for  $DV_m=1.75$  and  $W_{min}=0.13$  and with a binary encoding**

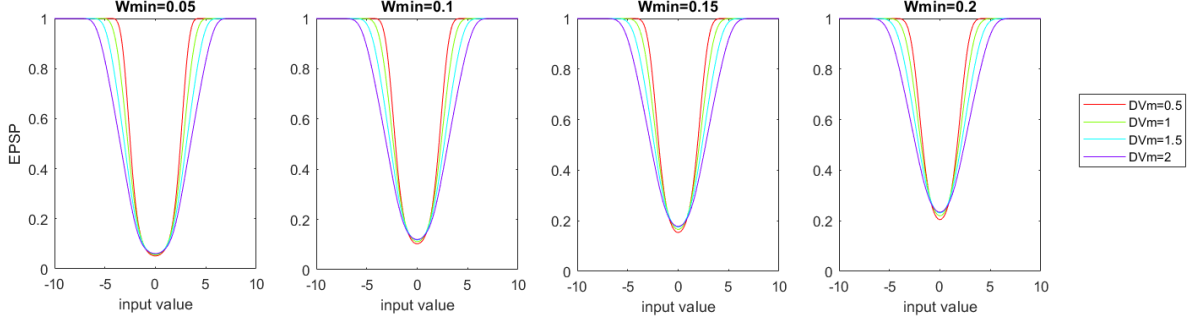


Figure IV-4: Effect of  $DV_m$  and  $W_{min}$  on the input value-EPSP relation, for a binary encoding.

## 2. Attention neuron

In the first version of the network the short-term plasticity was applied on the synapses connecting the input layer to the intermediate layer, in combination with the STDP rule. However, this method has the default to make low-amplitude waveforms difficult to recognize, as their low STP weights tends to shrink the receptive field size of the intermediate neurons learning it (see Section IV.C.3). A better idea is to implement an attention neuron, connected to the input layer through STP synapses, whose role is to specifically detect any action potential present in the input signal. When an action potential is present in the signal, this neuron fires continuously from the beginning to the end of the action potential. When no action potential occurs, it remains silent (Figure IV-5). Its spikes can then be used by the rest of the network as a signal that an action potential is present in the signal.

The attention neuron was implemented as a LIF neuron model. Its potential evolves dynamically according to the following equation:

$$\frac{dV}{dt} = -\frac{1}{\tau_m} * V(t) + \sum_i \sum_s w_i(t_{i,s}) \delta(t - t_{i,s}),$$

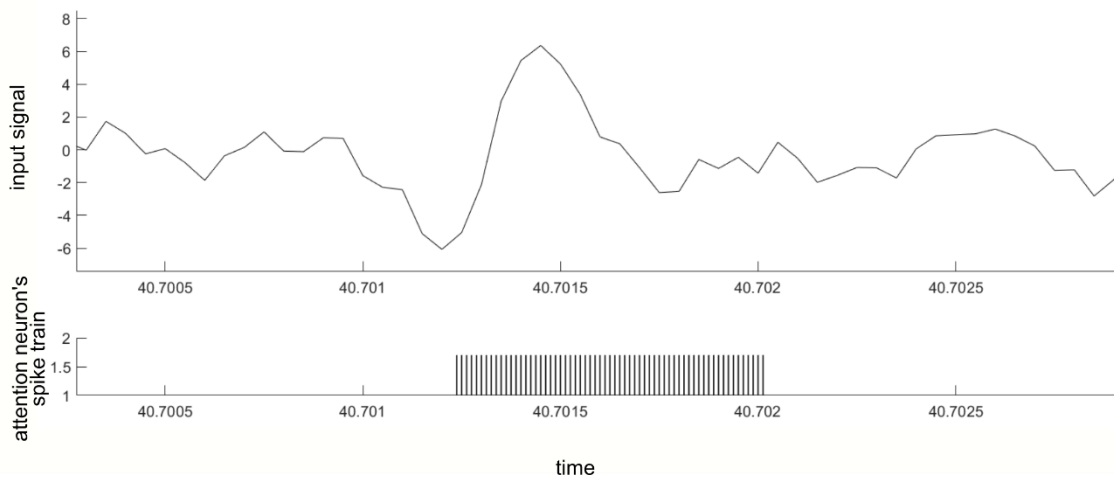
where  $V$  is the neuron potential,  $\tau_m$  is the membrane time constant,  $i$  indexes the incoming synapses,  $s$  indexes the spikes transmitted by the synapse,  $w_i$  is the synapse's weight and  $t_{i,s}$  the spike's time. Additionally, the neuron fires when the potential  $V$  reaches its threshold. In this case, the potential is not reset so that the potential value only depends on the input signal and not on the attention neuron's activity. As the EPSP received by the attention neuron increases when values outside the noise range occur within the signal, the attention neuron's potential also increases. By setting correctly its threshold, the attention neuron can thus detect action potentials (Figure IV-6).

The important information for the attention mechanism is the amplitude of the signal and not its exact shape. Thus, at first sight, it is not necessary to have different delays to connect the attention neuron to the input layer, as introduced in Section IV.A. Though the exact shape of the signal is not important, action potentials have the property of bringing the signal outside the noise range for a relatively long time. To take advantage of this, for a better detection, one possibility is to set the membrane's time constant to a value similar to the duration of an action potential, so that the neuron's potential depends on the input values in a time window corresponding to an action potential duration. The other possibility is to use the delayed synapses from the input layer. In this case, the potential is

already summed over such a time window, so a short membrane time constant insufficient. In practice, the later method proved to give better results.

The choice of the attention neuron's threshold is crucial, as it will determine the false positive and false negative rates. Our first idea was to fix a false positive rate, and to set the threshold to reach this goal. However it is difficult to precisely approximate the potential's distribution of the attention neuron for a noisy signal, especially for the tail of the distribution, which is key for having a low false positive rate. Only long simulations could help choosing the threshold with this criterion. However our goal is to have a network versatile for any situation, thus we would like to have a simpler criterion that allows to set the threshold easily. As an alternative idea, we define a stereotypic waveform with a minimum amplitude and duration, which would be the minimum waveform the attention neuron can detect. For simplicity this waveform has a square shape, which makes it easy to compute an approximation of the maximum potential reached by the attention neuron when encountering such a waveform, using the numerical approximation of the EPSP generated by each different input signal value (see Section IV.B.2).

In cases where a waveform has a peak followed by another peak of the opposite sign and thus crosses zero, we were confronted to the problem that the attention neuron potential decreased under the threshold in the middle of the action potential. Thus instead of firing continuously from the beginning to the end of the action potential, the attention neuron stopped firing in the middle of the action potential. To solve this problem, we implemented a hysteresis mechanism by increasing the attention neuron's threshold by a self-excitation value each time it fires. Thus the excitation due to the input signal needs to go down under a value that is lower than the neuron's threshold for it to stop firing.



**Figure IV-5: Detection of an action potential by the attention neuron. Top: input signal. Bottom: attention neuron's spike train.**



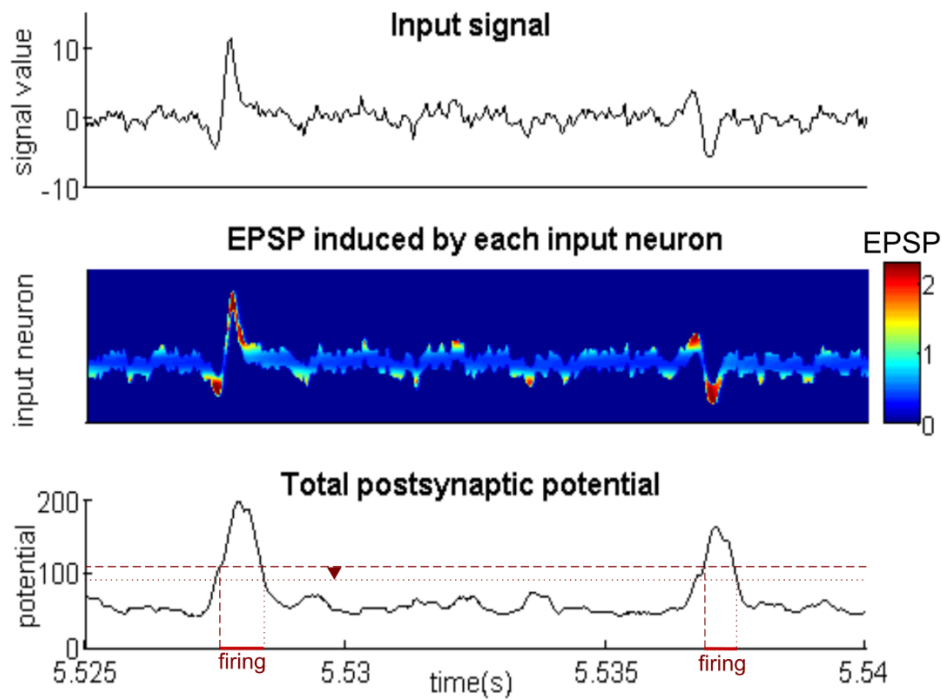


Figure IV-6: The attention neuron potential increases when an action potential is present in the input signal. Top: input signal. Middle: EPSP generated by the synapses implementing an STP rule. Bottom: attention neuron's potential. In red are represented the attention neuron threshold, the self-excitation effect (red arrow), and the time intervals when the neuron is firing

### C. Learning of waveform elements by the intermediate layer

The input layer generates a spike train representing the signal shape in a sliding time window. The intermediate layer receives this spike train through synaptic connections. Its purpose is thus to learn to recognize a pattern in this spike train, corresponding to parts of action potentials waveforms. To do so, an STDP rule is used on the synapses projecting from the input layer to the intermediate layer. Importantly, the intermediate layer is gated by an attention mechanism allowing it to fire and learn only when an action potential is present in the signal. In the different versions of the network, this attention mechanism was implemented either by using an STP rule on the synapses connecting the input and the intermediate layers or using an attention neuron. When the attention mechanism is implemented through the attention neuron, the latter projects excitatory synapses onto the intermediate layer, thus inducing an additional excitation when the attention neuron fires, due to presence of an action potential. In both cases, the intermediate neurons' threshold is adjusted so that the intermediate layer can fire only when an action potential is present in the signal.

#### 1. LIF neuron

The neurons in this intermediate layer follow a LIF neuron model, which internal potential  $V$  varies according to the following equation:

$$\frac{dV}{dt} = -\frac{1}{\tau_m} * V(t) + \sum_i \sum_s w_i(t_{i,s}) \delta(t - t_{i,s}),$$

where  $\tau_m$  is the membrane time constant,  $i$  indexes the incoming synapses,  $s$  indexes the spikes transmitted by the synapse,  $w_i$  is the synapse weight and  $t_{i,s}$  the spike times. Additionally when the potential reaches the neuron's threshold, it fires and its potential is reset to zero. Each time an intermediate neuron receives a spike from the input layer, its potential is increased proportionally to the corresponding synapse's weight. When no spike is received, the potential returns back to zero, with an exponential decay corresponding to the membrane time constant  $\tau_m$ . Thus if the neuron receives enough spikes in a short time compared to  $\tau_m$ , its potential increases enough for the neuron to fire. This type of neuron can thus detect spike coincidence, with a time precision that corresponds to  $\tau_m$ . We thus set  $\tau_m$  to be the same order of magnitude as the chosen time resolution for encoding. In our case, the number of spikes coming from the input layer is constant, thus the difference is made by the synapses weights, learnt with the STDP rule described in the next section. For a neuron to fire, enough spikes coming from potentiated synapses need to arrive simultaneously. Before the learning phase, intermediate neurons need to fire to begin the learning process. To do so, at start, the synapses weight are initialized randomly, with a mean weight that is high enough to be able to generate some spike on the intermediate layer. After the learning phase, the potentiated synapses come from different input neurons for each different intermediate neuron, thus each intermediate neuron can recognize a different spike pattern.

## 2. The STDP rule used

The principle of an STDP rule is that presynaptic spikes contributing to make the postsynaptic neuron fire trigger a potentiation of the synapses. This type of rule was observed in the brain by (Bi & Poo 1998) (see Section III.C.1). This well-known STDP rule induces a potentiation of the synapse when a presynaptic spikes occurs shortly before a postsynaptic spike and a depression when a presynaptic spike occurs shortly after a postsynaptic spike. Here, we chose to use a very simple STDP rule, where the synapse is potentiated when a presynaptic spike occurs within a given coincidence time window before a postsynaptic spike, and is depreciated if a postsynaptic spike occurs alone (Figure IV-7.a). The coincidence time window was chosen to match the time resolution of the input spike train, which is about 0.05ms (see previous section). Two versions of this STDP rule were tested: an additive version and a multiplicative version, which both have simple properties (Figure IV-7.b).

For the additive STDP, each time a postsynaptic spike occurs, the synapse's weight is decreased by  $\Delta w_-$ . Additionally if this postsynaptic spike coincides with a presynaptic spike, the weight is increased by  $\Delta w_+$ , resulting in a total weight change of  $-\Delta w_- + \Delta w_+$ . The synapse's weight is then clipped between 0 and 1. Following this rule the mean weight variation is given by:

$$\left\langle \frac{\Delta w}{\Delta t} \right\rangle = f_{post} (p_{pair} \Delta w_+ - \Delta w_-)$$

Where  $f_{post}$  is the postsynaptic neuron's firing rate and  $p_{pair}$  the probability that a presynaptic spike is present in the coincidence time window before each postsynaptic spike. The synapse's weight will

converge towards 1 if this mean variation is positive and towards zero otherwise. There is thus a threshold probability  $p_{lim} = \Delta w_- / \Delta w_+$ , for which if  $p_{pair} > p_{lim}$ , the synapse's weight converges towards 1, and if  $p_{pair} < p_{lim}$ , the synapse's weight converges towards 0 (Figure IV-7.b, left).

In the multiplicative case, the weight's change depends on the synapse's current weight. For a postsynaptic spike alone, the synapse's weight is increased by  $w \Delta w_+$ , where  $w$  is the current synapse's weight. For a postsynaptic spike coinciding with a presynaptic spike, the weight is increased by  $(1-w) \Delta w_+$ . In this case, the mean weight variation is:

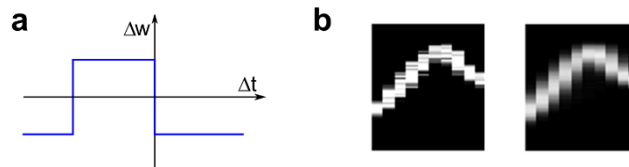
$$\left\langle \frac{\Delta w}{\Delta t} \right\rangle = f_{post} (p_{pair} (1-w) \Delta w_+ - (1-p_{pair}) w \Delta w_-)$$

The synapse's weight is naturally bounded and the equilibrium weight, for which the mean weight variation  $\left\langle \frac{\Delta w}{\Delta t} \right\rangle$  is null, is given by:

$$w_{eq} = \frac{1}{\frac{1-p_{pair}}{p_{pair}} \frac{\Delta w_-}{\Delta w_+} + 1}$$

In the case  $\Delta w_- = \Delta w_+$ , the equilibrium weight simply become  $w_{eq} = p_{pair}$  (Figure IV-7.b, right).

Thus, given  $p_{pair}$ , the equilibrium value of the synapse is known for these two STDP versions. We can then assume that, when the equilibrium has been reached, if a neuron of the intermediate layer has properly learnt a pattern from the input layer, it fires each time this pattern occurs and remains silent otherwise. Thus, for a specific synapse, the value of  $p_{pair}$  is close to the probability that the corresponding presynaptic neuron fires when the learnt pattern occurs. The multiplicative STDP seems to have nice properties with an equilibrium weight that is naturally bounded and can take continuous values. However, it can happen that a postsynaptic neuron learns two different patterns simultaneously, especially if they are partially overlapping. In this case, the weight of a synapse activated exclusively by one of the two learnt patterns will converge towards a lower weight as the synapse is depreciated when the other pattern occurs, and there is no simple way to force the neuron to choose one of the two patterns. With the additive STDP, this problem can be avoided by setting  $p_{lim} > 0.5$ . In this case, simultaneous learning cannot happen because for at least one of the two learnt patterns the corresponding synapses' weights would converge towards zero. For this reasons for the rest of this work we focused on the additive STDP, whose simple properties make it easy to manipulate.



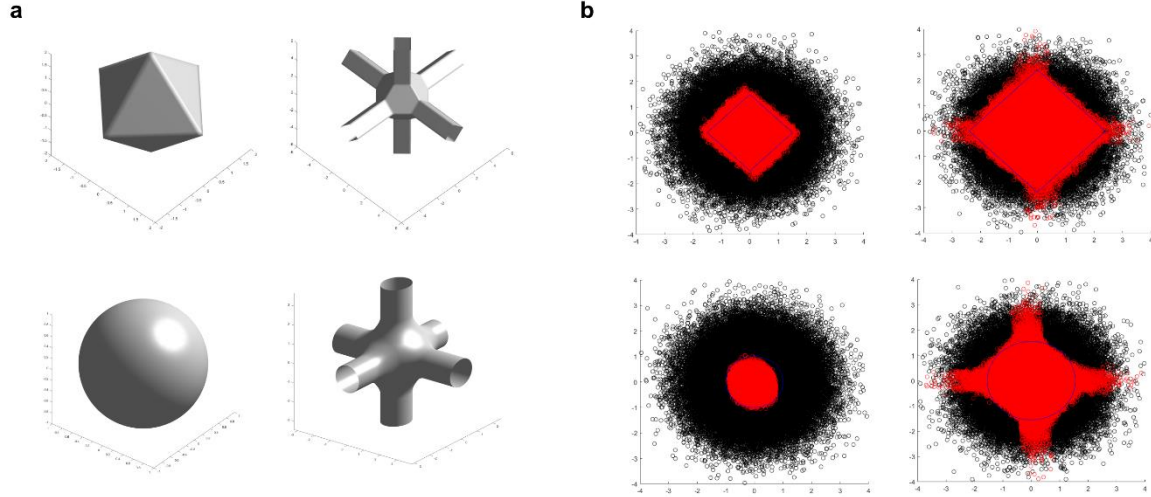
**Figure IV-7: STDP rule applied on the synapse connecting the input layer to the intermediate layer. (a) STDP rule.  $\Delta t$  is the interval between the presynaptic spike and the postsynaptic spike,  $\Delta w$  is the weight change. (b) Example of learnt weights for an additive STDP rule (left) and a multiplicative STDP rule (right). The two pictures represent the weight of the synapses connecting all input neurons to one particular intermediate neuron. Black is for a weight equal to zero and white for a weight equal to one.**

### 3. Receptive field size

At the beginning of the simulation, the synaptic weights connected to the input layer are initialized randomly. The mean weight is just high enough for neurons from the intermediate layer to fire and thus begin to learn patterns. A neuron has learnt a pattern once the weights of the synapses projecting on this neuron have all converged towards either one or zero. Then, for some input values the neuron reacts by firing whereas it remains silent for other values. The values for which the neuron fires constitute its receptive field.

The properties of the intermediate neurons' receptive fields are important for properly configuring this layer. They are studied in details in Annex A. Here, the input values are N-dimensional vectors corresponding to the signal values at the N encoded delays. The receptive field of each neuron tends to be centered on local density maxima of the possible input values, which typically corresponds to the mean shapes of action potentials emitted by different neural cells. Indeed the possible input values correspond to the different action potential waveforms, to which a Gaussian noise is added. We studied the receptive field's shape for both types of encoding methods, binary and triangular, described in Section IV.A. In both cases the receptive field has interesting properties when choosing  $\Delta w_+ = 2 \Delta w_-$  for the STDP rule. In both cases, it is bounded if  $Th > E_{max} * (N - 1)$ , where Th is the neuron's threshold, N the number of different synaptic delays and  $E_{max}$  is the maximum possible excitation generated by a single delay given the type of encoding. For the binary encoding method, when this bounded condition is met, the receptive field is an L1-norm ball of diameter  $2DV_mN-Th$ , where  $DV_m$  is half the sensitivity range size. For the triangular encoding, when  $Th > KD V_m(N-0.5)$  the receptive field is an L2-norm ball of diameter  $\sqrt{2}DV_m\sqrt{N - \frac{Th}{K*DV_m}}$ . Figure IV-8.a illustrates the different possible shapes taken by the receptive field.

These properties of the intermediate neuron receptive field have been confirmed by simulations (Figure IV-8.b). These simulation results, obtained for a simple case where the input values are low-dimensional and follows a Gaussian distribution, validate our theoretical results. They thus constitutes a first approximation of the real receptive field shape and allows us to choose the size of the receptive field to cover most of the occurrences of a learnt pattern, by adjusting either the size of the input neuron's sensitivity range or the intermediate neuron's threshold.



**Figure IV-8: Receptive field shape.** (a) Theoretical receptive fields obtained in a 3-dimensional case. The two at the top are obtained using the binary encoding, the two at the bottom for the triangular encoding. The two on the left are obtained on a bounded condition, and the two on the right for an unbounded condition. (b) Receptive fields obtained on simulation with Gaussian noise as input, in a 2-dimentional case. The two at the top are obtained using the binary encoding, the two at the bottom for the triangular encoding. The two on the left are obtained on a bounded condition, and the two on the right for an unbounded condition. The blues lines shows theoretical size of the receptive field, not taking into account the unbounded parts.

A receptive field with an L2-ball shape, obtained with the triangular encoding method, suits well the hypothesis of a white Gaussian noise. However this property is reached on conditions where the intermediate neuron's threshold is close to its maximum possible potential. In this condition, small relative variations on the neuron's potential greatly impact its behavior, which constitute a lack of robustness, especially since the results on the receptive field size were obtained using several approximations. Similarly, having a bounded receptive field is a nice property, but also requires a high threshold. We thus decided to use the binary encoding method, which is simpler to implement, and to allow an unbounded receptive field for more robustness in its size. The fact that the receptive field is unbounded should not impact the network behavior much, assuming that an input signal value falling in the "unbounded parts" of the receptive field has a low probability to occur. We chose to allow 3 out of the 10 used dimensions to be unbounded thus setting the threshold to  $2 \cdot 7 \cdot DV_m$ , or more generally  $2 \cdot 0.7 \cdot N \cdot DV_m$  for  $N$  dimensions,  $DV_m$  is half the sensitivity range for encoding. The receptive field size was then adjusted using  $DV_m$  to cover most of the pattern variations due to the noise. However the receptive field should not be too big, as we do not want a receptive field to cover two distinguishable patterns. In our different network implementations, different values of  $DV_m$  were tested, going from  $1.5\sigma_{noise}$  to  $2\sigma_{noise}$ , where  $\sigma_{noise}$  is the noise standard deviation.

Our receptive field study does not take into account the time integration of the consecutive sampling steps. To correct this, we assumed that the values received during an interval corresponding to the membrane time constant do not change significantly and we simply multiplied the threshold as if the same value was received at each sampling step.

#### 4. Winner-Take-All property mechanism

To ensure that only one neuron in the same layer fires at a time, and consequently that each neuron learns a different pattern, we implemented a classical winner-take-all (WTA) mechanism. Its principle is that when a neuron fires, the other neurons from the same layer are inhibited, preventing them from firing at the same time. Our implementation is a hard winner-take-all, in other words, at each simulation step, if at least one neuron has a potential higher than the firing threshold, the neuron with the maximum potential is selected, fires and inhibits the other neurons by resetting their potentials. As a result, instead of firing at the same time, the neurons fire one after the other, separated by a time interval that depends on their membrane time constant and the inhibition's strength. In our implementation, this time interval is about 0.05ms (Figure IV-9.a). The WTA mechanism ensures that the different neuron's receptive fields cover the possible values taken by the input without overlapping (Figure IV-9.b).

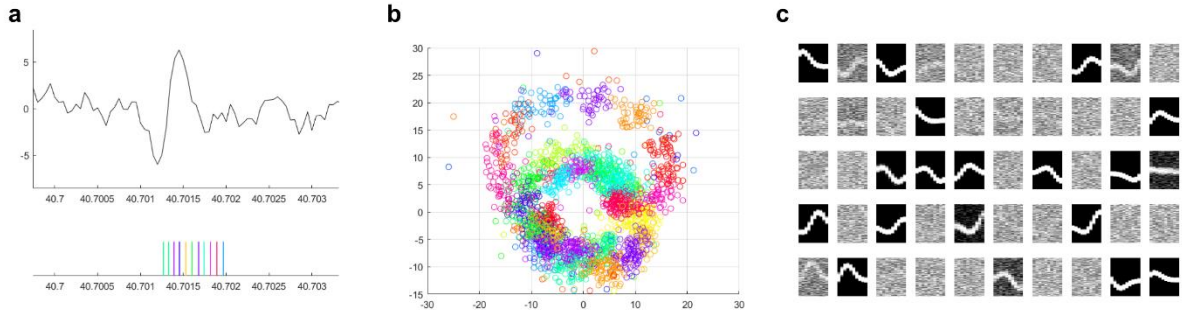
Each time the intermediate layer emits a spike, it means that the input value is in the receptive field of the neuron that fires. However, we have no information whether the input value is centered in the receptive field or close to its boundary. Two input values can be very close but recognized by two different neurons because they are from either side of a receptive field boundary. The information given by the intermediate layer would be improved if we allowed several neurons to fire at the same time, with partially overlapping receptive field. Then, the number of common spiking neurons for two different input values would give information about how close to each other these two values are.

To test this idea, we implemented a 2-WTA mechanism, where we allowed 2 neurons to fire before inhibiting all neurons in the layer. Even if the two neurons fire at the same time, they should not learn the same thing so that their receptive fields do not completely overlap. We assume that in fact the most excited neuron fires just before the other, and implemented a resource-based STDP inspired from (Hunzinger et al. 2012), where synaptic changes are proportional to a global resource variable. Each time a synaptic weight change occurs, the resource variable is decreased. The resource then progressively comes back to its initial value following a time constant. The evolution of the resource variable  $r$  is described by the following equation:

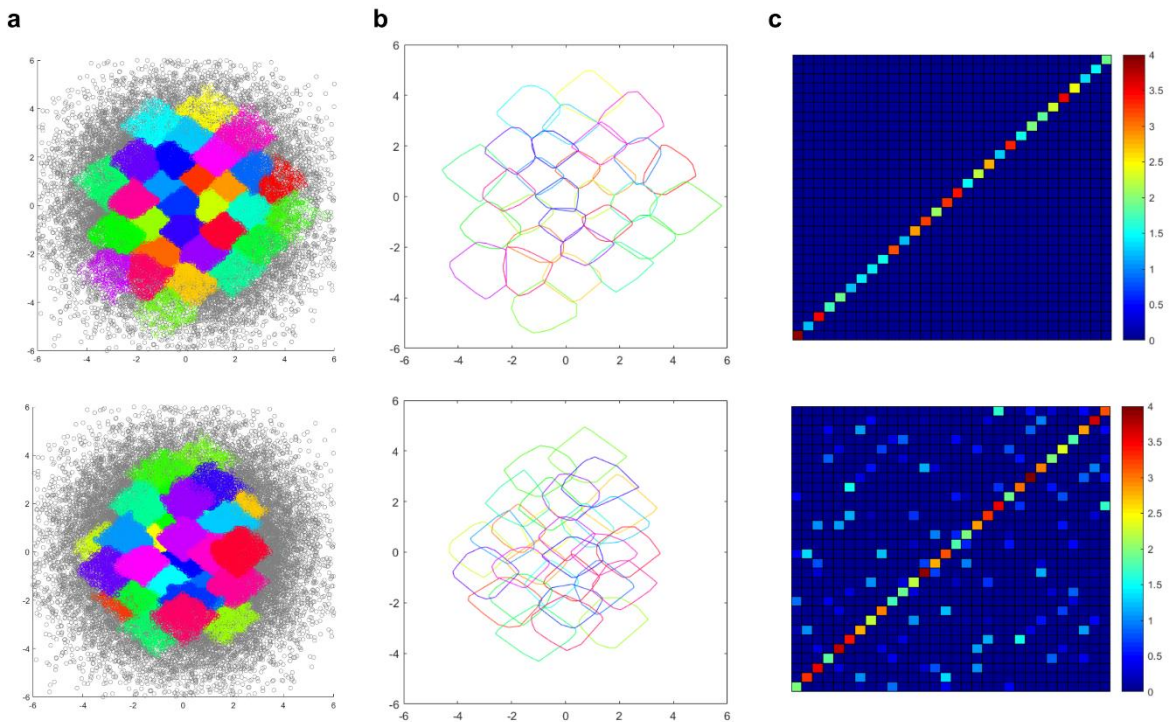
$$\frac{dr}{dt} = \frac{1}{\tau_{res}}(1 - r) - \sum_s r * f_{res} * \delta(t - t_s)$$

where  $\tau_{res}$  the resource recovery time constant,  $f_{res}$  the resource consumption factor and  $t_s$  the postsynaptic spike times of the intermediate layer. In our implementation, the time constant is set short enough so that between two firing steps of the intermediate layer, the resource completely recovers. Then the first spike triggers a synaptic weight change that takes almost all the resources. Thus, for an input value, two neurons fire but only the most excited learns. We showed by simulations with a random input signal that this indeed creates overlapping receptive fields (Figure IV-10). To test if this method actually improves the spike-sorting performance, we tested this intermediate layer implementation on our single electrode simulated data sets (see Section V.A.1). For this purpose we used as attention neuron's activity the attention neuron spike trains obtained with the one of our network implementation, ANNet. We then evaluated the intermediate layer output using the entropy criterion and the distance criterion described in Section V.B.3. These tests showed that the 2-WTA strategy gives better information for classification than the 1-WTA. We also tested different values for

$DV_m$  for the 2-WTA and 1-WTA. Indeed as receptive fields can overlap, they can be larger for the same distance separating their center. Though the entropy criterion did not show important differences, the distance indices show that the 2-WTA might lead to better results (Figure IV-11). Following these tests, in the last version of the network, we also decided to choose a sensitivity range with  $DV_m=2$ , which seems to give better results (Figure IV-11).

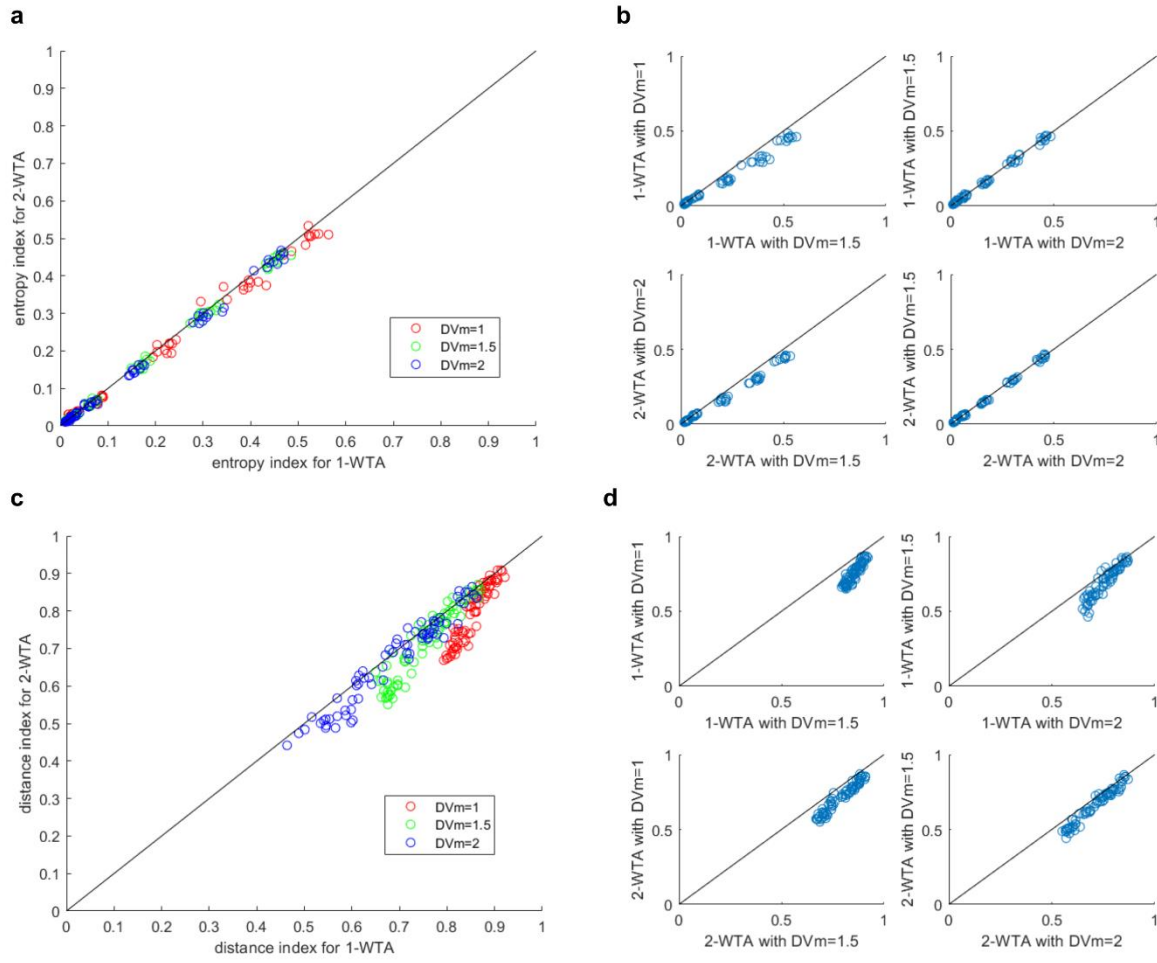


**Figure IV-9: Effect of the WTA mechanism. (a)** Intermediate neurons fire one after the other and not simultaneously. Top: input signal. Bottom: intermediate layer spike train, where the different colors stands for different neurons. **(b)** Receptive field of the intermediate neuron on data containing action potentials. The encoded input values have been represented in two dimension through a PCA. Each point represents an intermediate neuron firing. Different colors stands for different neurons **(c)** Learnt weights. Each square represent the weight of synapses projecting from all input neurons to one intermediate neuron. Black stands for 0 and white for 1.



**Figure IV-10: Effect of the k-WTA mechanism on the receptive fields. (a)** Receptive field obtained on simulation for a two-dimensional case. Different colors are for different neurons **(b)** Outline of the receptive fields shown in (a). **(c)** Measure of the common surface between two different receptive fields. Top: results for 1-WTA. Bottom: results for 2-WTA. The 2-WTA implementation shows an overlap between receptive fields.





**Figure IV-11: Comparison of the intermediate layer output quality for different k-WTA and different  $DV_m$ , on the simulated single electrode dataset. On each graph a comparison is made between two methods. Each point represents the two different results for one recording. A point below the diagonal means the method plotted vertical is better the one plotted in horizontal. (a) Comparison of the entropy index between the 1-WTA and 2-WTA implementations, for different values of  $DV_m$ . (b) Comparison of the entropy index between different values of  $DV_m$ , for the same k-WTA implementation. (c) Comparison of the distance index between the 1-WTA and 2-WTA implementations, for different values of  $DV_m$ . (d) Comparison of the distance index between different values of  $DV_m$ , for the same k-WTA implementation.**

## D. Output layer

As for the intermediate layer, the goal of the output layer is to recognize pattern in the spike train emitted by the previous layer, here the intermediate layer. It is also connected to the previous layer through an all-to-all synaptic connection, implementing an additive STDP rule, and is composed of LIF neurons in its first versions. For each occurrence of an action potential in the input signal, the intermediate layer outputs a spike sequence, which characterizes the action potential waveform. As action potentials with similar waveform recurrently occur in the signal, similar spike sequences are emitted by the intermediate layer, constituting a pattern the output layer should learn to recognize. The length of these spike sequences can vary depending on the amplitude and duration and the action

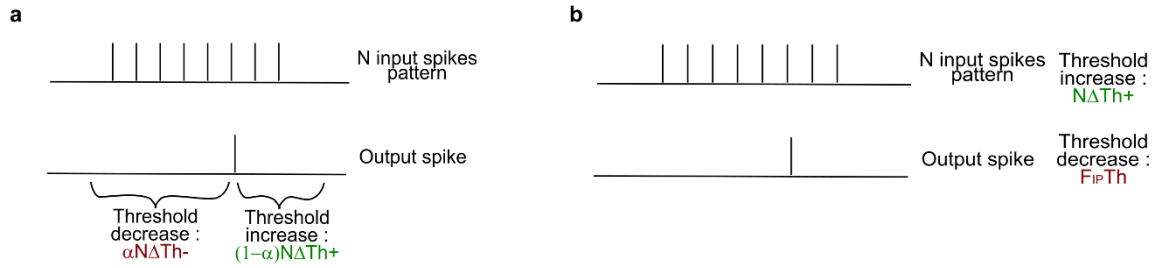


potential. In spite of these possible length variations, the output layer, which processes these spike sequences, should emit one spike for each spike sequence received. Moreover, the output layer should be robustly able to differentiate partially overlapping spike patterns, for example two sequences that have the same beginning but end differently. In an extreme case a spike pattern can contain a subsequence corresponding to another spike pattern to recognize. To achieve this goal, several mechanisms can be used, which will be described in the next sections.

## 1. Intrinsic plasticity

Intrinsic plasticity (IP) is a form of plasticity at the neuron level (Zhang & Linden 2003; Lazar 2009; Aswolinskiy & Pipa 2015). In our case, its purpose is to adapt the neurons' threshold to the size of the learnt pattern (e.g., length of the intermediate layer spike sequence). Indeed if the neuron's threshold is too low compared to the size of the pattern it has learnt, it will not only spike for this incoming patterns but also for shorter ones corresponding to other signal waveforms. In other words, it will be too tolerant to pattern differences and not be able to become specific to a single pattern. Also if the threshold is too low, the neuron might fire before the end of the intermediate spike sequence and be unable to discriminate patterns with the same beginning but different endings. By contrast, if the threshold is too high compared to the pattern size, it will not be able to detect the pattern. Thus, it is crucial to adapt the threshold of each output neuron to the size of the pattern it is tuned to, especially when this size is not the same across patterns.

To implement an IP, we considered two different mechanisms. In a first version, for each input spike received within a given time window before the neuron's spike, the threshold is decreased by a constant  $\Delta Th^-$ , and for each input spike received within a given time window after the neuron's spike, the threshold is increased by a constant  $\Delta Th^+$  (Figure IV-12.a). With this method, the neuron's threshold adjusts to fire after receiving a proportion  $\alpha = \Delta Th^+ / (\Delta Th^+ + \Delta Th^-)$  of the learnt pattern. This method has the strong inconvenient that it can only be used with architectures that allow the output neurons to fire before the end of the intermediate layer spike pattern. It was thus only used in our first network implementation MiniNet and soon replaced by the following IP mechanism. In our second method, each time the neuron fires, its threshold  $Th$  is decreased proportionally to its value, reaching a new value  $(1 - F_{IP})Th$ . Moreover, for each spike received within a given time window around the neuron's spike, the threshold is increased by a constant  $\Delta Th^+$  (Figure IV-12.b). The equilibrium is reached when the increase and the decrease compensate each other, giving an equilibrium threshold  $Th_{eq} = N * \Delta Th^+ / (1 - F_{IP})$  for a pattern containing on average  $N$  spikes. As a result, if the neuron has learnt to recognize a pattern, the equilibrium threshold will be proportional to the average size of this pattern. During our tests, we also implemented a variation of this mechanism, where the total threshold is the sum of a fixed threshold and a variable one subject to IP. However this variation was not used for any complete network implementation.



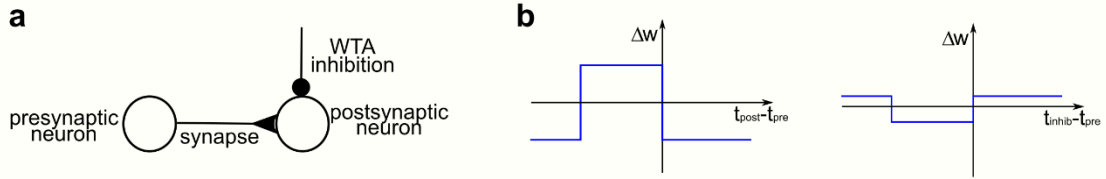
**Figure IV-12: Two different implementation of the intrinsic plasticity rule. (a) Input spikes arriving before an output spike decrease the threshold whereas input spikes arriving after an output spike increase the threshold. (b) Each output spike decreases the threshold whereas input spikes coinciding with an output spike increase the threshold.**

Is it thus possible to adapt each neuron's threshold individually to be proportional to the learnt pattern size. For commodity, and to be coherent with the WTA mechanism that is also applied on this output layer and selects the neuron with the highest potential, we can consider equivalently that the neuron's potential is divided by its threshold and that the neuron fires when its potential reaches 1. This threshold adaptation allows the neuron to be sensitive to the relative difference between the learnt pattern and the presented pattern rather than the absolute difference, as its potential is normalized by the learnt pattern size. Combining IP with synapses that can have negative as well as positive values would in theory allow to distinguish two patterns one included into the other. Indeed, a neuron that has learnt the large pattern will not fire for the small pattern as its threshold is high, and the neuron that has learnt the small pattern will not spike for the large pattern as synapses with negative weight, not present in the small learnt pattern, are activated and decrease the neuron potential.

The difficulty with this IP is how to deal with the neurons that have not yet learnt a pattern. Indeed these neurons should be able to learn new patterns, even the shortest ones, but should not fire before other neurons that have already learnt a pattern and might have a higher threshold. It is not possible to reach these conditions by just setting the initial threshold appropriately, and we thus need another mechanism to take full advantage of the IP.

## 2. Lateral STDP

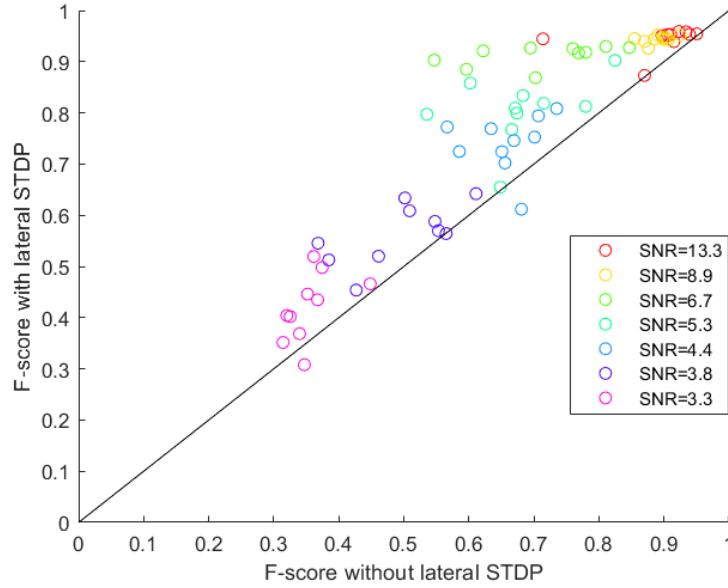
The idea of the lateral STDP is to implement a synaptic plasticity that prevents a neuron from recognizing a pattern when another neuron is learning this pattern. To achieve this, we took advantage of the fact that when a neuron fires, other neurons from the same layer receive an inhibition signal. Thus, when a neuron receives a lateral inhibition, it triggers synaptic weight changes on its incoming synapses (Figure IV-13).



**Figure IV-13: Lateral STDP principle.** (a) Elements involved in the lateral STDP: presynaptic spikes, postsynaptic spikes and inhibitions received by the postsynaptic neuron. (b) Weight changes induced by the lateral STDP. Left: classical STDP, postsynaptic spikes alone depress the synapse, pre and postsynaptic spike coincidences potentiate the synapse. Right: lateral STDP rule, presynaptic spikes alone potentiate the synapse, presynaptic spike and inhibition coincidences depress the synapse.

For the classical STDP implemented on the same synapse, the weight convergence depends on the probability to have a presynaptic spike when a postsynaptic spike occurs. For the lateral STDP we want the weight convergence to depend on the probability to have an inhibition each time a presynaptic spike occurs. Thus, each time a presynaptic spike occurs alone, the synapse weight is increased by a constant and when the presynaptic spike is paired with an inhibition the weight is decreased by a constant. As a result, similarly to the classical STDP, the synapse weight will converge towards 0 if the probability to have an inhibition each time a presynaptic spike occurs is above a threshold probability and towards 1 otherwise. This makes a neuron more likely to fire when a pattern has not been learnt by any neuron, as no lateral inhibition occurs in this case, and less likely to fire when a pattern is learnt by another neuron. However, when the neuron is learning a pattern, we want the synaptic weights to converge according to the classical STDP. A simple way to ensure this when the two types of STDP are implemented on the same synapses, is to set synaptic weight changes for the lateral STDP to be much lower than the weight change for the classical STDP, so that the classical STDP takes over the lateral STDP.

This lateral STDP, combined with the classical STDP, can also be combined with IP. Indeed, the neuron's threshold can be initialized low, so that neurons can learn new short patterns. Then, once a neuron begins to learn a pattern, even if its threshold increases, other neurons will be less likely to fire when the pattern is presented thanks to the lateral STDP. This solution was tested in our first implementation of the network (MiniNet), improving the F-score from 0.81 to 0.89 on our simulated preliminary dataset (see Section V.A.1). We also tested it on our last network implementation (LTSNet), where it shows a systematically better result on our simulated single electrode dataset (Figure IV-14).

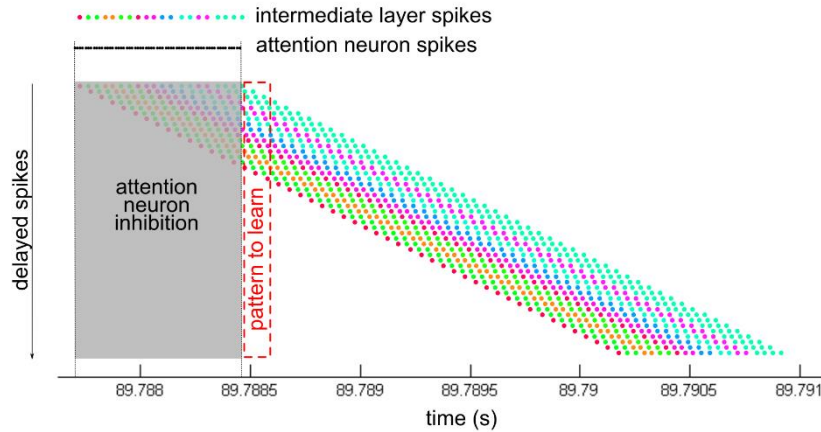


**Figure IV-14: Improvement of the performance with lateral STDP implemented on the output layer of the LTSNet network. Each point represents the performance with and without lateral STDP for one recording from the simulated single electrode dataset.**

### 3. Delays and inhibition by the attention neuron

One possibility to make the output neurons wait till the end of spike sequence before firing is to inhibit them when the attention neuron is firing. For the neurons to be able to fire after the end of the spike sequence, they need to receive an excitation after the inhibition by the attention neuron. To do so, we introduced delayed synapses between the intermediate layer and the output layer, the same way there are delays between the input layer and the intermediate layer. This also has the advantage to bring information about the spike timing inside the spike sequence. The interval between two consecutive synaptic delays determines the time resolution for this timing information. The membrane time constant of the output neuron is chosen of the same order as this time resolution, as we want to detect a coincidence between spikes arriving at the same time but with different delays.

With this structure, illustrated in Figure IV-15, the output layer receives spikes during a few milliseconds after the end of the detected action potential. Randomly initialized neurons are thus excited during a few milliseconds. By contrast neurons that have learnt to recognize a spike coincidence pattern are excited only when the learnt pattern occurs, that is during about 0.02ms, which is the time resolution used in the intermediate layer. As described in (Masquelier et al. 2008), during learning the neuron fires always earlier in the received spike sequence, until it recognizes the beginning of the sequence. To prevent the neuron from learning what comes before the end of the action potential, the inhibition we used from the attention neuron is a presynaptic inhibition that prevents the spikes from arriving to the neuron (see Figure IV-15). These spikes are thus not taken into account for the STDP rule.



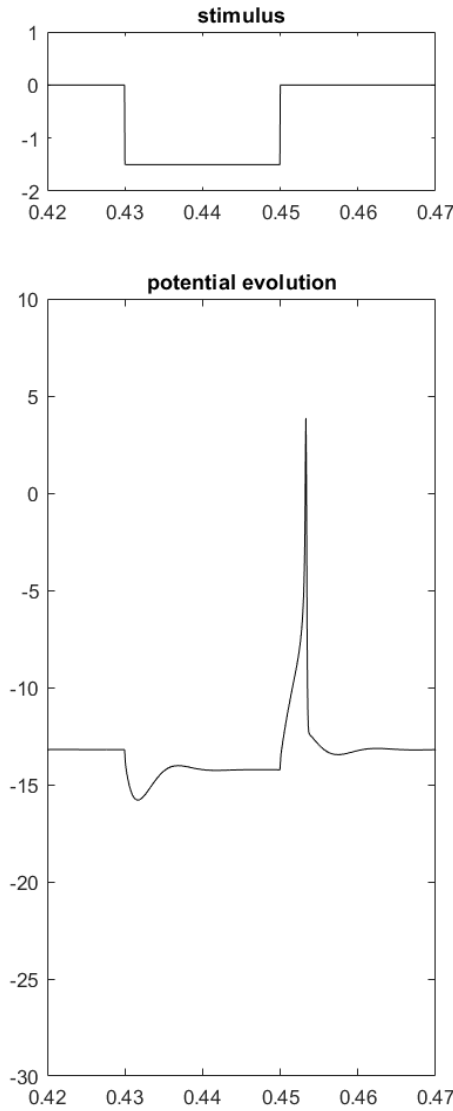
**Figure IV-15: Spike pattern received by the output layer when using a structure with transmission delays on synapses from the intermediate layer and inhibition from the attention neuron**

The neurons that have learnt a pattern are excited during a short coincidence time when this pattern is presented whereas neurons that have not learnt anything are excited during a few milliseconds. The neuron's threshold can thus be initialized high, so that before learning the neuron fires late. As a neuron begins to learn a pattern, it fires earlier in the sequence while its threshold decreases to become proportional to the pattern size.

This structure was used in our second version of the network (ANNet), but is not totally satisfying as it is very complex, rely on a precise timing from the attention neuron and require many synapses. The LTS neuron model presented in the next section allowed us to replace this complex structure with a much simpler solution.

#### 4. LTS neurons

The structure of the output layer and its plasticity rules make it complex to implement. We thus sought a more elegant and less power-consuming way to achieve the work performed by the output layer. The output layer described so far was designed based on the fact that the main properties of the LIF neuron model is to fire when it receives an excitation. However, other interesting behaviors have been observed in the brain. For example, LTS neurons have the property to generate a potential rebound, and even spikes if the rebound is high enough, after being inhibited (Nanami & Kohno 2016) (see Figure IV-16). This is particularly interesting in our case, as we want neurons from the output layer to wait the end of the pattern before firing.



**Figure IV-16: Example of potential rebound of the DSSN neuron model, generating a spike after the end of the stimulus**

Our idea to use a neuron model that has a rebound property was initiated during a collaboration with Timothée Lévi and Takashi Kohno from the University of Tokyo. Following a 2-week stay at their facility, focused on FPGA implementation (see Section VI.F), we tested if the properties of their neuron model, called DSSN (Nanami & Kohno 2016), fit our problem. After adjusting the membrane time constant to our needs, we found that, after an inhibitory stimulus, the latency of the first spike emitted by the neuron is a decreasing function of the stimulus integral and that the number of spikes emitted is roughly proportional to the stimulus integral (Figure IV-17.a). Assuming that the total stimulus reflects how well the presented pattern matches the learnt pattern, this latency property is very interesting for our problem. After adjusting the model's parameters, we managed to obtain only one spike for a large range of stimulus, which was the desired effect (Figure IV-17.b).

Though this model has interesting properties, simulating its equation requires a very short time step as it models the potential spiking dynamic, which is computationally demanding. We designed a simpler model, falling in the Integrate and Fire category, which means that even though the potential evolution is described by a differential equation, the spikes are modeled as discrete events that occur

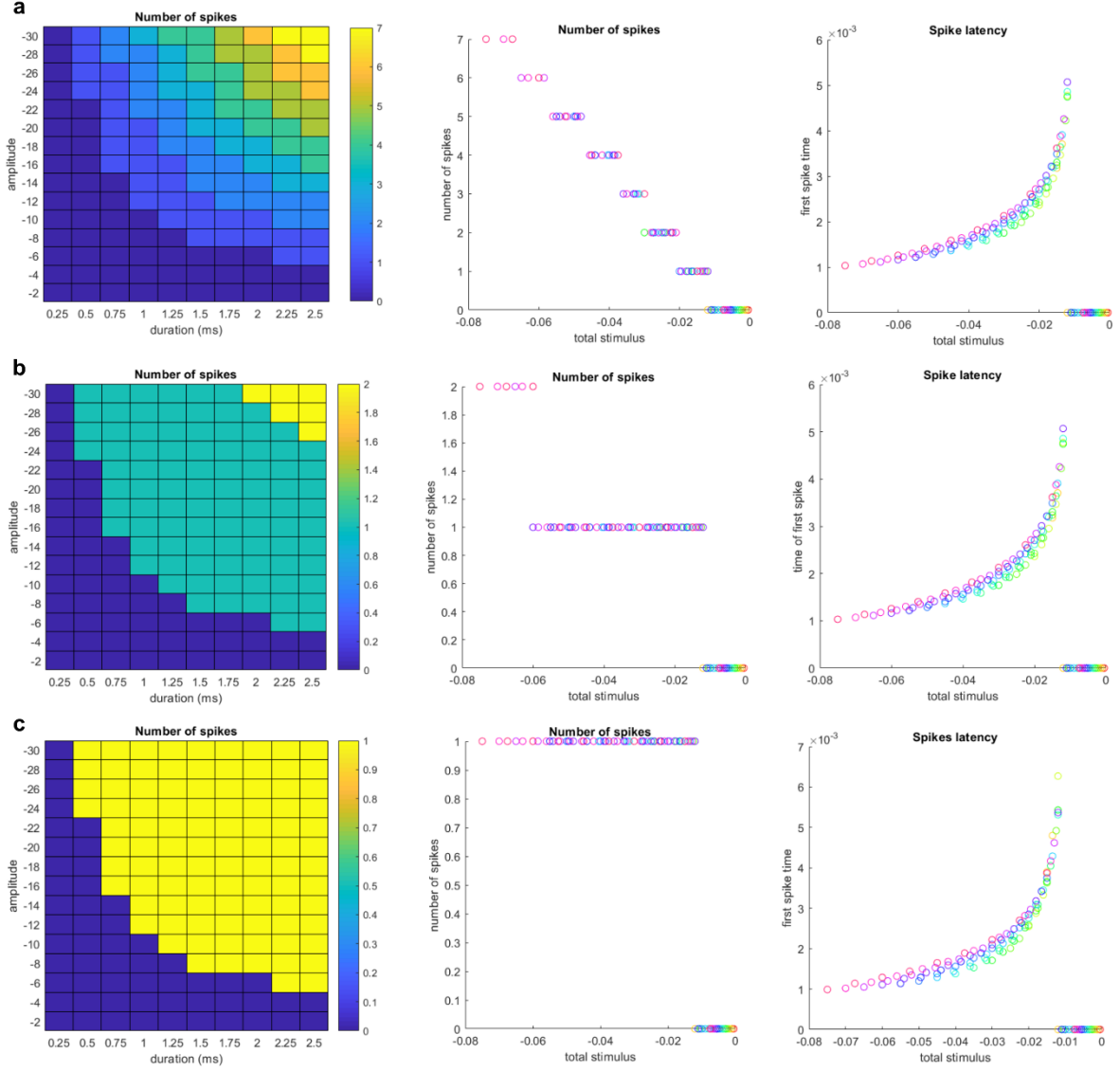
when the potential reaches a threshold. This way to model spikes also has the advantage that the spikes can be used to reset the neuron after firing or for the WTA mechanism. Indeed, a lateral inhibition cannot be used as WTA for this type a neuron, as inhibition can trigger spikes through a rebound.

In our simplified model, the neuron's potential is governed by the following equation:

$$\begin{cases} \tau_m \frac{dV}{dt} = -V + q + gI_{stim} \\ \frac{\tau_m}{\varepsilon} \frac{dq}{dt} = -q + f(V) \end{cases}$$

$$\text{with } f(V) = \begin{cases} \alpha_n V & \text{if } V < 0 \\ \alpha_p V & \text{if } V \geq 0 \end{cases}$$

where  $V$  is the neuron potential,  $q$  is an adaptation variable that triggers the rebound after inhibition,  $\tau_m$  is the membrane time constant,  $\varepsilon$  is a constant that makes  $q$  vary slower than  $V$ ,  $I_{stim}$  is the stimulus current that corresponds to the received spikes and  $g$  is a constant. When the potential  $V$  reaches the neuron's threshold, both  $V$  and  $q$  are reset. After adjusting the parameters to the values shown in Table IV-1, we obtained similar properties as the DSSN model (Figure IV-17.c). These parameters were used in all our tests, except for  $g$ , which plays the same role as the threshold for the LIF neuron model. Indeed, when  $g=1$ , the parameters are adjusted so that our LTS neuron fires for a total stimulus of 1. For the LTS neuron to fire for a threshold stimulus  $Q_{th}$ ,  $g$  is set to  $g=1/Q_{th}$ .



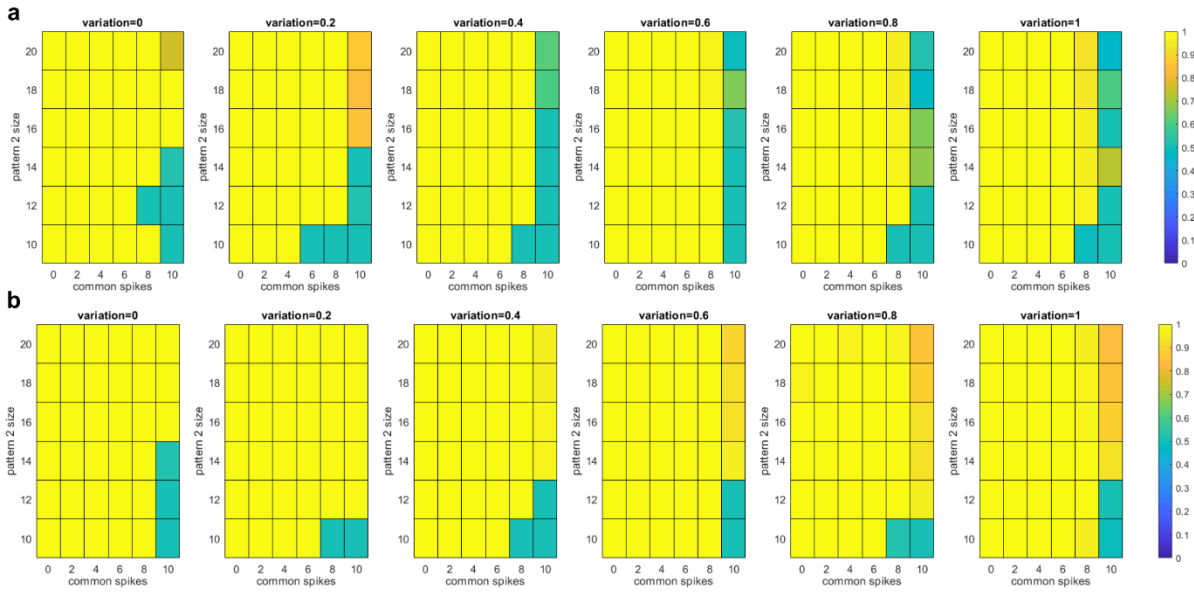
**Figure IV-17: Spiking properties of the LTS neuron after receiving a negative stimulus, with different models. (a) Original model from (Nanami & Kohno 2016). (b) Same model with modified parameters. (c) Simplified LTS model used in our network. Left: number of spikes emitted depending on the stimulus amplitude and duration. Middle: number of spikes emitted depending on the total stimulus (amplitude\*duration). Different colors represent different stimulus amplitudes. Right: time of the first spike after the end of the stimulus, depending on the total stimulus (amplitude\*duration). Different colors represent different stimulus amplitudes.**

**Table IV-1: Parameters used for the simplified LTS model**

| Parameter  | Value |
|------------|-------|
| $\epsilon$ | 0.03  |
| $\alpha_n$ | -200  |
| $\alpha_p$ | -10   |
| $Th$       | 480   |
| $g$        | 100   |



To test this neuron model independently from the rest of the network, we tested it on arbitrary patterns. We generated different datasets each containing two different spike patterns for the output layer to discriminate. These datasets are generated with the following process. 5000 pattern occurrences are generated, separated by a time interval of 20ms, each pattern occurrence lasting between 0.5 and 1ms. For each pattern occurrence, one of the two patterns is chosen randomly and variations are randomly introduced respecting a maximum difference with the reference pattern. In each dataset, one pattern is composed of ten spikes, mimicking 10 different afferents. The second pattern's size varies between 10 and 20. The two patterns share some common spikes (stemming from the same afferents), the number of which varies from 0 to 10. Additionally, at each occurrence of a pattern in the simulation, the presented pattern is randomly modified by adding and deleting some spikes. These initial tests show that an output layer constituted of simplified LTS neurons can distinguish patterns robustly (Figure IV-18.a). We also performed tests where the threshold was the sum of a fixed threshold of 5 and a variable threshold subject to IP. The parameters are set so that the variable threshold stabilizes to one quarter of the size of the learnt stimulus. With the IP, the output layer is even capable of distinguishing two patterns completely included one into the other (Figure IV-18.b). In our final single-electrode network implementation (LTSNet), we chose not to use a threshold adaptation, as it brings no obvious performance improvement for spike-sorting.

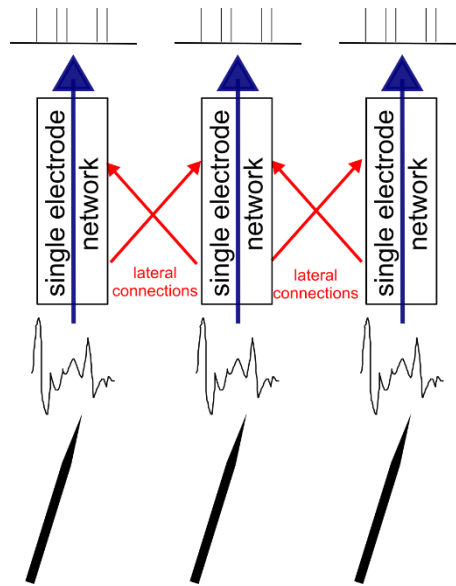


**Figure IV-18: Discrimination performance, assessed through an F-score, obtained with the LTS neuron on different conditions. The testing dataset contains two different patterns. Between each testing dataset, the size of one of the two patterns varies, the size of the common parts between the two patterns varies, and the maximum variation between occurrences of the same pattern varies. (a) Results obtained without IP, (b) Results obtained with IP.**

## E. Adaptation to polytrodes

After testing our STDP network on single electrode data, we worked on adapting the network architecture to multiple electrodes. The principle is to take advantage of the fact that an action potential might be recorded by several electrodes, which give additional information. First preliminary tests were done with the real tetrode data (see Section V.A.2), which are recordings from 4 electrodes

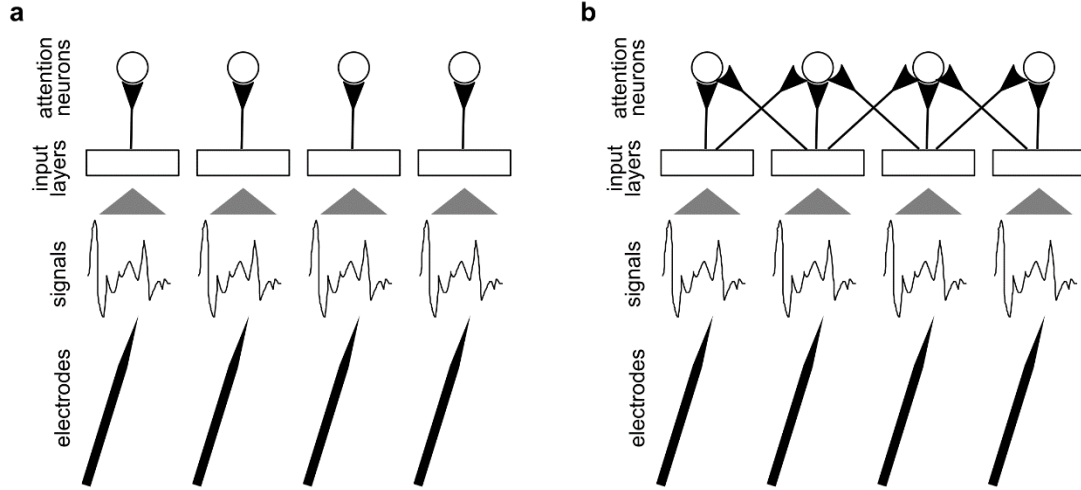
close enough from each other to record the same cell. These preliminary tests, done with the ANNet version of the network (see Section VI.B), show that adapting the structure can actually improve the performance. We then performed extended tests on a set of simulated polytrode recordings. The general idea to adapt the single network structure to multiple electrodes is the following. The basic network structure is constructed by duplicating the single network structure to process each electrode in parallel. With this basic structure, the shared information between electrodes is not used. Then, to take advantage of the redundancy between neighboring electrodes, lateral connections between the parallel single-electrode networks are introduced (Figure IV-19). We worked on the benefits of adapting the architecture for two specific parts of the network: the attention neuron and the output layer.



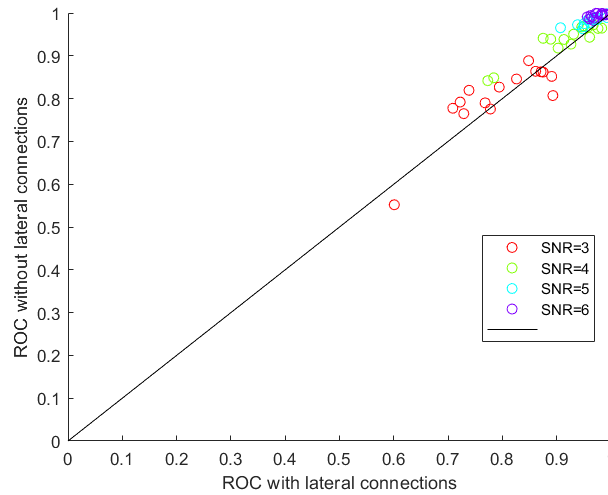
**Figure IV-19: General principle of our adaptation of the network to the case of multiple electrodes**

#### 1. Test on attention neuron

Each electrode signal is encoded into a spike train by an independent input layers. These input layers are projecting to several attention neurons, whose role is to detect action potentials at different positions in the electrode array. With a purely parallel structure, each input layer projects to one attention neuron whose role is then to detect action potentials on the corresponding electrode. We wanted to test if connecting each attention neuron to several input layers, encoding neighboring electrodes could improve the detection performance. To do so, we designed a network where there is as many attention neurons as input layers, and each attention neuron is connected to three input layers (see Figure IV-20). This network was tested on our simulated polytrode dataset (see Section V.A.1), and run with different thresholds for the attention neuron, in order to compute a ROC score (see Section V.B.2). The results show that the structure with lateral connections did not give better results than the purely parallel structure (see Figure IV-21), and was thus not used in our final implementation. However other structures could be tested, including for example inhibitory connections.



**Figure IV-20: Two different polytrode structures for the attention neuron. (a) Baseline purely parallel structure. (b) Structure with lateral connections.**

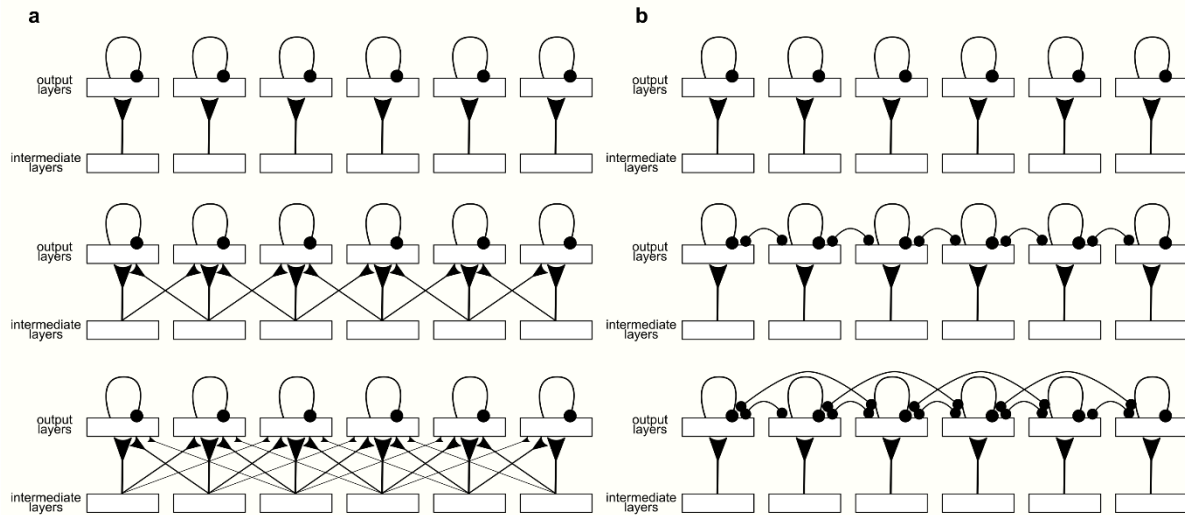


**Figure IV-21: Comparison of the ROC scores obtained with the two different structures for the attention neuron. Each point corresponds to a recording from the simulated polytrode dataset.**

## 2. Output layer structure

Our base structure for processing several electrodes is constituted of several parallel subnetworks, each of them processing the signal of one electrode. As for the attention neuron, at each layer of the network the different sublayers can be connected to several of the preceding sublayers to process more information. We decided to adopt such a structure for the output layer rather than the intermediate layer, as it requires fewer synapses. Ideally, the output layer should output one spike for each recorded action potential, regardless of the electrodes on which it is recorded. Each output sublayer implements a WTA mechanism which prevents several neurons from the same sublayer to fire at the same time. As an action potential can be recorded on several neighboring electrode, several output sublayers could fire for the same action potential. It is thus necessary to extend the WTA mechanism to neighboring output sublayers. Overall, we tested different structures for the output

layer, combining these two aspects. First, each output sublayer is connected to either one, three or five intermediate sublayers (see Figure IV-22.a), with a synaptic weight higher for connections corresponding to electrodes closer to each other. Second, the WTA mechanism is either unchanged or included lateral inhibition between three or five output layers. (see Figure IV-22.b).



**Figure IV-22: Different polytrode structures tested for the output layer. (a) Different possible structures for the connection between the intermediate and the output layer. (b) Different possible structures for the WTA mechanism. The two structural aspects were tested with all possible combinations.**

These structures were tested for the last version of our network (PolyNet), on our simulated polytrode dataset, which simulated a recording with ten electrodes arranged in a line. Table IV-2 shows the mean F-scores obtained. When there are less WTA connections between output sublayers than connections between intermediate and output sublayers, the performance is degraded. This can be explained by the fact that lateral connections between intermediate and output sublayers increase the number of output sublayers likely to fire for one action potential. Therefore the number of connections for the WTA mechanism needs to be adjusted to take this effect into account. Our hope was that introducing lateral connections for both the intermediate to output layer connection and the WTA mechanism would increase the performance. Though this is not the case, we can see on some simulated recordings that, when no lateral connections are used an action potential can be detected by several output sublayers (Figure IV-23.a), which lowers the final score. When using lateral connections, this effect is reduced as each different action potential is detected mainly by one output sublayer, but the recall is overall lower (Figure IV-23.b). In our final version, we chose to use the three to one intermediate to output connections in combination with a five sublayer WTA mechanism, which is the structure that give the best results apart from the one with no lateral connections. Future work should focus on improving the structure to get a better recall.

**Table IV-2: Mean F-scores obtained with the different structures tested on the simulated polytrode dataset**

| WTA connections                    | 1     | 3     | 5     |
|------------------------------------|-------|-------|-------|
| Intermediate to output connections |       |       |       |
| 1                                  | 0.476 | 0.448 | 0.448 |
| 3                                  | 0.332 | 0.421 | 0.456 |
| 5                                  | 0.177 | 0.24  | 0.448 |

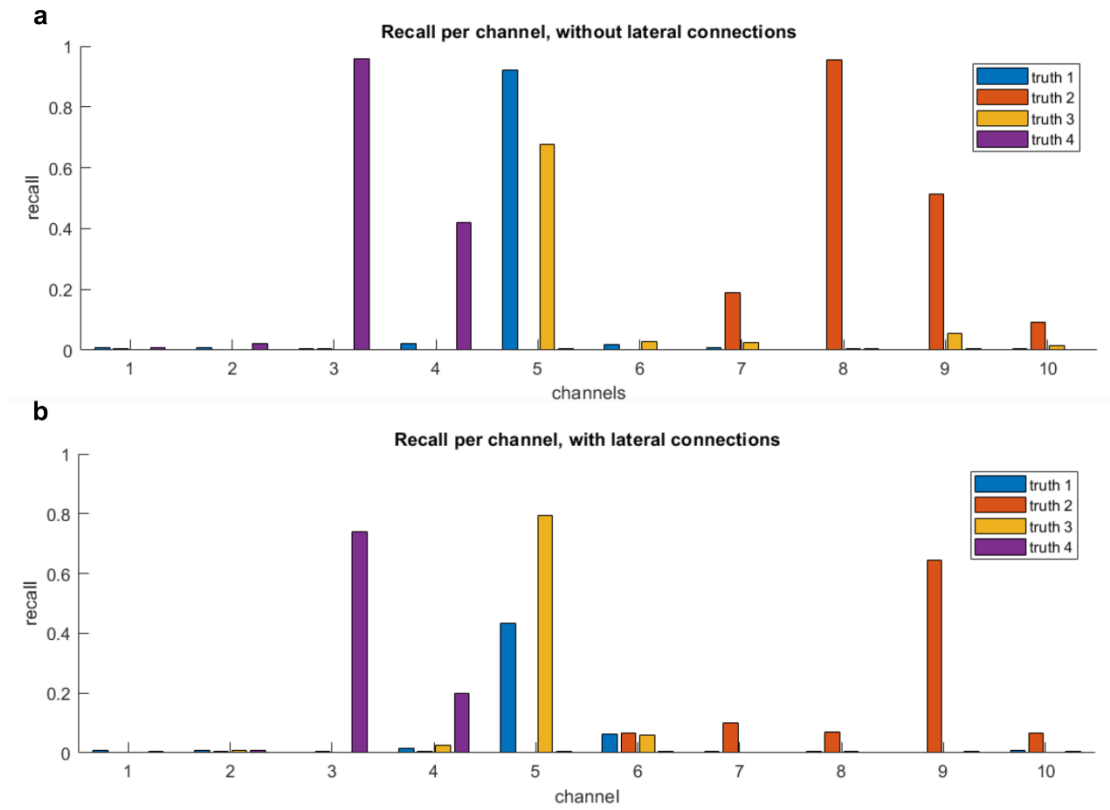


Figure IV-23: Recall of action potential from each different neural cell, on each output sublayer, on a simulated polytrode recording with four different neural cell and an SNR of 6. (a) Results obtained with the purely parallel structure. (b) Results obtained with a structure with a WTA connecting 5 output sublayers, and each output sublayer connected to three intermediate sublayers.

## References

- Abbott, L.F. et al., 1997. Synaptic Depression and Cortical Gain Control. *Science*, 275, pp.220–223.
- Aswolinskiy, W. & Pipa, G., 2015. RM-SORN: a reward-modulated self-organizing recurrent neural network. *Frontiers in Computational Neuroscience*, 9(March), pp.1–15. Available at: <http://journal.frontiersin.org/article/10.3389/fncom.2015.00036>.
- Bi, G.Q. & Poo, M.M., 1998. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(24), pp.10464–10472.
- Gerstner, W. et al., 1996. A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383(6595), pp.76–81.
- Gerstner, W., Ritz, R. & van Hemmen, J.L., 1993. Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns. *Biological Cybernetics*, 69(5–6), pp.503–515.
- Ghosh-Dastidar, S. & Adeli, H., 2007. Improved Spiking Neural Networks for EEG Classification and Epilepsy and Seizure Detection. *Integrated Computer-Aided Engineering*, 14, pp.187–212. Available at: <http://iospress.metapress.com/content/LW681773V3036222>.
- Hopfield, J.J., 1995. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376(6535), pp.33–36.
- Hunzinger, J.F., Chan, V.H. & Froemke, R.C., 2012. Learning complex temporal patterns with resource-dependent spike timing-dependent plasticity. *Journal of Neurophysiology*, 108(2), pp.551–566.
- Lazar, A., 2009. SORN: a Self-organizing Recurrent Neural Network. *Frontiers in Computational Neuroscience*, 3(October), pp.1–9. Available at: <http://journal.frontiersin.org/article/10.3389/neuro.10.023.2009/abstract>.
- Masquelier, T., Guyonneau, R. & Thorpe, S.J., 2008. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS ONE*, 3(1).
- Nanami, T. & Kohno, T., 2016. Simple cortical and thalamic neuron models for digital arithmetic circuit implementation. *Frontiers in Neuroscience*, 10(MAY), pp.1–12.
- Zhang, W. & Linden, D.J., 2003. The other side of the engram: experience-driven changes in neuronal intrinsic excitability. *Nature reviews. Neuroscience*, 4(11), pp.885–900.



## V. PERFORMANCE ASSESSMENT

### A. Testing datasets

#### 1. Simulated data

To assess the spike-sorting performance of the network, we need to test it on data with known ground truth and controlled parameters. To do so we generated simulated data, both in the case of a single electrode and of multiple electrodes.

##### a) *Single electrode datasets*

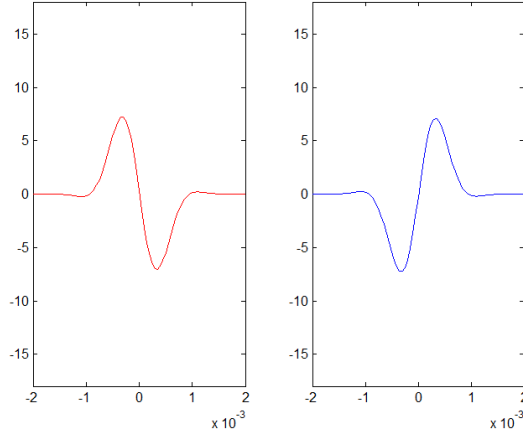
In case of a single electrode, our method was strongly inspired from (Adamos et al. 2008). For each simulated cell, the ground truth is generated according to a Poisson process. The action potential shape corresponding to each simulated cell is modeled according to the following equation:

$$V(t) = A * \cos\left(2\pi \frac{t - t_{ph}}{\tau_1}\right) * \exp\left(-\left(\frac{2.3548 t}{\tau_2}\right)^2\right)$$

Where  $A$ ,  $t_{ph}$ ,  $\tau_1$  and  $\tau_2$  are parameters that determine the shape of the action potential for one simulated cell. For commodity, we introduce an intermediate parameter  $A_{max}$ , which corresponds to the maximum amplitude of the waveform, from which  $A$  is computed. Each time a simulated neural cell fires, the corresponding waveform is added to the signal. A correlated Gaussian noise is also added to the signal, generated through a dynamical Ornstein-Uhlenbeck process, following the equation  $dX_t = -X_t dt / \tau_{noise} + dW_t$ , where  $\tau_{noise}$  is the time constant of the process, and  $W_t$  is a Wiener process, which means the  $dW_t$  are independent and identically distributed, following a Gaussian distribution. For our simulations, the time constant is set to  $\tau_{noise} = 0.1$  ms. We chose to generate all our recordings with a sampling frequency of 20 Hz. For each simulated recording the signal-to-noise ratio (SNR) is computed as  $SNR = \langle |A_{max}| \rangle / \sigma_{noise}$ , where  $\langle |A_{max}| \rangle$  is the average action potential amplitude and  $\sigma_{noise}$  is the noise level.

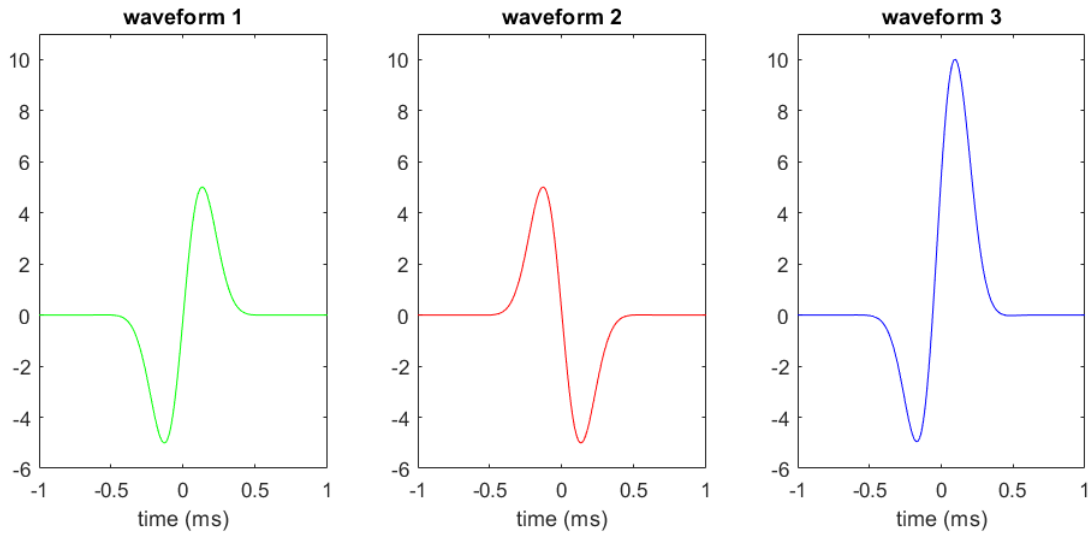
The very first tests on our STDP network were conducted on a preliminary dataset, generated through this method. Twenty simulated recordings were generated, lasting 200 s, each containing two different waveforms (see Figure V-1) occurring at 10Hz and 12Hz respectively, with a noise level  $\sigma_{noise}$  of 2, leading to an SNR of 3.5. This preliminary dataset has two main drawbacks. First the noise level does not vary between recordings, and second the chosen waveforms have a width of about 2ms, which is quite large for an action potential and makes them easier to detect.





**Figure V-1: The two waveforms used in the preliminary dataset**

We thus generated a more extensive dataset, which then constituted our main simulated single electrode dataset. The recordings of this dataset also lasts 200 s. They contain three different waveforms defined by the parameters shown in Table V-1 (see also Figure V-2), occurring at 3.3Hz each. Seven different noise levels were used, corresponding to seven different SNR (see Table V-2). For each noise level, ten recordings were simulated, leading to a total of 70 simulated recordings, each of 200 s duration.



**Figure V-2: The three waveforms used in the simulated single electrode dataset**

**Table V-1: Parameters used to generate the three waveforms of the simulated single electrode dataset**

|                   | $A_{\max}$ | $\tau_1(\text{ms})$ | $\tau_2(\text{ms})$ | $t_{\text{ph}}$ |
|-------------------|------------|---------------------|---------------------|-----------------|
| <b>Waveform 1</b> | 5          | 1                   | 0.5                 | 0.25            |
| <b>Waveform 2</b> | 5          | 1                   | 0.5                 | -0.25           |
| <b>Waveform 3</b> | 10         | 1                   | 0.5                 | 0.19            |

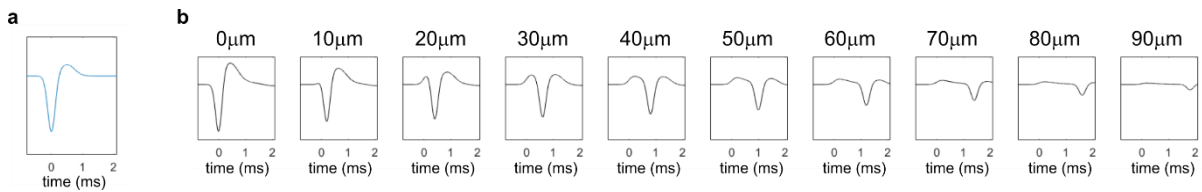
**Table V-2: Noise level used and corresponding SNR for the simulated single electrode dataset**

| $\sigma_{noise}$ | 0.5   | 0.75 | 1    | 1.25 | 1.5  | 1.75 | 2    |
|------------------|-------|------|------|------|------|------|------|
| SNR              | 13.33 | 8.89 | 6.67 | 5.33 | 4.44 | 3.81 | 3.33 |

*b) Polytrode dataset*

For tests with multiple electrodes, we needed to model the fact that an action potential generated by a neural cell is recorded by nearby electrodes with an amplitude and a shape that differ depending on the distance and the geometry of the neuron relatively to the electrode.

We thus implemented a simple model of extracellular action potentials, able to reproduce different variations of their temporal shape depending on the electrode position. An intracellular action potential is a positive peak of the membrane potential. This signal results from voltage-sensitive sodium and potassium channel ion fluxes flowing through the neuron's membrane at the initial segment, characterized by a positive transmembrane current that first flows inward and then outward. These active currents are compensated by capacitive and leak currents in the rest of the arborization. Detailed models of action potentials show that this transmembrane current peak, which is strongest in the soma, is also found in remote neuron parts, with a decreased amplitude, a larger width and a delayed peak (Gold et al. 2006). Considering that the extracellular space is homogeneous and conductive, the potential at one point of the extracellular space is proportional to the sum of the local transmembrane currents divided by the distance. The difference in shape of the same action potential recorded at different points of the extracellular space can thus be explained by the shape differences and the delays between the transmembrane current waveform in the different parts of the neuron. In our model the transmembrane currents follow a template waveform (Figure V-3.a) that propagates along one segment line, representing the neuron arborization. The waveform's amplitude decreases linearly along this line until being null. We also ensure that at any time the sum of the currents is null (Kirschhoff's law), by subtracting the total current sum to the local currents. The final current waveforms obtained along the propagation segment are shown in Figure V-3.b. The potential recorded by an electrode is modeled as the potential at one point of the extracellular space, which is the sum of the local currents for all simulated neurons divided by the distances, to which we add some noise.

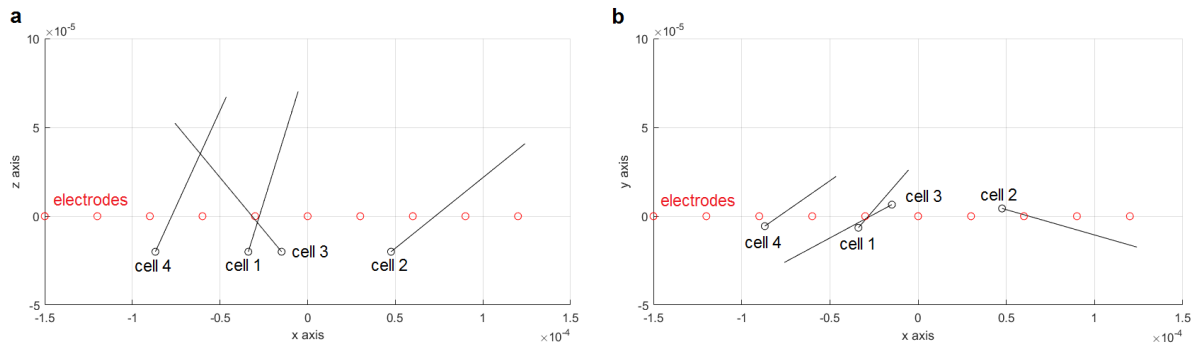


**Figure V-3: Model of the transmembrane current through the neuron. (a) Template waveform used to model the transmembrane current. (b) Transmembrane currents obtained at different points of the line representing the neuron, after applying a propagation, an amplitude decrease, and adjusting the currents to have a null sum.**

The noise we used is a Gaussian noise with spatial and temporal correlations. The spatial correlation between two points decreases exponentially with the distance. The exponential decrease is characterized by a constant  $d_{noise}$ . Thus, we computed the correlation matrix between all electrodes and generated a spatially correlated Gaussian noise at each time step. The temporal correlation was

then obtained using an Ornstein-Uhlenbeck process with a time constant  $\tau_{noise}$ , as for the single electrode simulations.

For our simulated polytrode dataset, we generated recordings lasting 100 s, sampled at 20 kHz, with a noise level  $\sigma_{noise} = 1$ . For this dataset, we chose a short spatial correlation for the noise ( $d_{noise} = 1 \mu\text{m}$ ), so that the correlation between two electrodes was negligible. The temporal noise correlation was the same as for the single electrode simulations ( $\tau_{noise} = 0.1 \text{ ms}$ ). For each recording we simulated 10 electrodes, arranged along a line, and spaced by  $30 \mu\text{m}$ . In the different recordings, we simulated 2, 3 or 4 neural cells, randomly positioned along a line distant of  $20 \mu\text{m}$  from the electrodes' line (Figure V-4). For each neural cell, the segment along which the intracellular action potential propagates is  $100\text{-}\mu\text{m}$  long and its direction is chosen randomly in the half-space directed towards the electrodes (Figure V-4).



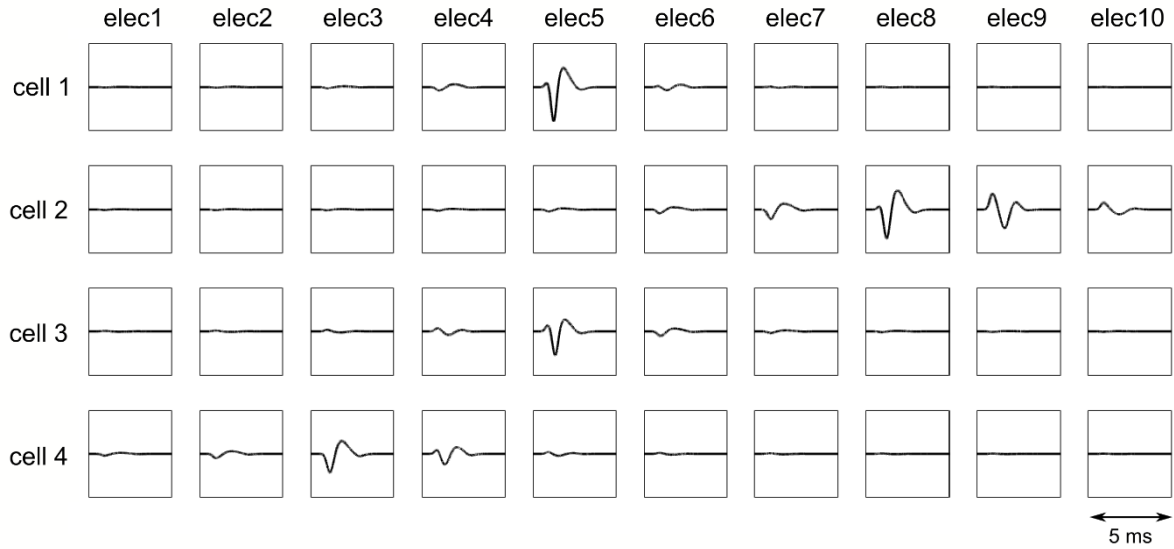
**Figure V-4: Example of cells position relatively to the electrode line. The cells are placed along a line  $20 \mu\text{m}$  away from the electrode line. (a) x-z view. (b) x-y view. Units are in meters. Red circles represent the electrodes, black circles represent the cells soma and black lines represent the propagation line.**

The transmembrane current waveform, shown in Figure V-3, is the same for all simulated cells and propagates at the speed of  $50\text{mm}\cdot\text{s}^{-1}$ , which is quite slow but allows to see significant variations in shape for the action potential waveforms. Given the characteristics of the simulated neural cells and the simulated electrodes, we can compute the action potential waveform generated on each electrode by each cell. Then for each neural cell, the waveforms' amplitude is adjusted so that the maximum amplitude for each cell matches the chosen SNR. We generated recordings with an SNR of 3, 4, 5 or 6. The maximum amplitude chosen for each neural cell to obtain the different SNR are detailed in Table V-3.

**Table V-3: Amplitude of the each neural cell's action potential for the different SNR**

| SNR   | 3   |     |     | 4   |     |     | 5 |   |     | 6   |     |     |
|---|-----|-----|-----|-----|-----|-----|---|---|-----|-----|-----|-----|
| Number of cells                                 | 2   | 3   | 4   | 2   | 3   | 4   | 2 | 3 | 4   | 2   | 3   | 4   |
| Action potential amplitude for each neural cell | 2.4 | 2.4 | 2.1 | 3.2 | 3.2 | 2.8 | 4 | 4 | 3.5 | 4.8 | 4.8 | 4.2 |
|   | 3.6 | 3   | 2.7 | 4.8 | 4   | 3.6 | 6 | 5 | 4.5 | 7.2 | 6   | 5.4 |
|   |     | 3.6 | 3.3 |     | 4.8 | 4.4 |   | 6 | 5.5 |     | 7.2 | 6.6 |
|   |     |     | 3.9 |     |     | 5.2 |   |   | 6.5 |     |     | 7.8 |

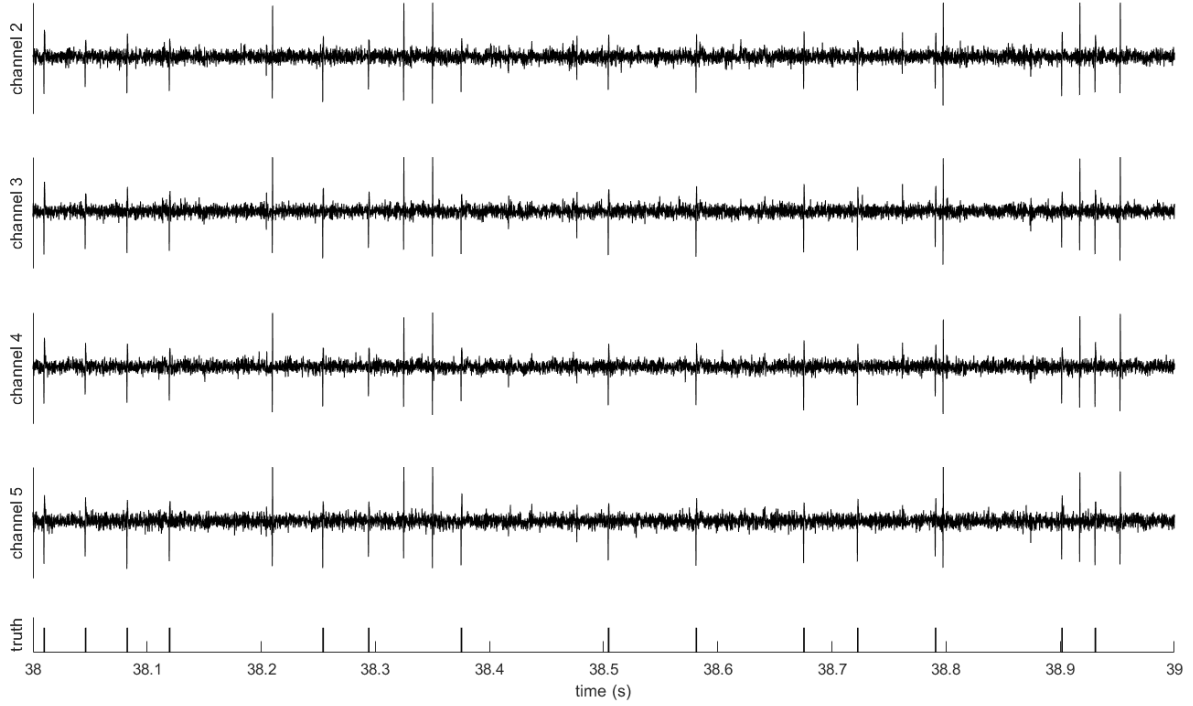
Figure V-5 shows an example of action potentials' waveforms generated with this method. Once the waveforms are computed, they are included in the recording following a Poisson process, with a firing rate of 10 Hz for each simulated cell. For each different number of cells and each different SNR, we generated 5 different recordings, each time with randomly placed neural cells, leading to a total of 60 simulated recordings, each of 100 s duration.



**Figure V-5: Example of action potential waveforms obtained for each simulated electrode and each simulated neural cell**

## 2. Real recordings

Our network was also evaluated with real recordings from hippocampus region CA1 of anesthetized rats, available from the Buzsaki Laboratory (Henze et al. 2000; Henze et al. 2009) (datasets d533101 and d11221.002). These two recordings are tetrode recordings, associated with an intracellular recording giving the ground truth for one neural cell (Figure V-6). The d11221.002 recording has a sampling frequency of 20 kHz. The d533101 recording, having an original 10-kHz sampling frequency, was up-sampled, for convenience, at 20 kHz using a Witter-Shannon interpolation. Before being fed to the network, the signals were band-pass-filtered using a first-order Butterworth filter (300Hz – 3000Hz).



**Figure V-6: Sample of the tetrode recording d533101, with the four channels of the tetrode after filtering and the ground truth extracted from the intracellular recording.**

## B. Performance indices

### 1. Spike-sorting performance

When assessing the performance of a spike-sorting algorithm, one can be confronted to three types of errors: 1) false positive errors, when the algorithm detects an action potential where there is none, 2) false negative errors, when an action potential is not detected, and 3) clustering errors, when an action potential is detected but classified in a wrong cluster. False positives and false negatives are both detection errors. We thus defined several performance indices that quantify each type of error as well as the global performance.

The ground truth and the algorithm output are respectively constituted of  $N$  and  $K$  clusters.. We denote  $H_{ij}$  the number of elements of true cluster  $i$  detected and classified in output cluster  $j$ ,  $FN_i$  the number of elements of true cluster  $i$  not detected by the algorithm and  $FP_j$  the number of elements in output cluster  $j$  not corresponding to any true action potential. The number of elements in true cluster  $i$  is denoted  $T_i$ , and the total number of true action potentials is denoted  $T$ . The number of elements in output cluster  $j$  is denoted  $O_j$ , and the total number of detected action potentials (including false positives) is denoted  $O$ .

To evaluate the detection performance of the algorithm, we take into account the total number of false positives  $FP = \sum_{j=1}^K FP_j$ , the total number of false negatives  $FN = \sum_{i=1}^N FN_i$ , and the total number of action potentials correctly detected without taking into account their classification,  $D =$

$\sum_{i=1}^N \sum_{j=1}^K H_{ij}$ . We can then define the precision  $P_D$ , recall  $R_D$  and F-score  $F_D$  as performance indices for detection:

$$\left\{ \begin{array}{l} P_D = \frac{D}{D + FN} = \frac{D}{O} \\ R_D = \frac{D}{D + FP} = \frac{D}{T} \\ F_D = \frac{2D}{2D + FP + FN} = \frac{2D}{T + O} = \frac{2}{\frac{1}{P_D} + \frac{1}{R_D}} \end{array} \right.$$

To evaluate the clustering performance, we first need to find a correspondence between the true clusters and the output clusters, as we do not know a priori which output cluster corresponds to which true cluster. We define a correspondence  $M$  as a set of  $(i, j)$  pairs, where  $i$  is a truth cluster index and  $j$  an output cluster index. To be valid,  $M$  should contain neither the same true cluster twice nor the same output cluster twice. Given a correspondence  $M$ , the number of correctly classified elements is  $H_M = \sum_{(i,j) \in M} H_{ij}$ . The optimal correspondence is the one maximizing  $H_M$ , computed through an exhaustive search, and we denote  $H^* = \max_M (H_M)$  the optimal number of hits.  $H^*$  thus corresponds to the number of correctly detected and correctly classified action potentials. We can then define a clustering performance index:

$$C = \frac{H^*}{D}$$

To evaluate the global performance of the algorithm, we combine  $C$  and  $F_D$  to obtain an F-score that takes into account all types of errors:

$$F = \frac{2H^*}{T + O} = F_D * C$$

Given a true cluster  $i$  and an output cluster  $j$ , we can also defined the precision  $P_{ij}$ , recall  $R_{ij}$  and F-score  $F_{ij}$  for this specific pair, as follows:

$$\left\{ \begin{array}{l} P_{ij} = \frac{H_{ij}}{O_j} \\ R_{ij} = \frac{H_{ij}}{T_i} \\ F_{ij} = \frac{2H_{ij}}{T_i + O_j} \end{array} \right.$$

These paired scores are used to analyze in detail how well each action potential waveform is detected and classified. They are also used to assess the performance with the real dataset, for which the truth is only known for one neural cell. The scores used in this case is the paired scores obtained for this true known cell and the best matching output neuron.

## 2. ROC curve for the attention neuron

The role of the attention neuron is to detect action potentials within the input signal. Depending on how its threshold is chosen, the attention neuron generates more or less false positive and false negative errors. To assess the efficiency of the attention neuron independently from the choice of its threshold, we decided to use a variation of the Receiver Operating Characteristic (ROC) curve. Usually, a ROC curve shows the evolutions of both the false positive rate and the true positive rate (or recall) when the threshold evolves. In our case, we cannot compute a false positive rate since we have no negative labels (all noisy part of the signal can be qualified as negative). Instead we used the recall (true positives over positives) and the precision (true positives over predicted positives). The idea of the ROC curve is the following. When the threshold is very low, the recall is 1 whereas the precision is 0. As the threshold increases, the recall decreases while the precision increases, until they reach respectively 0 and 1. This can be represented in a curve showing the trajectory of the precision and recall values (see Figure V-7). For perfectly separable data, there is a point for which both recall and precision are equal to 1. In this case, the area under the ROC curve is 1. Otherwise, the recall begins to decrease before the precision reaches 1, leading to an area under the curve lower than 1. Applied to the attention neuron, this area under the ROC curve thus constitutes an index of the performance on a particular recording, independently of the threshold choice. This index was used for example in Section IV.E.2 to compare different structures for the attention neuron.

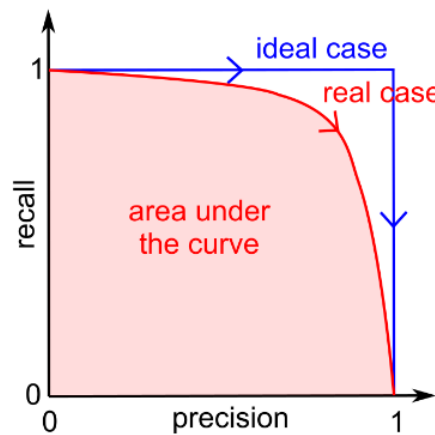


Figure V-7: Principle of the ROC curve

## 3. Intermediate layer quality

The output of the intermediate layer is an important intermediate result. To evaluate the quality of the intermediate layer, we defined several indices to evaluate its output knowing the ground truth.

Each time an action potential occurs in the input signal, the intermediate layer generates a sequence of spikes stemming from different intermediate neurons. Though the exact timing of these spikes could be exploited to classify the spike sequence, not all versions of our output layer use this information. Thus for the sake of simplicity, to describe the spike sequence generated for each action

potential occurrence, we simply count how many times each intermediate neuron fires. For each action potential occurrence we know its label  $L$  from ground truth and the corresponding intermediate layer output is described by a vector  $X$  of  $N$  natural integers where  $N$  is the number of intermediate neurons and  $X_i$  is the number of spikes emitted by the  $i^{th}$  intermediate neuron.

Our first quality index  $I_H$  is based on entropy. We want to compute the entropy of the action potential label given the intermediate layer output, which tells us how much information is missing in the intermediate layer output to find the correct label. To do so, we computed an approximation of the probability to have  $X=x$ , a priori or given a label, using the density of points around  $x$ . We can then compute the conditional entropy with the following formula:

$$H(L|X) = \sum_{x,l} p(x)p(x|l) \log\left(\frac{p(x)p(x|l)}{p(l)}\right)$$

We then normalize our index with respect to the entropy of the label:

$$I_H = \frac{H(L|X)}{H(L)}$$

This entropy index gives us an objective index on the intermediate layer quality in case of an ideal classification, however it does not reflect how the output layer works. Thus we constructed a more concrete index based on the distances between intermediate layer spike sequences. We grouped the intermediate layer outputs, described by the same  $N$ -dimensional natural integer vector, into  $K$  clusters  $C_i$  according to their ground truth label. We first computed the centroid of each cluster  $c_i = \text{mean}(\{X/X \in C_i\})$ . To estimate the size  $S_i$  of each cluster  $i$  we took the 0.9 quantile of the distance of each point of the cluster to its centroid:

$$S_i = q_{0.9}(\{|X - c_i| / X \in C_i\})$$

For each cluster  $i$  we also computed an inter-cluster distance  $D_i$  as the minimum distance to other clusters based on the centroids' distance:

$$D_i = \min(\{|c_i - c_j| / j \in \llbracket 1, K \rrbracket, j \neq i\})$$

We then took the ratio of the median cluster size and the median inter-cluster distance as our distance index  $I_D$ .

$$I_D = \frac{\text{median}(\{S_i\}_{i \in \llbracket 1, K \rrbracket})}{\text{median}(\{D_i\}_{i \in \llbracket 1, K \rrbracket})}$$

## C. Comparison with other spike-sorting methods

### 1. Tests with Osort and Wave\_clus

To assess our method, we chose to test two other spike-sorting methods: Osort (Rutishauser et al. 2006) and Wave\_clus (Quiroga et al. 2004). Both of these methods give the possibility to sort the data



automatically, without supervision. These two methods were tested on our single electrode simulated dataset (see Section V.A.1) and the real tetrode data (see Section V.A.2).

Concerning Osort, the detection was done using the positive amplitude thresholding for the simulated data and the negative amplitude thresholding for the real data. The extraction threshold was set to 4. The alignment was done, using the findPeak method, on the maximum for the simulated data and on the minimum for the real data. The sampling frequency was set to 20 000, to match the input data sampling frequency. Other parameters were set to their default values.

Concerning Wave\_clus, all the parameters were set to their default value, except for the threshold's type, which is set to positive for simulated data and negative for real data, and for the number of samples stored before and after the event that are both set to 30. Wave\_clus gives the option to adjust the temperature parameter after the first clustering pass. We did not use this option and used the automatically chosen value, as our goal is to compare fully automatic methods. Wave\_clus proposes an option to assign unsorted action potentials to found clusters. For each recording we used this option once, after the first clustering pass.

## 2. Statistical tests

The single electrode simulated data and the real tetrode dataset were both used to compare different methods, in particular to compare the STDP network to Osort and Wave\_clus. For the simulated data, several recordings with similar characteristics were generated (see Section V.A.1). Each of the compared methods are run once on each simulated recordings. The different methods' results on a set of recordings with similar characteristics are then compared. For the real tetrode data (see Section V.A.2), the STDP method was run 8 times on each channel, Wave\_clus was run 8 times on each channel, and Osort was run once on each channel as its result was deterministic. The different methods' results on the same recording are then compared. As the variances were significantly different for the different groups (as assessed by a Bartlett test), a Welch test was used for 2-by-2 comparisons, except for the comparison with Osort on the tetrode data where a one-sample t-test was used. A Bonferroni correction was used for each set of multiple comparisons.

## References

- Adamos, D.A., Kosmidis, E.K. & Theophilidis, G., 2008. Performance evaluation of PCA-based spike sorting algorithms. *computer methods and programs in biomedicine*, 91, pp.232–244.
- Gold, C. et al., 2006. On the Origin of the Extracellular Action Potential Waveform: A Modeling Study. *Journal of Neurophysiology*, 95(5), pp.3113–3128. Available at: <http://jn.physiology.org/cgi/doi/10.1152/jn.00979.2005>.
- Henze, D. et al., 2009. Simultaneous intracellular and extracellular recordings from hippocampus region CA1 of anesthetized rats. Available at: CRCNS.org.
- Henze, D.A. et al., 2000. Intracellular features predicted by extracellular recordings in the hippocampus in vivo. *Journal of neurophysiology*, 84(1), pp.390–400.
- Quiroga, R.Q., Nadasdy, Z. & Ben-Shaul, Y., 2004. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8), pp.1661–1687.
- Rutishauser, U., Schuman, E.M. & Mamelak, A.N., 2006. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of Neuroscience Methods*, 154(1–2), pp.204–224.



## VI. IMPLEMENTATIONS AND RESULTS

### A. MiniNet

#### 1. Implementation

The first version of our network, consisted in one input layer, one intermediate layer implementing both an STDP rule and an STP rule, and one output layer implementing an STDP rule (Figure VI-1). In this version of the network, the encoding frequency was 20 kHz, corresponding to our recordings' sampling frequency, without up-sampling. The number of encoding delays used in the input layer was large, corresponding to a signal window of 1.2 ms. As improvements we also implemented a lateral STDP and an intrinsic plasticity on the output layer. The main characteristics of this network are summarized in Table VI-1 and Figure VI-1. The detailed parameters are shown in Table VI-2. This method was subject to a patent (n° EN1654485).

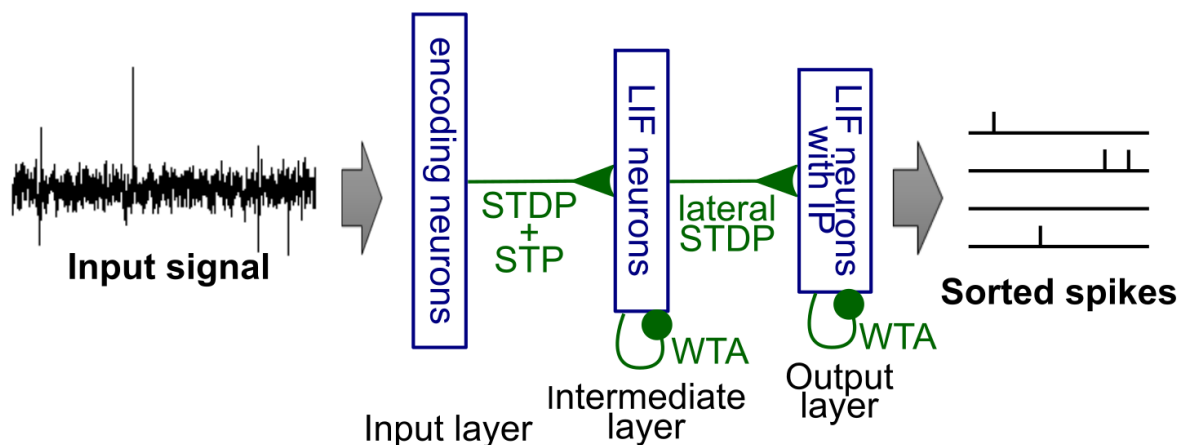


Figure VI-1: MiniNet structure

Table VI-1: Main features of MiniNet

|                            |  |
|----------------------------|--|
| <b>Input layer</b>         | Encoding frequency of 20Hz.<br>At each encoding step, 24 signal values are encoded, each separated by 0.05 ms (1.2ms time window).<br>The sensitivity margin $DV_m$ is $1.5\sigma_{noise}$ , overlap is 8. |
| <b>Attention mechanism</b> | STP plasticity rule implemented on the synapses connecting the input layer to the intermediate layer.  |
| <b>Intermediate layer</b>  | 30 LIF neurons. Synapses stemming from the input layer implement an STDP rule and an STP rule.   |
| <b>Output layer</b>        | 10 LIF neurons. Synapses stemming from the intermediate layer implement an STDP rule. Performance can be improved with a lateral STDP rule and an IP rule.   |

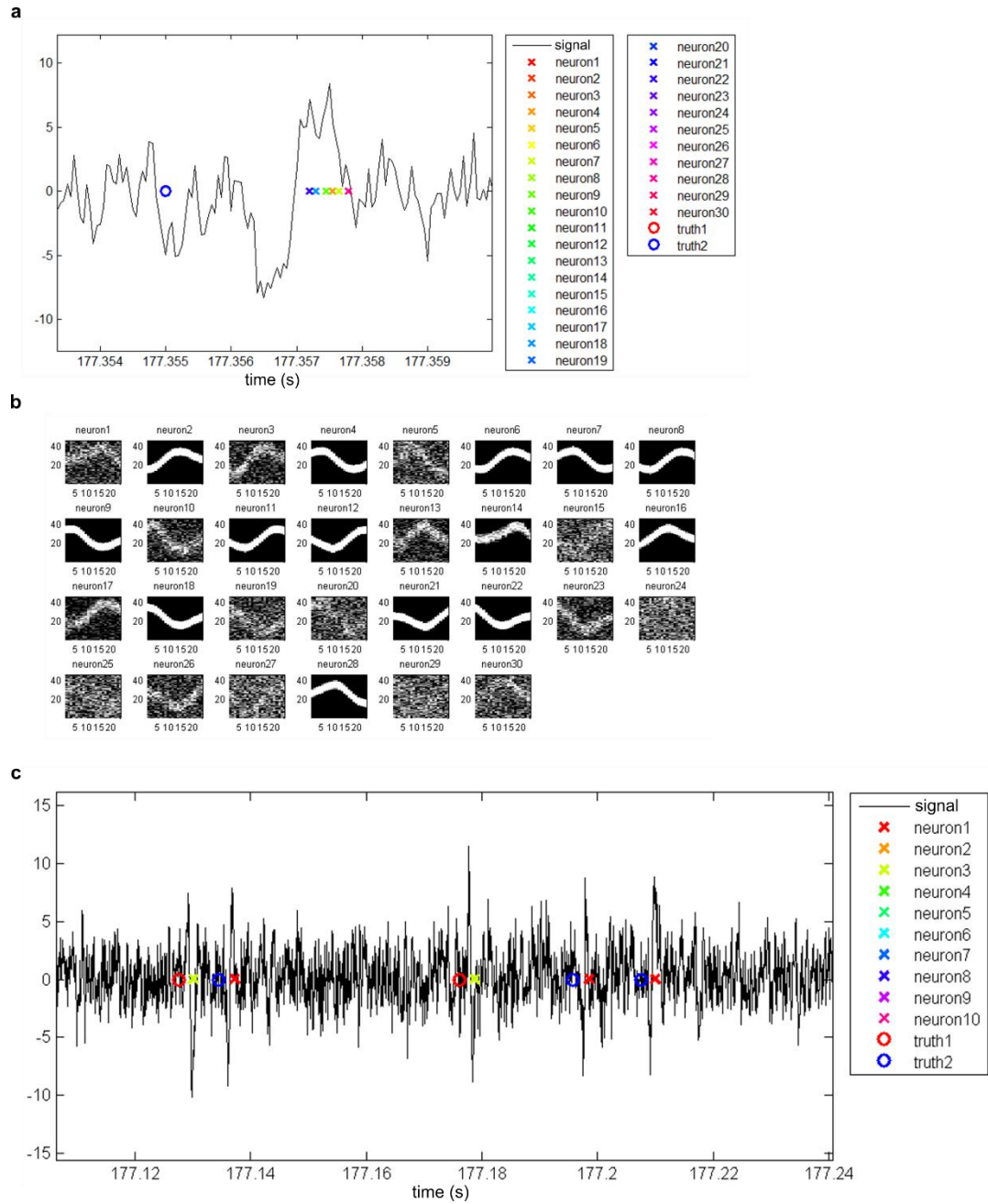
**Table VI-2: Detailed parameters of MiniNet**

| <b>Input layer parameters</b>        |  |   |
|--------------------------------------|--|---|
| Parameter                            | Description  | Value   |
| $\Delta V_m$                         | Sensitivity margin (half-size of the sensitivity range)                    | $1.5\sigma_{\text{noise}}$                      |
| $N_{\text{overlap}}$                 | Number of neuron active at the same time within one column                 | 8   |
| $\Delta t_s$                         | Input layer sampling period  | 0.05 ms   |
| $\Delta t_c$                         | Time interval between two encoding delays                                  | 0.05 ms   |
| $N_c$                                | Number of encoding delays  | 24  |
| <b>Intermediate layer parameters</b> |  |   |
| Parameter                            | Description  | Value   |
| $N_{\text{neur}}$                    | Number of neurons  | 30  |
| $\tau_m$                             | Membrane time constant   | $3 \cdot \Delta t_s = 0.15$ ms                  |
| $Th$                                 | Neurons threshold  | $N_{\text{overlap}} \cdot N_c \cdot 0.75 = 144$ |
| $\tau_{\text{refrac}}$               | Refractory period  | 0.025 ms  |
| $V_{\text{reset}}$                   | Reset potential  | $-20 \cdot Th = -2880$                          |
| $w_{\text{inhib}}$                   | Lateral inhibition weight for WTA  | $-0.2 \cdot Th = -28.8$                         |
| $w_0$                                | Average weight of the feedforward synapses at initialization.              | 0.5   |
| $\tau_{\text{stdp}+}$                | Positive STDP rule time window   | $\Delta t_c \cdot 1.1 = 0.055$ ms               |
| $\tau_{\text{stdp}-}$                | Negative STDP rule time window   | 0   |
| $\Delta w_{\text{pair}}$             | Weight change for a presynaptic spike coinciding with a postsynaptic spike | 0.01  |
| $\Delta w_{\text{post}}$             | Weight change for each postsynaptic spike                                  | $-0.65 \cdot \Delta w_{\text{pair}}$            |
| $\tau_{\text{stp}}$                  | Short term plasticity time constant  | 1.5 ms  |
| $f_d$                                | Short term plasticity depression factor                                    | 0.2   |
| <b>Output layer parameters</b>       |  |   |
| Parameter                            | Description  | Value   |

|                   |  |                          |
|-------------------|--|--------------------------|
| $N_{neur}$        | Number of neurons  | 10                       |
| $\tau_m$          | Membrane time constant   | 2.5 ms                   |
| $Th$              | Neurons' threshold   | 3                        |
| $\tau_{refrac}$   | Refractory period  | 2.5 ms                   |
| $V_{reset}$       | Reset potential  | $-10*Th = -30$           |
| $w_{inhib}$       | Lateral inhibition weight for WTA  | $-10*Th = -30$           |
| $w_0$             | Average weight of the feedforward synapses at initialization.              | 0.5                      |
| $\tau_{stdp+}$    | Positive STDP rule time window   | 1.5 ms                   |
| $\tau_{stdp-}$    | Negative STDP rule time window   | 1.5 ms                   |
| $\Delta w_{pair}$ | Weight change for a presynaptic spike coinciding with a postsynaptic spike | 0.01                     |
| $\Delta w_{post}$ | Weight change for each postsynaptic spike                                  | $-0.5*\Delta w_{pair}$   |
| $\Delta w_{lat}$  | Weight change for a presynaptic spike coinciding with an inhibition        | $-0.02* \Delta w_{pair}$ |
| $\Delta w_{pre}$  | Weight change for each presynaptic spike                                   | $-0.9* \Delta w_{lat}$   |

## 2. Results

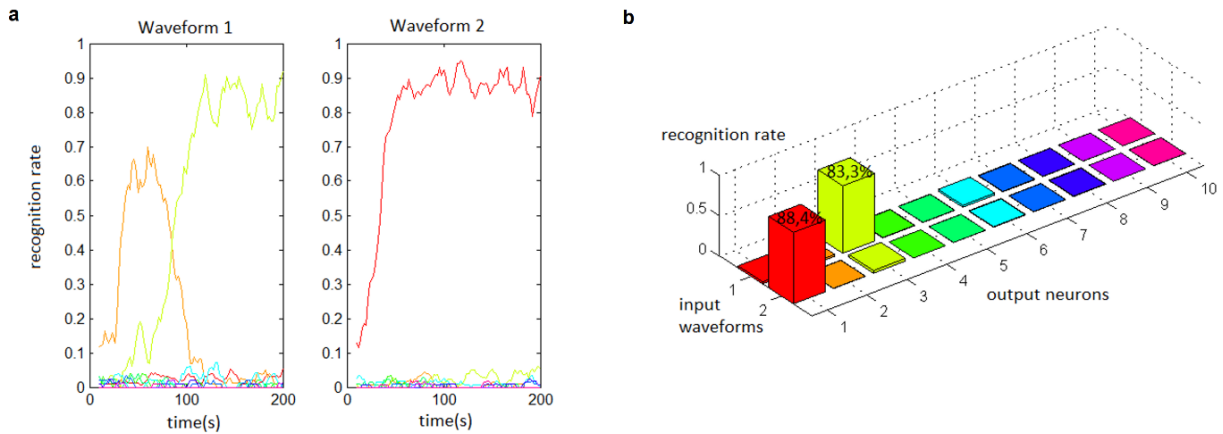
The network was first tested on our preliminary simulated dataset (see Section V.A.1), which consists in signals containing two action potentials. These action potentials have an SNR of 3.5 and a quite large duration of about 2 ms. Figure VI-2 shows the behavior of the network for one recording example. Figure VI-2.a shows the behavior of the intermediate layer. Thanks to the STP, intermediate neurons fire only when an action potential is present in the signal. During an action potential, different intermediate neurons fire in sequences, as they each recognize a different part of the action potential. Indeed, as shown in Figure VI-2.b, for each active intermediate neuron, the synapses stemming from the input layer have evolved to match a specific waveform, which the neuron is then able to recognize. Figure VI-2.c shows an example of the network's output. It can be seen that the output neuron 3 has learnt to recognize one action potential waveform, whereas the output neuron 1 has learnt to recognize the other one.



**Figure VI-2: Qualitative results of MiniNet on a recording from the preliminary dataset. (a) Example of intermediate layer spike train during an action potential. (b) Weights learnt by the intermediate layer. Each square represents the weights of the synapses from all input neurons, organized into a grid, to one specific intermediate neuron. The weights go from 0 (black) to 1 (white). (c) Example of output layer spike train. Output spikes match the different action potential occurrences.**

This first network, with neither lateral STDP nor IP, gave satisfactory results on the preliminary dataset, with a mean F-score of 0.81. Figure VI-3 shows an example of results on one recording from this dataset. It can be seen that the learning converge after about 100 s, and that the waveform 1 is recognized by the output neuron 3 with an F-score of 0.83, and the waveform 2 is recognized by the output neuron 1 with an F-score of 0.88.

Adding a lateral STDP to the output layer improved the performance of the network. Indeed, with lateral STDP, the mean score was brought from 0.81 to 0.88. As preliminary tests, an intrinsic plasticity was implemented on the output layer in combination with the lateral STDP. It was tested qualitatively, on signals containing action potentials with slightly different amplitudes and durations. These tests showed that the neuron's threshold is actually able to adapt to the size of the learnt pattern.



**Figure VI-3: Example of performance of MiniNet on one recording from the preliminary dataset. (a) Evolution of the F-scores during the 200-s simulation. The two graphs are for the two ground truth cells and the different colors correspond to two different output neurons. (b) Mean F-scores on the last 100s of the same recording, for each pair of ground truth cell and output neuron.**

Although this first version worked well on the preliminary dataset, the performances were not as good with more realistic data, especially when action potentials were shorter. One reason was that the input layer was adapted for slow action potentials, because of the slow sampling frequency used for encoding. This was corrected in the next versions. Another problem was the interaction between the STP and the STDP on the intermediate layer, which made it difficult to robustly recognize low-amplitude action potentials. Indeed, without STP, an intermediate neuron's potential depends on how close the input signal is to the shape it has learnt. When the STP is added, its potential also depends on the signal's amplitude. The reduced potential for low amplitudes requires low-amplitude waveforms to be very close to the learnt waveform to be recognized. This led us to introduce an attention neuron (which was added in the patent) in the next versions of the network.

## B. ANNet

### 1. Implementation

We first improved the MiniNet network with the introduction of an attention neuron, which gave its name to this new version. The attention neuron allows to isolate the network's detection function, so that the STP does not impair the intermediate layer's recognition function. The attention neuron then projects to the intermediate layer through fixed-weight synapses (see Figure VI-4), bringing an additional excitation to the intermediate layer, necessary for it to fire. Another important improvement was to set a shorter encoding step on the input layer, while keeping the same interval between two encoded signal values. This brings robustness against sampling time jitter without adding



unnecessary synapses. The number of delays in the input layer was also reduced to 10, corresponding to a time window of 0.5 ms. This is more suitable for realistic action potentials, as the encoding time window should be slightly shorter than an action potential's duration. The structure of the output layer was also more complex. It is constituted of LIF neurons with IP, and it receives spikes from the intermediate layer through synapses with different synaptic delays, presynaptically inhibited by the attention neuron, as described in Section IV.D.3. The synaptic weights can take positive as well as negative values. This structure forces the output layer to wait for the end of the spike sequence emitted by the intermediate layer before firing. This prevents the output layer from missing any important information for pattern separation. The delays also bring some information about the sequence timing. The characteristics of ANNet are summarized in Table VI-3 and Figure VI-4. The detailed implementation of ANNet and the results obtained can be found in an article (annex B), currently in review for publication.

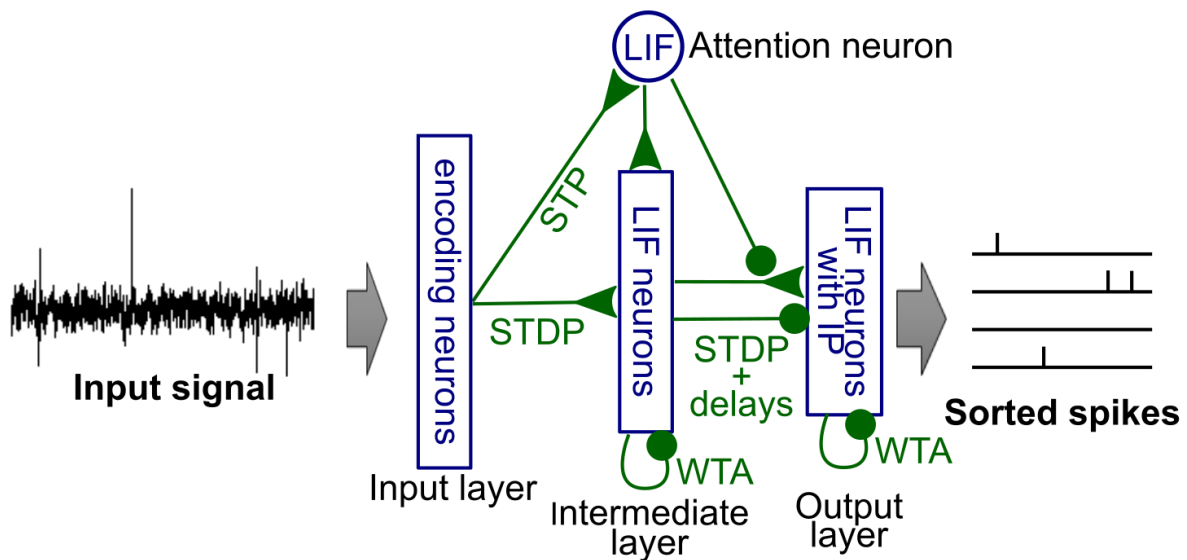


Figure VI-4: ANNet structure

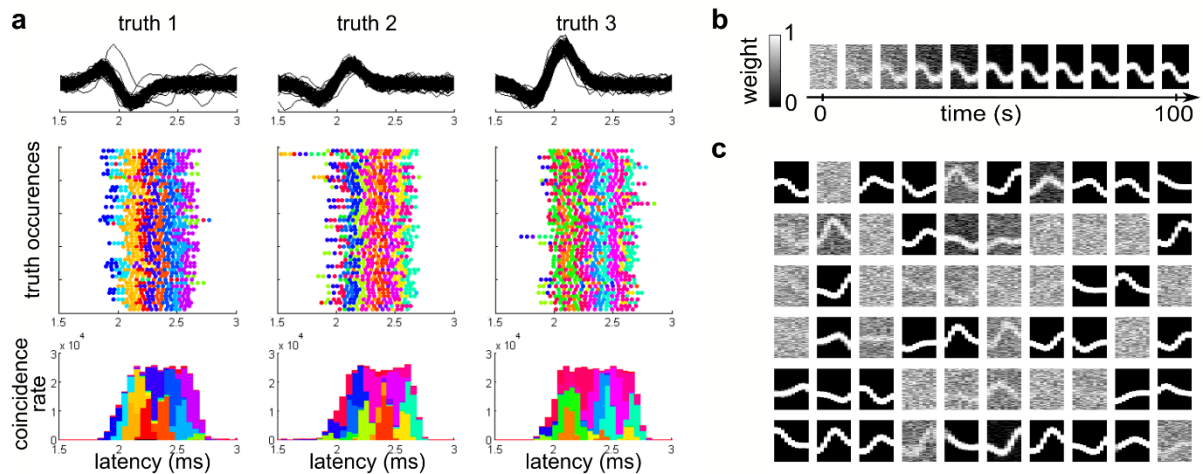
Table VI-3: Main features of ANNet

|                            |  |
|----------------------------|--|
| <b>Input layer</b>         | <p>Encoding frequency of 80Hz.</p> <p>At each encoding step, 10 signal values, separated by 0.05ms each are encoded (0.5ms time window).</p> <p>The sensitivity margin <math>DV_m</math> is <math>1.75\sigma_{noise}</math>, overlap is 10.</p>  |
| <b>Attention mechanism</b> | <p>Attention neuron. Input layer projects to the attention neuron through synapses implementing an STP rule.</p>   |
| <b>Intermediate layer</b>  | <p>60 LIF neurons. Synapses stemming from the input layer implement and STDP rule.</p> <p>The intermediate layer receives the attention neuron spike-train through fixed weight synapses. The threshold is set so that this layer cannot fire if the attention neuron does not fire.</p> |

|                     |  |
|---------------------|--|
| <b>Output layer</b> | <p>10 LIF neurons, implementing an IP rule.</p> <p>The output layer is connected to the intermediate layer through both excitatory and inhibitory synapses, duplicated with different transmission delays. All these synapses implement an STDP rule. The spike transmission is inhibited by the attention neuron, which thus also prevent STDP.</p> |
|---------------------|--|

## 2. Results

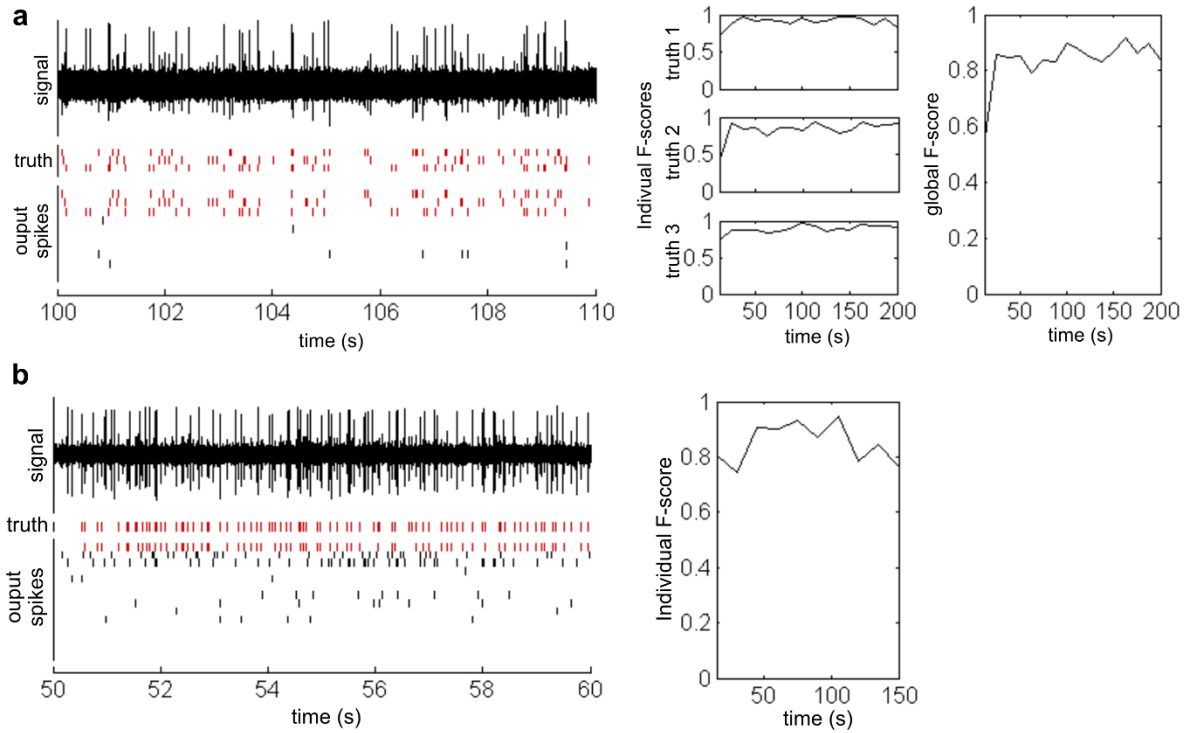
ANNet was tested both on our single-electrode simulated dataset (see Section V.A.1) and on real recordings (see Section V.A.2). The behavior of the intermediate layer is similar to the one obtained with the first implementation. Thanks to the attention neuron, intermediate neurons fire only when an action potential is present in the signal. The spike sequence generated by the intermediate layer matches for similar action potentials and does not for different action potentials (Figure VI-5.a). Indeed, each active intermediate neuron learns to recognize a specific shape in the signal (Figure VI-5.b and c). The intermediate layer spike sequence can thus be seen as a signature of the action potential occurring in the input signal.



**Figure VI-5: Behavior of the intermediate layer of ANNet. (a)** Spike sequences emitted by the intermediate layer for each action potential occurrence. Top: signal shapes for all action potential occurrences on the last 100 s of simulation. Middle row: spike trains of the intermediate layer synchronized with 50 different action potential occurrences. Bottom row: distribution of intermediate spike latency relative to each action potential occurrence (the histograms are cumulated). The different colors stand for different intermediate neurons. **(b)** Weight evolution of the synapses projecting on one specific intermediate neuron. Each square represents the weights of the synapses from all input neurons at a specific time. The weights go from 0 (black) to 1 (white). **(c)** Weights learnt by each intermediate neuron after a 200-s simulation. Each square represents the weights of the synapses from all input neurons, organized into a grid, to one specific intermediate neuron. The weights go from 0 (black) to 1 (white).

Figure VI-6 shows an example of the network's output for a simulated recording and for a real recording, as well as the F-score evolution along the whole recording. It can be seen that, for the simulated recording, three output neurons are active and accurately reproduce the true activity.

Concerning the real recording, for which the ground truth is only known for one neural cell, one of the output neuron reproduces the known ground truth. Though it cannot be objectively evaluated, two other action potentials seem to be detected by two other output neurons.



**Figure VI-6: Examples of ANNet output and performance (a) Output and scored on a simulated recording. Left: comparison of the output spike train, composed of ten neurons, with the truth spike train, composed of three neurons, on a 10-s segment of input signal. Matching spike trains have been highlighted in red. Right: evolution of the performance of the network over the 200-s simulation. (b) Same as (a) for a real recording (dataset d553101). Here the truth is only known for one cell thus only the performance relative to this cell can be computed.**

The performance of ANNet was compared to two other spike-sorting algorithms, Osort (Rutishauser et al. 2006) and Wave\_clus (Quiroga et al. 2004) (see Section V.C). On simulated data with a high SNR, the ANNet's scores were lower than the two other algorithms, though the difference remained relatively small. In contrast, for SNR lower than 6, ANNet performed significantly better than Osort and Wave\_clus (Figure VI-7.a). The performances on real recordings confirmed these results, except that Wave\_clus obtained quite low scores on data with high SNR (Figure VI-7.b). An insight on the detailed scores for the simulated recordings with an SNR of 4.5, where ANNet perform significantly better than the two other methods, shows that the STDP network has a better recall and a better clustering score (Figure VI-7.c).

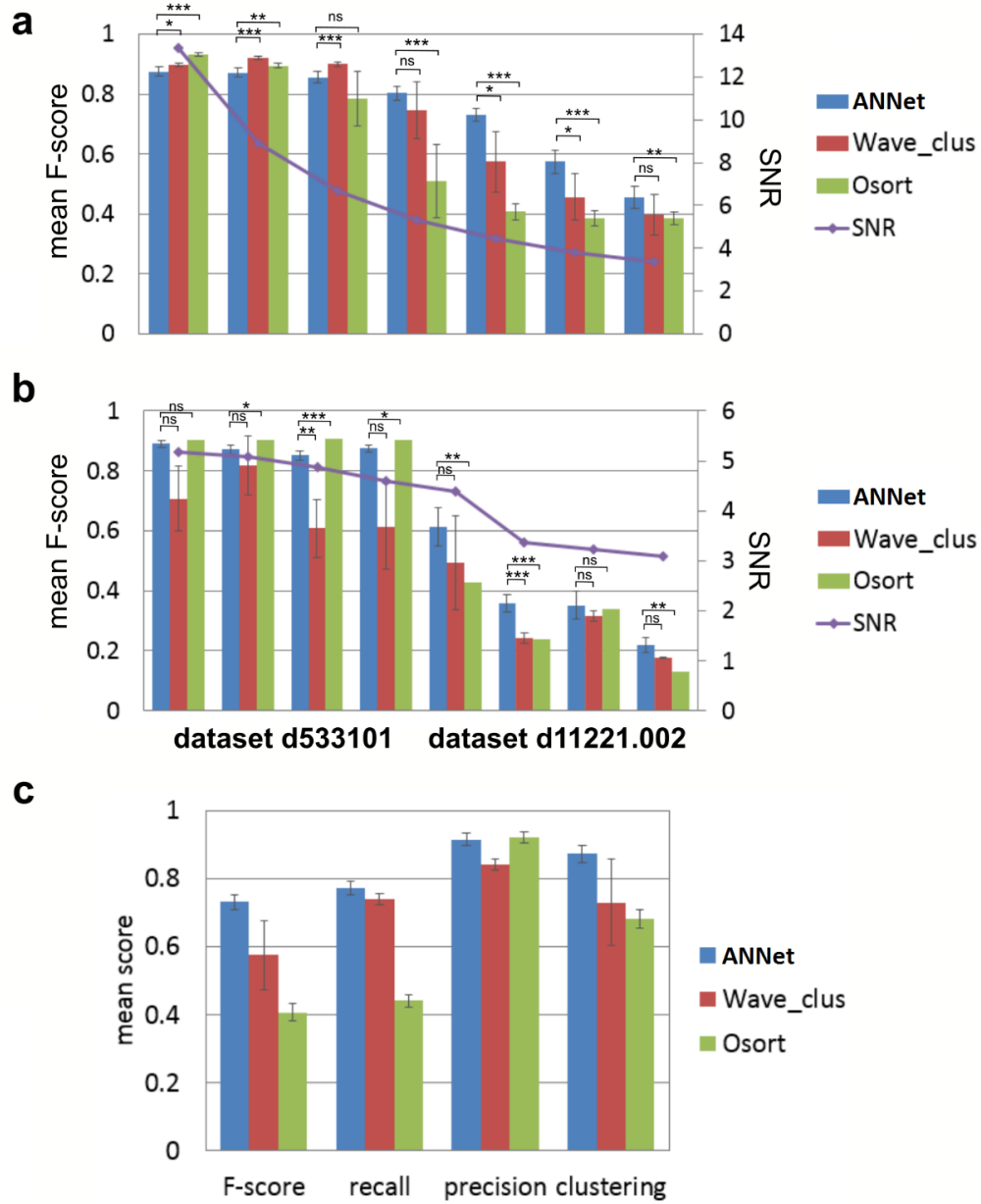


Figure VI-7: ANNet performances compared to Osort and Wave\_clus. (a) Performance on the single electrode simulated dataset. (b) Performance on real recordings. Error bars show the standard deviation. ANNet performance was compared to Wave\_clus and Osort through statistical tests (see Section V.C). ns stands for non-significant, \* for  $p < 0.05$ , \*\* for  $p < 0.01$ , \*\*\* for  $p < 0.001$ . (c) Detailed mean scores on the 10 simulated recordings with an SNR of 4.5.

Finally, the robustness of this network was also tested on two additional simulated recordings. In the first one, the firing rates were different between the three simulated cells (Figure VI-8.a). In the second, each cell became silent one after the other during a 50 s time interval (Figure VI-8.b). The results show qualitatively that the network is robust to such variations.

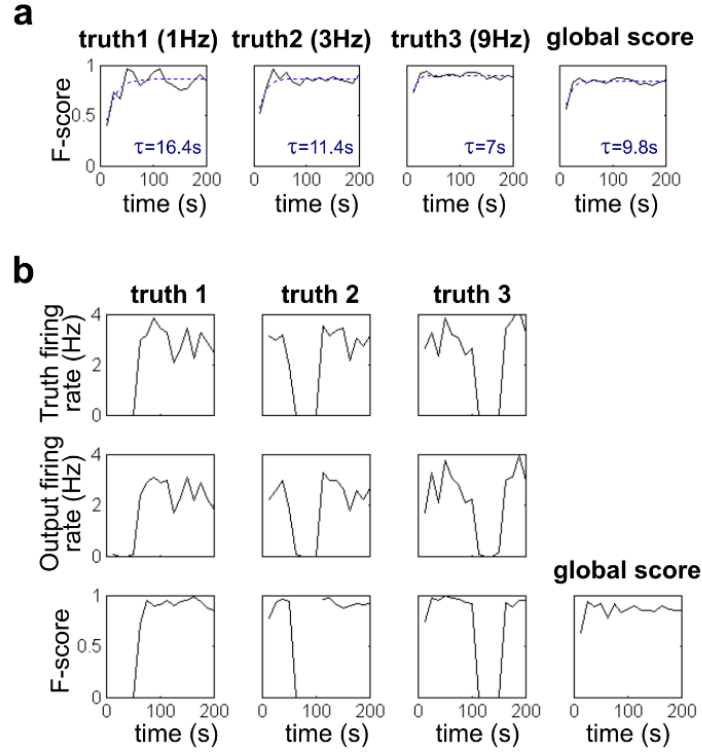


Figure VI-8: ANNet performance on simulated recordings with different firing rate scenarios.

### 3. Preliminary tests on tetrode data

Our ANNet implementation of the network, designed initially for single electrode, was tested on real tetrode recordings. We tested two different methods to adapt the feedforward network structure to process the tetrode data (see Section V.A.2). In a first adapted architecture, the input layer was duplicated four times for each electrode, and these four inputs layers project to a common attention neuron and a common intermediate layer (Figure VI-9.a). In a second architecture, the input layer, the attention neuron and the intermediate layer were all duplicated and connected to a common output layer (Figure VI-9.b). We compared the performances of these two methods to the performance obtained on the best single channel for each tetrode, with the single-electrode version of the network (Figure VI-10). On the d533101 recording, with the highest SNR, the tests show a clear improvement of the results when taking into account all four channels, with even better results when using the second structure. On the d11221.002 recording, there was no significant improvement, and the performance dropped with the second structure, with a high variation between different runs. This is thus difficult to conclude on these two datasets. The next version of our network, adapted to multiple electrode, was thus tested on more data generated through simulation, with more electrode and a controlled SNR and controlled number of cells, to check the effect of different network structures (see Section VI.D).

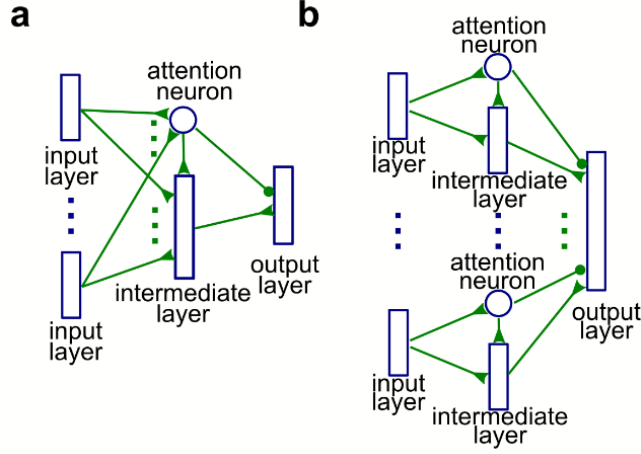


Figure VI-9 : Two different structures adapted for tetrode recordings, for the ANNet implementation.

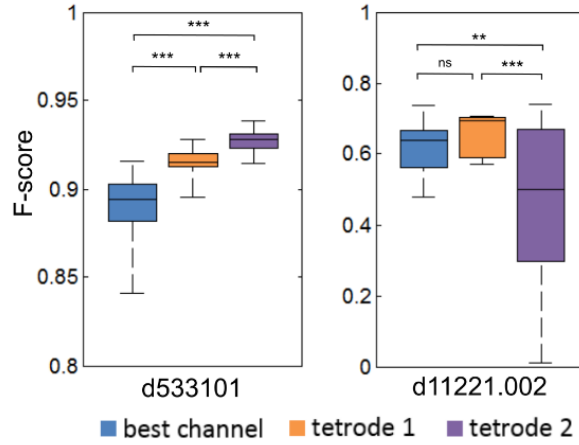


Figure VI-10 : F-scores obtained with the two tested structures, compared to the best single electrode performance.

## C. LTSNet

### 1. Implementation

The most advanced single-electrode version of our network, which we called LTSNet because of the introduction of LTS neurons, includes improvements on both the intermediate and the output layers (Figure VI-11). The intermediate layer implements a 2-WTA mechanism, allowing two neurons to fire almost simultaneously instead of one. This 2-WTA was combined with the implementation of a resource-dependent STDP on the synapses stemming from the input layer, which modulates the weight update depending on the number of spikes recently emitted. As described in Section IV.C.4, this mechanism allows the intermediate layer receptive fields to overlap. The neuron model used in the output layer mimics an LTS neuron, with the simplified model described in Section IV.D.4. This neuron model has a potential rebound after receiving a negative stimulus. The synapses coming from

the intermediate layer are thus inhibitory, and the output neurons fire after the stimulus (see Section IV.D.4), removing the need for an inhibition from the attention neuron used in ANNet. The attention neuron is thus only used to gate the intermediate layer through fixed excitatory synapses. The output layer structure is thus greatly simplified compared to ANNet. No intrinsic plasticity was used in this implementation. A lateral STPD was implemented on the output layer to improve the performance. The main characteristic of LTSNet are shown in Table VI-4 and Figure VI-11, and its detailed parameters in Table VI-5.

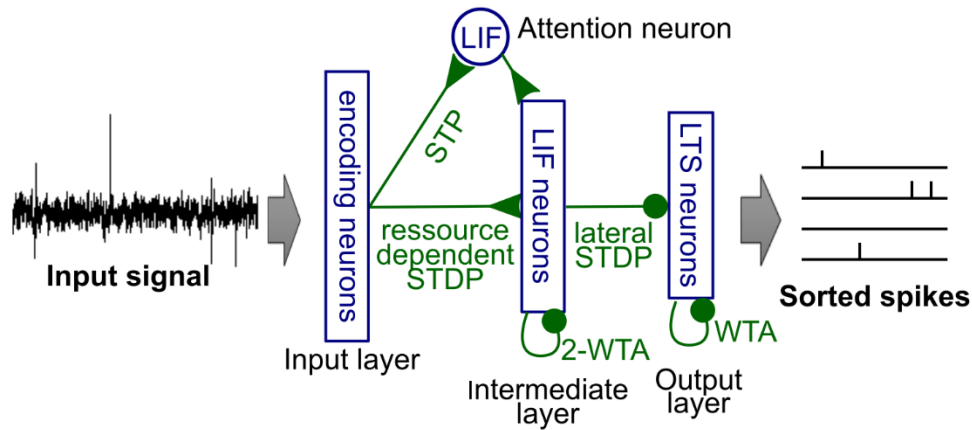


Figure VI-11: LTSNet structure

Table VI-4: Main features of LTSNet

|                            |   |
|----------------------------|---|
| <b>Input layer</b>         | <p>Encoding frequency of 80Hz.</p> <p>At each encoding step, 10 signal values, separated by 0.05ms each are encoded (0.5ms time window).</p> <p>The sensitivity margin <math>DV_m</math> is <math>2\sigma_{noise}</math>, overlap is 10.</p>  |
| <b>Attention mechanism</b> | <p>Attention neuron. Input layer projects to the attention neuron through synapses implementing an STP rule.</p>  |
| <b>Intermediate layer</b>  | <p>100 LIF neurons. Synapses stemming from the input layer implement and STDP rule.</p> <p>A 2-WTA mechanism is used.</p> <p>The intermediate layer receives the attention neuron spike train through fixed weight synapses. The threshold is set so that this layer cannot fire if the attention neuron does not fire.</p> |
| <b>Output layer</b>        | <p>15 simplified LTS neuron.</p> <p>Synapses stemming from the intermediate layer are inhibitory and implement a lateral STDP rule.</p>   |

Table VI-5: Detailed parameters of LTSNet

| Input layer parameters |             |       |
|------------------------|-------------|-------|
| Parameter              | Description | Value |
|                        |             |       |

|                      |  |                          |
|----------------------|--|--------------------------|
| $\Delta V_m$         | Sensitivity margin (half-size of the sensitivity range)    | $2\sigma_{\text{noise}}$ |
| $N_{\text{overlap}}$ | Number of neuron active at the same time within one column | 10                       |
| $\Delta t_s$         | Input layer sampling period                                | 0.0125 ms                |
| $\Delta t_c$         | Time interval between two encoding delays                  | 0.05 ms                  |
| $N_c$                | Number of encoding delays                                  | 10                       |

#### Attention neuron parameters

| Parameter              | Description  | Value                                     |
|------------------------|--|---|
| $\tau_m$               | Membrane time constant                               | $2 * \Delta t_s = 0.025$ ms               |
| $\tau_{\text{refrac}}$ | Refractory period                                    | 0   |
| $Th$                   | Neurons threshold                                    | $0.947 * N_{\text{overlap}} * N_c = 94.7$ |
| $w_{\text{self}}$      | Weight of the self-excitatory synapse                | $0.077 * N_{\text{overlap}} * N_c = 7.7$  |
| $\tau_{\text{stp}}$    | Short term plasticity time constant                  | 20 ms                                     |
| $w_{\text{min}}$       | Equilibrium weight for the input neuron coding for 0 | 0.13                                      |

#### Intermediate layer parameters

| Parameter              | Description   | Value   |
|------------------------|---|---|
| $N_{\text{neur}}$      | Number of neurons   | 100   |
| $\tau_m$               | Membrane time constant  | $2 * \Delta t_s = 0.025$ ms   |
| $\tau_{\text{refrac}}$ | Refractory period   | $\Delta t_c = 0.05$ ms  |
| $V_{\text{reset}}$     | Reset potential   | 0   |
| $V_{\text{inhib}}$     | Post-inhibition potential                                     | 0   |
| $w_{AN}$               | Weight of the synapses coming from the attention neuron       | $0.45 * N_c * N_{\text{overlap}} = 45$  |
| $w_0$                  | Average weight of the feedforward synapses at initialization. | 0.7   |
| $Th$                   | Neurons threshold   | $(w_{AN} + w_0 * N_c * N_{\text{overlap}}) * (1 - \exp(-\Delta t_c / \tau_m)) / (1 - \exp(-\Delta t_s / \tau_m)) = 252.7$ |
| $k_{WTA}$              | Maximum number of neuron that can fire simultaneously         | 2   |



|                   |  |   |
|-------------------|--|---|
| $\tau_{stdp+}$    | Positive STDP rule time window   | $\Delta t_c * 1.01 = 0.0505 \text{ ms}$ |
| $\tau_{stdp-}$    | Negative STDP rule time window   | 0                                       |
| $\Delta w_{pair}$ | Weight change for a presynaptic spike coinciding with a postsynaptic spike | 0.005                                   |
| $\Delta w_{post}$ | Weight change for each postsynaptic spike                                  | $-0.55 * \Delta w_{pair}$               |
| $\tau_{res}$      | STDP resource time constant  | $3.125 \mu\text{s}$                     |
| $f_{res}$         | STDP resource consumption factor   | 0.5                                     |

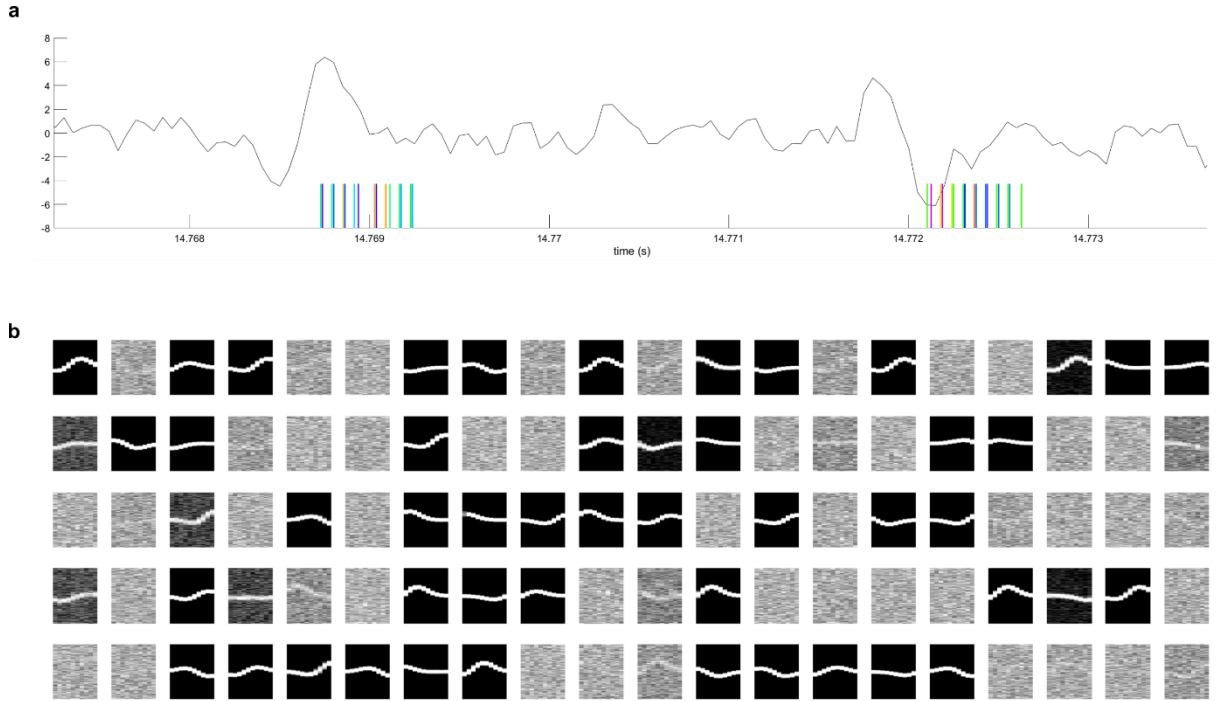
---

| Output layer parameters |  |                          |
|-------------------------|--|--------------------------|
| Parameter               | Description  | Value                    |
| $N_{neur}$              | Number of neurons  | 15                       |
| $\tau_m$                | Membrane time constant   | 2 ms                     |
| $\tau_{refrac}$         | Refractory period  | 0                        |
| $V_{reset}$             | Reset potential  | 0                        |
| $V_{inhib}$             | Post-inhibition potential  | 0                        |
| $q_{reset}$             | Reset value of q after firing  | 0                        |
| $q_{inhib}$             | Reset value of q after an inhibition                                       | 0                        |
| $w_{layer}$             | Weight factor applied to all feedforward synapses.                         | -0.25                    |
| $w_0$                   | Average weight of the feedforward synapses at initialization.              | 0.5                      |
| $\tau_{stdp+}$          | Positive STDP rule time window   | 10 ms                    |
| $\tau_{stdp-}$          | Negative STDP rule time window   | 0                        |
| $\Delta w_{pair}$       | Weight change for a presynaptic spike coinciding with a postsynaptic spike | 0.01                     |
| $\Delta w_{post}$       | Weight change for each postsynaptic spike                                  | $-0.6 * \Delta w_{pair}$ |
| $\Delta w_{lat}$        | Weight change for a presynaptic spike coinciding with an inhibition        | $-0.1 * \Delta w_{pair}$ |
| $\Delta w_{pre}$        | Weight change for each presynaptic spike                                   | $-0.2 * \Delta w_{lat}$  |

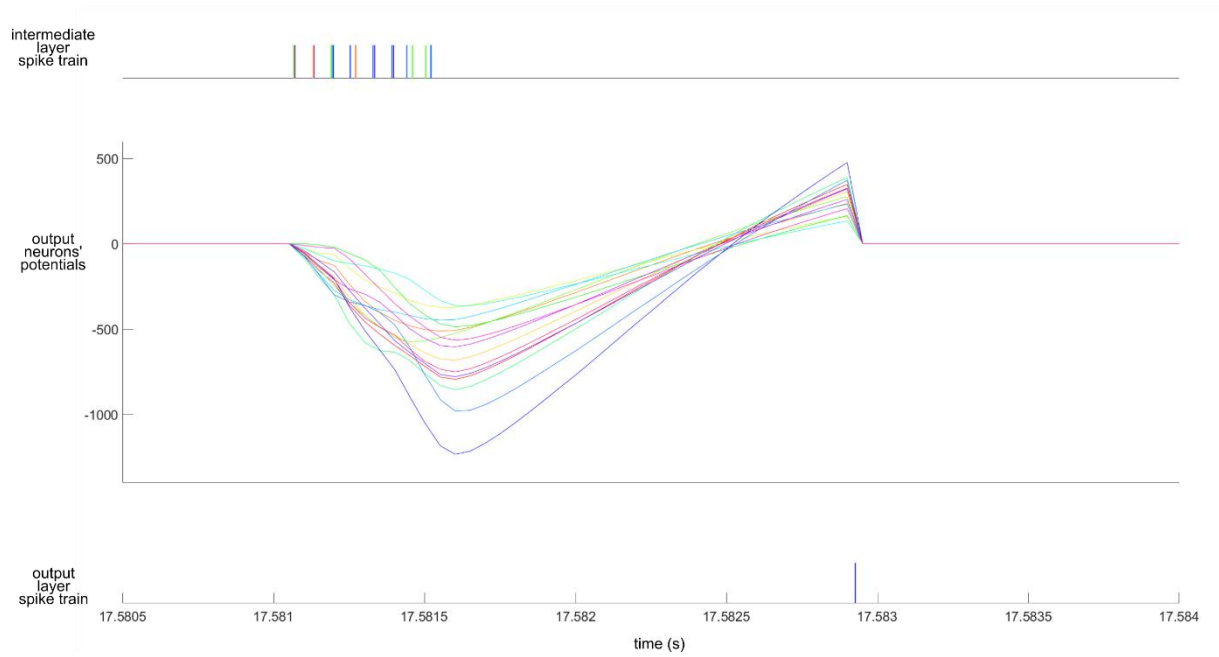
---

## 2. Results

LTSNet was tested on our single electrode simulated dataset (see Section V.A.1). The behavior of the intermediate layer is qualitatively similar to the previous version, except that two intermediate neurons fire each time simultaneously, due to the 2-WTA mechanism (Figure VI-12). Concerning the output layer, thanks to their rebound properties, LTS neurons fire after the inhibitory stimulus received from the intermediate layer. The neuron receiving the strongest stimulus fires first (Figure VI-13).

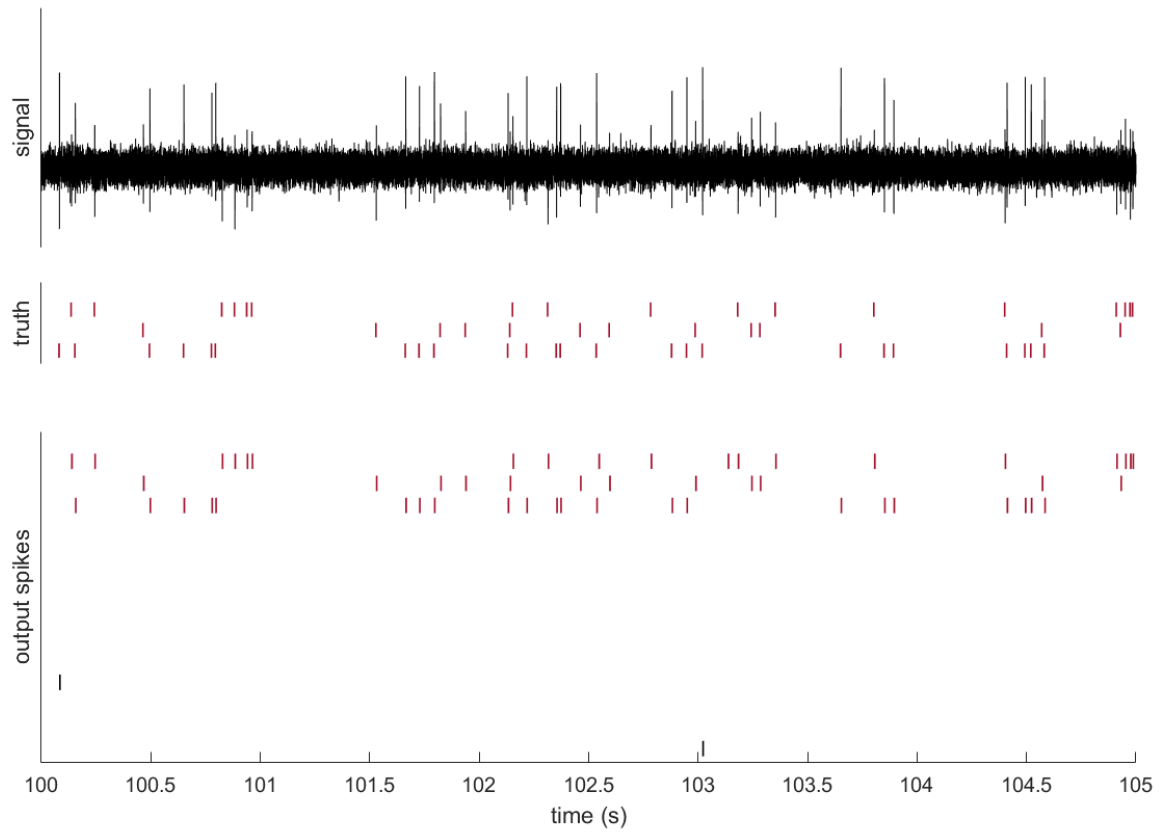


**Figure VI-12: LTSNet intermediate layer behavior. (a) Spikes emitted by the intermediate layer in response to action potential in the signal. The signal is shown in black. Spikes are shown by the colored vertical bars, different colors standing for different intermediate neurons. (b) Weights learnt by each intermediate neuron. Each square represents the weights of the synapses from all input neurons, organized into a grid, to one specific intermediate neuron. The weights go from 0 (black) to 1 (white).**

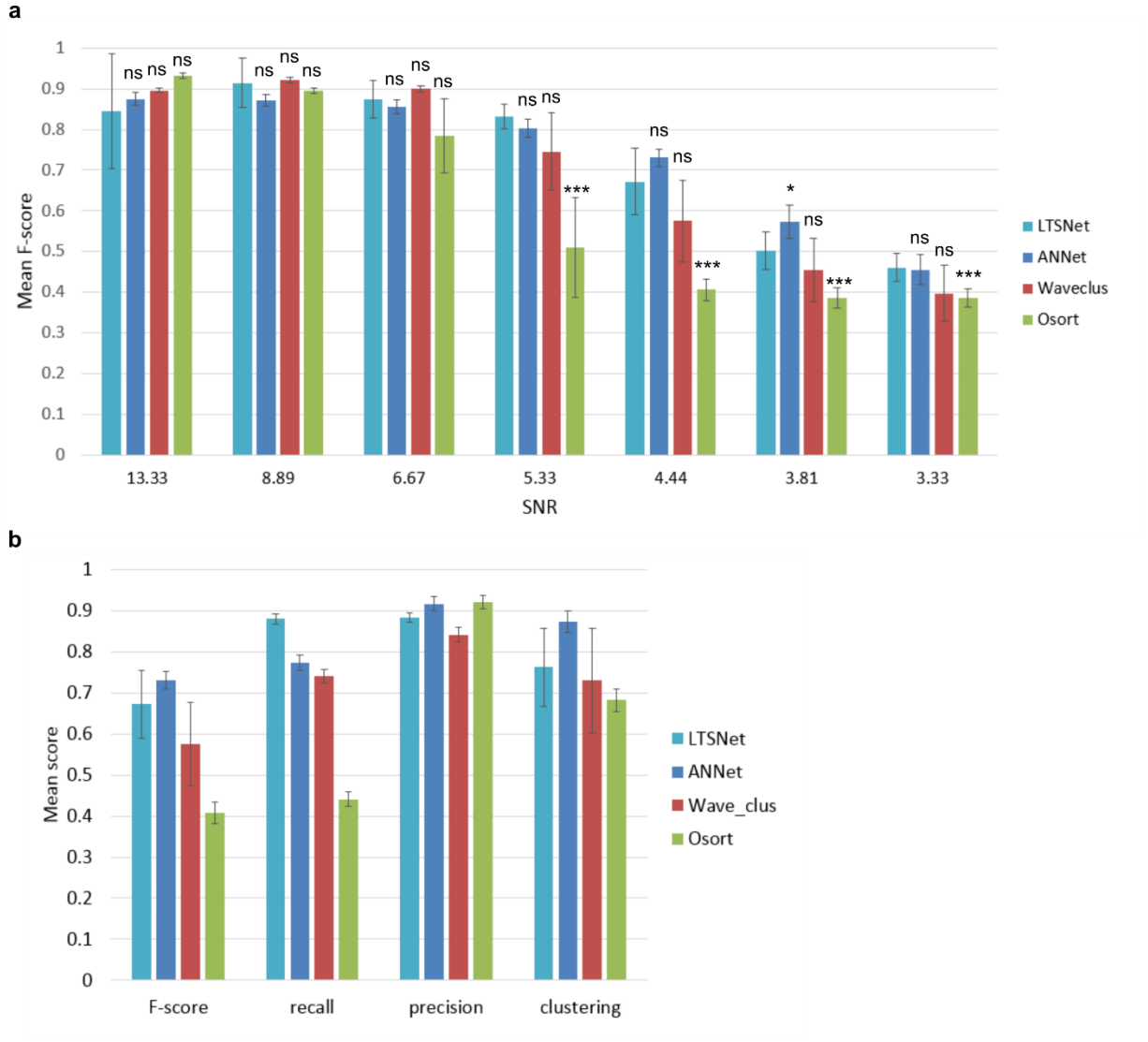


**Figure VI-13: Example of potential rebound of LTS neurons in the output layer. Top: Spike train generated by the intermediate layer. Different colors stands for different intermediate neurons. Middle: Potential evolution of the different output neurons in different colors. Bottom: output neurons spikes. Colors match the middle panel.**

Figure VI-14 shows an example of LTSNet output on a 5-s sample of a simulated recording. Over the 15 output neurons, three neurons are active, and reproduce the true activity. LTSNet performance was assessed on the single electrode simulated dataset and compare to ANNet, as well as Osort (Rutishauser et al. 2006) and Wave\_clus (Rutishauser et al. 2006). LTSNet has similar scores as ANNet and Wave\_clus, and perform significantly better than Osort on recordings with a low SNR (Figure VI-15.a). Looking at the detailed score on recordings with an SNR of 4.5, it can be seen that LTSNet has a better recall than ANNet, but the clustering performance is degraded (Figure VI-15.b).



**Figure VI-14:** Example of LTSNet output on a 5-s segment of simulated signal. Top: input signal Middle: spike train corresponding to true action potentials in the signal. Bottom: output spike train, composed of 15 neurons. Spike trains matching the truth have been highlighted in red.



**Figure VI-15: LTSNet results on the single electrode simulated dataset, compared to ANNet, Osort and Wave\_clus. (a) Mean F-scores on recordings with different SNR. Error bars show the standard deviation. A Welch test with a Bonferroni correction was used to compare LTSNet to each of the three other methods shown. ns stands for non-significant, \* for  $p < 0.05$ , \*\* for  $p < 0.01$ , \*\*\* for  $p < 0.001$ . (b) Detailed mean scores on the 10 simulated recordings with an SNR of 4.5**

## D. PolyNet

### 1. Implementation

We adapted the last version of our network, LTSNet, to process signal stemming from multiple electrodes. We named PolyNet this adapted version of the network illustrated in Figure VI-16. Except for structural adaptations, the parameters are the same as for LTSNet. The structure adaptation of the network depends on the geometry of the electrode array. Here we tested the network on our polytrode simulated dataset (see Section V.A.1). The recordings simulated the signals generated by

ten electrodes arranged in a line. With the chosen electrode geometry, an action potential can typically be seen on about three neighboring electrodes. To adapt the network to polytrode recordings, the initial network structure is duplicated ten times, to match the number of electrodes. We obtain a structure with ten parallel subnetworks (see Figure VI-16). This parallel structure is unchanged from the input layer to the intermediate layer. Lateral synapses are introduced for the connection between the intermediate layer and the output layer. Each intermediate sublayer projects to three output sublayers and each output sublayer receives synapses from three intermediate sublayers, with each time one straightforward connection corresponding to the parallel structure and two lateral connections (except at the edges of the array). Each all-to-all connection from one intermediate sublayer to one output sublayer is pondered by a layer weight  $w_L$ . This means that for each synapse, the learnt weight  $w$ , varying between 0 and 1, is multiplied by  $w_L$ , leading to a resulting weight  $w * w_L$ . The layer weight  $w_L$  is  $1/2$  for the straightforward connection and  $1/4$  for the each of the two lateral connections (see synapse symbols of different size in the Figure VI-14 diagram). The WTA mechanism is also extended between the different output layers, in order to avoid several sublayers to fire for the same action potential. To do so, each output sublayer is inhibited by itself and by the 4 closest output sublayers.

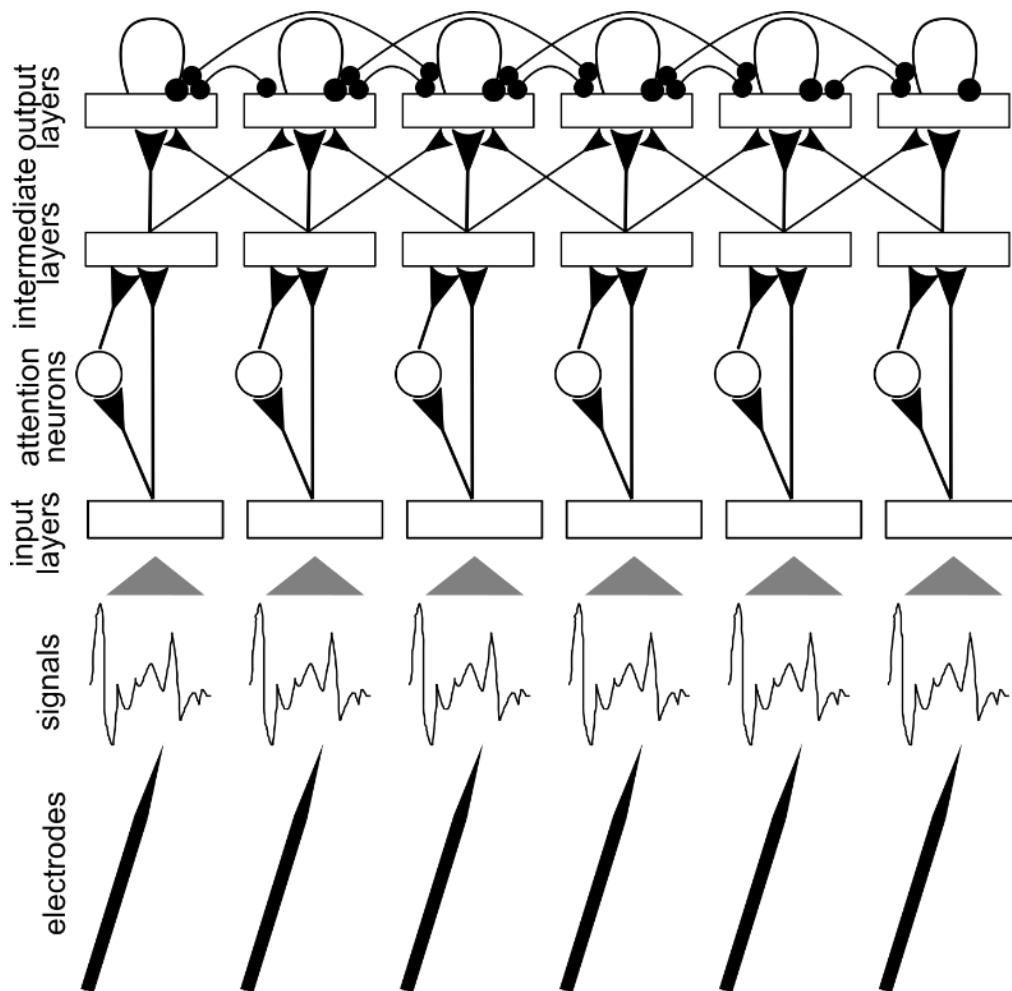
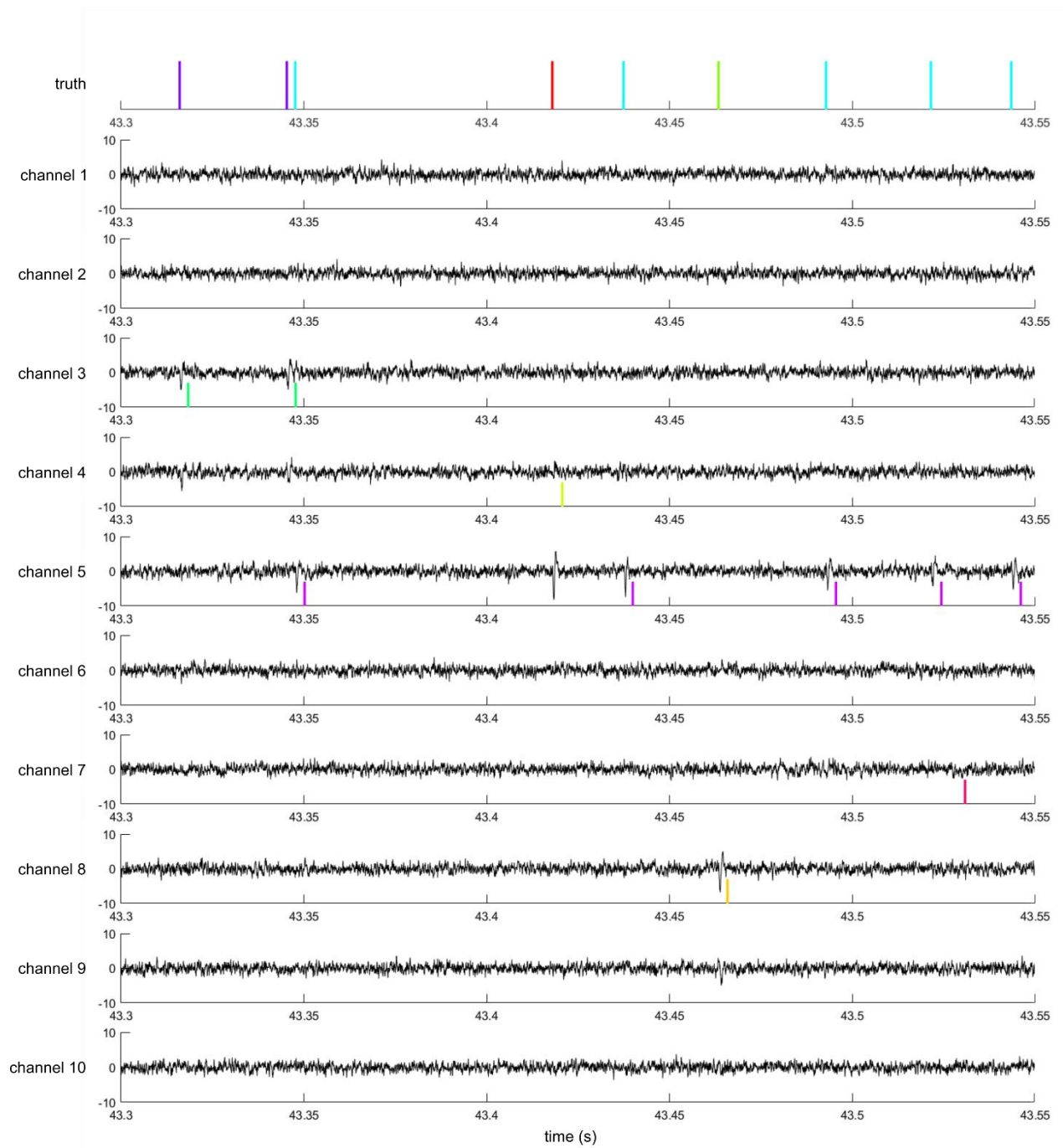


Figure VI-16: PolyNet structure for polytrode recordings (illustration for a line of 6 electrodes).

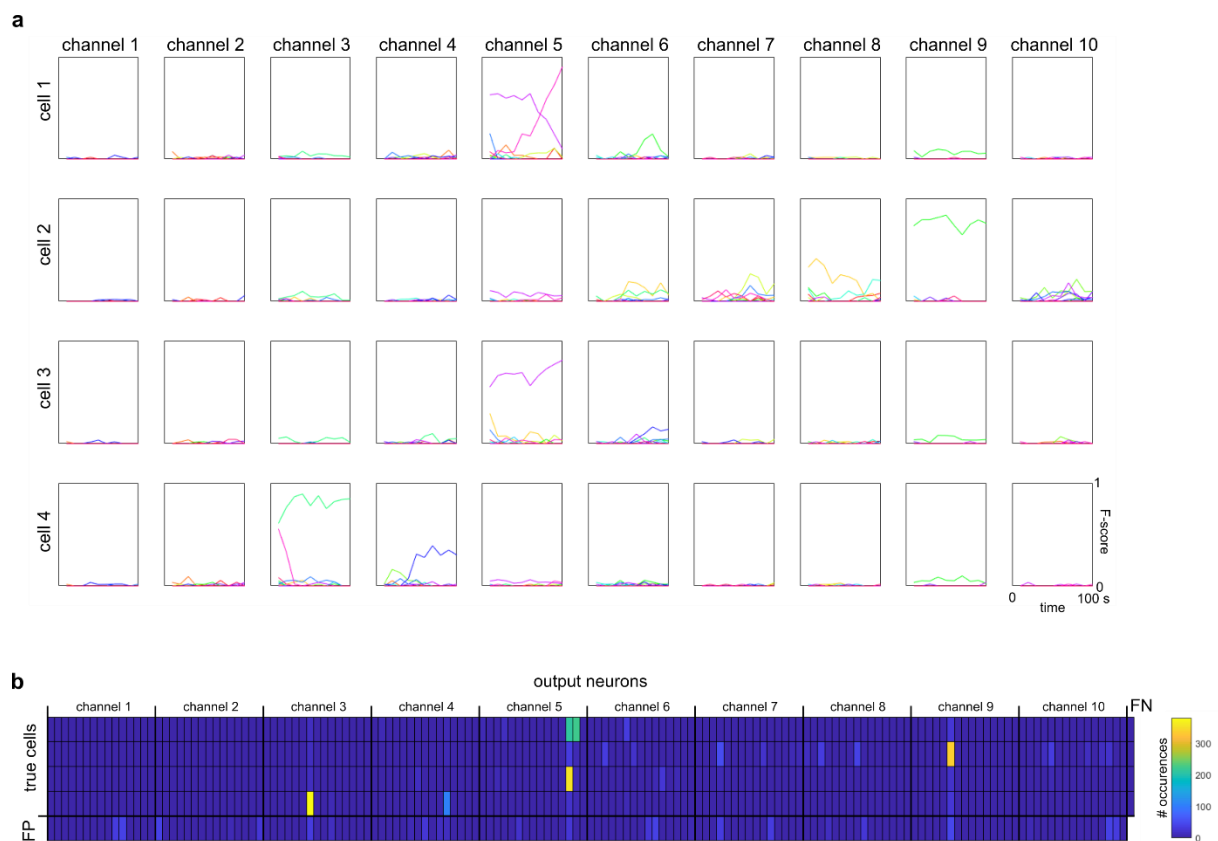
## 2. Results

This implementation was tested on our polytrode simulated dataset (see Section V.A.1). Figure VI-17 shows an example of output on a sample of 250 ms of a simulated polytrode signal with four simulated neural cells and an SNR of 6. Each action potential in the signal triggers a spike from a unique neuron in the entire output layer, most of the time in the sublayer corresponding to the channel where the action potential amplitude is the highest. Figure VI-18 shows the scores obtained on the same signal for each pair of true cells and output neurons. Most action potentials emitted by one specific cell are detected by one specific output neuron, though cell 1 is detected by the same neuron as cell 3 at the beginning of the simulation, and a few false positives and classification mismatches can be observed. This is consistent with the mean clustering, precision and recall scores observed with different SNR (Figure VI-19). Although the recall is quite good, the precision and the clustering score are low, due to the high number of false positives and classification mismatches. Indeed when taking each output neuron separately, the number of corresponding mismatches and false positives is low, as shown for example in Figure VI-18.b, but due to the high number of output neurons (150), the total number of mismatches and false positives become important. The misclassifications could be explained by the concurrence between neighboring output sublayers, as they do not receive exactly the same stimulus. The first output sublayer to fire is not systematically the one corresponding to the highest action potential amplitude, as we would ideally expect. This can be seen for example in channel 4 of Figure VI-17 on which the action potential detected for the true red waveform has a lower amplitude lower than the one occurring on channel 5.

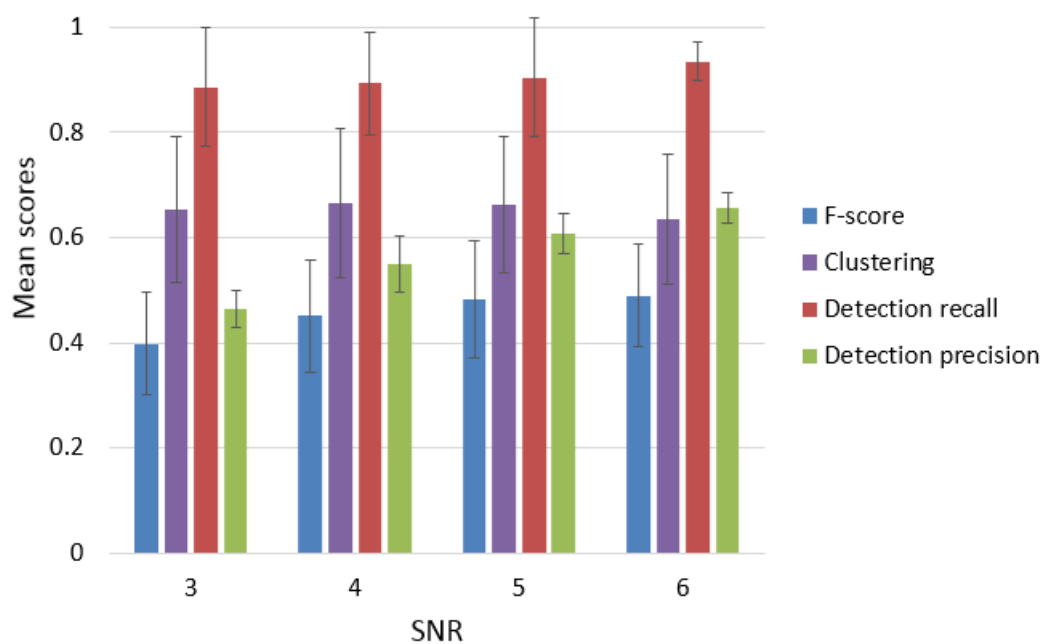


**Figure VI-17:** Example of output on a simulated polytrode recording with an SNR of 6. Truth is shown at the top. Each panel shows the recorded signal for each electrode (black line), and the spikes from the corresponding output sublayer (colored bars). Different colors correspond to different neurons within the same sublayer.





**Figure VI-18: Action potentials stemming from different cells are recognized by a different output neurons. (a) Evolution of the paired F-scores during the simulation. Different colors stand for different output neurons within the same output sublayer. (b) Number of action potentials detected by each output neurons and number of false positives and false negatives, within the last 50 seconds of the simulation.**



**Figure VI-19: Mean scores on the polytrode simulated recordings with different SNR, for each type of error.**

## E. Estimation of the resources used

For each of our different implementations, we made a rough estimation of the resources used for a neuromorphic implementation, in terms of number of neurons, synapses and transmitted spikes. For the input layer, the estimation was done for encoding a signal range of  $20\sigma_{noise}$ . The number of spikes generated and transmitted is expressed in spikes per second on the input layer, and then in spikes per action potential present in the signal. The results obtained are gathered in Table VI-6, Table VI-7 and Table VI-8, for the three single-electrode networks, respectively. Depending on how the neuromorphic implementation is done, some of these implantation bricks can be more limiting than others. However, with our network structure, the limiting factor is likely to be the synaptic connection between the input and the intermediate layers, as the numbers of synapses and of spikes transmitted are very high. Another aspect that can be relevant for power consumption in a neuromorphic implementation is the number of synaptic weights' changes. For the connection between the input layer and the intermediate layer, the number of weights' changes can be estimated at respectively 12720, 5700 and 10000 changes per action potential for respectively MiniNet, ANNet and LTSNet. An estimation of 1pJ per weight change (Xiong et al. 2011; Chin et al. 2013) and an electrode signal containing about 100 action potentials per second lead to an approximation power consumption of 1μW. For the polytrode implementation the quantity of resources used is multiplied by the number of electrodes, except for the intermediate to output layer connection, for which the number of synapses is also multiplied by the number of lateral connections between sublayers (3 in our case). However, this connection should not be the limiting element in this structure.

Table VI-6: Resources used for MiniNet

| NEURONS            |                   |                           | SYNAPSES               |                    |                             |
|--------------------|-------------------|---------------------------|------------------------|--------------------|-----------------------------|
| Layer              | Number of neurons | Number of spike generated | Connection             | Number of synapses | Number of spike transmitted |
| Input layer        | 53                | 160e3 /s                  | Input to intermediate  | 38 160             | 115e6 /s                    |
| Intermediate layer | 30                | 10 /AP                    | Intermediate to output | 300                | 100 /AP                     |
| Output layer       | 10                | 1 /AP                     |                        |                    |                             |

Table VI-7: Resources used for ANNet

| NEURONS | SYNAPSES |
|---------|----------|
|---------|----------|

| Layer              | Number of neurons | Number of spike generated | Connection                | Number of synapses | Number of spike transmitted |
|--------------------|-------------------|---------------------------|---------------------------|--------------------|-----------------------------|
| Input layer        | 57                | 800e3 /s                  | Input to intermediate     | 34 200             | 480e6 /s                    |
| Attention neuron   | 1                 | 40 /AP                    | Input to attention neuron | 570                | 8e6 /s                      |
| Intermediate layer | 60                | 10 /AP                    | Intermediate to output    | 60 000             | 100 000 /AP                 |
| Output layer       | 10                | 1 /AP                     |                           |                    |                             |

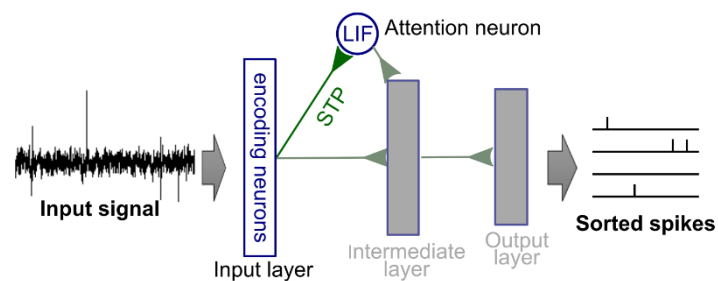
Table VI-8: Resources used for LTSNet

| NEURONS            |                   |                           | SYNAPSES                  |                    |                             |
|--------------------|-------------------|---------------------------|---------------------------|--------------------|-----------------------------|
| Layer              | Number of neurons | Number of spike generated | Connection                | Number of synapses | Number of spike transmitted |
| Input layer        | 50                | 800e3 /s                  | Input to intermediate     | 50 000             | 800e6 /s                    |
| Attention neuron   | 1                 | 40 /AP                    | Input to attention neuron | 500                | 8e6 /s                      |
| Intermediate layer | 100               | 20 /AP                    | Intermediate to output    | 1500               | 300 /AP                     |
| Output layer       | 15                | 1 /AP                     |                           |                    |                             |

## F. Preliminary FPGA Implementation

The different versions of the network were tested by simulations on a workstation. This allows to test the spike-sorting performance of the network but not the efficiency of the implementation, as it is designed to be executed on parallel hardware. Neuromorphic technologies not being mature enough to implement our network, we chose as an intermediate step towards an embedded implementation, to begin an implementation on FPGA devices. This technology has both advantages to be flexible and to offer parallelism architecture opportunities. To this end we started a collaboration with Takashi Kohno and Timothée Levi at the University of Tokyo, whose team has already been working on

implementing neural network on FPGA (Li et al. 2012). I stayed for 2 weeks in Tokyo for this purpose and the work started there was continued by Joel Fournier during a 4-month internship under my supervision. This collaboration and this internship resulted in the implementation of an input layer, a short term-plasticity and an attention neuron, which were tested all together (Figure VI-20). The implementation was done on a Kintex-7 device, on a Genesys 2 development board from Digilent (Figure VI-21). The results were qualitatively good, as this attention neuron was able to detect, in real time, action potentials in a testing signal generated on the fly (Figure VI-22). The parameters still need to be adjusted to optimize the detection and to match the characteristics of a realistic signal. This work should be continued with Takashi Kohno and Timothée Levi to add an STDP rule and LTS neurons to obtain a complete network. During his internship, Joel Fournier also implemented a protocol for receiving data from a computer via a USB-interface included in the development card we used. This interface will be very useful in the future to send electrode recordings in real-time to the FPGA for a real time spike-sorting.



**Figure VI-20: Parts of the network implemented on FPGA. The parts that were not implemented are grayed.**



**Figure VI-21: Genesys 2 board used for FPGA implementation**

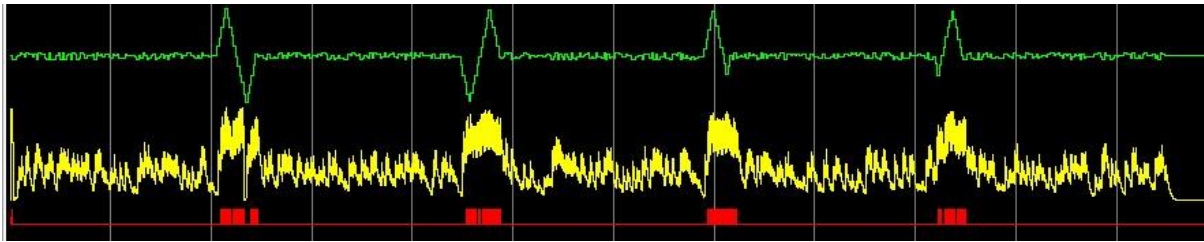


Figure VI-22: Attention neuron simulation implemented on an FPGA. Up: testing signal. Middle: attention neuron potential. Bottom: Attention neuron spikes.

## References

- Chin, A. et al., 2013. Ultra-Low Switching Power RRAM Using Hopping Conduction Mechanism. *ECS Transactions*, 50(4), pp.3–8. Available at: <http://ecst.ecsdl.org/cgi/doi/10.1149/05004.0003ecst>.
- Li, J., Katori, Y. & Kohno, T., 2012. An FPGA-Based Silicon Neuronal Network with Selectable Excitability Silicon Neurons. *Frontiers in Neuroscience*, 6(December), p.183. Available at: <http://journal.frontiersin.org/article/10.3389/fnins.2012.00183/abstract>.
- Quiroga, R.Q., Nadasdy, Z. & Ben-Shaul, Y., 2004. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8), pp.1661–1687.
- Rutishauser, U., Schuman, E.M. & Mamelak, A.N., 2006. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of Neuroscience Methods*, 154(1–2), pp.204–224.
- Xiong, F. et al., 2011. Low-power switching of phase-change materials with carbon nanotube electrodes. *Science*, 332(6029), pp.568–570. Available at: <http://www.nanotechnology.org.in/goto/http://poplab.ece.illinois.edu/pdfs/Xiong-LowPowerPCMCNT-sciexp11.pdf%5Cnhttp://www.ncbi.nlm.nih.gov/pubmed/21393510>.



## VII. CONCLUSION AND PERSPECTIVES

### A. Main contributions

The goal of this thesis was to design a fully automatic online spike-sorting algorithm using an STDP network, in the perspective of an implementation on neuromorphic hardware. The network developed during this work meets in large these requirements. First, the algorithm is entirely neuromorphic. Apart from a simple band-pass filtering, that can be achieved by simple passive analog circuitry, no preprocessing is done before feeding the input signal to the network. The conversion of the analog signal into spike trains is done by a set of simple units that can be assimilated to sensory neurons. No post-processing is needed either, as the network is designed to output spike trains that correspond to the sorted cells' spiking activity. Second, the method is fully automatic. The network has been designed and parameterized taking into account the main characteristics common to all extracellular recordings, in particular the duration of an action potential. The parameters of the network depend only on two characteristics of the input signal that may change between recordings: the noise level and the amplitude range, which can both be easily estimated. Third the algorithm can theoretically be executed online. Indeed the network is designed to process the input signal on the fly. The theoretical latency of the response depends on the dynamic of the neuron models and the synaptic transmission delays used in the network and does not exceed a few milliseconds. In practice, the execution time depends on the physical implementation of the network, in particular on what type of hardware it is executed. Though the algorithm was designed for a neuromorphic implementation, the practical implementation on neuromorphic hardware is beyond the scope of this work. Though, preliminary implementation on FPGA gave good results (see Section VI.F).

An important point when designing an algorithm to solve a problem is to check if the problem is solved correctly. This is not trivial for spike-sorting, as the ground truth is most of the time unknown. A common way to assess the performance of a spike-sorting algorithm is to test it on simulated data, mimicking real recordings, but for which the ground truth is thus known. In this work we generated our own simulated data, using two different methods (see Section V.A.1). The first method, used for single electrode recordings, uses predefined action potential waveforms for each true cell. The second method, used for multiple electrode recordings, uses an extracellular action potential model based on a simplified neural geometry. The simplicity of this model allows to quickly generate multiple electrode data without advanced knowledge in electrophysiology, in contrast with more realistic models that require to simulate complex neural structures (Hines & Carnevale 2001; Hagen et al. 2015). The performance assessment on simulated data was supplemented with tests on real recordings, with a partially known ground truth. These tests showed that our algorithm reaches the state-of-the-art performance on single electrode recordings, using a radically new method (see Sections VI.B and VI.C). It also has a correct qualitative behavior on multiple-electrode recordings (see Section VI.D). In particular we were able to ensure that each action potential in the signal generates only one output spike, in spite of being processed through several parallel channels. Moreover, as expected for an STDP network, our network needs only a few occurrences of a pattern (about a hundred) to learn to recognize it, in contrast to deep neural networks that require to be trained on large datasets.



Beyond the spike-sorting application, many mechanisms were developed for this network, to deal with the specificities of the spike-sorting problem. These mechanisms could be used for other applications. First the patterns to recognize in the signal had a strong temporal aspect as the relative times of the values taken by the signal are crucial to discriminate the patterns (i.e. the action potential waveforms). This is not common in STDP applications. Indeed LIF neurons are naturally able to detect coincidences, but need additional mechanisms to process this temporal aspect. This problem can be simply solved by using delays in the synaptic transmissions (Natschläger & Ruf 1998; Ghosh-Dastidar & Adeli 2007; Ghosh-Dastidar & Adeli 2009), which we did in our network (see Section IV.A and IV.D.3). The network was successfully able to distinguish two waveforms that were symmetric along time (one positive and then negative and the other negative and then positive), showing that the temporal aspect is correctly taken into account. Such network structure using transmission delays has already been used for EEG pattern recognition (Ghosh-Dastidar & Adeli 2007), though in this study, learning is not done through STDP. On the other hand, some studies applied STDP networks to auditory or visual pattern learning, but without using transmission delays (Suri et al. 2013), which limit the network to recognizing short timescale patterns and not complex sequences. The ability to recognize temporal patterns using transmission delays could thus be useful for applications such as video processing or sound and language processing. It would allow for example discriminating similar objects moving in different directions in a video stream, or recognizing whole words or sentences in an audio recording.

Another important aspect of the network is the attention mechanism (see Section IV.B). Artificial neural networks are often presented isolated patterns, removing the need to select the relevant parts of the input stimuli. Recognizing patterns in a composed input stimulus, possibly containing several patterns among irrelevant inputs, is a much more complex problem. Convolutional neural networks solve this problem by scanning all the input stimulus to find local patterns. However, it would be more interesting to process only the relevant parts of the input stimulus through an attention mechanism. In this work, we implemented an attention mechanism thanks to a short-term plasticity rule, able to detect changes in the input stimulus. This mechanism could possibly be extended to visual processing, for example to detect moving objects in a video stream.

Finally, when designing our spike-sorting network, we were confronted to the problem of input patterns with different sizes due to action potentials of different durations. A neuron learning a pattern should thus adapt its error tolerance to the size of the learnt pattern. This was done using an intrinsic plasticity rule that adapts the value of the neuron's threshold to the size of the learnt pattern (see Section IV.D.1). As our patterns are temporal, we also needed a mechanism to ensure that neurons would not fire before the end of the pattern and thus miss information to recognize it. Our most satisfying solution to solve this problem was to use an LTS neuron model that generates a potential rebound after an inhibitory stimulus. During tests on our output layer structure, we showed that combining these two mechanisms (see Section IV.D.4) makes the network model able to discriminate patterns with very different sizes, and even two patterns strictly included one into another. A possible alternative application of this approach could be the processing of complex visual scenes, where relevant objects can have different sizes in the input picture.

## B. Discussion and future work

Our STDP network works well to process single electrode signals. However, the long-term goal is to develop an STDP network able to process signals from dense MEAs. Our implementation of a network processing a multiple-electrode signal is preliminary. We were able to design a network architecture that fires one spike for each action potential, on a channel where this action potential is visible, which is the intended behavior. However, this network still generated too many misclassification errors, in part due to the fact that action potentials stemming from the same neural cell are not robustly classified in the same channel. Future efforts should thus focus on adjusting the architecture to solve this problem. Possibilities are to simply adjust parameters such as the connection weights between layers, to introduce new plasticity mechanisms, or to add a layer dedicated to this problem. For example, the lateral STDP already implemented on the output layer could be adjusted depending on which layer the WTA inhibition comes from. Another aspect that could be investigated is the influence of the number of neurons in each layer on the network performance. Our guess is that, provided that there is already enough neurons to learn all presented patterns, adding additional neurons should not impact the spike-sorting performance. Indeed, all inactive neurons that have not learnt any pattern behave the same way, and their number should not impact the behavior of neurons responding to a pattern. However this hypothesis has not been investigated experimentally.

In this work, we mostly investigated the simplest cases of spike-sorting situations. However, as it has been highlighted in Section II, some specific difficulties emerge when performing spike-sorting on realistic extracellular signals. One of them is the problem of non-stationary data, when the action potentials' shapes slowly change during the recording. As our network is continuously learning, we could expect that it would be robust to slow changes in waveforms. However this still needs to be confirmed by running the network on long non-stationary recordings. Another problem in spike-sorting is to robustly classify action potentials occurring in bursts, in which case they display decreasing amplitudes. Without modification, our network would classify as different action potentials occurring in a burst, although they would stem from the same cell. A possibility to solve the problem would be to add a layer that would classify them as stemming from the same neural cell, by taking advantage of the temporal correlation between them. Finally, a major difficulty is to be able to recognize action potentials overlapping in time. Methods able to do so explicitly model the fact that action potentials sum up. Our way to encode the signal into a spike train does not take into account this property, making it difficult to separate action potentials occurring at the same time on the same electrode. Solving this problem would require changing the encoding paradigm, or adding a mechanism to the input layer to take into account additivity.

All our simulations were done on a computer workstation whose sequential architecture is not adapted to the execution of artificial neural networks. As a first step towards a neuromorphic implementation, part of the network was implemented on an FPGA. Even though this work needs to be completed, it showed the feasibility of an FPGA implementation. Future work should implement first a complete single electrode network, then a multiple electrode network, and study how much electrodes can be simultaneously processed given the FPGA resources. On a long-term perspective, this network could be implemented on neuromorphic hardware using memristive devices. Synapses implemented through memristors or RRAMs are able to mimic STDP and short-term plasticity (Ohno et al. 2011; Saïghi et al. 2015; Werner et al. 2016), which are both used in our network. Neurons, often

implemented using CMOS technologies, can implement integrate-and-fire models as well as more complex models (Indiveri et al. 2013; Sourikopoulos et al. 2017). However mechanisms such as intrinsic plasticity have not yet been tested on neuromorphic device, and thus require further developments.

The limiting element in a neuromorphic implementation is the number of synapses. Though efforts were done to not use an excessive number of synapses, the use of several transmission delays between two layers increases their number. A first idea to reduce the number of synapses could be to reduce the number of delays between two layers, possibly compensating this loss by implementing more layers, each acting at a different time scale. Indeed, it is known in the field of deep-learning, using formal neural networks, that deep structures are more optimal than large structures. It would be interesting to investigate if the same effect is observed with STDP networks. Another possibility would be to implement a plasticity rule on the synaptic transmission delays, such as for example delay plasticity based on spike timing (Eurich et al. 1999; Lücken et al. 2017), removing the need to use several synapses with different delays.

### C. Impact and perspectives

Artificial neural networks, applied to pattern recognition problems, have undergone significant development in the last decades, thanks to the availability of huge datasets and powerful computers. The advent of neuromorphic chips could open a new era in pattern recognition, by allowing to implement such applications into embedded low power devices. However many developments remain to be done in this field, both from the hardware and software points of view, as the available techniques are in their early stages. In particular, STDP networks constitute a new field of artificial intelligence, with few concrete applications. Many different mechanisms are being explored to better regulate the learning dynamic of these networks. Some of them have been used in our network, such as short-term plasticity, intrinsic plasticity, transmission delays, or winner-take-all mechanism. Others were not explored in this work, such as metaplasticity, plasticity on inhibitory synapses, plasticity on delays, but could however be useful for the application to spike-sorting. Managing to implement a spike-sorting method into an embedded neuromorphic chip would strongly benefit to BCI applications, for which real-time processing is crucial. Indeed, no available spike-sorting methods completely meet the need of automatic real-time processing for the ever growing number of electrodes in recording devices. Thus human BCI experiments currently do not use spike-sorting in practice for sake of simplicity, although it has been shown to improve decoding performances (Todorova et al. 2014). Ideally, spike-sorting would be done at the electrode level thanks to such device, thus avoiding to transmit a massive flow of data to an external processing unit. Beyond spike-sorting, the STDP network paradigm could be extended to downstream processing of MEA recordings, such as behavior decoding for fundamental neurosciences or BCI, or to process other types of neural data recordings. Improving neural data processing through neuromorphic implementation would allow analyzing data from many neurons simultaneously and thus improving our understanding of brain functioning. More generally, spike-sorting is a pattern recognition problem and the mechanisms developed in this work to process MEA recordings could be used for completely different applications, as for instance visual pattern recognition or language processing.

## References

- Eurich, C.W. et al., 1999. Dynamics of self-organized delay adaptation. *Physical Review Letters*, 82(7), pp.1594–1597.
- Ghosh-Dastidar, S. & Adeli, H., 2009. A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks*, 22(10), pp.1419–1431.
- Ghosh-Dastidar, S. & Adeli, H., 2007. Improved Spiking Neural Networks for EEG Classification and Epilepsy and Seizure Detection. *Integrated Computer-Aided Engineering*, 14, pp.187–212. Available at: <http://iospress.metapress.com/content/LW681773V3036222>.
- Hagen, E. et al., 2015. ViSAPy: A Python tool for biophysics-based generation of virtual spiking activity for evaluation of spike-sorting algorithms. *Journal of Neuroscience Methods*, 245, pp.182–204. Available at: <http://dx.doi.org/10.1016/j.jneumeth.2015.01.029>.
- Hines, M.L. & Carnevale, N.T., 2001. NEURON : A Tool for Neuroscientists. *Neuroscientist*, 7(2), pp.123–135.
- Indiveri, G. et al., 2013. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology*, 24(38).
- Lücken, L. et al., 2017. Pattern reverberation in networks of excitable systems with connection delays. *Chaos*, 27(1), pp.1–21.
- Natschläger, T. & Ruf, B., 1998. Spatial and temporal pattern analysis via spiking neurons. *Network: Computation in Neural Systems*, 9(3), pp.319–332. Available at: <http://www.informaworld.com/openurl?genre=article&doi=10.1088/0954-898X/9/3/003&magic=crossref%7C%7CD404A21C5BB053405B1A640AFFD44AE3>.
- Ohno, T. et al., 2011. Short-term plasticity and long-term potentiation mimicked in single inorganic synapses. *Nature Materials*, 10(8), pp.591–595.
- Saighi, S. et al., 2015. Plasticity in memristive devices for spiking neural networks. *Frontiers in Neuroscience*, 9(MAR), pp.1–16.
- Sourikopoulos, I. et al., 2017. A 4-fJ/spike artificial neuron in 65 nm CMOS technology. *Frontiers in Neuroscience*, 11(MAR), pp.1–14.
- Suri, M. et al., 2013. Bio-inspired stochastic computing using binary CBRAM synapses. *IEEE Transactions on Electron Devices*, 60(7), pp.2402–2409.
- Todorova, S. et al., 2014. To sort or not to sort : the impact of spike- sorting on neural decoding performance. *Journal of Neural Engineering*, 11.
- Werner, T. et al., 2016. Experimental demonstration of short and long term synaptic plasticity using OxRAM multi k-bit arrays for reliable detection in highly noisy input data. In *2016 IEEE International Electron Devices Meeting (IEDM)*. IEEE, p. 16.6.1-16.6.4. Available at: <http://ieeexplore.ieee.org/document/7838433/>.



## VIII. ANNEXES

### A. Intermediate layer receptive field shape

The input signal is encoded into a spike train by the input layer, which is then transmitted to the intermediate layer through synapses implementing an STDP rule. At the level of the synapse, we know that the synapse weight will converge towards one or zero depending on the probability to have a presynaptic spike in the STDP coincidence window each time a postsynaptic spike occurs. Here we study the learning process steady state at the level of a neuron from the intermediate layer, in other word, the neuron receptive field and the weight of its incoming synapses after learning. At each sampling step, the spikes received by the intermediate, transmitted from the input layer through delayed synapses, represent the signal value at  $N$  different delays from current time. A pattern to learn can thus be seen as an  $N$  dimensional vector, encoded into spikes by the input layer. We assume in this section that the only variation between action potential waveforms from the same cell is due to a white Gaussian noise added to the signal. Thus at each occurrence of a pattern, a Gaussian noise is added to the  $N$  dimensional vector representing this pattern, which will induce variations in the set of spikes encoding it. Here we will also neglect the effect of time integration and assume that the potential of the neuron is mostly due to the last set of spikes received from the input layer.

Let's focus first on the one-dimensional case. We denote  $x$  the input value received by the network. Each sensory neuron  $n$  can be associated with a center value  $c_n$ , that corresponds to the center of its sensitivity range. For an input value  $x$ , the number of spikes emitted by a sensory input neuron is given by an activation function  $a_{c_n}(x)$  that is the same for each sensory neuron but shifted according to their center value:

$$a_{c_n}(x) = a_0(x - c_n)$$

Let  $W_n$  be the weight of the synapse connecting the  $n^{\text{th}}$  sensory neuron to the studied intermediate neuron. Given the way the sensory neurons are organized, we can define a weight density:

$$w(c_n) = W_n/DV_i$$

Where  $DV_i$  is the difference between two consecutive sensory neuron center values. In this one-dimensional case, assuming that the sensory neurons cover a value range large enough not to have border effects, the excitation of the intermediate neuron for an input value  $x$  can be written:

$$E_1(x) = \sum_n W_n a_{c_n}(x) = \sum_{n=-\infty}^{+\infty} w(n * DV_i) a_0(x - n * DV_i) DV_i$$

For simplicity and assuming that  $DV_i$  is small, we can rewrite this expression in a continuous form:

$$E_1(x) = \int_{-\infty}^{+\infty} w(c) a_0(x - c) dc$$

Whereas  $a_0$  is known by definition, we need to do hypotheses on the form of  $w$ . We assume for the moment that only one pattern  $x_0$  is presented to the network (or that the different patterns presented are far enough from each other not to interact). Without losing generality we can assume that  $x_0=0$ .

Thus the input value  $x$  of the network follows a normal distribution. Given the symmetry of the problem and the STDP rule properties, we assume that  $w(x)$  equals 1 for  $x \in [-V_{pot}; V_{pot}]$  and 0 otherwise for a value  $V_{pot}$  to be determined. Using the binary encoding presented in paragraph IV.A, we have  $a_0(x)=1$  for  $x \in [-DV_m; DV_m]$  and  $a_0(x)=0$  otherwise, where  $DV_m$  is half the size of the sensitivity range. We can then compute the value of the neuron's potential. Both  $a_0$  and  $w$  are equal to one on a segment and to 0 otherwise. When  $|x| < |DV_m - V_{pot}|$ , the segments  $[-V_{pot}; V_{pot}]$  and  $[-DV_m - x; DV_m - x]$  are included one into the other and  $E_1(x) = 2\min(DV_m, V_{pot})$ . When  $|x| > |DV_m + V_{pot}|$  the segments are disjoint and  $E_1(x) = 0$ . In between  $E_1(x)$  follow a linear evolution.  $E_1(x)$  can thus be written as follow:

$$E_1(x) = \begin{cases} 2\min(DV_m, V_{pot}) & \text{if } |x| < |DV_m - V_{pot}| \\ \max(DV_m + V_{pot} - |x|, 0) & \text{otherwise} \end{cases}$$

Visually,  $E_1(x)$  has a trapezoidal shape. Noticeably for  $DV_m=V_{pot}$ ,  $E_1(x)$  become triangular with  $E_1(x)=\max(2DV_m-|x|, 0)$ .

Knowing the neuron excitation in the one-dimensional case, it is easy to deduce the excitation for an N-dimensional input. For an input  $x = (x_i)_{i \in \llbracket 1; N \rrbracket} \in \mathbb{R}^N$  we have:

$$E_N(x) = \sum_{i=1}^N E_1(x_i)$$

The neuron fires when its potential reaches its threshold  $Th$ . Thus the receptive field  $D_s$  of the neuron is defined by:

$$D_s = \{x \in \mathbb{R}^N / E_N(x) \geq Th\}$$

When  $DV_m=V_{pot}$ , which is a condition that is easily reached as we will show later,  $D_s$  has some nice properties. Indeed  $D_s$  is bounded for  $Th > 2DV_m(N-1)$ , and is then an L1-norm ball of diameter  $2DV_mN - Th$ . Indeed when  $Th > 2DV_m(N-1)$  for  $x$  in  $D_s$  we have  $|x_i| \leq 2DV_m$  for all  $i$ , because otherwise  $E_N(x) < Th$ . Thus  $D_s$  is bounded and  $E(x)$  can be rewritten  $E(x) = 2 * DV_m * N - \|x\|_1$  and thus  $x \in D_s \Leftrightarrow x_1 \leq 2 * DV_m * N - Th$ . For lower values of  $Th$ ,  $D_s$  is not bounded, as all  $x$  such as  $x_1 = \dots = x_{N-1} = 0$  and  $x_N \in \mathbb{R}$  belong to  $D_s$ , but we can show that for any  $k \in \llbracket 1, N \rrbracket$ , if  $Th > 2(k-1)DV_m$ , then for all  $x \in D_s$  at least  $k$  components of  $x$  are inferior to  $2DV_m$ .

For the triangular encoding proposed in Section IV.A, the activation function has a triangular shape and can be written:

$$a_0(x) = K * \max(1 - \frac{|x|}{DV_m}, 0)$$

Where  $K$  is the maximum number of spikes emitted. In this case, and with the condition  $DV_m=V_{pot}$ , it can be shown, with a similar reasoning as previously, that  $D_s$  is bounded for  $Th > KDV_m(N-1)$ , and that for  $Th > KDV_m(N-0.5)$ ,  $D_s$  is and L2-norm ball of diameter  $\sqrt{2}DV_m \sqrt{N - \frac{Th}{K * DV_m}}$ .

Noticeably in both cases, if we denote  $E_{max}$  the maximum possible excitation for one dimension, the bounded condition is given by:

$$Th > E_{max} * (N - 1)$$

Knowing the receptive field of the neuron and the distribution of the input values, we can then deduce the synapse weight value according to the STDP rule, and thus the value of  $V_{pot}$ . The STDP rule state that for each input value  $x \in \mathbb{R}^N$ , the weight change for the synapse corresponding to the  $i^{th}$  delay and a sensory neuron with a center value  $c$  is:

$$\Delta w_i(c) = \begin{cases} 0 & \text{if } x \in D_S \\ \Delta w_+ - \Delta w_- & \text{if } x \in D_S \text{ and } x_i \in S_c \\ -\Delta w_- & \text{if } x \notin D_S \text{ and } x_i \in S_c \end{cases} \quad \text{with } S_c = [c - DV_m, c + DV_m]$$

We note  $p(x)$  the probability of  $x$  to be an input of the network. The mean variation of the synapse weight when the network receive an input is:

$$\langle \Delta w_i(c) \rangle = \Delta w_+ \int_{D_S \cap \{x/x_i \in S_c\}} p(x) dx - \Delta w_- \int_{D_S} p(x) dx$$

For  $x$  following a Gaussian distribution,  $\langle \Delta w_i(c) \rangle$  is maximum for  $c=0$  and decrease when  $|c|$  increase. This works more generally if  $p(x)$  is symmetric relatively to each of the dimensions and  $p(x)$  decreases when  $|x_i|$  increases, as  $D_S$  is also symmetric and its cross-section with the hyperplane define by  $x_i=c$  decreases when  $|c|$  increase. We suppose that for all  $x \in D_S$  such that  $\|x\|_\infty > 2DV_m$ ,  $p(x)=0$ , which is in particular true if for all  $x \in D_S$   $\|x\|_\infty \leq 2DV_m$ . Then  $\langle \Delta w_i(c) \rangle$  is positive for  $c=0$  and negative when  $|c|$  is high enough. Thus there exist a unique value  $V_{pot}$  for which  $\langle \Delta w_i(c) \rangle$  is positive if  $|c| < V_{pot}$ , and negative if  $|c| > V_{pot}$ , which correspond to the initial definition of  $V_{pot}$ . Additionally, if  $\Delta w_+ = 2 \Delta w_-$ , we can verify that  $V_{pot}=DV_m$ , thanks to the symmetry properties of  $D_S$  and  $p$ .



## B. Article about ANNet

The following pages are an article, recently accepted for publication in the International Journal of Neural Systems. This article sums up the implementation and results obtained with ANNet (see Section VI.B).

# AN ATTENTION-BASED SPIKING NEURAL NETWORK FOR UNSUPERVISED SPIKE-SORTING

**MARIE BERNERT**

*BrainTech Lab U1205, INSERM and University Grenoble Alpes  
2280 rue de la piscine, Saint Martin d'Hères 38400, France BrainTech Lab U1205  
LETI, CEA Grenoble  
17 rue des Martyrs, Genoble 38000, France*

**BLAISE YVERT\***

*BrainTech Lab U1205, INSERM and University Grenoble Alpes  
2280 rue de la piscine, Saint Martin d'Hères 38400, France  
E-mail: blaise.yvert@inserm.fr*

Bio-inspired computing using artificial spiking neural networks promises performances outperforming currently available computational approaches. Yet, the number of applications of such networks remains limited due to the absence of generic training procedures for complex pattern recognition, which require the design of dedicated architectures for each situation. We developed a spike-timing-dependent plasticity (STDP) spiking neural network to address spike-sorting, a central pattern recognition problem in neuroscience. This network is designed to process an extracellular neural signal in an online and unsupervised fashion. The signal stream is continuously fed to the network and processed through several layers to output spike trains matching the truth after a short learning period requiring only few data. The network features an attention mechanism to handle the scarcity of action potential occurrences in the signal, and a threshold adaptation mechanism to handle patterns with different sizes. This method outperforms two existing spike-sorting algorithms at low signal-to-noise ratio (SNR) and can be adapted to process several channels simultaneously in the case of tetrode recordings. Such attention-based STDP network applied to spike-sorting opens perspectives to embed neuromorphic processing of neural data in future brain implants.

**Keywords:** Spike-timing-dependent synaptic plasticity, spiking neural network, spike-sorting, unsupervised learning, attention mechanism

## 1. Introduction

Pattern recognition is a fundamental task performed very efficiently by the brain. Artificial intelligence is making quick progress in reproducing these performances with artificial neural networks. In particular, deep neural networks have been successful both for static and sequential patterns recognition in many applications such as image recognition, analysis of video streams or natural language processing. However, these types of neural networks require extensive training on large data sets and heavy computations. A promising alternative are spiking neural networks (SNNs), also known as the third generation of artificial neural networks<sup>1</sup>, which have in theory more computational efficiency<sup>2</sup>. Some class of

SNNs such as spiking neural P systems have been shown to be Turing universal<sup>3</sup>. Whereas formal neurons output a real value modelling a firing rate, spiking neuron models have a temporal dynamics and output spike trains, thus keeping the spike timing information. Importantly they offer an opportunity to be implemented in very low power analog neuromorphic hardware that currently undergoes important developments, in particular with memristive devices that are able to mimic synaptic plasticity, such as spike timing dependent plasticity (STDP), at a highly miniaturized scale<sup>4-11</sup>. STDP rules are local learning rules that change the synapse weight depending on the time difference between pre and postsynaptic spikes. It has been shown that incorporating such rule in SNNs allows to perform

---

\* Corresponding author

unsupervised learning<sup>12,13</sup>, and this strategy has been successfully applied for visual or auditory pattern recognition<sup>14–17</sup>. Yet, for now, the range of applications of spiking neural networks in pattern recognition remains limited, most studies being focused on biological modeling<sup>18–22</sup>. Whereas second-generation networks, and some specific spiking neural network<sup>23–25</sup>, can be trained using backpropagation, there is no well-established way to train a spiking neural network implementing an STDP rule. New architectures thus remain to be proposed to address a wider range of applications of these networks.

Here, we present an STDP spiking neural network to perform spike-sorting, a central problem in neuroscience. Spike-sorting consists in determining from a common extracellular signal how many cells emit action potentials and, for each cell, when these events take place. Action potentials occur sparsely at discrete time points, and appear in the signal as temporal waveforms whose shapes typically differ from one neuron to another according to the geometry and position of the cells with respect to the recording electrode. Thus, spike-sorting can be seen as an unsupervised pattern recognition task, where the goal is to detect and classify different temporal waveforms, occurring sparsely in a noisy signal. Currently available spike-sorting approaches typically use three separate steps.

- The first step is the detection of action potentials, usually done by thresholding the extracellular signal directly or after suitable filtering, for example using an energy operator<sup>26</sup> or templates<sup>27</sup>.
- The second step is the extraction of features from the action potential waveforms, such as amplitudes<sup>27</sup>, wavelet coefficients<sup>28</sup>, or reduced representations obtained with dimensionality reduction algorithms such as PCA<sup>29–31</sup>.
- Finally, the last and most computation-demanding step consists in clustering these features. Many clustering algorithms have been proposed, such as expectation-maximization<sup>29,30</sup>, K-means<sup>32</sup>, C-means<sup>33</sup>, mean-shift<sup>31,34</sup>, or superparamagnetic clustering<sup>28</sup>. Noticeably, ref 35 used an STDP network for this clustering step.

Although some spike-sorting algorithms have been designed to process neural data online<sup>26,35,36</sup>, most methods remain offline or require an offline preprocessing step, precluding their use in online applications such as brain-computer interfaces (BCIs). Moreover, the current availability of very dense arrays of

microelectrodes<sup>37–42</sup> creates the need for efficient ways to handle the important data flow generated by these devices and in particular to automate spike-sorting processing<sup>30,31,34,36,43</sup>. Ideally, implantable neural interfaces would strongly benefit from fully automatic spike-sorting algorithms compatible with very low-power hardware implementation for future embedding at the electrode site. However, no such algorithm is available yet.

In this context, we propose to use a spiking neural network as a radically new way to handle the problem of spike-sorting. We present an STDP network architecture designed for this specific problem. The network processes continuously the stream of data recorded by a microelectrode and directly outputs trains of artificial spikes that correspond, after a short learning period, to the sorted activity of the recorded cells. Thus, the entire processing is done by a single network in a fully unsupervised manner, without explicit implementation of the conventional three-step procedure and without requiring a readout post processing step. Beyond the classical STDP learning rule, the network combines an attention mechanism, delayed synapses, and threshold adaptation to handle simultaneously the problems of detecting and classifying time-varying patterns with different sizes. This approach was tested on both simulated and real data and compared to two available spike-sorting software, Osort<sup>26</sup> and Wave\_clus<sup>28</sup>, showing better performance at low signal to noise ratio (SNR).

## 2. Materials and methods

### 2.1. Network model

We considered an artificial spiking neural network composed of three layers (input, intermediate and output) and one “attention” neuron, all connected by feedforward synapses implementing specific plasticity rules (Figure 1). This network, composed of “sensory” and Leaky-Integrate-and-Fire (LIF) neurons, is bio-inspired rather than realistic. In particular the characteristic times are much shorter than realistic ones, which allows processing the information contained in the action potential shapes and makes the approach compatible with real-time spike-sorting at the millisecond time scale. After learning, the output spikes of the network correspond to the detected and sorted action potentials present in the input signal. To avoid any confusion in the following, the term “spikes”

thereafter denotes action potentials emitted by neurons of the artificial neural network, while “action potentials” denotes those of real neurons embedded in the input signal analyzed by the network.

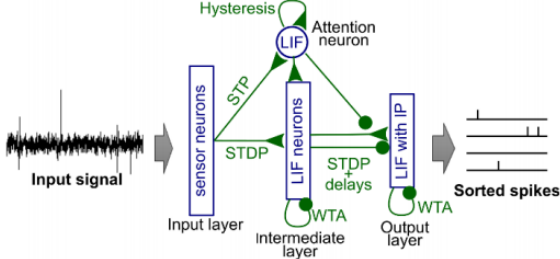


Fig. 1. Overview of the network structure.

### 2.1.1. Input layer

The first layer, called the input layer, encodes the incoming neural signal within a sliding time window preceding the current time point into artificial spikes that are passed on to the next layers. Neurons of this input layer act as “sensory” neurons.

The input neurons are organized into a grid with  $N_c$  columns corresponding to different processing delays of the input signal, starting from zero and regularly spaced by  $\Delta t_c$  (see Figure 2a). Within each column, each neuron is sensitive to a given range of signal values and fires at regular time steps  $\Delta t_s$  when the signal falls within this sensitivity range (see Figure 2b). Thus, at each of these time steps, each input neuron performs a simple

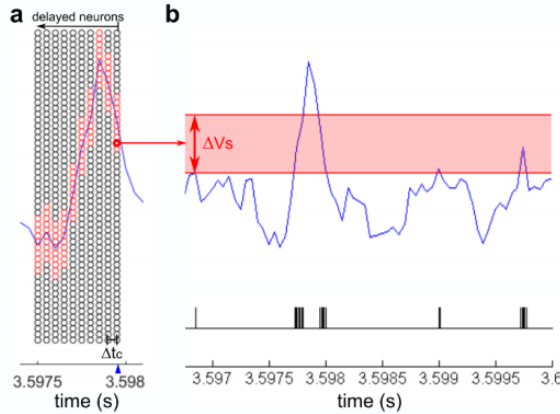


Fig. 2. Input layer structure (a) Input layer activation at a specific time step (blue arrow). (b) Spike train emitted by one specific input neuron, highlighted in bold red in (a).

computation that consists in comparing the input signal value to the extrema of its sensitivity range and fires accordingly. The size of this range,  $\Delta V_s$ , is the same for each neuron, and is set proportional to the noise level. The sensitivity ranges of neurons from one column are regularly overlapping so that, for each possible signal value,  $N_{overlap}$  neurons of this column fire. Within one column, the step between two consecutive sensitivity ranges is determined by the size of the sensitivity range  $\Delta V_s$  and the overlapping factor  $N_{overlap}$ , and the total number of neurons is determined by the total signal value range to cover. The input layer processes the input signal at regular time steps,  $\Delta t_s$ . This time step is chosen sufficiently short so that the time jitter due to sampling does not impact the performance of the network. As this time step is shorter than the original signal sampling time step, the signal is linearly interpolated for steps falling between samples. Assuming that the timing properties are consistent between different neural recordings, the sensitivity range size is the only parameter of the network that needs to be adjusted depending on the signal properties. The values of the input layer parameters are given in Table 1.

As a result, the input layer converts continuously the input signal into an artificial spike train encoding at each time step the shape of the signal within a sliding window. The role of the rest of the network is to detect different patterns into this input spike train that reflect the shape of action potentials stemming from different cells.

Table 1. Input layer parameters.

| Parameter     | Description  | Value               |
|---------------|--|---------------------|
| $\Delta V_s$  | Sensitivity range size                                     | $3.5\sigma_{noise}$ |
| $N_{overlap}$ | Number of neuron active at the same time within one column | 10                  |
| $\Delta t_s$  | Delay interval between two activations of the input layer  | 0.0125 ms           |
| $\Delta t_c$  | Time interval between two columns                          | 0.05 ms             |
| $N_c$         | Number of column   | 10                  |

### 2.1.2. Attention neuron

The role of the “attention” neuron is to detect every action potential occurrences embedded in the input signal. Each time an action potential is present in the input signal, the attention neuron fires a sequence of spikes from the beginning to the end of the action potential. This spiking activity in turn modulates the



intermediate and output layers, thus acting as an attention mechanism gating the learning mechanisms occurring on these two subsequent layers.

The attention neuron receives all spikes emitted by the input layer through excitatory synapses that implement a short-term plasticity (STP) rule, governed by the following equation:

$$\frac{dw}{dt} = \frac{1}{\tau_{stp}}(1 - w) - \sum_s w * f_d * \delta(t - t_s) \quad (1)$$

where  $w$  is the synaptic weight,  $\tau_{stp}$  a time constant,  $f_d$  a depression factor, and  $t_s$  the presynaptic spike times. This STP rule weakens the weights of synapses for which presynaptic spikes occur at high frequency<sup>44</sup>. As a result, the weights of the synapses corresponding to presynaptic input neurons encoding signal amplitudes within the range of noise (and thus often activated) are weakened, while synapses corresponding to presynaptic input neurons encoding signal amplitudes outside this range remain strong. The time constant  $\tau_{stp}$  is set long compared to the duration of an action potential, so that the synaptic weights do not change significantly during an action potential. The depression factor  $f_d$  is set so that the asymptotic weight of a synapse stemming from an input neuron firing at each time step is significantly lower (x0.13 in our network implementation) than that stemming from an input neuron never firing. Thanks to these STP synapses, the excitation received by the attention neuron is much more important for infrequent signal amplitudes than for signal values close to zero. (Figure 3, middle).

The attention neuron is a Leaky-Integrate and Fire (LIF) neuron, which membrane potential  $V$  is governed by the following equation:

$$\frac{dV}{dt} = -\frac{1}{\tau_m} * V(t) + \sum_i \sum_s w_i(t_{i,s}) \delta(t - t_{i,s}) \quad (2)$$

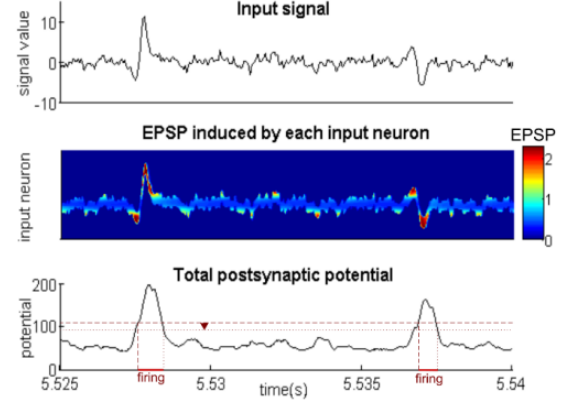


Fig. 3. Short-term plasticity effect on the attention neuron. Top: example of input signal. Middle: corresponding EPSP induced on the attention neuron by each input neuron of one input layer column. Bottom: time evolution of the attention neuron potential and its firing times given its threshold (dash line) and self-excitation (arrow).

where  $V$  stands for the neuron potential,  $\tau_m$  for the membrane time constant,  $i$  indexes the incoming synapses with synaptic weight  $w_i$ ,  $s$  indexes the received spikes from each synapse  $i$  with arrival time  $t_{i,s}$ , and  $\delta$  is the Dirac delta function. The membrane time constant of the attention neuron is chosen relatively short, in order to detect an action potential as early as possible. Combined with the STP rule, this LIF dynamic leads to an increased membrane potential when an action potential occurs in the input signal (Figure 3 bottom). The attention neuron threshold value is then set empirically to obtain a compromise between false negative and false positive detection errors. The neuron has no refractory period and no reset potential is applied. Instead, after firing, the potential is increased through a self-excitatory synapse, which constitutes a hysteresis mechanism. The self-excitatory synapse ensures that, even if the action potential waveform crosses zero, thus lowering temporarily the attention neuron potential, the detection spike train is continuous from the beginning to the end of

Table 2. Attention neuron parameters.

| Parameter       | Description                             | Value   |
|-----------------|---|---|
| $\tau_m$        | Membrane time constant                  | $0.5 * \Delta t_c = 0.025$ ms   |
| $\tau_{refrac}$ | Refractory period                       | 0   |
| $Th$            | Neurons threshold                       | $0.43 * N_{overlap} * N_c / (1 - \exp(-\Delta t_s / \tau_m)) = 108.1$ |
| $w_{self}$      | Weight of the self-excitatory synapse   | $0.3 * (1 / \exp(-\Delta t_s / \tau_m) - 1) * Th = 10.5$              |
| $\tau_{stp}$    | Short term plasticity time constant     | $10 * \Delta t_c * N_c = 5$ ms  |
| $f_d$           | Short term plasticity depression factor | $1 - \exp(-6.5 * \Delta t_s / \tau_{stp}) = 0.0161$                   |

the action potential. The values of the attention neuron parameters are given in Table 2.

### 2.1.3. Intermediate layer

The role of each intermediate layer neuron is to learn to recognize specific fragments of action potential waveforms present in the input signal so that the layer emits different firing sequences for different action potential waveforms. The activity of intermediate layer neurons is gated by the attention neuron through fixed-weight excitatory synapses, ensuring that these neurons fire when the attention neuron fires and remain silent otherwise. Thus, for each action potential occurrence in the signal, the intermediate layer fires a sequence of spikes characterizing the action potential shape, as it is different for different action potential waveforms and stable across occurrences of the same waveform (Figure 4a). These different sequences are then processed by the output layer, to sort the action potentials.

Neurons of the intermediate layer are LIF neurons following a similar equation as the attention neuron (Equation 2). They receive all spikes emitted by the input layer through excitatory synapses whose weights vary according to an STDP rule. The membrane time constant  $\tau_m$  of the intermediate neurons is the same order of magnitude as  $\Delta t_s$ , so they are sensitive to the shape of the

signal at current time, which they have to recognize. As part of a winner-take-all (WTA) mechanism, each time an intermediate neuron fires, all intermediate neurons reset their potential to zero. This ensures that only one neuron fires at a time. Intermediate neurons thus fire one after the other, separated by a time interval that depends on the received excitation and the threshold, and is here adjusted to be approximately equal to  $\Delta t_c$ .

The STDP rule is based on a coincidence time window defined by the value  $\tau_{stdp+}$  (Figure 4b). A presynaptic spike is considered to coincide with a postsynaptic spike if it arrives at most  $\tau_{stdp+}$  before the postsynaptic spike. For each post-synaptic spike occurrence, the synapse weight is decreased by  $\Delta w_{post}$ , and additionally, if a presynaptic spike coincides with the postsynaptic spike, the synapse weight is increased by  $\Delta w_{pair}$  (resulting in a total change of  $\Delta w_{post} + \Delta w_{pair}$ ). Consequently, the synaptic weight globally increases toward one if the ratio of pre- and postsynaptic spike coincidences over postsynaptic spike occurrences is superior to the ratio  $|\Delta w_{post}/\Delta w_{pair}|$ , and decreases to zero otherwise.

The synapses weights are initialized randomly according to a uniform distribution. The mean weight is chosen so that the neurons potential is just high enough to reach the threshold when the attention neuron also

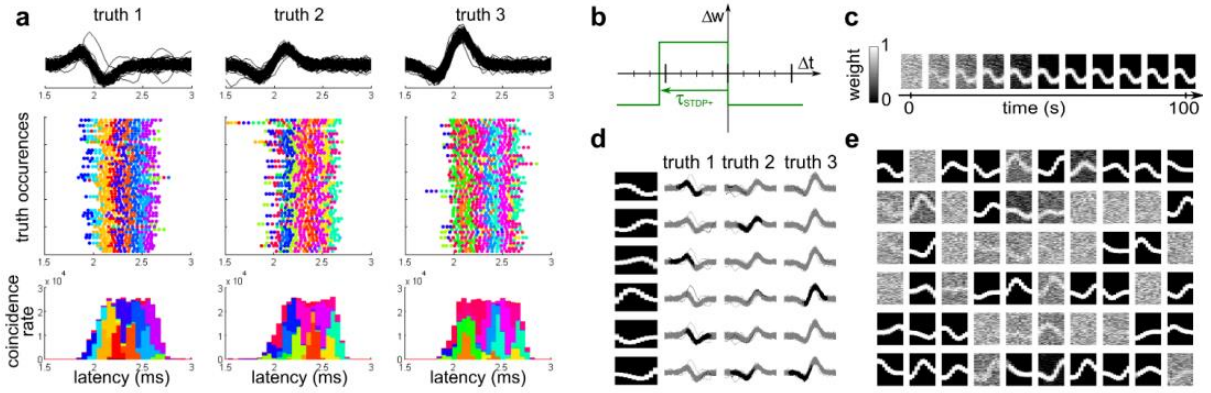


Fig. 4. The intermediate layer learns waveform fragments. (a) Spike sequence emitted by the intermediate layer for each action potential occurrence. Top: signal shapes for all action potential occurrences on the last 100 s of simulation. Middle row: spike trains of the intermediate layer synchronized with 50 different action potential occurrences. Bottom row: distribution of intermediate spike latencies relative to each action potential occurrence (the histograms are cumulated). The different colors stand for different intermediate neurons. (b) STDP rule applied on the synapses stemming from the input layer. (c) Weight evolution of the synapses projecting on one specific intermediate neuron. (d) Learnt shapes correspond to fragments of waveforms occurring in the signal. The first column shows the learnt weights for 6 different intermediate neurons. The three other columns show the signal shape at each action potential occurrence, the black lines show the part of the signal encoded by the input layer at the moment when the intermediate neuron fires. (e) Weights learnt by each 60 intermediate neurons after a 200-s simulation.



Table 3. Intermediate layer parameters.

| Parameter         | Description   | Value  |
|-------------------|---|--|
| $N_{neur}$        | Number of neurons   | 60   |
| $\tau_m$          | Membrane time constant  | $2 * \Delta t_s = 0.025$ ms  |
| $\tau_{refrac}$   | Refractory period   | $\Delta t_c = 0.05$ ms   |
| $V_{reset}$       | Reset potential   | 0  |
| $V_{inhib}$       | Post-inhibition potential   | 0  |
| $w_{AN}$          | Weight of the synapses coming from the attention neuron                     | $0.6 * N_c * N_{overlap} = 60$   |
| $Th$              | Neurons threshold   | $(w_{AN} + 0.7 * N_c * N_{overlap}) * (1 - \exp(-\Delta t_c / \tau_m)) / (1 - \exp(-\Delta t_s / \tau_m)) = 252.7$ |
| $w_{init-}$       | Minimal value for the feedforward synapses weight initialization.           | 0.4  |
| $w_{init+}$       | Maximal value for the feedforward synapses weight initialization.           | 1  |
| $\tau_{stdp+}$    | Positive STDP rule time window  | $\Delta t_c + 0.5 * \Delta t_s = 0.0563$ ms  |
| $\tau_{stdp-}$    | Negative STDP rule time window  | 0  |
| $\Delta w_{pair}$ | Weight change for a presynaptic spike coinciding with a post synaptic spike | 0.005  |
| $\Delta w_{post}$ | Weight change for each postsynaptic spike                                   | $-0.55 * \Delta w_{pair}$  |

fires. Thus, at the beginning of the signal processing, when an unknown action potential waveform is presented, one intermediate neuron arbitrarily fires first and inhibits the others. This neuron updates its synaptic weights whereas synapses projecting to the other intermediate neurons remain unchanged. The STDP time window being equal to  $\Delta t_c$ , the potentiated synapses reflect the shape of the signal at this specific time with the same time resolution as the input layer. Consequently, when a similar waveform fragment occurs again, this neuron is more likely to fire and thus reinforce the specificity of its response to this particular waveform pattern (Figure 4c,d). Once learning has been achieved, several intermediate neurons become specifically responsive to different waveform fragments (Figure 4d,e). The intermediate layer is constituted of 60 neurons, which we found enough to learn the different possible input patterns. This was confirmed by the fact that some intermediate neurons did not learn any pattern (Figure 4e). It is possible to increase this number if a great variety of waveforms is expected in the signal. The values of the intermediate layer parameters are given in Table 3.

#### 2.1.4. Output layer

The purpose of the output layer is to finalize the spike-sorting process, by learning to recognize spike sequences produced by the intermediate layer that correspond to combinations of waveforms fragments. If the input signal contains action potentials of  $N$  different cells, a successful spike-sorting results in exactly  $N$  active

neurons of the output layer, each firing once for each occurrence of an action potential of a given cell in the input signal. An important problem to overcome is the ability to learn patterns of different sizes and to differentiate between patterns that may overlap or include one into another. Overlapping patterns can occur for example if two action potential waveforms have the same beginning but different endings.

The output layer is designed here to overcome these problems using four features in addition to the STDP rule and the WTA mechanism.

- First, each pair of intermediate and output neurons are connected through multiple synapses with different transmission delays, as in refs 45–47. This gives information to the output layer about the spike times of the intermediate layer (Figure 5a).
- Second, the output layer is inhibited by the attention neuron. This inhibition is applied on the feedforward synapses from the intermediate layer, after the transmission delay (mimicking a biological presynaptic inhibition), so that intermediate spikes arriving at the output layer are prevented from being transmitted to the output neuron if the attention neuron is firing at the same time (Figure 5b). These spikes are thus not taken into account for the plasticity rules. This forces the output neurons to wait until the end of a pattern before firing and thus to take into account the entire pattern to adjust the weights of incoming synapses.

Third, each neuron of the output layer receives the spikes of each neuron of the intermediate layer through both excitatory and inhibitory synapses implementing an STDP rule, allowing the overall weight to take negative or positive values. As a result, a neuron that has learnt a pattern gets excited by spikes belonging to this pattern and inhibited by spikes not belonging to the pattern. Thus, its potential is maximal when the pattern is exactly reproduced, without missing or additional intermediate neuron spikes.

Fourth, the output neurons implement an intrinsic plasticity (IP) rule (Figure 5c), which enables them to adapt their threshold to the learnt pattern size. This ensures that an output neuron that has learnt to recognize a long pattern has a sufficiently high threshold preventing it to fire if the incoming pattern is incomplete, whereas a neuron that has learnt a short pattern has a low threshold and is still able to detect it. The introduction of this IP rule on the output neurons in conjunction with the use of

excitatory and inhibitory synapses is key to solve the problem of recognition of overlapping patterns or patterns included one into another.

The output neurons thresholds are initialized at a high value, and the feedforward synapses weights are initialized randomly, according to a uniform distribution, with an average positive weight. Before learning, when an output neuron receives a spike pattern from the intermediate layer, it fires late due to its high threshold. Once a neuron fires, its threshold begin to evolve according to the intrinsic plasticity rule (Figure 5c). Every time an output neuron emits a postsynaptic spike, its threshold  $Th$  is decreased by  $\Delta Th_{post} = F_{post}^{\Delta Th} * Th$ . For each presynaptic spike received within a coincidence window around the postsynaptic spike  $[-\tau_{IP+}, \tau_{IP-}]$ , the threshold is increased by  $\Delta Th_{pair}$  multiplied by the synaptic weight. Additionally, the threshold is bounded between a minimum and a maximum value (see Table 4). It can be shown that the equilibrium threshold value is proportional by a factor  $F_{post}^{\Delta Th} / \Delta Th_{pair}$  to the average weighted sum of the presynaptic spikes received within

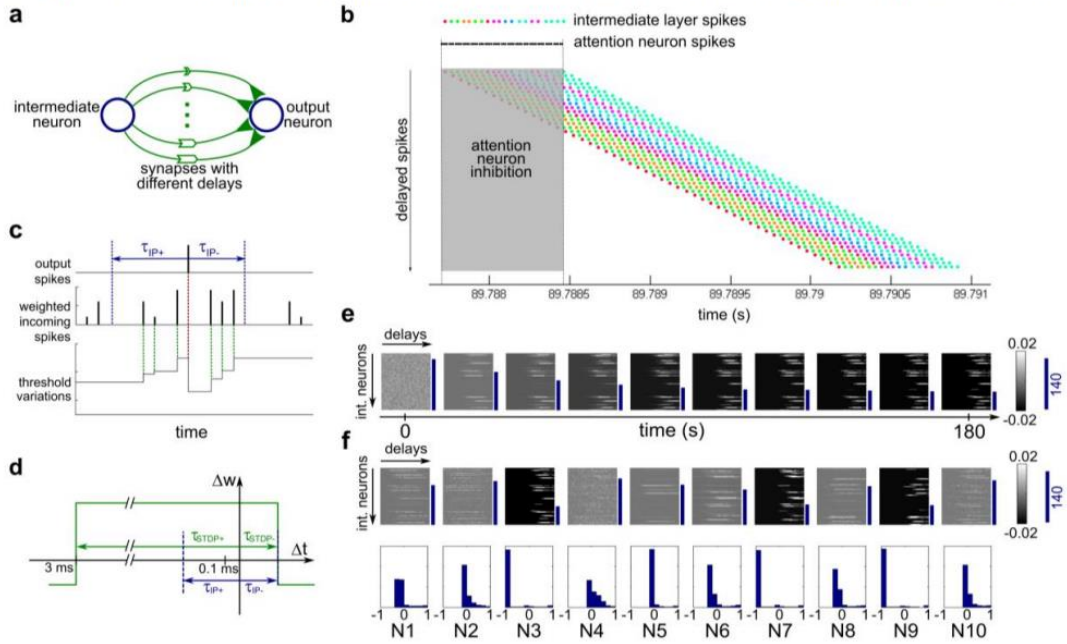


Fig. 5. The output layer learns to recognize different intermediate layer patterns. (a) Connection between intermediate and output neurons, through several synapses with different transmission delays (b) Spike train received by the output layer, depending on the intermediate layer and attention neuron spike train. The different colors stand for different intermediate neurons. (c) Intrinsic plasticity principle. (d) STDP rule applied on the synapses stemming from the intermediate layer. (e) Evolution of weights (in grayscale) and thresholds (blue bars) of one output neuron. The weights are organized according to their delays and their presynaptic intermediate neuron. Shown weights are the sum of the inhibitory and excitatory synapse weights divided by the neuron's threshold. (f) Synaptic weights and thresholds learned by each of the 10 output neurons. Top row: final synaptic weights and thresholds at the end of a 200-s simulation, represented similarly as in (e). Bottom row: distributions of final synaptic weights for each output neuron, without threshold normalization.



Table 4. Output layer parameters.

| Parameter               | Description   | Value                                |
|-------------------------|---|--------------------------------------|
| $N_{neur}$              | Number of neurons   | 10                                   |
| $\tau_m$                | Membrane time constant  | 3 ms                                 |
| $\tau_{refrac}$         | Refractory period   | 3 ms                                 |
| $F_{reset}$             | Reset potential factor. The reset potential is $V_{reset} = F_{reset} * Th$                           | -10                                  |
| $F_{inhib}$             | Lateral inhibition potential factor. The lateral inhibition potential is $V_{inhib} = F_{inhib} * Th$ | -10                                  |
| $N_{delays}$            | Number of synaptic delays   | 50                                   |
| $\Delta t_d$            | Time interval between synaptic delays   | $\Delta t_c = 0.05$ ms               |
| $w_{init+}^{pos}$       | Minimal value for the excitatory feedforward synapses weight initialization.                          | 0                                    |
| $w_{init+}^{pos}$       | Maximal value for the excitatory feedforward synapses weight initialization.                          | 1                                    |
| $w_{init-}^{neg}$       | Minimal value for the inhibitory feedforward synapses weight initialization.                          | 0                                    |
| $w_{init+}^{neg}$       | Maximal value for the inhibitory feedforward synapses weight initialization.                          | 0                                    |
| $\tau_{stdp+}$          | Positive STDP rule time window  | 3 ms                                 |
| $\tau_{stdp-}$          | Negative STDP rule time window  | $4 * \Delta t_c = 0.2$ ms            |
| $\Delta w_{pair}^{pos}$ | Excitatory synapses weight change for a presynaptic spike coinciding with a post synaptic spike       | 0.05                                 |
| $\Delta w_{post}^{pos}$ | Excitatory synapses weight change for each postsynaptic spike   | $-0.7 * \Delta w_{pair}^{pos}$       |
| $\Delta w_{pair}^{neg}$ | Inhibitory synapses weight change for a presynaptic spike coinciding with a post synaptic spike       | -0.05                                |
| $\Delta w_{post}^{neg}$ | Inhibitory synapses weight change for each postsynaptic spike   | $-0.1 * \Delta w_{pair}^{neg}$       |
| $Th_{min}$              | Lower bound of the neuron threshold.  | 8                                    |
| $Th_{max}$              | Upper bound of the neurons threshold, also used as initialization value                               | $0.35 * N_{delays} * Th_{min} = 140$ |
| $\tau_{ip+}$            | Positive IP rule time window  | $6 * \Delta t_c = 0.3$ ms            |
| $\tau_{ip-}$            | Negative IP rule time window  | $4 * \Delta t_c = 0.2$ ms            |
| $F^{\Delta Th}_{post}$  | Proportional threshold decrease for each postsynaptic spike   | 0.01                                 |
| $\Delta Th_{pair}$      | Threshold increase for each presynaptic spike coinciding with a postsynaptic spike                    | $0.6 * F^{\Delta Th}_{post}$         |

the coincidence window around a postsynaptic spike. As a consequence, the learning neuron's threshold, which is initialized at its maximum value, progressively decreases and the neuron fires earlier than the neurons that have not learnt any pattern (Figure 5e).

The synaptic weights evolve in parallel to the threshold according to an STDP rule (Figure 5d). The rule used is similar to the one used for the intermediate layer, with a different coincidence time window and different weight change values. The coincidence time window is defined by both  $\tau_{stdp+}$ , for presynaptic spikes occurring before a postsynaptic spike, and  $\tau_{stdp-}$ , for presynaptic spikes occurring after a postsynaptic spike.  $\tau_{stdp+}$  is chosen quite long to facilitate learning at the beginning, when the neuron fires late due to its high threshold. The inhibitory and excitatory feedforward synapses follow an STDP rule with the same coincidence time window, but with a different  $|\Delta w_{post}/\Delta w_{pair}|$  ratio, so that after learning, the summed weight converges to either 1, 0 or -1 depending on the ratio of the number of pre- and postsynaptic spike coincidences over the number of postsynaptic spike occurrences. Once an output neuron has learnt a pattern (see neurons 3, 7 and 9 in the example of Figure 5f), most of its incoming synapses have converged to the minimum negative

weight value. Only the synapses corresponding to the intermediate neurons and delays constituting the pattern converge to the maximum positive weight, and the few remaining synapses not relevant for discriminating the pattern converge to a null weight (Figure 5f). By contrast, the incoming synaptic weights of neurons that did not learn any pattern remain distributed near the zero value.

The output layer is constituted of 10 LIF neurons. The number of neurons is chosen higher than the maximum number of expected action potential waveforms in the signal. These neurons have their potential reset after firing or after a lateral WTA inhibition. The reset potential and the lateral inhibition potential are proportional to the neuron threshold, which vary according to the IP rule. The values of the output layer parameters are given in Table 4.

Overall, the proposed architecture of the network allows the output neurons to learn to recognize different patterns generated on the intermediate layer and to emit one spike for each occurrence of an action potential waveform in the input neural signal. Thus, each active output neuron directly predicts the activity of one real cell (Figure 6).

## 2.2. Simulated data

The spike-sorting methods were first tested on simulated data generated using a method adapted from the literature<sup>48</sup>. Simulated signals are the superposition of correlated noise and action potentials occurring at known time stamps thus providing a known ground truth. The simulated sampling frequency is 20 kHz. The waveform of each truth neuron action potential is defined according to the following template equation:

$$V(t) = A \cos\left(2\pi \frac{t - t_{ph}}{\tau_1}\right) \exp\left(-\left(\frac{2.3548t}{\tau_2}\right)^2\right) \quad (3)$$

The parameters of this equation are given in Table 5 for the 3 neurons that were simulated. The coefficient  $A$  is computed so that the maximum value of the template matches the parameter  $A_{max}$ . The time occurrences of these waveforms are defined according to a Poisson process, each with a mean firing rate of 3.3Hz (unless otherwise stated), with a simulation time step of 0.05 ms corresponding to a 20 kHz sampling frequency. Additionally, a 3-ms refractory period is applied for each neuron. The noise added to the signal is a correlated noise generated by a dynamical Ornstein-Uhlenbeck process, with a 0.1-ms time constant and a simulation step of 0.002 ms. Simulated signal were generated with different noise levels to achieve different signal-to-noise ratios defined as  $\langle |A_{max}| \rangle / \sigma_{noise}$ , where  $\langle |A_{max}| \rangle$  is the mean of the peak amplitudes of the 3 action potentials and  $\sigma_{noise}$  is the standard deviation of the noise. Seven different noise levels were defined (from 0.5 to 2) and ten 200-s datasets were generated for each of them.

Table 5. Waveform templates parameters.

|            | $A_{max}$ | $\tau_1$ (ms) | $\tau_2$ (ms) | $t_{ph}$ (ms) |
|------------|-----------|---------------|---------------|---------------|
| Waveform 1 | 5         | 1             | 0.5           | -0.25         |
| Waveform 2 | 5         | 1             | 0.5           | 0.25          |
| Waveform 3 | 10        | 1             | 0.5           | -0.19         |

## 2.3. Tetrode data

The spike sorting methods were also evaluated on real recordings from the hippocampus region CA1 of anesthetized rats, available from the Buszaki Laboratory<sup>49,50</sup> (datasets d533101 and d11221.002). Before sorting, the signals were band-pass-filtered using a first-order butterworth filter (300Hz – 3000Hz). The d533101 dataset, having an original 10-kHz sampling frequency, was up-sampled, for convenience, at 20 kHz

using a Wittaker-Shannon interpolation. However the signal was always resampled to 80 kHz at the input layer level (see Table 1).

## 2.4. Spike-sorting performance evaluation

To assess the performance of the spike-sorting method, we computed indices based on an F-score. For each pair of truth neuron  $i$  and output neuron  $j$ , we computed the F-score as:

$$F_{ij} = \frac{2 * H_{ij}}{T_i + O_j} \quad (4)$$

where  $T_i$  is the number of spikes emitted by the  $i$ th truth neuron,  $O_j$  is the number of spikes emitted by the  $j$ th output neuron, and  $H_{ij}$  is the number of output spikes coinciding with a truth spike within a 3-ms coincidence time window. Note that the F-score combines the recall and the precision of a prediction.

We also computed a global F-score across all truth neurons and all output neurons as:

$$F = \frac{2 * H}{T + O} \quad (5)$$

where  $T$  is the total number of truth spikes,  $O$  the total number of output spikes and  $H$  the total number of hits. To compute  $H$ , it is necessary to choose a correspondence between output neurons and truth neurons. Then  $H$  is the number of output spikes coinciding with a corresponding truth spike within a 3-ms coincidence time window. The pairing between output neurons and truth neurons was chosen to maximize  $H$ .

We also computed the recall as the number of correctly detected action potentials over the number of true action potentials, the precision as the number of correctly detected action potentials over the number of detected action potentials, and a clustering score as the number of correctly classified action potentials over the number of correctly detected action potentials.

## 2.5. Statistical tests

The STDP network was compared to two other spike-sorting methods, Wave\_clus<sup>28</sup> and Osort<sup>26</sup>, on both the simulated and real data. For the simulated data, ten 200-s datasets were generated for each of the seven different noise levels, and each of the three compared methods were run once on each dataset. For the tetrode data, the STDP method was run 8 times on each channel, Wave\_clus was run 8 times on each channel, and Osort

was run once on each channel as its result was deterministic. We then used the tetrode data to evaluate the performance of the STDP network when adapted to process all channels simultaneously. In this case, the STDP network was run 40 times both on each channel separately and on all channels simultaneously using two different tetrode network architectures (see Figure 8). Results obtained by the two tetrode methods were then compared to the best result obtained using only a single channel. As the variances were significantly different for the different groups (as assessed by a Bartlett test), a Welch test was used for 2-by-2 comparisons, except for the comparison with Osort on the tetrode data where a one-sample t-test was used. A Bonferroni correction was used for each set of multiple comparisons. All tests were performed using Matlab R2014a.

### 3. Results

#### 3.1. Performance of the network on simulated and real extracellular data

Figure 6 shows two examples of spike sorting results obtained on both artificial data with known ground truth and real extracellular tetrode recordings associated with

an intracellular recording providing ground truth for one cell<sup>49,50</sup>. Different types of error can occur when comparing the ground truth spike trains to the spike trains predicted by the sorting method: false negative, false positive and wrong classification. The performance of the method was assessed using the F-score, which accounts for all types of error. Figure 6a shows an example of 10 seconds of simulated data embedding 3 different action potential waveforms, each firing at known time stamps according to a Poisson distribution with an average rate of 3.3 Hz. As shown in this figure, the output spiking pattern of the STDP network showed 3 mainly active output neurons whose spike trains closely resembled those of the three embedded signal waveforms. The network quickly converged to provide high classification rates as assessed over the last 100 s by an overall F-score of 85%, with individual scores of 90, 89, and 93% for each of the waveforms. Figure 6b shows an example of 10 s of real extracellular data for which the activity of one neuron was known from a simultaneous intracellular recording. The output of the STDP network showed 3 main active neurons, the activity of one being close to that of the intracellularly identified cell. For this cell, the

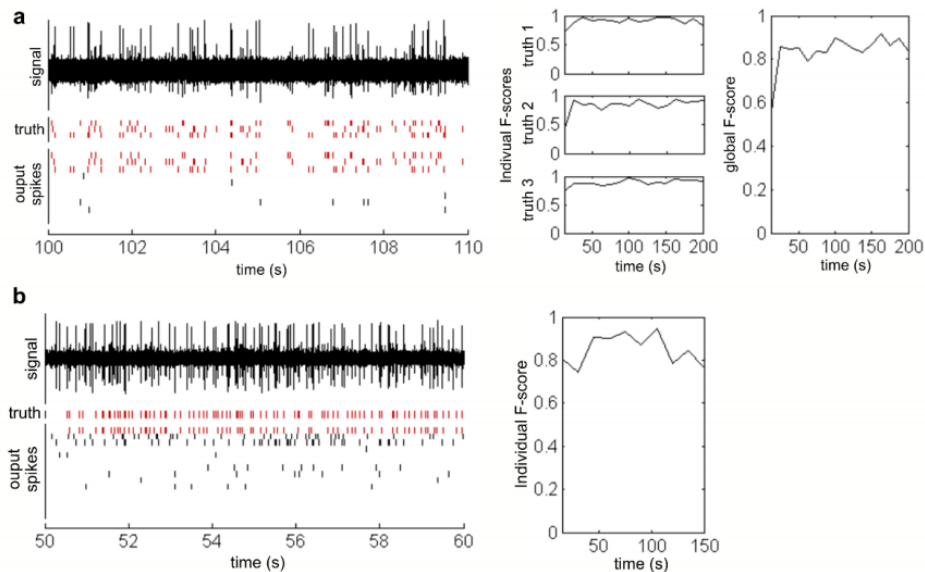


Fig. 6. Examples of spike-sorting achieved by the artificial network on simulated and real data. (a) Performance on simulated data. Left: comparison of the output spike trains generated by the ten output neurons, with the truth spike trains generated by the three simulated neurons, on a 10-s segment of input signal. Matching spike trains have been highlighted in red. Right: evolution of the performance of the network over the 200-s simulation. (b) Same as (a) for a real recording (dataset d553101). Here the truth is only known for one cell thus only the performance relative to this cell can be computed.



sorting F-score of the network was stable for 150 s with an average value of 89% on the last 100 s.

We further compared our method with two open-source software able to perform unsupervised spike-sorting: Osort<sup>26</sup> and Wave\_clus<sup>28</sup>. Figure 7a shows confusion results for the three methods on a simulated dataset. In this example, the STDP network made only a few false negative errors, while Wave\_clus made both false negative and false positive errors and Osort failed to detect one of the waveform and misclassified the two others in the same cluster, with an important number of false negative errors (Figure 7a). We further considered artificial data with varying signal-to-noise ratios (SNR). At very high SNRs, the STDP method gave slightly lower performance than Osort and Wave\_clus, but the F-score differences remained low ( $<0.06$ ). At lower SNRs more comparable to those encountered in real cortical recordings, the STDP method became more efficient than the two other methods with strong F-score differences up to 0.3 for an SNR of 4.4 (Figure 7b). These performance were further observed on real neural signals (Figure 7c). In this case, the STDP approach performed similarly or better than Wave\_clus in all cases and performed better than Osort at SNR lower than 4. The superior performance of the STDP approach at low SNR seems to be due to a better recall (with a fewer false negatives), and, more relevantly, a better clustering (Figure 7d).

### 3.2. Stability of the network

A spike-sorting method should generate robust classification irrespective of the level of activity of the cells. In particular, classification should be correct for highly active neurons as well as for poorly active neurons. We thus tested the STDP-based sorting method on artificial data where 3 neurons were simulated with different firing rates: 1Hz for one cell, 3Hz for another and 9Hz for the third cell (Figure 8a). The network successfully classified the three different neurons with an overall F1 score of 84%. Noticeably, the network learned faster the waveform of the most active cell and more slowly the waveform of the least active cell. We found that the network needed about 50 occurrences of a given waveform to reach a correct and stable classification. The proposed method is thus able to correctly classify different waveforms corresponding to neurons having different firing rates, as soon as each waveform has occurred enough times.

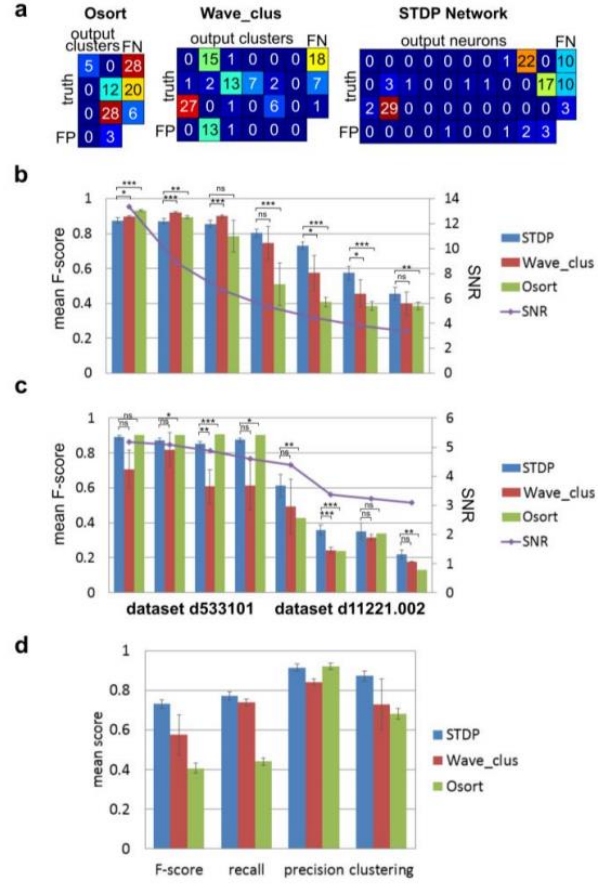


Fig. 7. Comparison of the performance of the STDP network with those of two other algorithms (Osort and Wave\_Clus). (a) Confusion matrices for the 3 sorting methods for a SNR=4.4 simulated dataset. The values are normalized relatively to 100 ground truth spikes i.e. 33 occurrences of each waveforms. (b) Mean F-score obtained on simulated data. (c) Mean F-score obtained on real tetrode recordings. “ns” stands for non-significant, \* for  $P<0.05$ , \*\* for  $P<0.01$ , and \*\*\* for  $P<0.001$ . (d) Detailed mean scores on the 10 simulated datasets with an SNR of 4.5.

A robust spike-sorting method then requires that a classification remains correct when the firing rates of the cells fluctuate. Indeed the activity of a neuron during a behavioral task can be subject to fluctuations depending on the involvement of the cell into the task. We thus also tested the STDP-based sorting method on artificial data where 3 neurons had intermittent firing activity (Figure 8b). All cells had a firing rate of 3 Hz but one became active only 50 s after the others and then these two other cells became silent one after the other before all became active again. We found that these firing rate variations

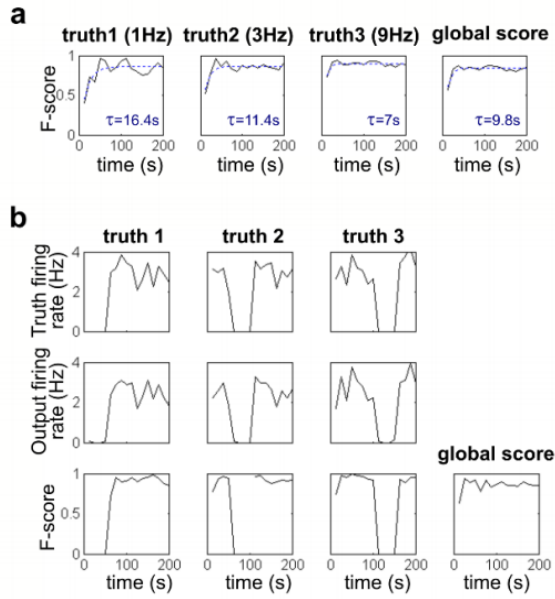


Fig. 8. Robustness of the network to different firing rates scenarios. (a) Performance of the network along time for ground truth neurons with different firing rates. Blue dashed line are an exponential fit (b) Evolution of the performance of the network for neurons with varying firing rates.

did not impair the algorithm performance (Figure 8b). In particular, after the network had stabilized learning of two waveforms, it could still learn a new one. Moreover, when a waveform that the network had learnt became transiently absent in the signal, it could still classify it immediately and successfully as soon as it reappeared, meaning that the network keeps memory of the waveforms that have been learned.

### 3.3. Extension of the network to tetrode recording

The recent advances of neural interfacing has benefited from novel micro and nanotechnologies allowing the fabrication of high-density multi-electrode systems<sup>51–54</sup>. Dense arrays of microelectrodes may provide neural recordings displaying partially overlapping information from one electrode site to the next. In particular, when recording sites are separated by less than a few tens of microns, action potentials from a same cell can be recorded on several sites and spike-sorting methods can benefit from combining neighboring electrodes instead of processing recording sites independently<sup>30,34,55,56</sup>. In this context, we thus adapted our method to the case of tetrode

recordings, processing all channels at once instead of separately. To do so, we considered two variations of our network architecture. The first one had four input layers processing in parallel each of the four input signals and projecting to one common attention neuron and one common intermediate layer (Figure 9a). In this configuration, the attention neuron's threshold, the intermediate neurons' threshold and the weights of the synapses linking the attention neuron and the intermediate layer were multiplied by four. The second configuration had four input layers, four attention neurons and four intermediate layers working in parallel. The four intermediate layers and attention neurons project to one common output layer (Figure 9b). In this case the network parameters were unchanged. On the two datasets tested, we found that combining the four signals with the first configuration gave better classification than

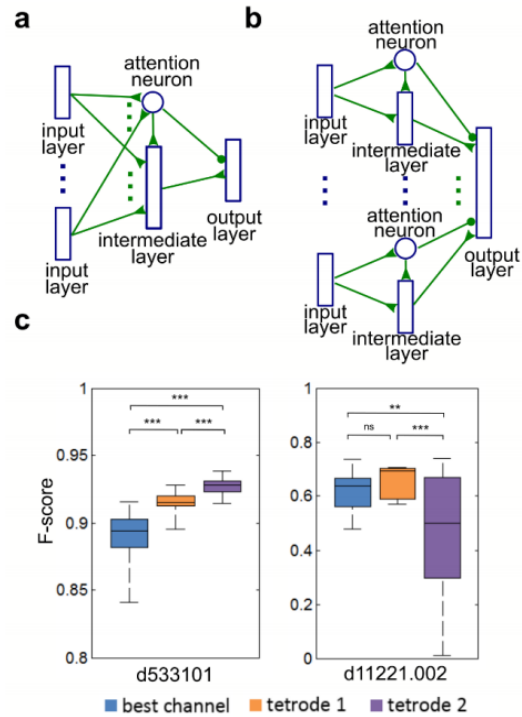


Fig. 9. Adaptation of the network for the processing of tetrode data. (a) First network adaptation to tetrodes (tetrode 1) where one input layer is used independently for each recording site and the intermediate and output layers are further shared. (b) Second adaptation (tetrode 2) where one input and one intermediate layers are used independently for each recording site and only the output layer is further shared. (c) Performance of the two different tetrode architectures compared with the best single electrode performance. “ns” stands for non-significant, \* for  $P<0.05$ , \*\* for  $P<0.01$ , and \*\*\* for  $P<0.001$ .



using each electrode separately. The second configuration, requiring a higher number of neurons and thus more computations, was only better for the first dataset having a higher SNR (Figure 9c).

#### 4. Conclusion

We show that a 3-layer STDP network with an attention-based mechanism is successful on a spike-sorting task. The network needs only few data to configure in a fully unsupervised manner. Moreover, the parameters of the network did not need to be adjusted between the different tested datasets. Indeed, only the noise level of the input data needs to be known to parameter the network correctly (see section 2.1.1). In a preliminary study<sup>57</sup>, extracellular action potentials could be successfully sorted in case of a very high SNR using a single-layer STDP network but this architecture failed at SNR corresponding to standard cortical recordings. Here, we propose a multi-layer network architecture incorporating several plasticity rules, which shows performances comparable to existing methods with the advantage to process the stream of neural signal continuously. In contrast to conventional spike-sorting algorithms that use separate steps for action potential detection, feature extraction and clustering, here the entire process is considered as a whole. The neural network directly processes the raw signal to output time stamps of sorted action potentials.

#### 5. Discussion

Bio-inspired computing using artificial spiking neural networks promises performances outperforming currently available computational approaches. Yet, the number of applications of such networks remains limited due to the lack of generic training method for complex pattern recognition. This study shows encouraging results on the application of STDP networks to spike-sorting, with classification results even better than those obtained with Osort and Wave\_clus software. This latter finding yet only applies to the datasets tested in this study and would require more extensive testing on numerous situations to be confirmed. This study proposes a radically new approach to handle spike-sorting offering perspective on the long term for very low power embedding of this central neural signal processing step in brain implants. From the point of view of unsupervised pattern recognition using STDP networks, these results

also represent significant advances with respect to previous achievements obtained so far. STDP networks have been shown to be successful for unsupervised learning<sup>13</sup> and pattern recognition from sets of static images such as digits<sup>17</sup> or objects or faces<sup>14</sup>, with performances improved by considering deep architectures<sup>58</sup>. When considering dynamical inputs such as sounds or videos, pattern recognition using STDP networks has so far been restricted to cases where the time of occurrence of the pattern fragments across the different input units did not need to be taken into account<sup>15</sup>. Indeed, a LIF neuron combined with STDP can act as a coincidence detector, which is not adequate to discriminate between patterns differing by the time ordering of their fragments across the input units. Here we solved this problem using a set of delaying synapses, to take into account time ordering. In our case, two layers with delays were used, but this mechanism could be extended to more layers, the length of the recognized time-series going up in scale at each layer. Another difficulty which had to be overcome to solve the problem of spike-sorting is that the pattern processed by the output layer might vary in size. As different patterns can share common parts, this can lead in the worst case to patterns including one into another. This situation was handled by using an intrinsic plasticity rule which allows the neuron threshold to adapt to the size of the learnt pattern, combined with synapses that can have negative as well as positive weights, which ensures that the maximum excitation is reached when the presented pattern exactly matches the learnt pattern. It has been shown that STDP network can recognize spike patterns occurring sparsely in a random spike background<sup>12</sup>. However, here, we face a different problem as the background signal fluctuates around zero and thus gets encoded as consistent spike trains that could be learnt by the network. This problem was handled with an attention mechanism, which takes advantage of the fact that the relevant patterns occur only sparsely, whereas background patterns that should be ignored occur repeatedly. This attention mechanism thus detects the relevant parts of the signal and then gates the rest of the network to only learn from these relevant periods.

This approach has been designed to be compatible with a neuromorphic implementation, which opens perspectives for low-power real time processing, thanks to the development of neuromorphic chips. Real-time processing requires fast computations, especially when

hundreds of recording sites need to be processed simultaneously. Artificial STDP neural networks have been implemented on FPGAs to demonstrate embedding neuromorphic computing<sup>59–62</sup>. However, although the real-time transposition of artificial STDP spike-sorting networks could be envisioned with GPUs or FPGAs, such strategies would still require high power consumptions not compatible with future embedding in implantable devices. Future brain implants embedding spike-sorting at the electrode level will thus need to rely on other types of very low power implementations. The approach proposed here was designed specifically in such perspective. It complies with neuromorphic circuits and in particular with scalable nonvolatile resistive memory (memristive) devices that can mimic artificial synapses with embedded STDP plasticity and offer the perspective of very low power implementation of spiking neural networks in analog hardware for pattern recognition<sup>4–10,35,63,64</sup>. Indeed, synapses are several orders of magnitude more numerous than neurons in neural networks. Thus, low-power computing based on artificial neural networks should be conceived to be compatible with low-power synapses. Here, the proposed method relies on two types of plasticity rules at the synaptic level, STP and STDP. Both have been modeled into RRAM synapses, making the method compliant with currently developed nanoscale neuromorphic hardware<sup>10,15,65,66</sup>. The more power-consuming part of our network is the synaptic connection between the input layer and the intermediate layer with  $8 \times 10^6$  spikes per second generated by the input layer, each transmitted through 60 synapses. With a raw estimation of about 200 pJ of energy consumed for each weight change<sup>57,67</sup>, this lead to a total power of less than 1 W. This however remains a very rough estimation as, depending on the technology used, the switching power of a synapse can be decreased under 1 pJ<sup>68,69</sup>, which would result in a power consumption of the network of about 5 mW, hence 1 W for 200 microelectrodes. This approach thus opens new perspectives to achieve very power-efficient spike-sorting in miniaturized analog neuromorphic circuits, which should benefit to future fully implantable electronics and intelligent and standalone autonomous implants for seamless neural function monitoring and neurorehabilitation.

The present study may further open new ways to build large-scale hybrid neural networks. Hybrid networks connect real neurons to artificial neurons, ideally in a

one-to-one bidirectional scheme. This field has been pioneered by dynamic clamp experiments at the single cell level using the patch-clamp technique<sup>70</sup>. Microelectrode arrays now open the way to build larger hybrid networks connecting multiple living cells to multiple artificial units. Achieving a complete bidirectional hybridization requires two types of transformations: converting real neural network activity into artificial neural activity and, conversely, converting artificial activity into real activity. Artificial-to-real conversion has been achieved using electrical microstimulation at a network level<sup>71</sup> but remains unachieved at the level of one-to-one artificial-real neuron pairs. Here, the spike-sorting STDP network offers a simple solution for a real-to-artificial one-to-one conversion of the spike trains of multiple biological single units into spike trains of an equivalent number of artificial spiking neurons. Such direct real-to-artificial conversion should thus ease the exploitation of neural activity by downstream neuromorphic architectures that have been proposed for neural signal decoding in rehabilitation devices<sup>72</sup>.

#### Data availability

The simulated data used in this work is available on zenodo at doi 10.5281/zenodo.888977. The code used to simulate the STDP network is available on zenodo at doi 10.5281/zenodo.2248525.

#### Acknowledgements

This work was supported in part by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 732032 (BrainCom), by the Fondation pour la Recherche Médicale (FRM) under Grant agreement No DBS20140930785, and by the French National Research Agency under Grant Agreement No. ANR-16-CE19-0005-01 (Brainspeak). The authors wish to thank Elisa Vianello and Jean-François Bêche for helpful discussions, and George Malliaras for his feedback on the manuscript.

#### References

1. Ghosh-dastidar, S. & Adeli, H. Spiking Neural Networks. *Int. J. Neural Syst.* **19**, (2009).
2. Maass, W. Networks of spiking neurons: the third generation of neural network models. *Neural networks* **10**, 1659–1671 (1997).
3. Wu, T., Bilbîe, F.-D., Păun, A., Pan, L. & Neri, F. Simplified and Yet Turing Universal Spiking Neural



- P Systems with Communication on Request. *Int. J. Neural Syst.* (2018). doi:10.1142/S0129065718500132
4. Jo, S. H. *et al.* Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Lett.* **10**, 1297–1301 (2010).
  5. Indiveri, G., Linares-Barranco, B., Legenstein, R., Deligeorgis, G. & Prodromakis, T. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology* **24**, (2013).
  6. Indiveri, G., Corradi, F. & Qiao, N. Neuromorphic architectures for spiking deep neural networks. in *Technical Digest - International Electron Devices Meeting, IEDM 2016-Febru*, 4.2.1–4.2.4 (2015).
  7. Park, S. *et al.* Electronic system with memristive synapses for pattern recognition. *Sci. Rep.* **5**, 10123 (2015).
  8. Saïghi, S. *et al.* Plasticity in memristive devices for spiking neural networks. *Front. Neurosci.* **9**, 1–16 (2015).
  9. Rajendran, B. & Alibart, F. Neuromorphic Computing Based on Emerging Memory Technologies. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **6**, 198–211 (2016).
  10. La Barbera, S., Vincent, A. F., Vuillaume, D., Querlioz, D. & Alibart, F. Interplay of multiple synaptic plasticity features in filamentary memristive devices for neuromorphic computing. *Sci. Rep.* **6**, 39216 (2016).
  11. Sourikopoulos, I. *et al.* A 4-fJ/spike artificial neuron in 65 nm CMOS technology. *Front. Neurosci.* **11**, 1–14 (2017).
  12. Masquelier, T., Guyonneau, R. & Thorpe, S. J. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS One* **3**, (2008).
  13. Masquelier, T., Guyonneau, R. & Thorpe, S. J. Competitive STDP-based spike pattern learning. *Neural Comput.* **21**, 1259–1276 (2009).
  14. Masquelier, T. & Thorpe, S. J. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Comput. Biol.* **3**, 0247–0257 (2007).
  15. Suri, M. *et al.* Bio-inspired stochastic computing using binary CBRAM synapses. *IEEE Trans. Electron Devices* **60**, 2402–2409 (2013).
  16. Srinivasa, N., Cho, Y. & Hennequin, G. Unsupervised discrimination of patterns in spiking neural networks with excitatory and inhibitory synaptic plasticity. **8**, 1–23 (2014).
  17. Diehl, P. & Cook, M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* **9**, 99 (2015).
  18. Antonietti, A., Monaco, J., D’Angelo, E., Pedrocchi, A. & Casellato, C. Dynamic Redistribution of Plasticity in a Cerebellar Spiking Neural Network Reproducing an Associative Learning Task Perturbed by TMS. *Int. J. Neural Syst.* **28**, 1850020 (2018).
  19. Antunes, G., Faria da Silva, S. F. & Simoes de Souza, F. M. Mirror Neurons Modeled Through Spike-Timing-Dependent Plasticity are Affected by Channelopathies Associated with Autism Spectrum Disorder. *Int. J. Neural Syst.* **28**, 1750058 (2018).
  20. Geminiani, A., Casellato, C., Antonietti, A., D’Angelo, E. & Pedrocchi, A. A Multiple-Plasticity Spiking Neural Network Embedded in a Closed-Loop Control System to Model Cerebellar Pathologies. *Int. J. Neural Syst.* **28**, 1750017 (2018).
  21. Naro, A., Bramanti, A., Leo, A., Bramanti, P. & Calabrò, R. S. Metaplasticity: A Promising Tool to Disentangle Chronic Disorders of Consciousness Differential Diagnosis. *Int. J. Neural Syst.* **28**, 1750059 (2017).
  22. Zhang, X., Foderaro, G., Henriquez, C. & Ferrari, S. A Scalable Weight-Free Learning Algorithm for Regulatory Control of Cell Activity in Spiking Neuronal Networks. *Int. J. Neural Syst.* **28**, 1750015 (2018).
  23. Bohte, S. M., Kok, J. N. & La Poutré, H. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* **48**, 17–37 (2002).
  24. Mohammed, A., Schliebs, S., Matsuda, S. & Kasabov, N. SPAN: Spike Pattern Association Neuron for Learning Spatio-Temporal Spike Patterns. *Int. J. Neural Syst.* **22**, 1250012 (2012).
  25. Guo, L., Wang, Z., Cabrerizo, M. & Adjouadi, M. A Cross-Correlated Delay Shift Supervised Learning Method for Spiking Neurons with Application to Interictal Spike Detection in Epilepsy. *Int. J. Neural Syst.* **27**, 1750002 (2017).
  26. Rutishauser, U., Schuman, E. M. & Mamelak, A. N. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *J. Neurosci. Methods* **154**, 204–224 (2006).
  27. Delescluse, M. & Pouzat, C. Efficient spike-sorting of multi-state neurons using inter-spike intervals information. **150**, 16–29 (2006).
  28. Quiroga, R. Q., Nadasdy, Z. & Ben-Shaul, Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Comput.* **16**, 1661–1687 (2004).
  29. Shoham, S., Fellows, M. R. & Normann, R. a. Robust, automatic spike sorting using mixtures of multivariate t-distributions. *J. Neurosci. Methods* **127**, 111–122 (2003).
  30. Rossant, C. *et al.* Spike sorting for large, dense electrode arrays. *Nat. Neurosci.* **19**, 634–641 (2016).
  31. Hilgen, G. *et al.* Unsupervised Spike Sorting for Large-Scale, High-Density Multielectrode Arrays. *Cell Rep.* **18**, 2521–2532 (2017).
  32. Chah, E. *et al.* Automated spike sorting algorithm based on Laplacian eigenmaps and k-means clustering. *J. Neural Eng.* **8**, 016006 (2011).
  33. Oliynyk, A., Bonifazzi, C., Montani, F. & Fadiga, L. Automatic online spike sorting with singular value decomposition and fuzzy C-mean clustering. *BMC Neurosci.* **13**, 96 (2012).
  34. Marre, O. *et al.* Mapping a Complete Neural Population in the Retina. *J. Neurosci.* **32**, 14859–14873 (2012).
  35. Zhang, B., Jiang, Z., Wang, Q., Seo, J. & Seok, M. A



- neuromorphic neural spike clustering processor for deep-brain sensing and stimulation systems. in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)* 91–97 (IEEE, 2015). doi:10.1109/ISLPED.2015.7273496
36. Chung, J. E. *et al.* A Fully Automated Approach to Spike Sorting. *Neuron* **95**, 1381–1394.e6 (2017).
37. Seidl, K. *et al.* CMOS-based high-density silicon microprobe arrays for electronic depth control in intracortical neural recording-characterization and application. *J. Microelectromechanical Syst.* **21**, 1426–1435 (2012).
38. Alivisatos, A. P. *et al.* The Brain Activity Map. *Science* (80-. ). **339**, 1284–1285 (2013).
39. Angotzi, G. N., Malerba, M., Zucca, S. & Berdondini, L. A 512-channels, whole array readout, CMOS implantable probe for acute recordings from the brain. in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* 877–880 (IEEE, 2015). doi:10.1109/EMBC.2015.7318502
40. Pothof, F. *et al.* Chronic neural probe for simultaneous recording of single-unit , multi-unit , and local field potential activity from multiple brain sites. *J. Neural Eng.* **13**, (2016).
41. Rios, G., Lubenov, E. V., Chi, D., Roukes, M. L. & Siapas, A. G. Nanofabricated Neural Probes for Dense 3-D Recordings of Brain Activity. *Nano Lett.* **16**, 6857–6862 (2016).
42. Lopez, C. M. *et al.* 22.7 A 966-electrode neural probe with 384 configurable channels in 0.13 $\mu$ m SOI CMOS. in *2016 IEEE International Solid-State Circuits Conference (ISSCC)* 392–393 (IEEE, 2016). doi:10.1109/ISSCC.2016.7418072
43. Lee, J. *et al.* YASS: Yet Another Spike Sorter. *bioRxiv* 1–24 (2017). doi:10.1101/151928
44. Abbott, L. F., Varela, J. A., Sen, K. & Nelson, S. B. Synaptic Depression and Cortical Gain Control. *Science* (80-. ). **275**, 220–223 (1997).
45. Natschläger, T. & Ruf, B. Spatial and temporal pattern analysis via spiking neurons. *Netw. Comput. Neural Syst.* **9**, 319–332 (1998).
46. Ghosh-Dastidar, S. & Adeli, H. Improved Spiking Neural Networks for EEG Classification and Epilepsy and Seizure Detection. *Integr. Comput. Aided. Eng.* **14**, 187–212 (2007).
47. Ghosh-Dastidar, S. & Adeli, H. A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks* **22**, 1419–1431 (2009).
48. Adamos, D. A., Kosmidis, E. K. & Theophilidis, G. Performance evaluation of PCA-based spike sorting algorithms. *Comput. Methods Programs Biomed.* **91**, 232–244 (2008).
49. Henze, D. A. *et al.* Intracellular features predicted by extracellular recordings in the hippocampus in vivo. *J. Neurophysiol.* **84**, 390–400 (2000).
50. Henze, D. *et al.* Simultaneous intracellular and extracellular recordings from hippocampus region CA1 of anesthetized rats. (2009). doi:http://dx.doi.org/10.6080/K02Z13FP
51. Berdondini, L. *et al.* Extracellular recordings from locally dense microelectrode arrays coupled to dissociated cortical cultures. *J. Neurosci. Methods* **177**, 386–396 (2009).
52. Charvet, G. *et al.* BioMEA: A versatile high-density 3D microelectrode array system using integrated electronics. *Biosens. Bioelectron.* **25**, 1889–1896 (2010).
53. Viventi, J. *et al.* Flexible, foldable, actively multiplexed, high-density electrode array for mapping brain activity in vivo. *Nat. Neurosci.* **14**, 1599–1605 (2011).
54. Khodagholy, D. *et al.* NeuroGrid: recording action potentials from the surface of the brain. *Nat. Neurosci.* **18**, 310–315 (2014).
55. McNaughton, B. L., O’Keefe, J. & Barnes, C. A. The stereotrode: A new technique for simultaneous isolation of several single units in the central nervous system from multiple unit records. *J. Neurosci. Methods* **8**, 391–397 (1983).
56. Gray, C. M., Maldonado, P. E., Wilson, M. & McNaughton, B. Tetorodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. *J. Neurosci. Methods* **63**, 43–54 (1995).
57. Werner, T. *et al.* Spiking Neural Networks Based on OxRAM Synapses for Real-Time Unsupervised Spike Sorting. *Front. Neurosci.* **10**, 474 (2016).
58. Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J. & Masquelier, T. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks* **99**, 56–67 (2018).
59. Li, J., Katori, Y. & Kohno, T. An FPGA-Based Silicon Neuronal Network with Selectable Excitability Silicon Neurons. *Front. Neurosci.* **6**, 183 (2012).
60. Ambroise, M., Levi, T., Joucla, S., Yvert, B. & Saïghi, S. Real-time biomimetic central pattern generators in an FPGA for hybrid experiments. *Front. Neurosci.* **7**, 1–11 (2013).
61. Maguire, L. P. *et al.* Challenges for large-scale implementations of spiking neural networks on FPGAs. *Neurocomputing* **71**, 13–29 (2007).
62. Rossello, J. L., Canals, V., Morro, A. & Oliver, A. Hardware implementation of stochastic spiking neural networks. *Int. J. Neural Syst.* **22**, 1250014 (2012).
63. Yao, P. *et al.* Face classification using electronic synapses. *Nat. Commun.* **8**, (2017).
64. Al-Shedivat, M., Naous, R., Cauwenberghs, G. & Salama, K. N. Memristors empower spiking neurons with stochasticity. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **5**, 242–253 (2015).
65. Ohno, T. *et al.* Short-term plasticity and long-term potentiation mimicked in single inorganic synapses. *Nat. Mater.* **10**, 591–595 (2011).
66. Kuzum, D., Jeyasingh, R. G. D., Lee, B. & Wong, H. S. P. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Lett.* **12**, 2179–2186 (2012).
67. Bichler, O. *et al.* Visual pattern extraction using

- energy-efficient ‘2-PCM synapse’ neuromorphic architecture. *IEEE Trans. Electron Devices* **59**, 2206–2214 (2012).
68. Xiong, F., Liao, A. D., Estrada, D. & Pop, E. Low-power switching of phase-change materials with carbon nanotube electrodes. *Science (80-. )*. **332**, 568–570 (2011).
  69. Chin, A., Cheng, C. H., Chiu, Y. C., Zheng, Z. W. & Liu, M. Ultra-Low Switching Power RRAM Using Hopping Conduction Mechanism. *ECS Trans.* **50**, 3–8 (2013).
  70. Sharp, a a, O’Neil, M. B., Abbott, L. F. & Marder, E. Dynamic clamp: computer-generated conductances in real neurons. *J. Neurophysiol.* **69**, 992–995 (1993).
  71. Joucla, S. *et al.* Generation of locomotor-like activity in the isolated rat spinal cord using intraspinal electrical microstimulation driven by a digital neuromorphic CPG. *Front. Neurosci.* **10**, 1–9 (2016).
  72. Dethier, J. *et al.* A Brain-Machine Interface Operating with a Real-Time Spiking Neural Network Control Algorithm. *Adv. Neural Inf. Process. Syst.* **2011**, 2213–2221 (2011).



## IX. RÉSUMÉ EN FRANÇAIS

### A. Introduction

Enregistrer le cerveau est important à la fois pour en étudier le fonctionnement, mais également pour des applications telles que les interfaces cerveau-machine. Les microélectrodes extracellulaires permettent l'enregistrement de l'activité individuelle des cellules neurales. Avec les évolutions techniques, les matrices d'électrodes comportent de plus en plus de sites, permettant d'enregistrer de nombreux neurones simultanément. Cela ouvre beaucoup d'opportunités mais nécessite des algorithmes adaptés, notamment pour les interfaces cerveau machine qui nécessitent un traitement en temps réel des données. En particulier, pour les enregistrements extracellulaires, il est préférable de trier les potentiels d'action émis par différents neurones et enregistrés sur une même électrode, une opération appelée spike-sorting. De nombreuses méthodes existent, mais peu permettent un traitement en temps réel des données. Ce travail de thèse est focalisé sur le développement d'une méthode radicalement nouvelle, utilisant un réseau de neurones artificiel « spike-timing-dependent plasticity » (STDP). Ce type de réseau, encore peu utilisé pour des tâches de reconnaissance, a des propriétés d'apprentissage non-supervisé intéressantes pour le spike-sorting. Un tel algorithme pourrait être implémenté dans des puces neuromorphiques très basse consommation, qui connaissent aujourd'hui d'importants développements.

### B. Etat de l'art des méthodes de spike-sorting

Une microélectrode extracellulaire enregistre les potentiels d'action émis par des neurones proches. La forme de ces potentiels d'action diffère selon la position du neurone qui l'émet, ce qui permet de les trier par une méthode de spike-sorting.

La plupart des méthodes se décomposent en trois étapes principales : la détection des potentiels d'action, l'extraction de traits caractéristiques de leur forme, puis leur classification en groupes qui correspondent alors aux différents neurones enregistrés. La détection se fait par un seuillage, soit directement sur le signal, soit après un prétraitement basé par exemple sur un calcul d'énergie, l'application d'ondelettes ou l'utilisation d'un gabarit de potentiel d'action. La forme du potentiel d'action détecté peut alors être directement utilisée pour la classification, mais la plupart des méthodes en extraient quelques caractéristiques, afin de réduire le nombre de dimensions à traiter. Ces caractéristiques peuvent être prédéfinies, telles que l'amplitude ou la largeur du potentiel d'action ou des coefficients d'ondelette, ou bien déterminées automatiquement par un algorithme de réduction de dimension tel que l'analyse en composantes principales (ACP). La dernière étape consiste à classer les vecteurs de caractéristiques obtenus dans différents groupes. Parmi les algorithmes de regroupement utilisés on peut citer l'« expectation-maximization », le « K-mean », le « mean-shift », le « superparamagnetic clustering ». Une étude a notamment utilisé un réseau STDP monocouche pour cette étape de classification.

D'autres méthodes utilisent une approche plus globale, en modélisant le signal enregistré comme la somme de potentiel d'actions. C'est le cas des méthodes dites de « template matching ». Différents gabarits de potentiels d'action sont déterminés lors d'une étape de prétraitement semblable aux

méthodes décrites précédemment. Ces gabarits sont ensuite utilisés pour détecter et classer les potentiels d'action présents dans le signal.

La plupart des méthodes en trois étapes nécessitent un traitement offline, notamment à cause de l'étape de classification, qui requiert souvent de connaître au préalable tous les points à classer. Les méthodes par « template matching » sont plus adaptées à une exécution online. En effet, bien que l'étape de détermination des gabarits se fasse souvent offline, une fois les gabarits connus ces méthodes traitent le signal localement, permettant donc un traitement online. Un autre aspect important pour une exécution temps-réel est le temps de calcul, qui peut s'avérer limitant, en particulier pour l'étape de classification.

Lors de l'utilisation de matrices d'électrodes denses, les potentiels d'action d'une même cellule peuvent être enregistrés sur plusieurs électrodes. Cela apporte plus d'information pour le tri, à condition d'exploiter correctement l'aspect spatial des potentiels d'action. L'étape de détection doit déterminer non seulement à quel moment a lieu un potentiel d'action mais également sur quelles électrodes. L'étape de classification doit tenir compte de cette position, soit par un algorithme adapté, soit en prenant cette position comme caractéristique. Les méthodes par « template matching » ne nécessitent pas forcément d'adaptation, les gabarits étant nuls sur les électrodes où le potentiel d'action n'est pas visible.

Quelques difficultés connues peuvent surgir lorsque que l'on cherche à trier des potentiels d'action. Premièrement une cellule peut décharger en « burst », et les potentiels d'action générés diminuent alors en amplitude. Il faut donc modéliser cette diminution ou utiliser des caractéristiques indépendantes de l'amplitude. Deuxièmement, lors d'enregistrements longs, la forme des potentiels d'action peut changer au fil du temps. Une façon de pallier ce problème est de traiter le signal par morceaux suffisamment courts, dont il faudra ensuite lier les résultats. Enfin, un des problèmes les plus difficiles à résoudre est la superposition temporelle de potentiels d'action. Seules les méthodes modélisant la sommation de potentiels d'action sont robustes à ce problème.

De nombreuses implémentations de méthodes de spike-sorting existent, mais très peu permettent pour le moment le traitement en temps réel de données de matrices électrodes dense.

## C. Etat de l'art des réseaux de neurones STDP

Contrairement aux réseaux de neurones formels où les neurones sont statiques et donnent en sortie une valeur numérique, les réseaux STDP, inspirés de la réalité biologique, sont constitués de neurones ayant une dynamique temporelle et générant des trains de spikes. Une autre différence importante est que les réseaux STDP utilisent des lois d'apprentissage locales.

Dans ces réseaux STDP, l'information est donc véhiculée par des trains de spikes et peut être interprétée de différentes manières. L'encodage par taux de décharge considère que l'information est portée par le taux de décharge moyen des neurones. Cet encodage permet une formalisation plus simple mais induit une perte d'information et ne permet pas d'expliquer la rapidité d'exécution de certaines tâches par le cerveau. À l'inverse, avec l'encodage par impulsions, l'information est portée par les temps d'émission de chaque spike. Cette information temporelle peut être utilisée de différentes façons selon que l'information importante se trouve dans l'identité des neurones déchargeant en premier ou dans la synchronisation de différents neurones.

Les neurones utilisés dans les réseaux STDP peuvent avoir différents types de comportements tel que décharger de manière persistante pendant un stimulus ou seulement au début du stimulus, décharger par rebond après un stimulus inhibiteur, ou être sensibles seulement aux changements brusques dans un stimulus. On distingue deux grands types de modèles : les modèles où toute la dynamique du potentiel interne au neurone est mise en équation, ce qui peut être couteux en temps de calcul, et les modèles où les spikes sont modélisés comme des événements discrets déclenchés par un dépassement de seuil et seule la dynamique en dessous de ce seuil est mise en équation. Le choix d'un modèle dépend du comportement que l'on souhaite reproduire, avec quel degré de réalisme et quel coût calculatoire.

Si les modèles de neurones sont variés, les lois de plasticité synaptique le sont aussi. La loi STDP la plus connue consiste en un changement persistant du poids de la synapse déclenché par l'occurrence d'un spike postsynaptique et d'un spike présynaptique. L'amplitude du changement dépend alors de l'intervalle de temps entre ces deux spikes. Les observations expérimentales montrent que le poids augmente pour un spike présynaptique précédant un spike postsynaptique et diminue pour un spike postsynaptique précédant un spike présynaptique. De plus l'amplitude du changement diminue avec la longueur de l'intervalle de temps. Cependant, d'autres types de dépendances ont été observés et utilisés dans des modèles. Les lois STDP basées sur des paires de spikes ne parviennent pas à expliquer toutes les observations. Aussi, d'autres modèles de plasticité ont été développés, prenant en compte plus de deux spikes. Il existe aussi des plasticités à court terme, pour lesquels les changements de poids ne sont pas persistants. Elles sont souvent induites par les spikes présynaptiques seulement, et se traduisent par une augmentation (facilitation) ou une diminution (dépression) des poids après plusieurs spikes présynaptiques consécutifs rapprochés. La plasticité à court terme permet une régulation du potentiel postsynaptique. Les lois STDP induisant un renforcement positif, il peut être difficile de les paramétrer pour être à la fois stables et discriminatives. Les mécanismes d'homéostasie, intervenant à des échelles de temps différentes et sur plusieurs synapses, permettent une meilleure stabilité. Ces mécanismes interviennent soit directement sur le poids des synapses, par exemple en assurant une normalisation de la somme des poids, soit sur l'excitabilité du neurone postsynaptique (on parle alors de plasticité intrinsèque), soit en modulant les paramètres de plasticité synaptique (on parle alors de méta-plasticité).

Dans un réseau, les neurones et les synapses obéissent à des règles simples mais leurs interactions sont complexes. Les études de réseaux aléatoires à poids synaptiques fixes montrent plusieurs dynamiques possibles, dont le régime asynchrone où chaque neurone décharge indépendamment selon une loi de Poisson. Ce régime ne peut être obtenu qu'avec un équilibre entre l'excitation et l'inhibition. D'autres études se focalisent sur l'évolution du poids des synapses selon différentes hypothèses de corrélation entre l'activité des neurones postsynaptiques et présynaptiques. Il a par exemple été montré que l'on peut obtenir un équilibre entre l'excitation et l'inhibition par une loi STDP symétrique, ce qui améliore les performances lors de tâches de reconnaissance de motifs.

Bien qu'il n'existe pas de méthode universelle pour entraîner un réseau STDP à effectuer une tâche de reconnaissance, il a été montré que ceux-ci ont des propriétés d'apprentissage non-supervisé. On trouve des exemples d'application à la reconnaissance de motifs visuels ou auditifs.

## D. Modèle de réseau développé dans la thèse

Le réseau STDP qui a été développé est un réseau « feedforward », constitué de trois couches de neurones, ayant chacune un rôle spécifique, ainsi que d'un mécanisme d'attention. Cette structure, construite pour traiter un signal mono-électrode, a ensuite été adaptée pour traiter un signal provenant d'électrodes multiples.

La première couche, dite couche d'entrée, a pour but de transformer le signal d'entrée en train de spikes. Chaque neurone de cette couche agit comme un neurone sensoriel sensible à une certaine plage de valeur. Il décharge à intervalles réguliers lorsque le signal est dans sa plage de sensibilité. Les neurones d'entrée ont différentes plages de sensibilité et prennent en compte la valeur du signal avec différents délais, ce qui permet d'encoder à chaque instant la forme du signal dans une fenêtre de temps de 0,5 ms.

Les potentiels d'action ne sont présents que de manière ponctuelle dans le signal. Un mécanisme d'attention a été implémenté afin que le réseau ne traite que les parties du signal comportant un potentiel d'action. Ce mécanisme d'attention est implémenté grâce à une plasticité à court terme, qui permet de diminuer le poids des neurones d'entrée déchargeant fréquemment, c'est-à-dire ceux codant pour des valeurs proches de zéros. Dans la première version du réseau (MiniNet) cette plasticité a été implémentée sur les synapses reliant la couche d'entrée à la couche intermédiaire. Par la suite, un neurone d'attention a été implémenté pour isoler cette fonction de détection. Celui-ci reçoit les spikes émis par la couche d'entrée via des synapses implémentant une plasticité à court terme et émet une série de spikes pendant la durée d'un potentiel d'action présent dans le signal.

La deuxième couche du réseau, appelée couche intermédiaire, a pour but de reconnaître des motifs dans le train de spikes de la couche d'entrée, qui correspondent à différentes formes de signal. Les neurones intermédiaires implémentent un modèle « Leaky integrate and fire » (LIF), dont le potentiel augmente à chaque réception de spike et revient sinon à zéro selon une décroissance exponentielle. Un neurone LIF émet un spike (décharge) lorsque son potentiel atteint un seuil. Les poids des synapses provenant de la couche d'entrée sont initialisés aléatoirement. Avant apprentissage, chaque neurone reçoit donc une excitation similaire, insuffisante pour décharger. Les neurones reçoivent en plus les spikes du neurone d'attention, ce qui leur permet de décharger lorsque ce dernier décharge. Les synapses provenant de la couche d'entrée suivent une loi STDP. Leurs poids diminuent à chaque spike postsynaptique, mais augmentent si le spike postsynaptique coïncide avec un spike présynaptique. Grâce à cette loi les neurones intermédiaires deviennent sensibles à une forme spécifique du signal. La taille du champ récepteur de ces neurones, c'est-à-dire leur tolérance à la différence entre le signal en entrée et la forme apprise, peut être ajustée soit en modifiant leur seuil de décharge, soit en modifiant la largeur de la plage de sensibilité des neurones d'entrée. Afin d'éviter que plusieurs neurones déchargent simultanément et apprennent un même motif, un mécanisme dit « Winner-take-all » (WTA) est implémenté : lorsqu'un neurone décharge, les autres neurones de cette couche sont inhibés. Dans les dernières versions du réseau (LTSNet et PolyNet), une variante du WTA et de la loi STDP est implémentée, permettant à deux neurones de décharger simultanément, sans pour autant apprendre le même motif. Cela permet d'avoir des champs récepteurs se chevauchant partiellement, ce qui améliore la reconnaissance par la dernière couche du réseau. Lorsqu'un potentiel d'action est présent dans le signal, les neurones de la couche intermédiaire déchargent donc les uns à la suite des autres selon la forme du signal à différents moments du potentiel d'action.

La dernière couche est la couche de sortie. Celle-ci doit émettre un spike pour chaque potentiel d'action dans le signal, reflétant ainsi l'activité neuronale enregistrée. Elle doit notamment être robuste

aux différences dans la longueur des séquences émises par la couche intermédiaire. Pour parvenir à cet objectif, plusieurs mécanismes ont été testés. Premièrement une plasticité intrinsèque a été implémentée pour permettre au seuil des neurones de s'adapter à la taille des séquences apprises. Celle-ci a été testée sur les premières versions du réseau (MiniNet et ANNet). La loi STDP utilisée est similaire à celle utilisée sur la couche intermédiaire. Une variante a été testée, appelée STDP latérale, selon laquelle les poids sont modifiés lorsque le neurone postsynaptique reçoit une inhibition par WTA. Les neurones inhibés apprennent à ne pas reconnaître un motif reconnu par d'autres neurones, ce qui améliore les performances. Dans une des versions du réseau (ANNet), afin de forcer les neurones de sortie à décharger après la fin de la séquence de spikes intermédiaires, ceux-ci sont inhibés par le neurone d'attention. En plus de cela les synapses provenant de la couche intermédiaire sont dupliquées avec différents délais, ce qui permet à la fois de garder une excitation après la fin de la séquence et d'avoir une information sur les temps d'émission des spikes. Cette structure est cependant complexe. Dans les dernières versions du réseau (LTSNet et PolyNet), le modèle de neurone LIF a été remplacé par un modèle dit « LTS » (low threshold spiking) ayant la propriété de décharger par rebond après une inhibition. Les synapses provenant de la couche intermédiaire sont alors inhibitrices, ce qui permet aux neurones de sortie de décharger naturellement après la fin de la séquence.

Enfin, ce modèle de réseau, conçu pour traiter le signal d'une électrode, a été adapté au cas de plusieurs électrodes suffisamment proches pour qu'un potentiel d'action soit détectable sur plusieurs d'entre elles. La structure de base est dupliquée pour chaque électrode à traiter, formant ainsi plusieurs sous-réseaux parallèles. Des connections synaptiques entre les différents sous-réseaux sont ensuite introduites, pour prendre en compte la redondance d'information entre les électrodes voisines. Plusieurs structures ont été testées, au niveau de chaque couche du réseau. La solution retenue est de connecter la couche de sortie de chaque sous-réseau aux couches intermédiaires de plusieurs sous-réseaux correspondants à des électrodes voisines. Le mécanisme WTA de la couche de sortie est également étendu sur plusieurs sous-réseaux voisins.

## E. Evaluation des performances

Afin de tester ses performances, le réseau développé a été testé sur plusieurs jeux de données. Premièrement des jeux de données simulées pour lesquelles la vérité est connue. Pour générer des signaux mono-électrode, des gabarits de potentiel d'action sont placés dans le signal puis du bruit est ajouté. Pour générer des signaux multi-électrodes, un modèle de potentiel d'action extracellulaire basé sur une structure de cellule neuronale simplifiée a été développé. Deuxièmement, des jeux de données réelles, pour lesquels des enregistrements de tétrodes ont été utilisés, associés à un enregistrement intracellulaire donnant la vérité pour une cellule.

Pour évaluer la qualité du spike-sorting, nous avons choisi d'utiliser un F-score, qui prend en compte à la fois les faux positifs, les faux négatifs et les erreurs de détection. Ce score peut être subdivisé en un score de classification, un score de détection, un score de précision et un score de rappel. Nous avons également utilisé différents indices pour évaluer chaque sous partie du réseau. La détection par le neurone d'attention a été évaluée par une variante de l'aire sous la courbe ROC. La qualité de la couche intermédiaire a été évaluée selon deux indices différents : l'un basé sur l'entropie conditionnelle de la vérité connaissant la sortie de la couche intermédiaire, l'autre basé sur les distances entre les motifs intermédiaires.



Notre réseau a été comparé à deux autres logiciels de spike-sorting : Osort et Wave\_clus, tous deux pouvant, comme le réseau STDP, s'utiliser de manière automatique. Chacune des méthodes testées a été exécutée sur différents jeux de données avec différentes caractéristiques, et la significativité des différences de performance a été évaluée sur chaque jeu à l'aide de tests statistiques (test de Welch).

## F. Implémentations et résultats

Différentes versions du réseau STDP ont été implémentées suivant les modèles décrits au paragraphe D, chacune présentant des améliorations. La première version implémentée, MiniNet, est aussi la plus simple. Il n'y a pas de neurone d'attention, la plasticité à court terme étant implémentée sur les synapses reliant les couches d'entrée et intermédiaire. Ce réseau donne déjà de bons résultats sur des données simulées simples. Des tests additionnels sur la couche de sortie montrent que la STDP latérale améliore les résultats et que la plasticité intrinsèque permet de mieux discriminer des potentiels d'action de durées différentes.

Le réseau a ensuite été amélioré par l'augmentation de la fréquence d'encodage par la couche d'entrée et l'introduction d'un neurone d'attention. Cette version, nommée ANNet, comprend également une couche de sortie inhibée par le neurone d'attention, recevant les spikes de la couche intermédiaire par plusieurs délais synaptiques et implémentant une plasticité intrinsèque. Ce réseau a été comparé à Osort et Wave\_clus sur des jeux de données réelles et simulées et donne de meilleurs résultats sur les signaux avec un rapport signal-sur-bruit inférieur à 4, typique des enregistrements réels. Des tests préliminaires ont également été menés sur les données tétrodes, avec une structure de réseau en entonnoir, montrant l'intérêt de croiser les informations provenant de plusieurs électrodes voisines.

Enfin la dernière version mono-électrode du réseau, LTSNet, a pour principale amélioration l'introduction d'un modèle de neurone LTS sur la couche de sortie, qui permet de s'affranchir de la structure complexe d'ANNet sur cette couche (délais synaptiques, inhibition par le neurone d'attention, et plasticité intrinsèque). Ce réseau implémente également un 2-WTA sur la couche intermédiaire et une STDP latérale sur la couche de sortie. Les performances obtenues sont similaires avec celles d'ANNet, avec une structure plus simple.

Cette dernière version a été adaptée pour traiter des signaux multi-électrodes, en dupliquant la structure initiale pour chaque électrode et en introduisant des connexions synaptiques supplémentaires au niveau de la couche de sortie. Cette structure, appelée PolyNet, permet de détecter chaque potentiel d'action au niveau de l'électrode présentant la plus forte amplitude, mais des améliorations restent à apporter pour plus de robustesse dans la classification.

Le réseau STDP développé a été testé sur ordinateur. Un travail préliminaire d'implémentation sur FPGA a été effectué à l'occasion d'un encadrement de stage. L'implémentation FPGA comprend pour le moment une couche d'entrée et un neurone d'attention fonctionnels. Ce travail pourra être poursuivi pour parvenir à une implémentation embarquée complète du réseau. Une implémentation sur une puce neuromorphique nécessitera plus de développements, mais on peut déjà estimer la consommation du réseau à moins d'un microwatt par électrode.

## G. Conclusion et perspectives

Le travail effectué a permis de montrer la faisabilité d'appliquer un réseau STDP au problème de spike-sorting. Les performances obtenues sont en effet comparables, voire supérieures, à celle de l'état de l'art pour des signaux mono-électrodes. Les mécanismes développés pour répondre aux différents aspects du problème, à savoir reconnaître des motifs temporels de tailles variables parmi des stimuli non pertinents, pourraient être appliqués à des problèmes similaires. Des améliorations peuvent encore être apportées à ce réseau, notamment pour le traitement de signaux multi-électrodes, par des modifications de structure ou l'ajout de nouveaux mécanismes. Si l'implémentation sur FPGA marque un premier pas vers une application embarquée, l'utilisation de puces neuromorphiques pour un tel algorithme marquerait une grande avancée pour le spike-sorting et pour le traitement de signaux neuronaux en général. Il y a en effet, pour ce type de traitement, un besoin important d'implémentations embarquées et peu consommatrices, étant données les contraintes sur les implants d'enregistrement.