University of Genoa

Master's Program in Bioengineering

Thesis submitted in partial fulfillment of the requirements for the title of

Master of Bioengineering

Exploring Interaction Of Artificial And Biological Neural Networks In Biohybrid Experiments

Tommaso Lambresa

October 2024

Thesis advisor: Prof.ssa Michela Chiappalone

Thesis co-advisor: Prof. Federico Barban

Abstract

Brain prostheses are neuroengineering devices that interface with the central nervous system to supplement or restore function in individuals with disabilities through electrical stimulation. Recently, neuromorphic brain prostheses, such as spiking neural networks (SNNs), have received significant research attention for their ability to mimic neurobiological computations and process data in real time.

This thesis is divided into two parts. In the first part, we assess the effects of a novel type of intracortical microstimulation on *in vivo* models using healthy, anesthetized rats. This stimulation is driven by two types of SNNs. The results show that SNN-driven stimulation engages neural networks to varying degrees, depending on the stimulation pattern. Additionally, the findings align with previous studies exploring other forms of stimulation, positioning this method as a promising approach for achieving same functional recovery outcomes.

In the second part, we propose a multi-objective indicator-based evolutionary algorithm to fine-tune the SNN, optimizing it to replicate the electrophysiological parameters observed in the *in vivo* experiments. This algorithm is a crucial tool for personalizing stimulation based on the target, potentially improving the interaction between the prosthetic device and the biological system for more effective therapeutic results.

Contents

Introduction

The power and complexity of the human brain are mind-blowing. It is not only the container of our thoughts, emotions, and consciousness but also the foundation of every individual's identity. Within approximately 1.3 cubic decimeters of matter, the brain contains the entirety of our self-perceived world. And at the core of this wonder lies one fundamental truth: it's all driven by electricity. Every thought, action, and sensation is the result of intricate electrical impulses firing across neural networks. Neurons, the foundational units of the brain, encode everything through the depolarization and hyperpolarization of their membrane, effectively forming an intricate "electrical code" that represents sensations and actions. This "code" is not left to the case: each neuron is connected to others, ensuring that no neuron acts in isolation, but they must work in harmony, constantly considering the state of the entire network. This careful interaction makes sure that everyone is up to the job and no one is left behind, maintaining a temporal order as the large number of instruments in an orchestra (Buzsáki 2006).

The brain is also redundant, meaning that vastly different configurations of cellular and synaptic components can enable the same neural circuit functions (Mizusaki and O'Donnell 2021). However, despite these properties, any disruption to the integrity of the brain can lead to significant impairments in everyday life, affecting not only the individuals but also those around them.

From a report of the World Health Organization, diseases and injuries of the nervous systems constitute about 6.3% of the Global Burden of Disease. This, of course, has and will have a massive impact on society and economy (Chin and Vora 2014). As a result, there has been a rapidly growing interest in therapies and technologies aimed at preventing or treating brain disorders. Several large-scale international research initiatives, such as the Human Brain Project (HBP) and Brain Research through Advancing Innovative Neurotechnologies (BRAIN), have

emerged in recent years to address these challenges. One of the most promising technologies focusing on the possibility to remove and retrain people from cerebral impairments are the *electroceuticals*: since brain acts using electricity, targeting specific nerve fibers or specific brain circuits with electrodes makes possible the interaction with its functioning, allowing for pinpoint interventions (Famm et al. 2013; Reardon 2014). When applied correctly, electrical stimulation of specific brain areas can be more cost-effective and precise than pharmaceutical treatments, with the added potential to address drug-resistant conditions. In this context, several strategies can be adopted, each one specific for the disease: the most famous and consolidated tool is the Deep Brain Stimulator (DBS) which stimulates deep areas of the brain to alleviate or eliminate tremors induced by Parkinson's Disease or symptoms related to epilepsy and psychiatric conditions such as obsessive-compulsive disorder and major depression. One other approach to understand and treat neural pathology is to partially substitute or assist damaged brain areas. This kind of technologies are named *neuroprosthetics*.

Neuroprosthetics

A neuroprosthetic is a device or system that has an interface with the nervous system and supplements or substitutes functionality in the patient's body (Wright et al. 2016). A specific category within neuroprosthetics are *brain prosthesis*, devices directly connected with the central nervous system in order to replace a damaged area or bridge disconnected areas and regain the lost functionality (Panuccio et al. 2018). These can be categorized into two main types: open-loop and closed-loop systems. In open-loop systems, the device delivers a stimulus directly to the brain without any feedback from the neural activity. In contrast, closed-loop systems rely on feedback mechanisms, where brain activity (or its processed version) serves as input to the device, which in turn generates an output that becomes an input back to the brain. This creates a continuous I/O loop that operates indefinitely.

A brain prosthesis implementing an architecture following a closed-loop policy has been presented for the first time by Kansas University Medical Center (Guggenmos et al. 2013). Another promising example is represented by the hippocampal memory prosthesis, in which the neural activity of specific hippocampus areas suitably processed can be used to manipulate and thus restore (through ad hoc electrical stimulation) cognitive mnemonic processes (Berger et al. 2011).

To reach a perfect interaction between the device and the brain, different aspects should be taken into account: the device should be able to "read" the neural code (decoding), and subsequently "write" on the brain (coding) the processed information. This of course takes a lot of operations that should go almost real-time, to maintain physiological plausibility of interaction. For this reasons, recently a novel kind of neuroprosthetics has been developed: *neuromorphic neuroprosthetics* (Chiappalone, Cota, et al. 2022).

Neuromorphic neuroprosthetics and biohybrid systems

Neuromorphic hardware and computing are specialized fields that take inspiration from the structure and function of the human brain to design hardware and algorithms capable of more efficient, brain-like information processing. These systems often utilize spiking neural networks, which mimic the way neurons communicate and behave.

Due to their inherent brain-like architecture, neuromorphic systems are well-suited for interfacing with biological networks, enabling what are known as "biohybrid experiments" (Chiappalone, Cota, et al. 2022; Beaubois et al. 2024).

Biohybrid systems rely on the functional interaction between biological networks, such as neural systems, and artificial devices. A notable example is the "BrainBow" project, which developed a novel neuromorphic neuroprosthesis capable of artificially connecting two previously disconnected neuronal populations in vitro, enabling fully bi-directional communication between them (Chiappalone, Cota, et al. 2022). Translating this knowledge to in vivo applications in animals is a critical step toward developing devices that seamlessly integrate with biological systems to restore or enhance neural functions.

Real-Time Interaction of Artificial and Biological Neural Networks

This study aims to evaluate the effects of a novel "neural-like" open-loop stimulation method using a neuromorphic devices. Specifically, two FPGA-based Spiking Neural Networks (SNNs) are used to stimulate anesthetized rats *in vivo*. The impact of the stimulation will be assessed by measuring changes in electrophysiological parameters, with a comparison between two SNNs. Additionally, an optimization algorithm will be developed to fine-tune the FPGA configuration, ensuring the network closely resembles the biological counterpart being stimulated, thereby achieving a stimulation that is as physiologically plausible as possible. These are

preliminary experiments, primarily aimed at evaluating the driving capability of this type of stimulation, with the potential to modulate neural activity, especially impacting on its dynamic. The insights gained from this research could mark an important step toward the development of an efficient neuromorphic neuroprosthesis. Such advancements have the potential to enhance our ability to interface with the nervous system, providing innovative solutions for restoring lost functionality in patients with neurological impairments.

Part I

Biohybrid experiments

Chapter 1

BIOEMUS

To conduct bi-directional bio-hybrid experiments and develop bioelectrical applications for healthcare, such as electroceuticals (Famm et al. 2013; Reardon 2014), aiming at substituting an injured brain area, it is essential to have a real-time spiking neural network (SNN) to ensure interaction at the biological time scale. Most used approaches for biomimetic SNN simulations rely on software platforms like NEURON (Hines and Carnevale 2001) or NEST (Gewaltig and Diesmann 2007). However, these tools often suffer from long computation times, especially when simulating complex neuron models with synaptic plasticity. In contrast, hardware-based SNNs are better equipped for real-time processing and offer the additional advantage of enabling large-scale parallel simulations.

In neuromorphic engineering, we can have two different approaches to design an hardware-based SNN: the biomimetic and the bioinspired.

The bioinspired method wants to replicate the computational power and lower consuption of our brain, regardless of the biological mechanism that arises. In this context, simple and abstract models of neurons are involved, such as the Leaky-Integrate and Fire model (Brunel and van Rossum 2007). These models roughly replicate the shape of an action potential, in favour of a faster simulation of the dynamic especially relying on a high number of neurons. They are mainly employed in computations or Artificial Intelligence applications.

The biomimetic approach, in contrast, focuses primarily on the biophysical replication of electrical dynamics within neurons, including the modeling of membrane channel behavior, such as their opening and closing, which adds biological significance to its components. The most biologically accurate single-compartment model to date is the Hodgkin-Huxley (HH) model

(Hodgkin and Huxley 1952). It can simulate a wide range of neural dynamics by tweaking its parameters or adding extra differential equations to the system. Using this model as a foundational component of a neural network is an effective way to replicate electrophysiological activity, making it a strong candidate for enabling biohybrid experiments.
Hardware-based SNN can be analog or digital. Implementing a digital SNN allows quicker and more flexible design, being more suitable to prototyping (Beaubois et al. 2024). In the following sections an overall presentation of the Bioemus FPGA-based SNN is provided.

1.1 Hodgkin-Huxley (HH) model

The Hodgkin-Huxley model, introduced by Alan Hodgkin and Andrew Huxley in 1952, is one of the most influential mathematical models in neuroscience (Hodgkin and Huxley 1952). It describes the electrical characteristics of excitable cells, such as neurons, and explains how action potentials (nerve impulses) are initiated and propagated along the axon. The model is based on a set of differential equations that describe how ion channels in the neuron's membrane control the flow of ions, leading to changes in membrane potential.
It is a conductance-based model of a neuron, which means it describes the neuron's electrical properties in terms of conductances and currents Figure 1.1. In this family of models, the lipid bi-layer of the neuron's membrane is modeled as a capacitor because it separates ion charges across the membrane, storing electrical energy as potential difference between the inside and outside of the neuron. The ion channels embedded in the membrane are represented by conductances because they allow ions (such as sodium, potassium, and others) to pass through the membrane, similar to electrical resistors that control current flow in a circuit.
The conductance of these ion channels changes dynamically in response to the membrane potential, and this directly affects the flow of ions, driving the neuron's activity. The original HH model precisely described with differential equations the Na^+ and K^+ ions current, going from the inside to the outside of the membrane and vice versa. They saw that the permeability of the membrane to these ions was not fixed but variable basing on the voltage. As a consequence, they modeled this dynamical permeability as *gating particles*, other variables that describes the portions of opened or closed channels, governed by other differential equations.

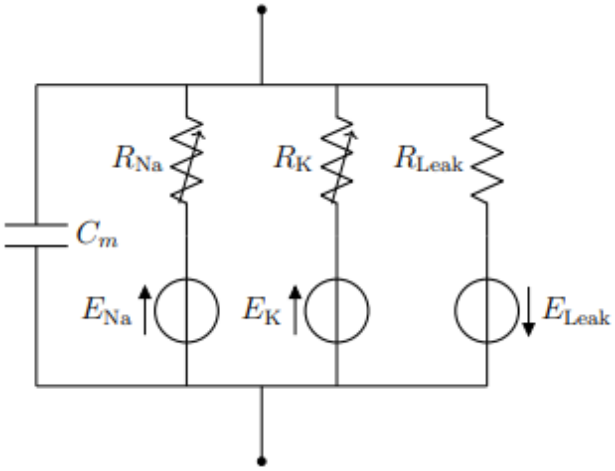

Figure 1.1: *Hudgkin-Huxley equivalent circuit of the neuronal membrane*

The model is represented as an RC circuit, where variable resistances correspond to the flow (current) of potassium (K) and sodium (Na) ions through voltage-gated channels. The membrane's phospholipid bi-layer is modeled as a capacitor, reflecting its ability to store charge across the membrane. A third resistance represents the leakage current, accounting for the passive, non-voltage-dependent diffusion of ions across the membrane. Additionally, the circuit includes three voltage sources, which represent the Nernst potentials for potassium, sodium, and the leakage, driven by the concentration gradients of these ions inside and outside the cell.

The equation describing the membrane voltage dynamic in time is:

$$-C_m \frac{dV}{dt} = g_{Na} m^3 h (V - E_{Na}) + g_K n^4 (V - E_K) + g_{leak}(V - E_{leak}) \tag{1.1}$$

where g_{Na}, g_K, g_{leak} are the average conductances, m, n, h, are the gating particles of activation and inactivation of the voltage-dependent channels, C_m is the membrane capacitance and E_{Na} E_K and E_{leak} are the Nernst potentials for sodium, potassium, and leakage, respectively.
The gating variables m, n and h follow their own voltage- and time-dependent differential equations, which describe how they evolve in response to changes in membrane potential:

$$\frac{dp(v,t)}{dt} = \alpha_p[1 - p(v,t)] - \beta_p(v,t) \; for \; p = m, n, h. \tag{1.2}$$

α and β are measured empirically as the open/close channels ratio, and are also voltage dependent.
The three addends at the right of the equal are the ion currents, so the same equation can be written as:

$$-C_m \frac{dV}{dt} = I_{Na} + I_K + I_{leak}. \tag{1.3}$$

Following a similar approach, several studies have extended the Hodgkin-Huxley model to incorporate additional types of ion channels, such as persistent sodium channels, calcium channels, and calcium-dependent potassium channels (*Computational Modeling Methods for Neuroscientists* 2024; Huguenard and McCormick 1992). This is achieved by introducing additional resistances in parallel, each corresponding to a different ion channel, and each governed by its own set of gating variables. As a result, each modeled current for the j-th ion becomes:

$$I_j = g_j m^{p_j} h^{q_j} (V - E_j), \tag{1.4}$$

These currents are then summed to represent the total ionic current flowing through the membrane. By including these additional channels, the model can capture more complex neuronal behaviors, such as bursting, adaptation, and oscillations, thereby providing a more detailed and biologically realistic representation of the neuron's electrical activity.

1.2 Bioemus: Adaptable integrated real-time biomimetic SNN

Bioemus, standing for BIOmimetic EMUlation Single compartment, is an accessible low cost platform for real-time emulation of biomimetic SNN (Beaubois et al. 2024). It engages detailed biophysical models of neurons or synapses within a fully customizable network, and its versatility stems from its use of a Field Programmable Gate Array (FPGA).

An FPGA in digital electronics is a programmable logic device, where the integrated circuit can be reprogrammed multiple times after manufacturing. It consists of an array of programmable logic blocks connected by a grid that can be reconfigured to link different blocks together, enabling various digital functions. These logic blocks are programmed using hardware description languages like VHDL (Sulaiman et al. 2009). The low-cost platform used is based on a System on Chip (SoC), which combines both Programmable Logic (PL, i.e., FPGA) and processors in a Processing System (PS). This platform can support up to 1,024 fully connected neurons, managing a total of 2^{20} synapses. Neurons are connected using biomimetic synapses to allow fast and slow synaptic excitation or inhibition. The PS part of the board runs the canonical Ubuntu 22.04 operating system, enabling direct FPGA configuration from the board itself thanks to the python interpreter installed. For remote control, the board offers various connectivity options, including Ethernet, Wi-Fi, and expansion PMODs. This allows users to connect and monitor the board from a host PC via serial or SSH protocols Figure 1.3.

A Python script generates JSON and text configuration files for the C++ applications. These files can be created on an external PC and transferred to the board using protocols like SCP. When interacting with a biological counterpart in biohybrid experiments, the user can simulate the entire SNN in real time. This includes the option to receive or provide stimulation to/from the board and selecting which specific neurons (between 1024) will deliver the stimulus train. Several outputs are provided at the end of the simulation: the user can choose to save the membrane potential waveform of a certain number (up to 8) of neurons or the entire rasterplot of the whole network; both are saved in binary files by default.

The binary spike data file is structured as follows:

- Time Stamp: A 32-bit unsigned integer representing the time stamp in milliseconds.

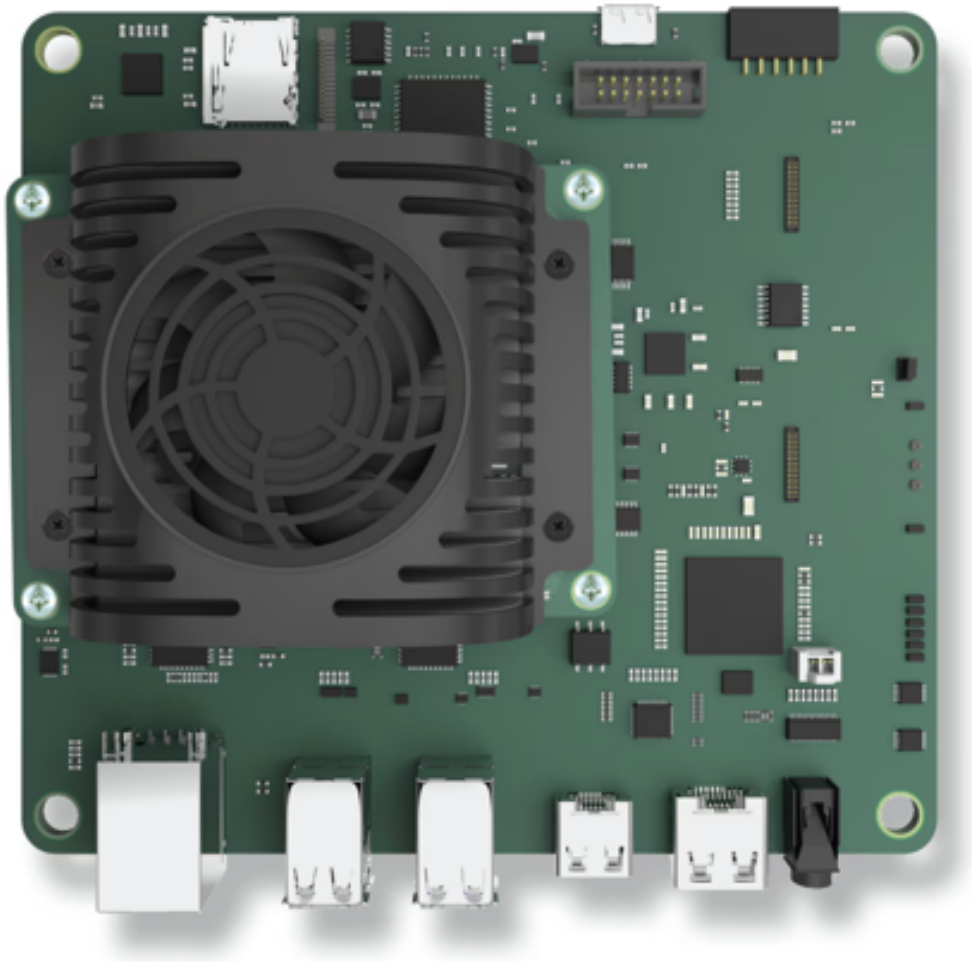

Figure 1.2: *Kria K26 SOM from AMD Xilinx embedded on the development platform Kria KV260 Robotics Starter Kit. From* `https://www.amd.com/en/products/system-on-modules/kria/k26/kv260-vision-starter-kit.html`

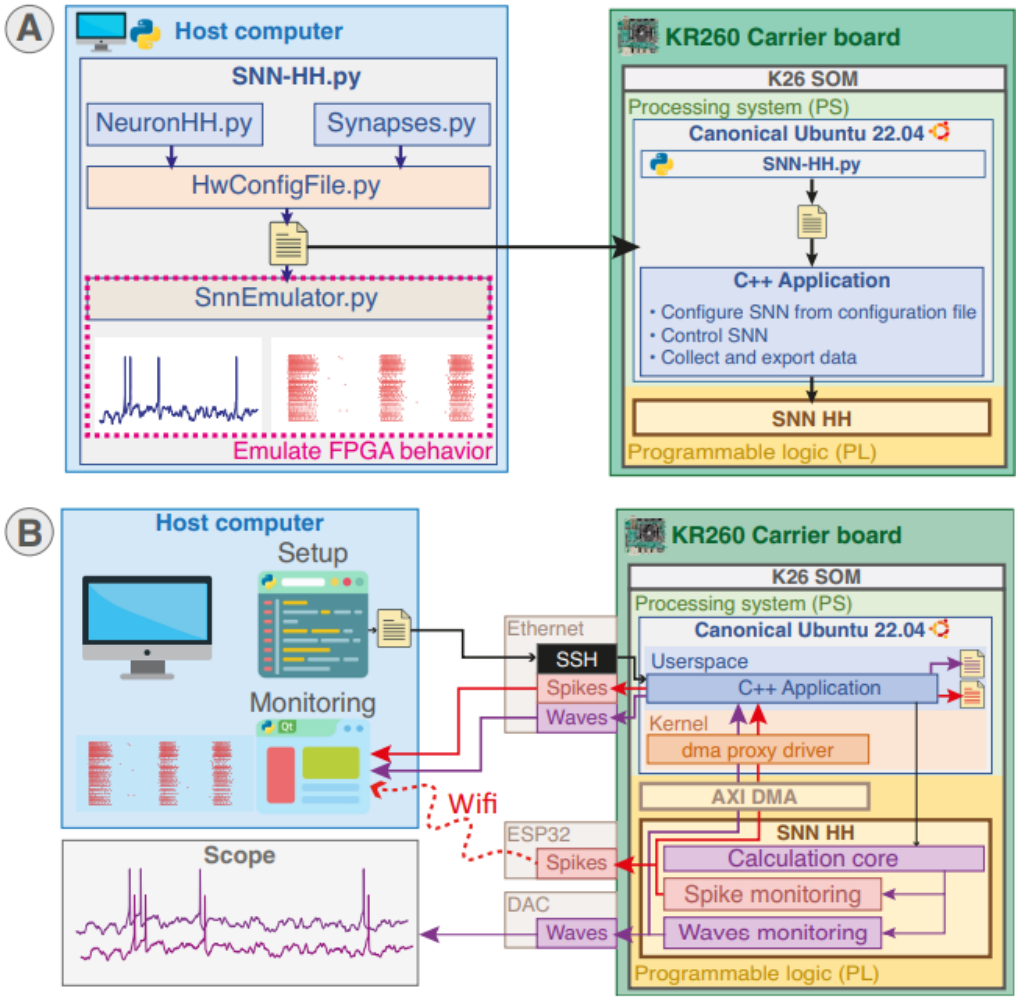

Figure 1.3: *(Beaubois et al. 2024)* ***Architecture and System Integration of the platform***

A) *Overview: The system configuration, generated by Python scripts running either on-board or on a separate computer, is read by a C++ application on the Processing System (PS) to configure the SNN in the Programmable Logic (PL) part. Emulator allows for predicting network behavior. Sample membrane voltage and raster plots can be exported from the Bioemus software.*

B) *System Communication: The system is controlled through the C++ application, either remotely via SSH or directly from the on-board Ubuntu desktop. Spikes can be monitored concurrently using Ethernet, Wi-Fi, and on-board file saving. Waveforms can be monitored using Ethernet, visualized on an oscilloscope by probing the Digital-to-Analog Converter (DAC), and saved on-board. Sample membrane voltage data can also be exported from Bioemus.*

- Neuron ID: A stream of 1024 (or whatever user-defined number) bits, where each bit indicates if the corresponding neuron fired (1) or not (0).

This format ensures that file size is determined by simulation duration rather than the number of spikes produced.

Additionally, a graphical user interface (GUI) has been developed to enable real-time monitoring of simulation results, offering a clear and interactive way to observe spiking activity as the simulation progresses. Opening this monitoring window gives the possibility to export the recorded data, whether spike timings or voltage traces, as a .csv file. Finally, the system offers the option to perform a purely software-based emulation of the network without using the FPGA. This emulation calculates membrane potential using the forward Euler method, resulting in significantly longer simulation times. Euler method is a numerical approach used to solve ordinary differential equations: if the ODE is for example

$$\frac{dy}{dx} = f(x, y) \tag{1.5}$$

the Euler method approximates the solution at subsequent points using:

$$y_{n+1} = y_n + h \cdot f(x_n, y_n) \tag{1.6}$$

where h is the step size (a small interval over which the function is approximated), $f(x_n, y_n)$ is the slope at the current point (x_n, y_n) and y_{n+1} is the next approximation of the solution.

1.3 Configurable Parameters

The neurons in the Spiking Neural Network (SNN) are modeled with high biological fidelity using the Hodgkin-Huxley (HH) framework (Hodgkin and Huxley 1952) as implemented in the Pospichil model (Pospischil et al. 2008), which includes six conductance-based currents. Synaptic noise is simulated by an injected current following an Ornstein–Uhlenbeck process (Uhlenbeck and Ornstein 1930), generating spontaneous activity through random action potentials. The HH model parameters and those of the synaptic noise are adjustable through 25 parameters provided in the Python scripts. The scripts include four predefined neuron types: Fast Spiking (FS), Regular Spiking (RS), Intrinsic Burst (IB), and Low Threshold Spiking (LTS), with the option for users to define additional types. Ionic channel state equations are pre-computed and stored using the Euler method (Equation 1.6), allowing modifications to

channel dynamics without affecting system performance or restricting the use of mathematical functions.

Neurons are interconnected using biomimetic synapses that emulate AMPA, NMDA, GABAa, and GABAb receptors (Destexhe, Z. Mainen, and Sejnowski 1998), enabling both rapid and slow forms of synaptic excitation or inhibition. The parameters for these synaptic models can be adjusted in the same manner as the Hodgkin-Huxley parameters through the Python scripts. Synaptic connections can be established among all neurons and weighted independently, providing flexibility for users to design custom functions for setting up these connections.

To emulate the neural architecture of a rat brain, a specialized model was previously developed (Beaubois et al. 2024). This model specifically incorporates only RS neurons, which function as excitatory neurons, and FS neurons, which serve as inhibitory neurons. The RS neurons are connected through AMPA and NMDA synapses, while the FS neurons utilize GABAa synapses for inhibitory interactions.

The probability of connection between two neurons is determined by their spatial distance: neurons are randomly positioned in space while maintaining a high degree of clustering. As the distance between neurons increases, the probability of connection decreases, and this relationship follows a nonlinear dependence:

$$P = P_{max}(1 - \frac{d_{npre-npost}}{d_{net}}) \tag{1.7}$$

where P is the probability of connection between two neurons, P_{max} is the maximum probability of connection between two neurons, $d_{npre-npost}$ is the distance between the presynaptic and postsynaptic neuron, d_{net} is the diameter of the neuronal network.

This design mimics a "Small-World" network topology (Watts and Strogatz 1998), characterized by high density of connections within clusters of neurons and relatively few connections between different clusters Figure 1.4. This approach effectively reproduces the spatial organization and connectivity patterns found in real neural systems (Bassett and Bullmore 2017).

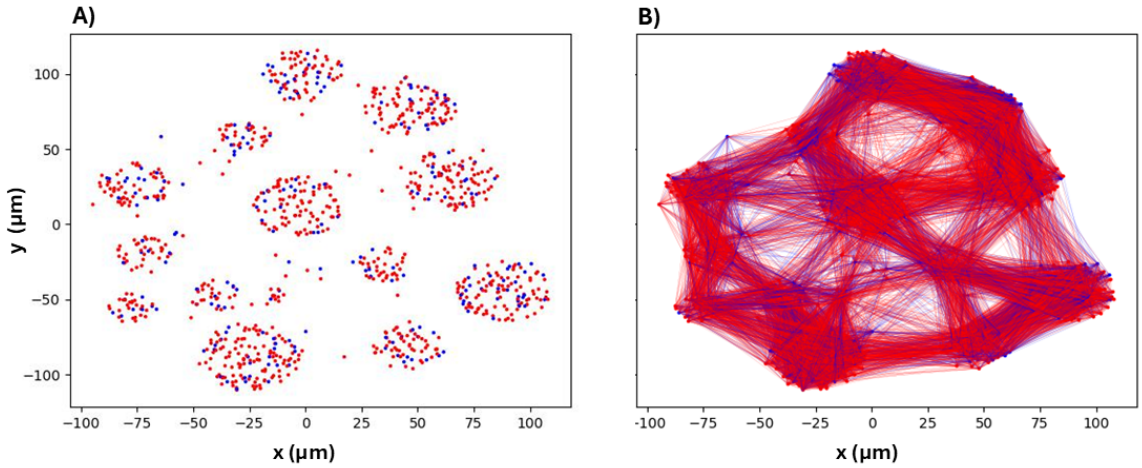

Figure 1.4: *(Beaubois et al. 2024) Small-world topology of the rat-brain configuration*

A) Position of each neuron. Red points are excitatory Regular Spiking neurons, blue points are inhibitory Fast Spiking neurons.

B) Synapses between neurons. Red lines are gluatamatergic AMPA or NMDA synapses (not distinguished), blue are GABAergic GABAa synapses.

Chapter 2

Analysis of Biohybrid Experiments

The following biohybrid experiments aim to evaluate the effects of SNN-driven neurostimulation, with the ultimate goal of utilizing this system as a neural prosthesis (Chiappalone, Cota, et al. 2022). Previous studies have demonstrated promising outcomes with this type of *"neural-like"* stimulation, leveraging various FPGA-based neuromorphic devices (Di Florio et al. 2023). However, no thorough assessment of the effects during the intra-stimulation phases has been conducted, leaving a critical gap in understanding the real-time impacts of the stimulation on neural activity. Consequently, these experiments aim to extend the analysis by evaluating the capacity of the stimulation to drive and modulate neural activity. To achieve this, two different FPGA boards are employed. First, the experiments are performed using the same board as in Di Florio et al. 2023, the Zybo Z7-20. The Zybo Z7-20 is a mid-range evaluation board that contains the same FPGA as the Kria KV260, but in this case its configurable network is composed by up to 100 neurons all independent and disconnected (Khoyratee et al. 2019). Subsequently, a new set of experiments is conducted with the Kria KV260 board, which implements Bioemus, allowing for a comparison between the two systems and providing insights into the performance and potential improvements of the newer platform. The stimulation pattern follows the activity of one neuron chosen at random.

2.1 Animals and Dataset

All experiments were approved by the Italian Ministry of Health and Animal Care granted prior authorization (Italy: authorization n. 509/2020-PR). In total, twenty-one adult male Long-Evans rats (weight: 350–400 g, age: 4–5 months; Charles River Laboratories, Wilmington, MA,

USA) are used in this study. Stimulation experiments were performed on healthy anesthetized preparations following the protocol described in Section 2.3
Rats are divided in three groups:

1. SHAM: animals that do not receive any stimulation. (7 animals)
2. Z-SNN: animals stimulated with the Zybo Z7-20 board, which is referred to simply as the Z group. (7 animals)
3. K-SNN: animals stimulated with the Kria KV260 board, which is referred to simply as the K group. (7 animals)

2.2 Surgical Procedure

Anesthesia is induced by placing the rat in a vaporizing chamber and administering gaseous isoflurane (5% at 1 lpm). To achieve a surgical level of anesthesia, ketamine (80-100 mg/kg IM) and xylazine (5-10 mg/kg IM) are injected. The rat is then secured in a stereotaxic frame, and vital parameters are continuously monitored throughout the procedure. The surgical process begins with the application of lidocaine cream, a topical analgesic, followed by a midline skin incision to expose the skull. A laminectomy is performed at the Cisterna Magna to allow for the drainage of cerebrospinal fluid (CSF). Based on stereotaxic measurements (Kleim et al. 2003) +3.5, +2.5 AP, and –1.25, +4.25 ML, burr holes (3 mm in diameter) are carefully drilled over the primary somatosensory area (S1) and rostral forelimb area (RFA). The dura mater is then removed from both burr holes to facilitate the insertion of microelectrode arrays (MEAs; A4x4-5 mm-100-125-703-A16, NeuroNexus).

2.3 Experimental Protocol and recordings

Wide-band signals are recorded from the RFA and S1 using 16-channel MEAs (A4x4-5 mm-100-125-703-A16, NeuroNexus), which are connected to Intan RHS headstages (RHS 16-Channel Stim/Recording Headstages, Intan Techonologies). Communication with the headstages happens over the SPI protocol, interfacing with the Intan RHS controller, an FPGA-based electrophysiology data acquisition system (Figure 2.1).
Concerning the stimulation site, an impedance analysis was performed and a single channel

from S1 is selected according to the lowest value ($\sim 200 - 300k\Omega$). Each time a spike is generated from the chosen SNN channel, a pulse is sent to the digital input of the Intan RHS to trigger the stimulation (Figure 2.2). The stimulation pulse is always designed to be a single squared 60 μA biphasic, cathodal-leading pulse (200 μs positive, 200 μs negative). The experimental protocol consisted of 20 minutes of pre-stimulation (PreS) recording, 6 minutes of connectivity mapping (CM1), 60 minutes of stimulation via SNN-generated activity (apart for SHAM), 20 minutes of post-stimulation (PoS) recording and 6 minutes of final connectivity mapping (CM2). Connectivity mapping is an excitability test, to see how the neurons react in response to a single stimulus. To do so, two different channels have been chosen to investigate whether the effects of stimulation could be location-dependent. During these 6 minutes, a stimulus was delivered every 5 seconds (corresponding to a frequency of 0.2Hz). Extracellular signals have been continuously sampled at a rate of 25 kHz and 16 bits of depth. .

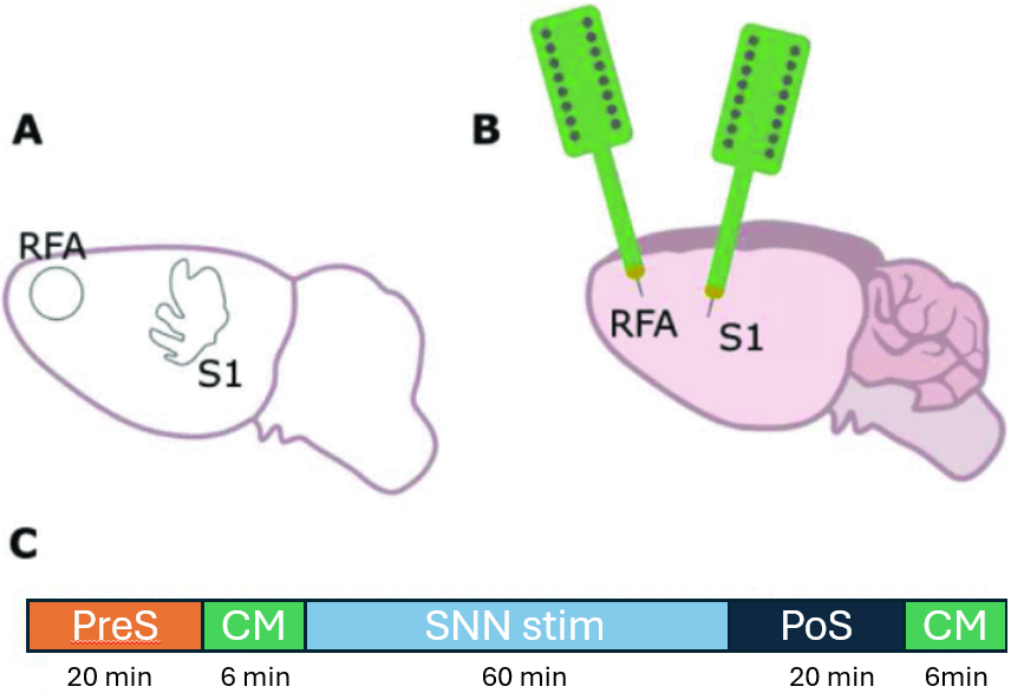

Figure 2.1: *Protocol of the experiment. A: identification of the areas of interest by means of a stereotaxic frame. B: placement of the electrode arrays in the rostral forelimb area (RFA) and in the primary somatosensory area (S1). C: timeline of experiments includes 20 minutes of pre-stimulation (PreS) recording, 6 minutes of connectivity mapping, 60 minutes of SNN-based stimulation, 20 minutes of post-stimulation (PoS) recording and 6 minutes of connectivity mapping.*

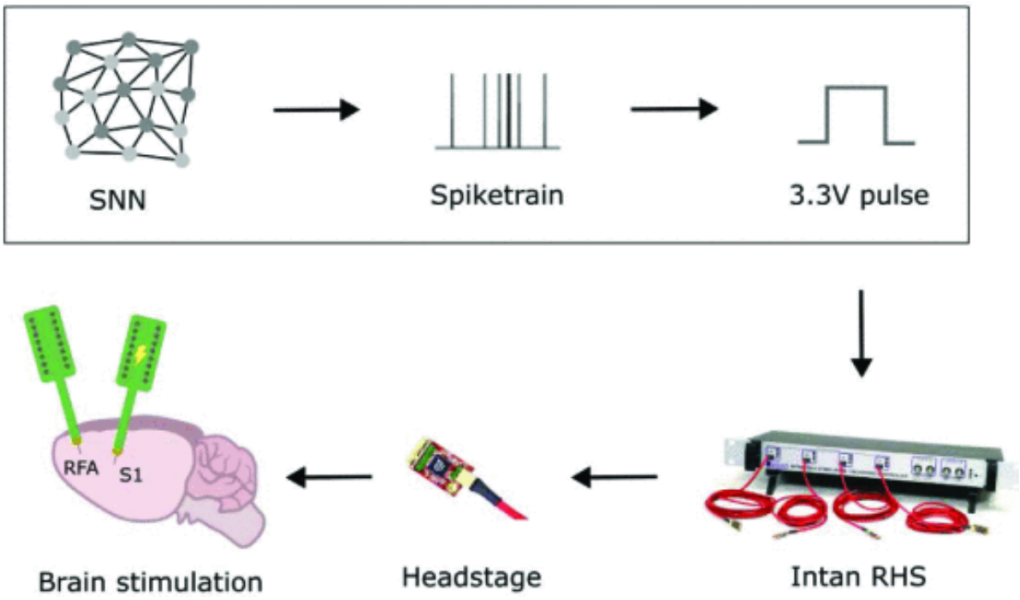

Figure 2.2: *Schematization of the experimental setup for stimulation. Starting from the spontaneous activity generated by a neuron of the SNN, a spike train is generated. Each time a spike occurs, a 3.3V pulse is sent to the digital input port of the Intan RHS system. This pulse triggers the stimulation process. A command is sent to the headstage from the Intan RHS, which generates a biphasic stimulation pulse for the electrode placed in S1.*

2.4 Data and Statistical Analysis

All the analyses on the in-vivo recordings, up to their spike detection, were performed in MATLAB (The MathWorks, Natick, MA, USA). The electrophysiological data underwent pre-processing through a custom MATLAB pipeline in the online software NigeLab (`https://github.com/barbaLab/nigeLab`). Briefly, data were formatted into a MATLAB-readable structure and organized on a per-channel basis. Subsequently, a 4th order elliptic bandpass filter (300-3000 Hz) was applied to eliminate low-frequency components in the signal, and all the data were re-referenced with respect to the overall mean value, to avoid common-mode noise. During this phase, a novel artifact rejection algorithm was applied to mitigate stimulation-induced artifacts (Negri 2023).

Following artifact suppression, a modified version of energy-based spike detection algorithm known as Stationary Wavelet Transform Teager Energy Operator (SWTTEO) was employed to identify spikes from the filtered data. The SWTTEO involved two levels of Stationary Wavelet Transform (SWT) followed by the use of the Teager Energy Operator (TEO). The

TEO outputs were smoothed, summed, and their combination was thresholded (Lieb, Stark, and Thielemann 2017).

The need for a modified version of SWTTEO arises from the characteristic ON-OFF dynamics of neural recordings from anesthetized rats. ON-OFF dynamic is a recurrent pattern found in intracortical recordings during deep sleep, anesthesia, coma, and other neurological conditions. They are mainly related with the raising of slow-waves oscillations (high-amplitude oscillations in the delta (0.5–4 Hz) band) which are the main biomarker to distinguish between an awake or a sleepy brain. This dynamics presents a period of profound membrane hyperpolarization and silence in cortical neurons, lasting for a few hundred milliseconds, where almost no spikes are detected, whereas during ON phase the neurons fires coherently. .(Massimini et al. 2024). The standard SWTTEO relies on a global threshold based on the 99th percentile of the transformed signal, causing it to miss spikes during the OFF phases (Figure 2.3). To address this limitation, an adaptive version of the algorithm was developed, which calculates thresholds locally within a moving window based on the signal's activity. This adaptive approach significantly improves spike detection accuracy, albeit with a slight increase in computational cost (see Appendix 1 for details).

The analysis was conducted using a multi-unit approach, without the application of any spike-sorting algorithms.

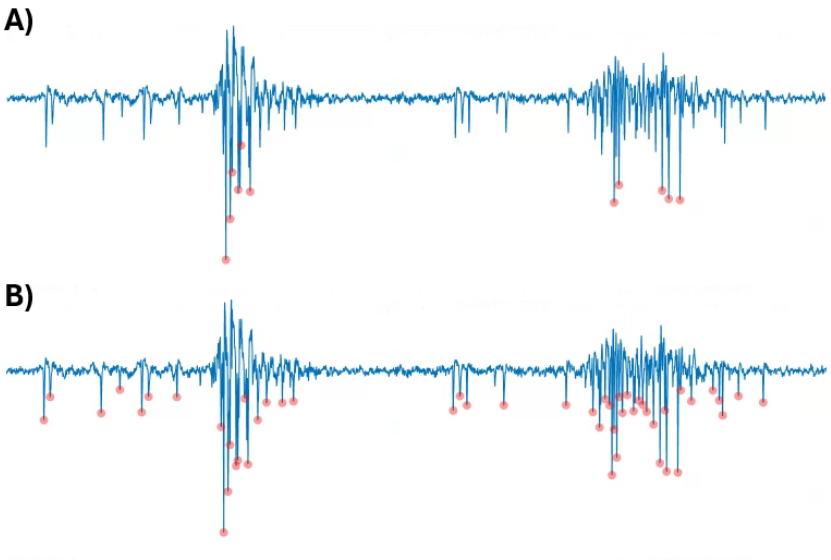

Figure 2.3: *Comparison of spike detection using SWTTEO with global (A) or local (B) threshold. The adaptive version is able to capture spikes in the OFF phase of the neural signal*

2.4.1 Metrics

To evaluate the effects induced by the stimulation, several metrics are analyzed:

1. Mean Firing Rate (MFR): This metric represents the frequency of neural firing, calculated as the total number of spikes divided by the recording duration, for each channel (expressed as spikes/sec).

2. Local Variation compensate for Refractoriness (LvR): This metric, introduced by (Shinomoto et al. 2009), is a measure used to quantify the variability in interspike intervals (ISIs) on a local timescale. It is calculated by comparing consecutive ISIs to assess how consistent the timing of spikes is within a spike train. It is mathematically computed as:

$$LvR = \frac{3}{n-1}\sum_{i=1}^{n-1}\left(-\frac{4I_iI_{i+1}}{(I_i+I_{i+1})^2}\right)\left(1+\frac{4R}{I_i+I_{i+1}}\right) \tag{2.1}$$

   A lower LvR (closer to 0.5) indicates more regular, periodic spiking activity, whereas higher values (about 1.5) suggest more compact, bursty spike timing. Values around 1 indicates an irregular, random spiking patterns (Figure 2.4).

Figure 2.4: *(Example spike sequences of representative neurons with LvR of 0.5 (Regular), 1 (Random), and 1.5 (Bursty). Taken from (Averna, Pasquale, et al. 2020).*

3. Post-Stimulus Time Histogram (PSTH): This metric measures the average evoked response to stimuli evaluated in a fixed binned time window (in this case 0.3 seconds binned at 1 ms) (Chiappalone, Vato, et al. 2007; Rieke 1999). Specifically, a time window, divided into predetermined bins, is captured after each stimulus and the number of spikes within each bin is counted. These individual windows are then averaged across all stimuli. This method is commonly used quantify the effectiveness of a stimulation pattern, particularly during the connectivity mapping phase.

4. Input-Output (I/O) Correlation: Since "neural-like" stimulation involves a variable inter-stimulus interval, the standard PSTH approach becomes impractical during the stimulation-phase due to variable inter-stimulus interval. Instead, the cross-correlation between the stimulus and the evoked activity in each channel is used to quantify neural responses (Averna, Hayley, et al. 2021). This was calculated using a window of 50ms binned at 1ms (see Appendix 2 for details), on the assumption that this should be a sufficient window to capture evoked monosynaptic spiking responses.

2.4.2 Statistics

The statistical analysis is conducted using MATLAB as well, with differences deemed statistically significant at $p < 0.05$. Dataset is split in two intersecting groups and two different statistical model are fitted:

1. Two-Way Repeated Measures ANOVA: This model assesses differences between the two types of stimulation (Z and K), independent of the SHAM group, following a visual inspection of the normality of residuals via QQ plots. The two factors considered are the areas (S1 and RFA) and the types of stimulation (Z and K).

2. Friedman Test: This model evaluates the differences between the SHAM and stimulated groups. The Friedman test is employed due to the inability to model the data with linear models and normal residuals. It serves as the non-parametric equivalent of the one-way repeated measures ANOVA. In this analysis, the single factor is the group (SHAM, Z, K), with only S1 recordings included since the SHAM group lacks recordings from RFA.

On both tests results, a post hoc multiple comparison using Tukey's honestly significant difference (HSD) test is performed. The box plots representation indicates the 25–75 percentile (box), the $Q1 - 1.5 \times IQR$ to $Q3 + 1.5 \times IQR$ (whiskers), and the median (line) values. The notch represents the confidence interval at $\alpha = 0.05$.

2.5 Results

2.5.1 Differences in types of stimulation

The stimulation provided by the two boards differs significantly. The newer board, the Kria KV260, allows for the selection of the stimulating neuron, introducing a complex decision-making process. The neurons on this board are modeled using two distinct computational models and are interconnected via three types of synapses, leading to considerable variations in both the frequency and occurrence of spikes. In contrast, the Zybo Z7-20 board features only fast-spiking or regular-spiking neurons, which are not connected. Consequently, the activity is uniform across all neurons on this board.

In these experiments, the choice of the stimulating neuron on both boards is made without a strict selection criterion. The only consideration, particularly for the Kria KV260, is ensuring the neuron's firing rate is not excessively high for stimulation purposes. This results in very different neural dynamics between the two systems. An analysis of the Mean Firing Rate (MFR), Local Variation of Refractoriness (LvR), and Inter-Spike Interval (ISI) of the stimulating neurons, averaged across all stimulated rats for both the Kria and Zybo boards, reveals several differences. The Zybo Z7-20 board produces a fourfold higher MFR (4.348 ± 0.011 Hz) compared to the Kria KV260 (1.213 ± 0.085 Hz). Additionally, the LvR is lower on the Zybo Z7-20 (0.919 ± 0.006) than on the Kria KV260 (1.165 ± 0.123), indicating that while the Z7-20 board generates more stimuli, the stimulation pattern is more random. In contrast, the Kria KV260 shows more burst-like behavior.

These results are further confirmed by the ISI histogram, which displays two distinct patterns: the Zybo neuron exhibits a much more spread-out histogram, whereas the Kria shows a peak within the first 20 ms, indicating that the majority of spikes occur approximately 15 ms apart (Figure 2.5).

2.5.2 Different trends in firing rate

When evaluating the differences in MFR before and after stimulation, alongside those from the non-stimulated SHAM animals, three distinct trends emerge across the groups: the SHAM group exhibits a spontaneous increase in firing rate over time, the Z group maintains a steady firing rate, and stimulation from the Kria board appears to decrease the firing rate (Figure 2.6).

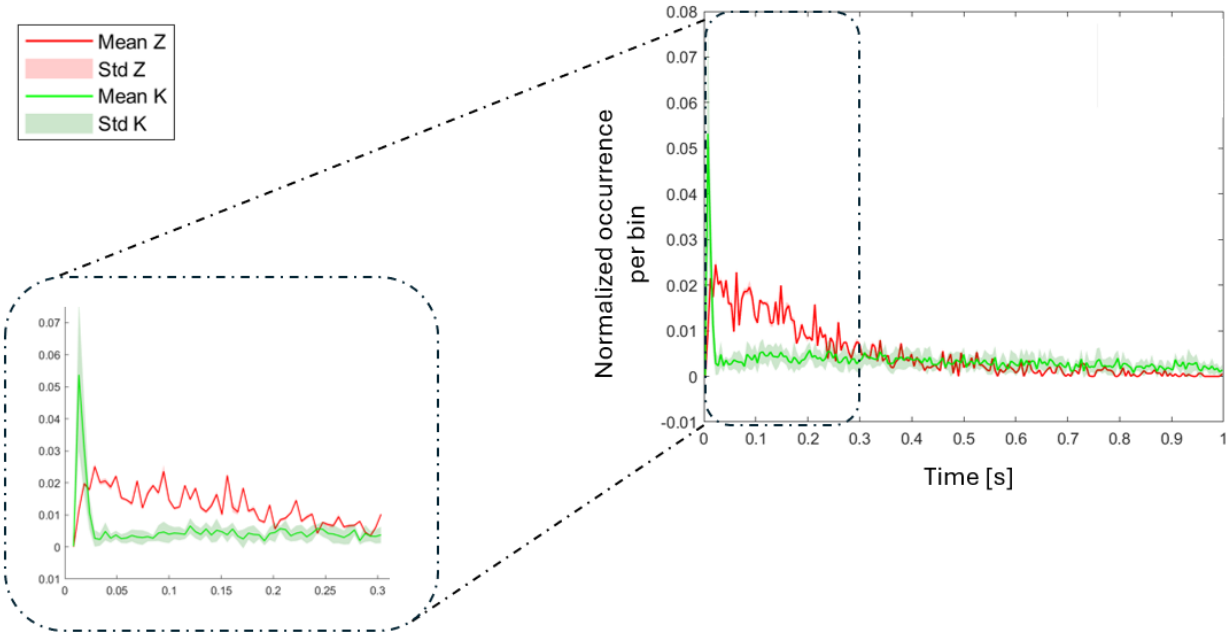

Figure 2.5: *Inter-Stimulus Interval (ISI) histogram for the two boards, Kria (green) and Zybo (red), covering intervals of 1 and 0.2 seconds binned at 5 ms. The solid line represents the mean, while the shaded area indicates the standard deviation. The histogram is constructed by calculating the time intervals between consecutive spikes and counting how often these intervals fall into each bin. The result is then normalized by the total number of spikes. The Kria KV260 (green) shows a more burst-like firing pattern, as evidenced by the distinct peak in the zoomed-in window. Additionally, the Kria board exhibits a larger standard deviation compared to the Zybo, suggesting greater variability in the neural-like stimulation across different experiments.*

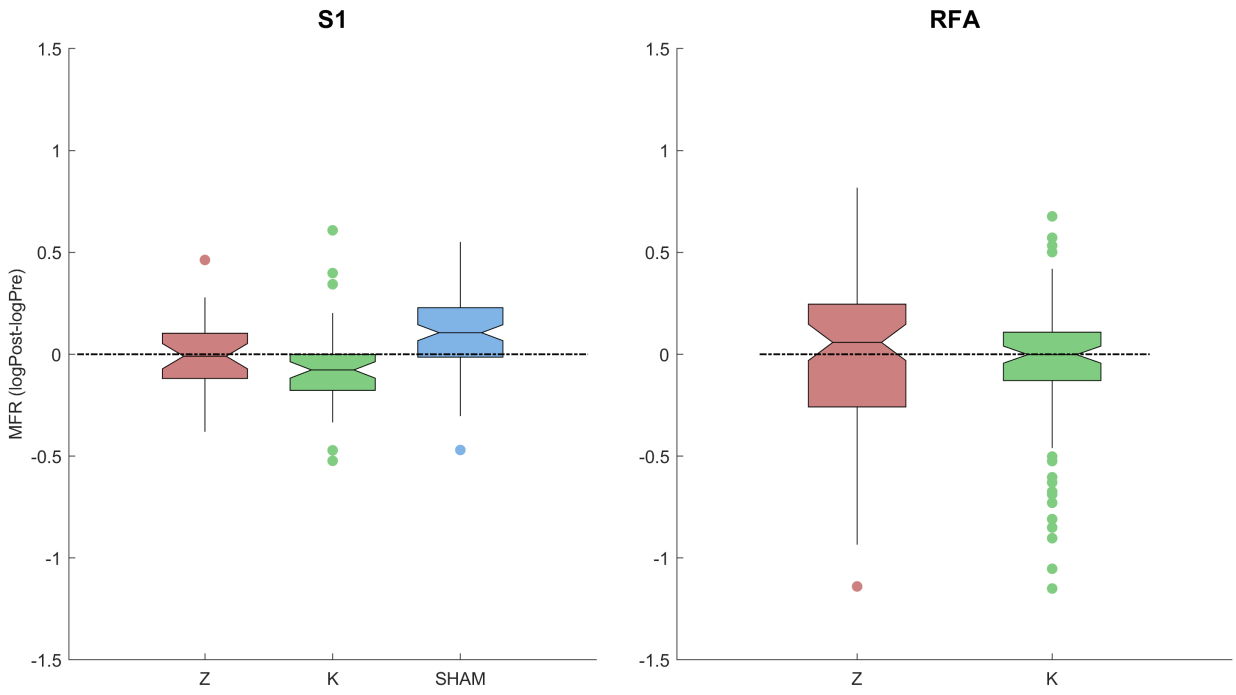

Figure 2.6: *MFR variation of three groups: each box represents the concatenated vector of channel-wise variation in each channel of each animal computed as* $log(MFR_{post}) - log(MFR_{pre})$*. We rejected the null hypothesis of equal median across groups using Friedman's test (*$p = 0.0203$*,* $\chi = 7.7965$*). In S1, the SHAM group (blue) has a clearly different trend compared to the others. This differences are not confirmed by the post hoc Tukey's HSD test. No statistically significant differences are found in RFA*

Friedman's test indicated a significant difference among the groups, rejecting the null hypothesis that all groups have the same mean (p = 0.0203, F=0.0042). However, post hoc pairwise comparisons using Tukey's HSD test did not reveal any significant differences between the groups. Two-ways repeated measures ANOVA does not reveal any statistical differences between the two kinds of stimulation (p=0.949, F=0.0042).

2.5.3 Stimulation changes the variability in firing pattern

Assessing the variations in the LvR, both types of stimulation resulted in an increase in this metric compared to the SHAM group, where LvR remained relatively stable. An increase in LvR indicates a shift toward a more "bursty" activity pattern. The differences between the groups were confirmed by the Friedman test ($p = 1.6463 \times 10^{-9}$, $\chi = 40.4494$), though post hoc pairwise comparisons using Tukey's HSD test failed to detect significant differences

between the groups (Figure 2.7). Two-ways repeated measures ANOVA does not reveal any statistical differences between the two kinds of stimulation (p=0.417, F = 0.6801).

2.5.4 Excitability of the tissue decreases after stimulation

Trying to quantify the excitability of the tissue pre- and post- stimulation, the area of the normalized PSTH in a window of 0.3 seconds binned at 0.001 seconds after the stimulus is calculated channel by channel. This estimate is done only in the connectivity mapping phase, since there the inter-stimulus interval is constant. In general, PSTH seems to spontaneously decrease across the recording in S1, but the stimulation induces major decreasing that the spontaneous one. Also here, these differences were statistically confirmed by the Friedman's test ($p = 3.5027 \times 10^{-7}$, $\chi = 29.7292$) but no positive results are provided by the post hoc Tukey's HSD test. In RFA no noticeable differences can be observed. Two-ways repeated measures ANOVA does not reveal any statistical differences between the two kinds of stimulation (p=0.495, F = 0.4279).

2.5.5 Stronger engagement of the network using Zybo Z7-20 board

In contrast with the expected results, the Zybo Z7-20 provides an higher I/O correlation with respect to the Kria KV260 board (two-ways repeated measures ANOVA, $p = 4.601 \times 10^{-14}$, F = 33.1015). We tested the null hypothesis that the two boards invoke the same network response, in terms of correlated firing of the network (considered here as the "Output") with the stimulus time-series (considered here as the "Input"). Figure 2.9a reveals a distinct difference between evoked responses in both S1 and RFA, with the Z group showing a notably stronger network engagement. This suggests that the Zybo board has a greater capacity to activate the network. This finding can be attributed to the fundamentally different stimulus trains: as mentioned earlier, the Z group received four times as many stimuli as the K group. Moreover, the temporal dynamics of these stimuli differ significantly, with the K group experiencing a more compressed stimulus pattern compared to the Z group.

The statistical analysis shows no significant interaction between area and group (p = 0.2724, F = 3.311), indicating that the stimulation effect occurs in both areas without distinguishing between the two boards. Notably, the induced activity in RFA is much lower than in S1.

In conclusion, contrary to expectations, the Kria board did not effectively engage the network

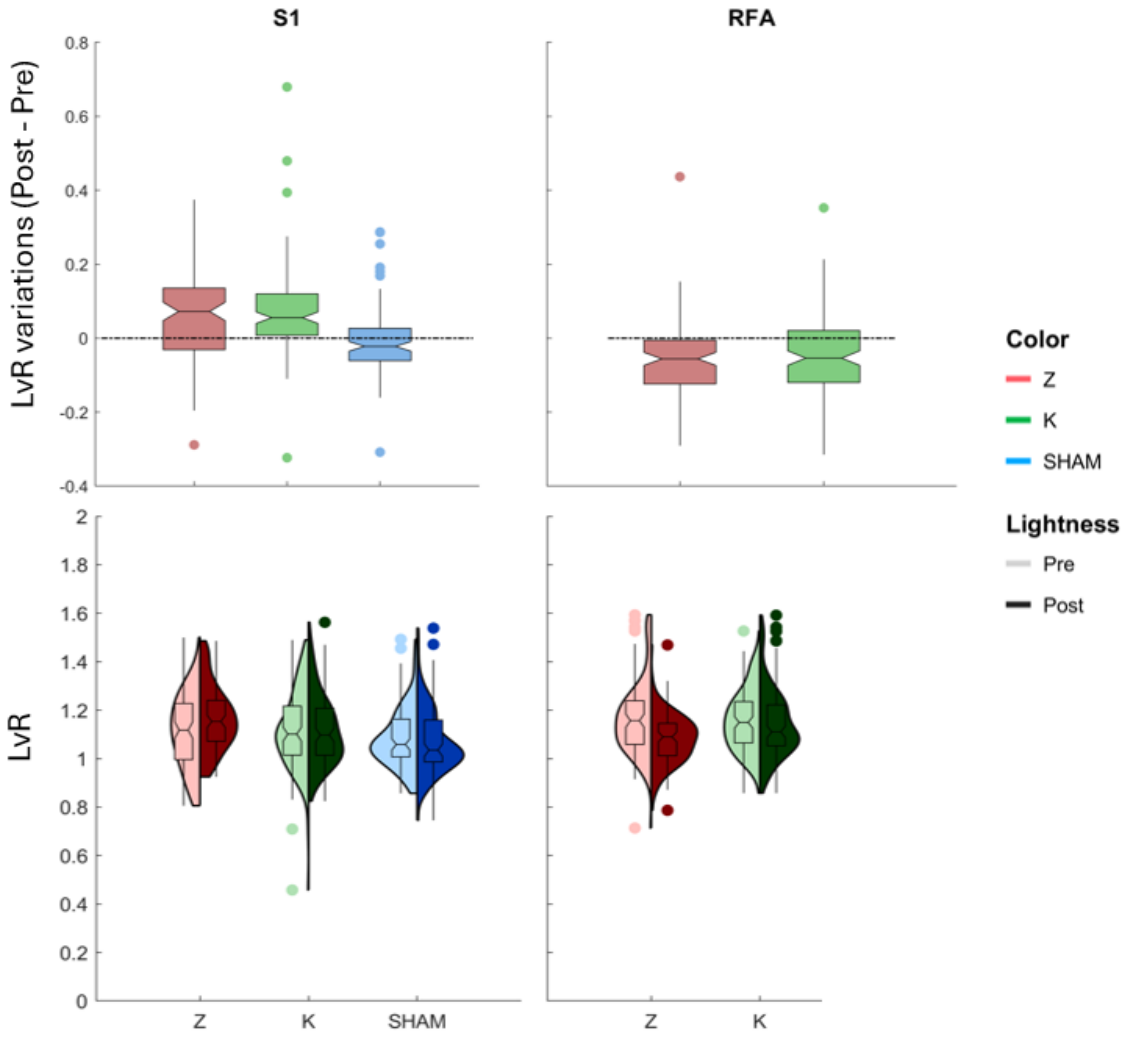

Figure 2.7: *LvR variations (top) and pre- vs. post- distributions (bottom) of the LvR. LvR changes significantly among groups, (Friedman's test,* $p = 1.6463 \times 10^{-9}$, $\chi = 40.4494$*), but post hoc evaluation failed in detecting the group-wise statistics. After stimulation, it seems to increase in S1 (compared to SHAM), pushing the network toward a more "bursty" spiking activity (closer to 1.5) and decrease in RFA, indicating a more random activity (closer to 1).*

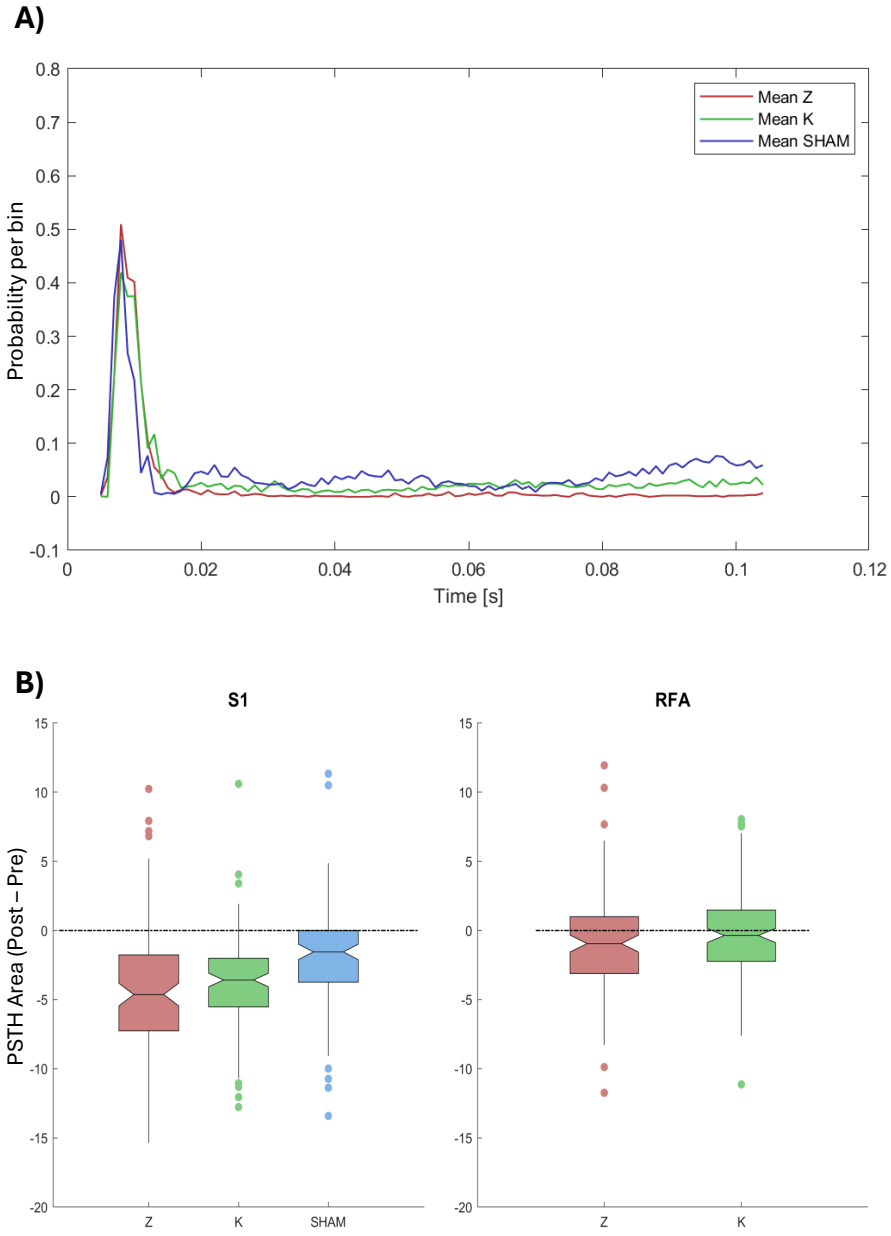

Figure 2.8: ***A)*** *Example traces of the mean PSTH across channels in S1 taken from one random animal per group, during the post stimulation connectivity mapping phase.*

B) *Variation of the PSTH area across groups, each box represents the normalized value taken channel by channel concatenating each animal belonging to this group. PSTH area changes significantly among groups, as the Friedman's test rejected the null hypothesis that all data have equal median ($p = 3.5027 \times 10^{-7}$, $\chi = 29.7292$). Again, post hoc testing failed in finding statistically significant variations pairwise. Looking at the boxes, stimulation seems to induce slightly bigger decreasing in the PSTH area, indicating a decreasing in the excitability of the tissue.*

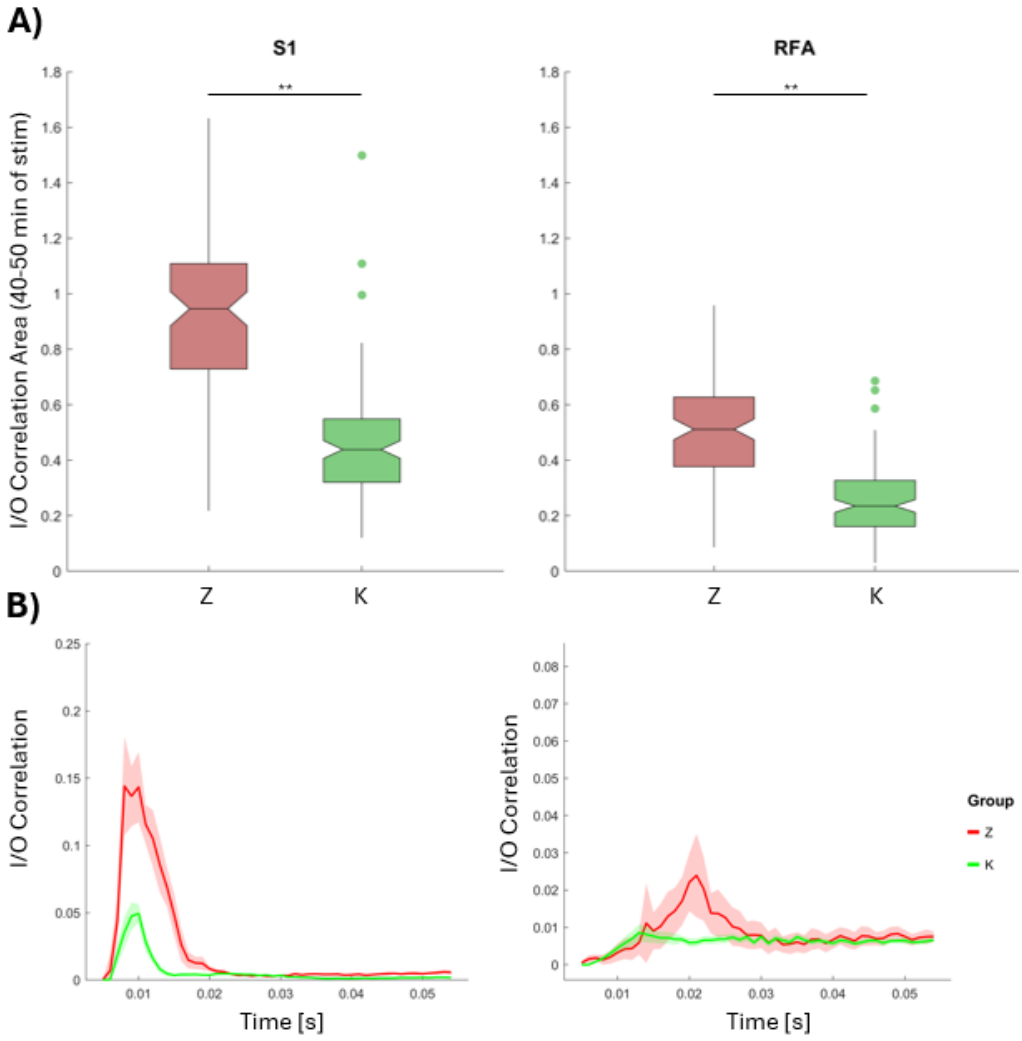

Figure 2.9: *Input/Output correlation between the stimuli and the recorded activity.*

A) *Area of the I/O correlation taken during the 40 to 50-minute stimulation period. Two-ways repeated measures ANOVA reveals a significant difference between the two kinds of stimulation (** means p¡0.001, $p = 4.601 \times 10^{-14}$, $F = 33.1015$). In conclusion, the Zybo board is noticeably better in the engagement of the network.*

B) *Example I/O correlation vectors in both S1 and RFA for the Z group (red) and the K group (green). The solid lines represent the mean, while the shaded areas indicate the 95% confidence intervals around the mean, calculated across all channels from a randomly selected animal, during the 40 to 50- minutes of stimulation. Clearly the evoked response is bigger in the rats stimulated with the Zybo Z7-20 board. Axes here are voluntarily different to appreciate the different amplitudes of correlation peaks in the two areas: S1 (the stimulated area) has a high correlation with the stimulus while in RFA this correlation is almost absent.*

to the same extent as the Zybo board. However, the findings confirm that different types of stimulation can have varying impacts on the network's evoked activity. This suggests that improving the compatibility between the board and the biological network being stimulated could enhance communication between the two. To achieve this, fine-tuning the interaction is necessary, which could likely be accomplished through automatic optimization. In the following section, the design of an Indicator-Based multi-objective Evolutionary Algorithm is introduced, along with its results in mimicking the network.

Part II

Optimization

Chapter 3

Multi-Objective Optimization using Indicator-Based Evolutionary Algorithm

One of the main challenges in computational neuroscience is to tune the parameters of a model it is able to reproduce some specific patterns of activity derived from experimental data.
The milestone of this process is the *hand* tuning: neuroscientists experiment with different sets of parameters, randomly navigating the parameter space, relying mainly on their prior knowledge to approximate the target activity.
However, as the network intricacy and the complexity of the neuronal model increase, the parameter space becomes increasingly high-dimensional, making *hand* tuning progressively more challenging. For this reason, automated tuning of neuronal model parameters is an active topic of research and different algorithms have been published (Van Geit, De Schutter, and Achard 2008).
The construction of a tuning algorithm for neural models can be divided into two stages: the first involves the choice of an error function (also called fitness function or cost function) that quantifies how well the model output matches the experimental data, the second is to pick an optimization algorithm to find minima of the error function. These two decisions are often independent, as most optimization algorithms can be paired with any error function and vice versa.
Recent studies have demonstrated the effectiveness of evolutionary algorithms in solving this

kind of problems (Druckmann et al. 2007; Masoli et al. 2017).

3.1 Introduction to Evolutionary Algorithms

Evolutionary algorithms (EA) are stochastic algorithms that are based on on the concepts of natural selection and survival of the fittest (Eiben and Smith 2015). Drawing from Darwin's concept of *"natural selection"*, these algorithms treat combination of parameters as individuals, each with a specific fitness (or error value, representing how closely the individual approaches the optimal solution) that influences its likelihood of survival and reproduction.

All evolutionary algorithms follow a similar structure. Initially, a random population of individuals is generated. The fitness of each individual is then assessed. Based on their fitness, certain individuals are chosen as parents, which will reproduce and undergo mutations. Reproduction involves combining the parameters of multiple parents, while mutation refers to a random alteration in the parameters of a single individual. The new individuals created through reproduction and mutation are evaluated and compared to the original population. A selection of the fittest individuals is then made to give rise to a new population. This process is repeated until a specified stopping condition is satisfied (Figure 3.1).

In the many variations of these algorithms, we can distinguish two main branches: *genetic algorithms* and *evolution strategies*.

Genetic algorithms (Holland 1992) are the most popular type of EA. They are termed "genetic" because each individual here is represented by a single finite binary string and the recombination step takes place by combining the bit strings of parents, while the mutation happens by flipping one or more bits, mimicking what happens to genetic materials during *meiosis*. For example, if the parameters of one individual are vectors of integers, each individual can be represented as a concatenated string of bit.

This encoding step makes genetic algorithms particularly well-suited for handling integers, while their application becomes more complex when dealing with floating-point numbers, as their binary representation and performing operations like recombination can be less straightforward.

Evolution strategies (Rechenberg 1989), on the other hand, work with real numbers, making the steps of the algorithm more natural than flipping bits. From an algorithmic point of view,

evolution strategies are optimization methods that sample new candidate solutions stochastically, most commonly from a multivariate normal distribution, as described in Section 3.2 (Hansen, Arnold, and Auger 2015).

For this reason, they are more appropriate for neuroscience applications, since the neuronal model parameters are usually floating-point numbers.

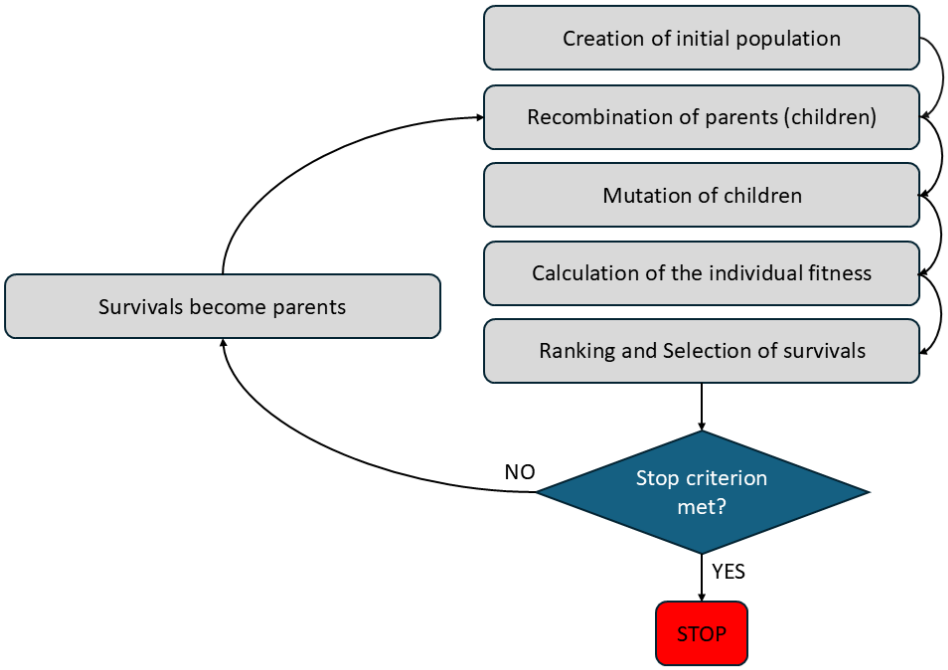

Figure 3.1: *General structure of an Evolutionary Algorithm*

3.2 Recombination and Mutation

Recombination and mutation are the key processes in an EA. They serve as the primary mechanisms by which the algorithm navigates the solution space, making the choice of how to proceed in these steps crucial.

3.2.1 Mutation

Regarding mutation, the simplest approach is the *"Isotropic Mutation"* where a point-symmetric perturbation is introduced to the individual x_i by adding a vector r_i drawn from a Gaussian distribution. As a result, the new individual is: $x_i + \mathcal{N}(0, \sigma)$. The standard deviation σ is one of the hyperparameters of the algorithm, since it represents the mutation rate: a higher σ encourages broader exploration of the solution space, though it risks deviating too far from

the optimum.

Various mutation strategies have been developed, including those with adaptive mutation rate, which starts high and decreases towards the end of the optimization process (Hansen, Arnold, and Auger 2015), such as the Non-Local Derandomized Step-Size Control (CSA).

In CSA, the authors successfully implemented a method that takes into account the global evolution of the search through the vector parameter $\mathbf{s}_\sigma$, referred to as *search path*, to optimize the convergence of the algorithm (Ostermeier, Gawelczyk, and Hansen 1994). This parameter memorizes the previous mutations, changing the mutation rate σ (that is in general n-dimensional) basing on the sum of consecutive successfully mutation steps, which are the $\mathbf{z} = \mathcal{N}(0, \mathbf{I})$ summed to the old individual $\mathbf{x_2} = \mathbf{x_1} + \mathbf{z}\sigma$. As a result the algorithm adapts the mutation rate σ after the selection steps, to drive the search optimally toward the objective (Hansen, Arnold, and Auger 2015). See Appendix 3 for details.

3.2.2 Recombination

Recombination, also called cross-over, is the process that combines information from two parents to generate new offspring. In the context of evolution strategies, where the search deals with real numbers, designing an algorithm able to obtain an individual from with values slightly different from one of the parents is not straightforward. In most cases, a probability distribution centering the parent solutions is assumed and two children solutions are created based on that probability distribution.

One method that has gained significant recognition is the *"Simulated binary cross-over"* (SBX) (Deb 1995). To understand SBX, it's essential to first introduce the single-point binary crossover, a technique from Genetic Algorithms that leverages binary string encoding in individuals. In this method, each parent string is split at a designated point, and the segments are swapped to generate offspring.

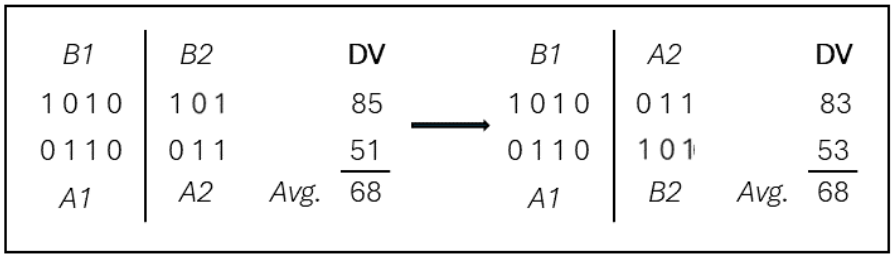

Figure 3.2: *Single-point cross-over on two random strings. DV is the Decoded Value. The average DV is the same for parents and children.*

The SBX operator has been developed based on the above method, trying to replicate it in terms of search power, measure of how flexible the operator is in creating an arbitrary point in the search space. In order to implement this crossover operator with those characteristics, for any two parent solutions p_1 and p_2 a non-dimensionalized spread factor β has been defined as the ratio of the spread of created children solutions c_1 and c_2 to that of the parent solutions as follows:

$$\beta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right| \tag{3.1}$$

Basing on this value we can classify the cross-over operator in three different classes:

1. If $\beta < 1$ it is a contracting cross-over: the children points are enclosed by the parents points.

2. If $\beta > 1$ it is an expanding cross-over: the children points enclose the parent points.

3. If $\beta = 1$ it is a stationary cross-over: the children are as distant as the parents.

It has been observed that in the binary single-point crossover operator, both children solutions lie either inside (contracting crossover) or outside (expanding crossover) the region bounded by the parent solutions (Deb 1995). Thus, on an average, the overall probabilities of contracting and expanding crossovers are the same. Again referring to single-point cross-over, it has also been possible to calculate the probability of creating a pair of children solutions having a certain β (Deb 1995). That probability distribution has been approximated by a polynomial

probability distribution:

$$P(\beta) = \begin{cases} 0.5(n+1)\beta^n & \text{if } \beta \leq 1; \\ 0.5(n+1)\frac{1}{\beta^{n+2}} & \text{otherwise.} \end{cases} \tag{3.2}$$

The advantage of using a probability distribution as a function of β is that the created children solutions are relative to the parent solutions. If the parent solutions are distant, children solutions far away from the parent solutions can be created. On the other hand , if the parent solutions are close to each other, the children solutions, in general, cannot be far away from the parent solutions. The n variable is the one that controls this property: the higher n the higher the probability of children close to the parents.

More specifically, in order to create two children solutions c_1 and c_2 from the parent solutions p_1 and p_2 using the above probability distribution, the following procedure is used.

- Create a random number u between 0 and 1
- Find a β' for which the cumulative probability

$$\int_0^{\beta'} P(\beta)\, d\beta = u \tag{3.3}$$

- Knowing the value of β, the children points are calculated as

$$\begin{aligned} c_1 &= 0.5[(p_1 + p_2) - \beta'|p_2 - p_1|] \\ c_2 &= 0.5[(p_1 + p_2) + \beta'|p_2 - p_1|] \end{aligned} \tag{3.4}$$

This algorithm can be also adapted to generate offspring within predefined bounds, if those bounds are known in advance, by restricting the distribution of the spread factor β to a specific range (Deb and Kumar 1995). Due to its flexibility and control, this operator has been selected for use in this thesis.

3.3 Multi-Objective Optimization (MOO)

Considering the problem of finding a neuronal network model able to reproduce experimental neural activity, a multi-objective approach can be adopted, as it has been successfully applied in recent years with remarkable results (Druckmann et al. 2007; Masoli et al. 2017).

An optimization problem is defined as a Multi-Objective Optimization problem (MOO) if more than one error function is used and one considers them in parallel, not by simply summing them. In this way several criteria can be optimized almost separately, avoiding the conflicts between them.

For example comparing the Mean Firing Rate and the Inter-Spike Interval distribution of two networks can be not straightforward because it's not guaranteed that every Mean Firing Rate can be associated with every Inter-Spike Interval, leading to potential incompatibilities. However, considering each objective's distance separately enables the search for a more balanced and optimal final solution.

The main difference between single objective optimization and the MOO is in the possible relations between two solutions. In a single objective problem, a solution can be either better or worse than another, depending on whether its error value is lower or higher. This is not the case in multiple objective problems. The relation of better or worse is replaced by that of *domination*.

Considering M objective functions, one solution x_1 is said to dominate the another x_2 if:

$$f_j(x_1) \leq f_j(x_2) \; for \; all \; j = 1 \ldots M \tag{3.5}$$

$$f_k(x_1) \leq f_k(x_2) \; for \; at \; least \; one \; k \in \{1 \ldots M\} \tag{3.6}$$

In other words a solution dominates on another if it has at least one better objective than the other solution, and it does not worse than the other for all the remaining objectives. If solutions do not dominate each other, it means that they represent different trade-offs in approximating the final goal. A MOO does not yield a single optimal solution but rather a set of optimal solutions that are not dominated by any other. This set is known as the *Pareto front*. The user has the final word to select one or several of them as the best fit for his specific issue.

3.4 Indicator-Based Evolutionary Algorithm (IBEA)

Starting from '90, the challenge of designing a successful Multi-Objective Evolutionary algorithm was by far a prominent research topic (Falcón-Cardona and Coello 2020). Several algorithms were developed, each with its advantages and disadvantages (Deb 2001).

One solution that received a remarkable attention is the one proposed by Zitzler and Künzli 2004 at the ETH of Zurich: an *Indicator-Based Evolutionary Algorithm* (IBEA).

In EA, ranking the entire population is necessary for selecting survivors. However, in a MOO scenario, as previously introduced, this is not possible. To address this, a metric is needed to evaluate the quality of each solution. This is where the concept of an Indicator (I) comes into play—a function that maps a Multi-Objective solution to a real number, allowing solutions to be ranked. Zitzler et al. introduced two types of Binary Indicator (binary means that it compares two solutions). The simplest one is the binary additive ϵ-indicator:

$$I_{\epsilon^+}(x^1, x^2) = min_\epsilon\{f_i(x^1) - \epsilon \leq f_i(x^2) \; for \; i \in \{1, ..., n\}\}, \tag{3.7}$$

with f_i the i-th objective function to be minimized. It finds the minimum value ϵ for which the individual x^1 dominates individual x^2 as the maximum distance between the objectives of the two solutions. As a consequence, the smaller is ϵ the more x^1 is close to the optimum. Indeed, this value is positive if x^1 does not dominate on x_2 and negative in the other case. This indicator is not commutative so:

$$I_{\epsilon^+}(x^1, x^2) \neq I_{\epsilon^+}(x^2, x^1) \tag{3.8}$$

To rank the solutions preserving the ones with the highest fitness value compared to the whole population, this fitness operator is introduced:

$$F(x^1) = \sum_{x^2 \in P/x^1} -e^{-I_{\epsilon^+}(x^2, x^1)/\kappa}, \tag{3.9}$$

which sums all the exponentially mapped indicators of the whole population with respect to the single individual. Being negated, the individual with the highest F goes first in the ranking. The κ is a scaling factor, that is very sensitive to the scale of the problem. For this reasons each objectives and the successive indicators are scaled over their maximum and minimum values. In this case, $\kappa = 0.05$ was assessed to be the best value.

The complete algorithm, as proposed in Zitzler and Künzli 2004, can be described as follows:

Input:

α (population size)

N (maximum number of generations)

κ (fitness scaling factor) = 0.05

Output:

A (Pareto set approximation)

Step 1: Initialization:

Generate an initial population P of size α;

Set the generation counter m to 0;

Step 2: Fitness assignment:

First scale objective and indicator values, and then use scaled values to assign fitness values:

1. Compute objectives $f_i(x)$ for all $x \in P$ for $i = 1 \ldots M$ (M=number of objectives)
2. Determine for each objective f_i its lower bound $\underline{b_i} = min_{x \in P}\ f_i(x)$ and its upper bound $\overline{b_i} = max_{x \in P}\ f_i(x)$.
3. Scale each objective to the interval $[0, 1]$, i.e., $f_i(x) = (f_i(x) - \underline{b_i})/(\overline{b_i} - \underline{b_i}))$.
4. Calculate indicator values $I(x_1, x_2)$ using the scaled objective values f_i' instead of the original f_i, and determine the maximum absolute indicator value $c = max_{x_1, x_2 \in P}\ |I(x_1, x_2)|$.
5. For all $x_1 \in P$ set $F(x^1) = \sum_{x_2 \in P/x^1} -e^{-I(x^2, x^1)/\kappa}$.

Step 3: Environmental selection:

Iterate the following three steps until the size of population P does not exceed α:

1. Choose an individual $x^* \in P$ with the smallest fitness value, i.e., $F(x^*) \leq F(x)$ for all $x \in P$.

2. Remove x^* from the population.

3. Update the fitness values of the remaining individuals, i.e., $F(x) = F(x) + e^{-I(x^*,x)/(c \cdot \kappa)}$ for all $x \in P$.

Step 4: Termination:

If $m \geq N$ or another stopping criterion is satisfied then set A to the set of decision vectors represented by the non-dominated individuals in P . Stop.

Step 5: Mating selection:

Perform binary tournament selection with replacement on P in order to fill the temporary mating pool P.

Step 6: Variation:

Apply recombination and mutation operators to the mating pool P and add the resulting offspring to P . Increment the generation counter ($m = m + 1$) and go to Step 2.

In summary: the algorithm firstly generates a random population α, each individual objective function is computed separately and, based on that, the indicator based fitness value is determined; the individuals are chosen randomly in couples and the one with the highest fitness is added to the mating pool; inside this pool the individuals are again matched and the recombination/mutation steps take place, generating the offspring; finally the whole population is ranked and only the α best individuals are kept; this steps are repeated until a criteria is satisfied. The running-time complexity is $\mathcal{O}(\alpha^2)$ (Zitzler and Künzli 2004).

3.5 Objective Functions: Median Distance and Kullback-Leibler Divergency

Last but not least critical step is choosing the method to determine the distances from the objective. Various objective functions can be employed, ranging from simple linear Euclidean distances to more complex non-linear functions. In the context of comparing two populations

of neurons - one artificial and one biological - it's important to use metrics that specifically compare the distributions of values (or their statistical parameters) between the populations. In this case, two different strategies were adopted:

1. Distance between the median values of the distributions.

2. Kullback-Leibler Divergency between the distributions.

Commonly used in information theory, the Kullback-Leibler Divergency (Kullback and Leibler 1951) is an asymmetric measurement that quantifies the distance between two probability distributions. The DKL of Q from P (written as $D_{\mathrm{KL}}(P\|Q)$ is formally defined as:

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log_2 \left(\frac{P(i)}{Q(i)} \right). \tag{3.10}$$

Using these two metrics as objective functions enables to differently compare individuals and objective: since, in this case study, the final goal is to match the activity of two neural networks, the median allows to quantify a single statistical summary that captures the overall behavior of the entire network, whereas the DKL offers a more precise assessment, relying on the single neuron contribution.

3.6 IBEAforBioemus - Optimization Setup

This section will present the implemented version of the algorithm applied on the FPGA-based Spiking Neural Network *Bioemus*.

As said in Chapter 1, *Bioemus* is implemented directly on the low-cost platform AMD Xilinx Kria KV260 carrier board, which features both a Programmable Logic part (the FPGA) and a Processing System part running a Canonical Ubuntu operating system. The algorithm is designed to execute directly on this Processing System. This approach offers performance comparable to using an external Windows client PC that communicates with the board to transfer configuration files. While the Windows PC can generate these files more quickly, the process is slowed by the file transfer between the PC and the Ubuntu system on the board, creating a bottleneck.

Summarizing previous information, this algorithm is an Indicator-Based Evolutionary Algorithm, derived formally from Zitzler and Künzli 2004 (Section 4.1 $IBEA_{\epsilon+}$); two version are devised: one that computes objectives as distances between medians and the other using

Kullback-Leibler Divergency.

Procedure is the same as described in Section 3.4, using Simulated Binary Cross-Over and Isotropic Mutation as recombination and mutation strategies, respectively with probability of 1 and 0.2, to promote exploration of the search space Section 3.2.

Each time an individual is generated (in the initial population or as a generated child), several steps are involved to compute the objectives:

1. Each individual has a unique set of parameter that changes the software and hardware configuration files for the FPGA.
2. Simulate each individual (configuration) for a specified duration.
3. Analyze the resulting SNN activity to extract various electrophysiological biomarkers.
4. Calculate the distances between each biomarker and the corresponding goal (the biomarkers of the BNN) individually and use these as objectives.

These objectives are then used as inputs to determine the fitness of each individual, enabling the population to be ranked accordingly. The general schema can be seen as followed:

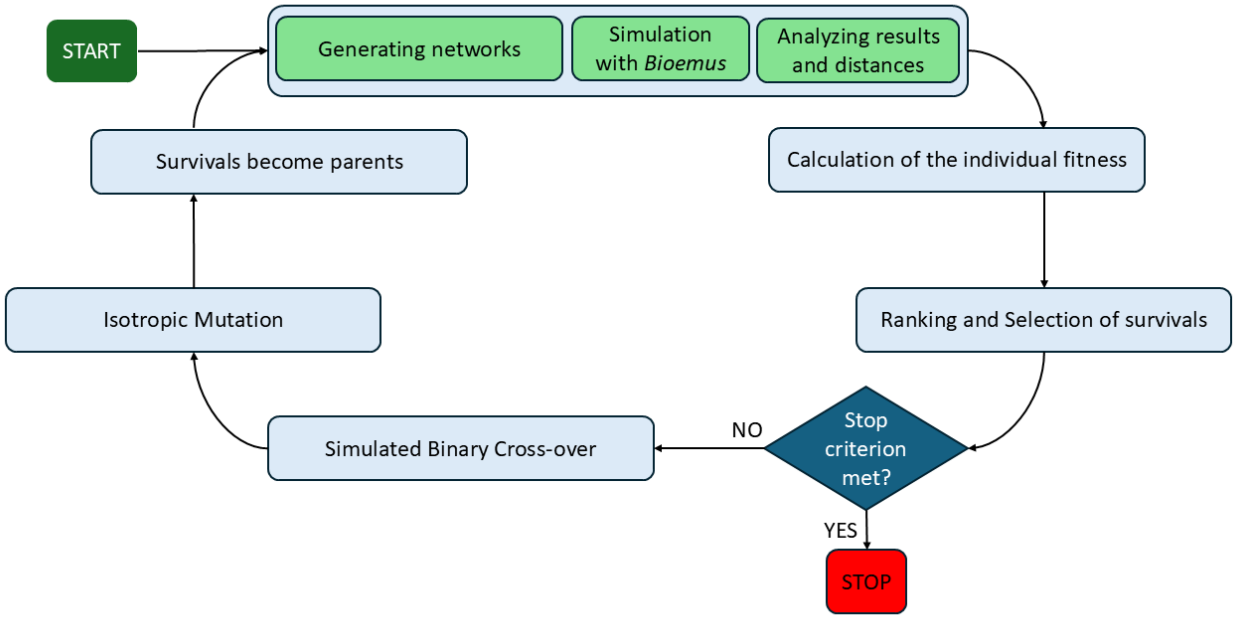

Figure 3.3: *General structure of IBEA for Bioemus. The algorithm generates an initial random population, where each individual is a different configuration of the SNN. The simulation produces a file where time stamps of spikes are saved. This file is analyzed and the distances from the goals are computed. Then, the process goes on in the same way as in Figure 3.1. This algorithm adopts the simulated binary crossover and the isotropic mutation as recombination strategies*

All the individual networks are composed by 1024 Hodgkin-Huxley neurons, that are randomly assigned as excitatory or inhibitory. Inhibition is modeled using Fast Spiking neurons connecting by GABAa and excitation by Regular Spiking neurons connecting by AMPA and/or NMDA. Among all the configurable parameters in *Bioemus*, six are selected as the most likely to influence population dynamics, and their search is limited in an interval (according to physiological values(Uhlenbeck and Ornstein 1930; Destexhe, Z. F. Mainen, et al. n.d.; Isaacson and Scanziani 2011; Myme et al. 2003)).

These parameters are: the ratio of inhibitory to excitatory neurons (in the interval 0.1-0.5), the maximum probability of connectivity used in calculating network connections (in the interval 0.1-1), considering that it should have a Small-World topology), the ratio of AMPA to NMDA excitatory synapses (in the interval 0.1-1), the synaptic weight of AMPA synapses (in the interval 0.1-1), the synaptic weight of NMDA synapses (in the interval 0.1-1), and the

mean value applied in the Ornstein-Uhlenbeck process to model synaptic noise (in the interval 0.01-0.1). As standard deviation of the isotropic mutation step is used the interval divided by 15.

Regarding biomarkers, three are chosen: the Mean Firing Rate (MFR), the Inter-Spike Interval(ISI) and the Mean Bursting Rate (MBR). These same biomarkers are extracted from a 10-minute recording of spontaneous intracortical activity in the S1 area of five adult male Long-Evans rats, using a 16-channel electrode array. The data are pre-processed and sorted to isolate single-unit activity. Across the five rats, a total of 194 neurons are identified and concatenated to form a single, larger biological neural network (BNN).

When measuring distances using DKL, to ensure comparability between the simulated neural network (SNN) and the BNN, which contain different numbers of neurons, the biomarker distributions for both networks were generated using Kernel Density Estimation (KDE). Each neuron contributed a single MFR and MBR value, which were aggregated into a single vector, from which a continuous probability distribution was computed for each biomarker. A different procedure was applied with ISI. Instead of generating a single value per neuron, each neuron produced its own distribution of inter-spike intervals. These distributions were then summed element-wise and normalized by the total number of neurons, resulting in a single, overall ISI distribution for both the simulated neural network (SNN) and the biological neural network (BNN). This approach ensures that the ISI biomarker captures the collective spiking behavior of all neurons in the network. These distributions were directly used as arguments of the DKL.

In contrast, for the median-based comparison, no Kernel Density Estimation (KDE) was applied; instead, the raw data were used directly to compute the medians. The approach for ISI was again different: since ISI is a distribution itself, the Root-Mean Square Error (RMSE) was used to compare the two traces directly. Before this comparison, the ISI distributions were smoothed using a Gaussian filter to account for variability and make the comparison more robust.

Transitioning from the generation of one network to the next takes approximately 25 seconds. The total time required to simulate and process the results of each network is therefore the sum of this 25-second plus the selected simulation time.

3.7 Results

3.7.1 Hyper-parameters tuning

Several trials were conducted to fine-tune the algorithm's hyperparameters. The initial population size was set to 200 individuals (α), as this number strikes a balance between limiting execution time and reducing the occurrence of repeated individuals in the population. A smaller initial population would lead to the same high-fitness parents being repeatedly selected for the mating pool, which hinders exploration of the search space.

Additionally, 20 offspring were generated per iteration, a value found to offer a good trade-off between ensuring reasonable convergence and maintaining efficient runtime.

Regarding the simulation duration, trials were conducted with 5, 30, and 60 seconds, with different seeds. While longer simulations increase the variability of potential activity patterns, the distributions of key metrics (MFR, MBR, ISI) remained consistent across all three durations. Therefore, 5 seconds is chosen as the optimal simulation time, allowing for faster processing without compromising the reliability of the results.

Some trials resulted in several winning solutions where many neurons remained largely silent (producing no spikes or just one spike). This led to poor MFR and MBR values. However, ISI metric sometimes appeared favorable, as ISI is normalized by definition and does not depend on the number of active neurons. To ensure an adequate level of network activation, a threshold was introduced, requiring at least 850 neurons to fire a minimum of 2 spikes within the 5-second simulation. Choosing the optimal number of iterations is one of the most challenging aspects of tuning the algorithm. A sufficient number of iterations is crucial to reach a good approximation of the target solution. However, the number of iterations is also a key factor in increasing execution time, as each additional generation requires the simulation of 20 new networks. To balance this, three trials were conducted with 100 iterations to assess whether the algorithm consistently converges. All three trials reached approximately the same non-zero , stable fitness value, at 45 iterations, suggesting that the generated offspring are generally performing at a similar level to their parent solutions, saturating the solutions (Figure 3.4).

Given these considerations, the total execution time stabilizes at around 8 hours to produce reasonable results. However, this duration can vary significantly depending on the specific use case. If less stringent final outcomes are acceptable, the process can require far fewer than

45 iterations. In fact, even after just 10 iterations, the algorithm was capable of generating a few acceptable solutions, allowing for the possibility of selecting the best configurations from the initial results. This suggests that, depending on the required precision, the execution time can be shortened substantially.

The main bottleneck of the algorithm lies in writing and saving the .txt files containing the network configuration details, possibly due to the SSD card used in the Kria KV260 board where the files are stored. Utilizing a faster SSD or optimizing the file saving and reading process in the C++ code could further reduce the overall execution time.

3.7.2 Optimization outcomes and conflicts between objectives

As mentioned earlier, each optimization process generates 200 individuals. However, not all of these individuals are non-dominated, meaning some are outperformed by others. Therefore, an additional selection step is required to filter out only the non-dominated individuals for further analysis. After the selection step, the number of remaining individuals varies depending on the run, with each individual outperforming others in at least one of the three metrics, making the interpretation of the outcome not straightforward. To visualize these results, a radar plot is generated to provide an immediate overview of the run (Figure 3.5). In this radar plot, the outer triangle has vertexes representing the three metrics: MFR, MBR, and ISI. The level of 1 is set as the goal, while each inner triangle represents a different network configuration, showing how close it comes to the optimum.

The position of each vertex is determined by $1-distance_i$, where i corresponds to MFR, MBR, or ISI, and $distance_i$ is the DKL divergence between the target objectives and the extracted features. Configurations with the best distance for each metric are highlighted with distinct colors. As expected, the configuration that performs best in MFR is often not the best in the other two metrics, presenting a trade-off between them. This allows for prioritizing the most important metric based on specific goals.

To identify the best compromise, the configuration that maximizes the area within the triangle is selected as the best overall, representing strong performance across all three metrics. Occasionally, this best configuration may coincide with the configuration that excels in one specific metric, but this is not always the case. The optimization algorithm can also highlight a challenge in achieving a high level of approximation across different metrics simultaneously.

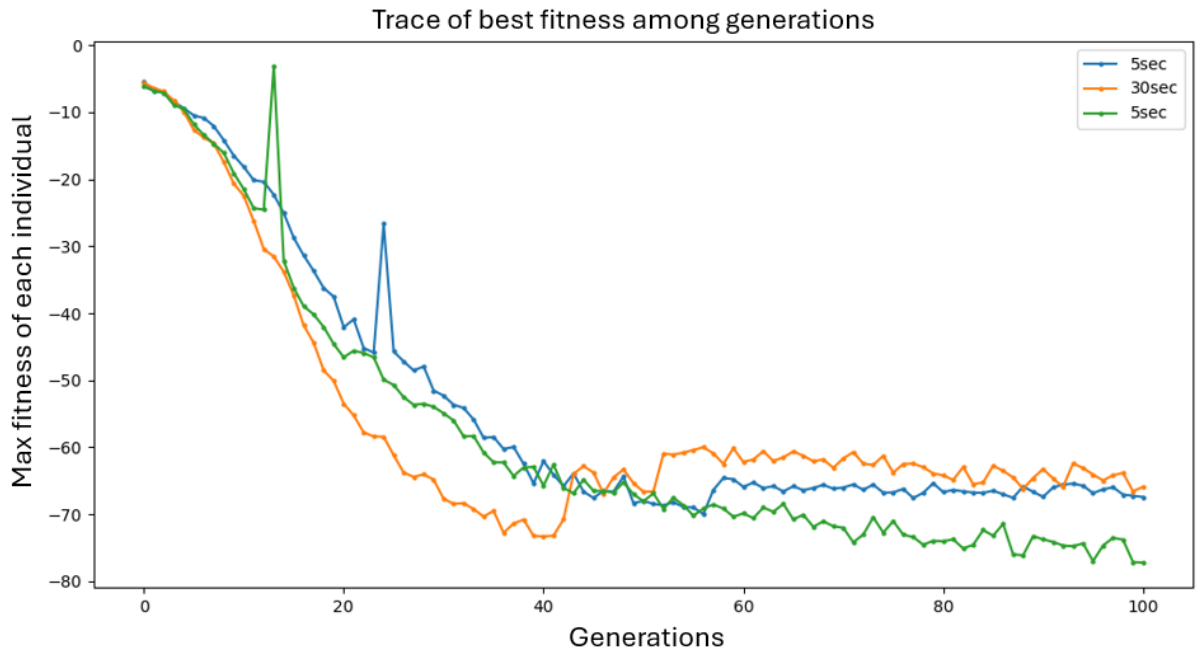

Figure 3.4: *Fitness of best individual across generations. The fitness of the best individual across generations is plotted to track the individual with the highest fitness in each generation. Since fitness is a relative metric, an individual's fitness can fluctuate depending on the population that survives each iteration. As a result, the fitness curve is not strictly monotonic. It is possible for the best fitness value in a previous generation to be higher than in the subsequent one. This reflects that, in some iterations, the best individual may be less dominant compared to others in the population than the previous best in earlier iterations. All three traces (including one with a 30-second simulation time) appear to exhibit an elbow at around the 45th generation. This suggests that 45 iterations could be a reasonable trade-off, balancing performance with computational efficiency, and serving as a good choice for the number of generations.*

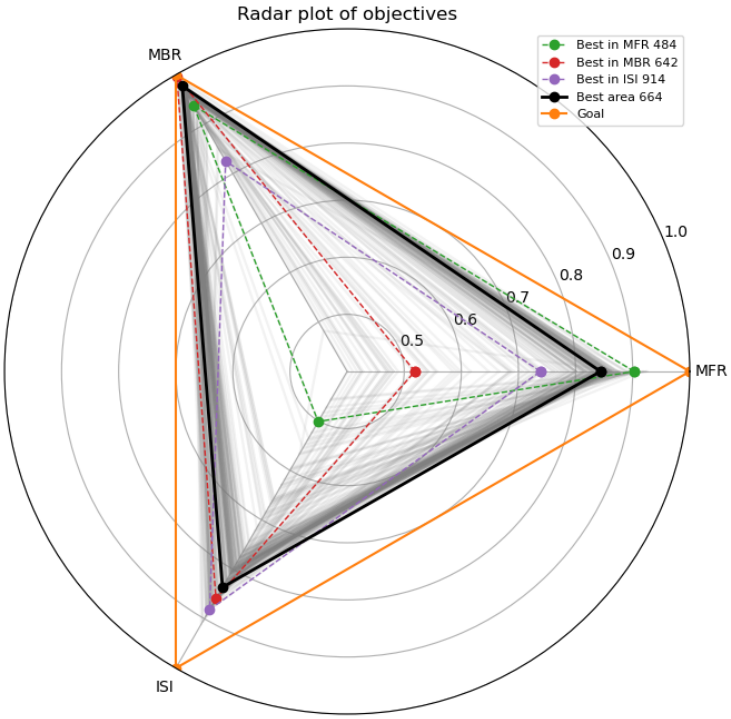

Figure 3.5: *Visualization of the optimization results. In this radar plot, the outer orange triangle represents the target values, where each vertex has a distance of 1 from the center, corresponding to the goal for all three metrics: MFR, MBR, and ISI. The inner triangles represent the performance of individual configurations, with each vertex's distance from the center set to* $1 - distance_i$ *where* i *is the metric and* $distance_i$ *is calculated using the Kullback-Leibler (DKL) divergence or the difference between the medians of the goal and individual distributions. Dashed lines highlight the configurations that have the smallest distance from the goal for each metric. The legend also displays the number of the corresponding winner configuration, indicating the order in which they were generated. For instance, a label like "664" means that the configuration closest to the MFR goal is the 664th configuration generated by the algorithm. Solid black line indicates the configuration with the largest triangle area, representing the network that achieves the best compromise across all three metrics (MFR, MBR, and ISI). In this specific run, we can observe that there is a conflict between reaching a good MFR together with a good ISI. The best solution represents a good trade off between the two metrics.*

As seen in Figure 3.5, configurations that perform best in terms of MFR often perform worst in ISI, indicating a conflict between these two metrics. This issue may be related to the limited neuronal types used in these networks, specifically regular spiking and fast spiking neurons. To address this, incorporating additional neuronal types available in Bioemus, such as intrinsic burst and low-threshold spiking neurons, could help resolve the conflict and introduce greater variability in the possible patterns, leading to better overall optimizations.

3.7.3 Comparison between DKL and Median distance

Optimization process is developed using two types of objective functions: DKL between distributions or distance between the median of these. Even though the median distance could be an overall good solution, the DKL outperforms in finding more similar distributions, making a direct comparison between distributions. Indeed, having a precise median value of the distributions does not necessarily ensure that the two distributions are similar (Figure 3.6). To make a meaningful comparison between the two methods, the best distribution (as the largest area in the radar plot) from each of the two different optimization runs is chosen as a representative of its population. As anticipated, when computing the DKL between these distributions and the goals, the distributions resulting from the run using DKL as an objective function show better performance. Additionally, the time required to process the two distances is comparable and does not act as a bottleneck in the system. Furthermore, the DKL faster converges toward better solutions.

3.7.4 Parameter convergence

Even if what parameters are chosen as the best is not fundamental to the purpose of mimicking the biological network, with this kind of optimization algorithm it is possible to investigate the parameter space and look which combination of parameters are more suited to reproduce the biological counterpart. The parameter space is composed by: the ratio of inhibitory to excitatory neurons , the maximum probability of connectivity used in calculating network connections, considering that it should have a Small-World topology), the ratio of AMPA to NMDA excitatory synapses, the synaptic weight of AMPA synapses, the synaptic weight of NMDA synapses, and the mean value applied in the Ornstein-Uhlenbeck process to model synaptic noise. As expected, there is no single combination of parameters that leads to a

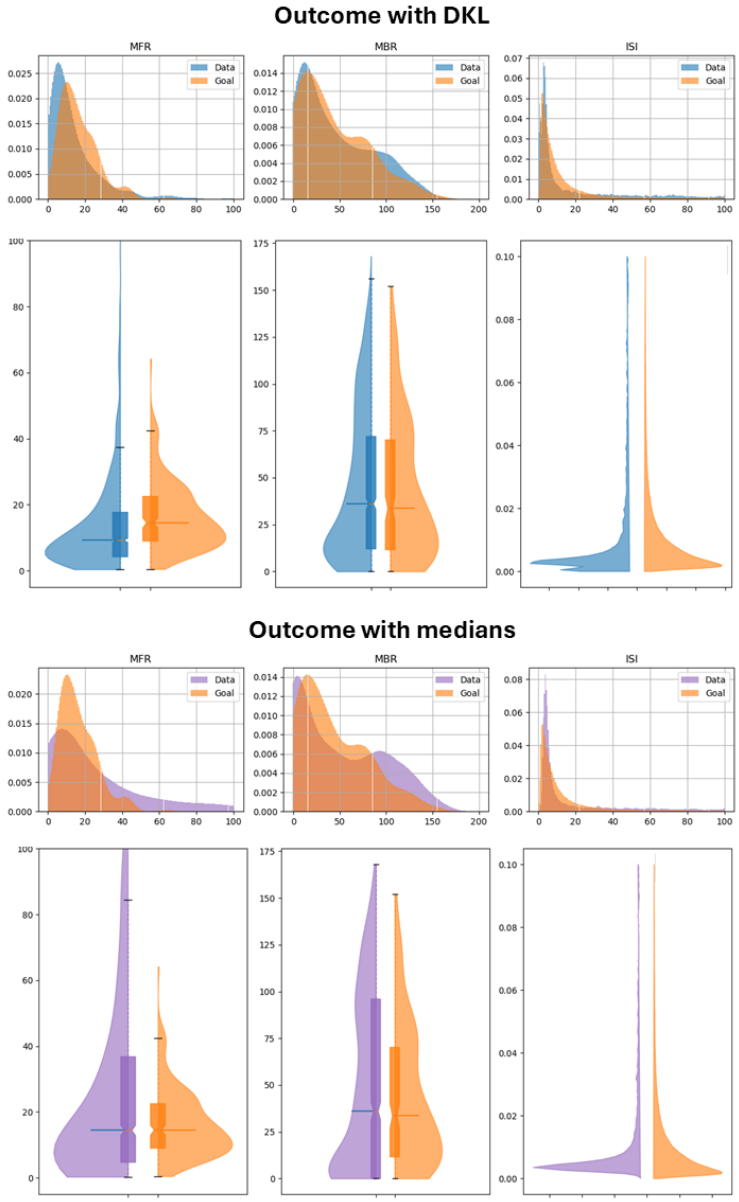

Figure 3.6: *DKL vs. Median distances as objectives functions. Distributions of MFR, MBR, and ISI are visualized using both bar and violin plots. The goal distributions are represented in orange, while the distributions obtained using the DKL method are shown in blue, and those from the median method are depicted in violet. The DKL method performs better in aligning the overall distributions closely to the goal, whereas the median method ensures proximity in terms of the median values but overlooks the shape and spread of the entire distribution.*

specific activity pattern; rather, different parameter combinations can have similar effects on the firing dynamics of the SNN. For example, a higher level of inhibitory neurons could be partially compensated by an increased weight of excitatory synapses. Examining how the parameters of the best-performing individuals are distributed within their ranges may provide insights. If parameters cluster tightly around a single value, this suggests that the value is critical for achieving the goals. In contrast, a wide spread of parameter values may indicate that the parameter has little impact on the optimization process, making it a potential candidate for exclusion in favor of more influential parameters. For instance, as shown in Figure 3.7, the mean value of synaptic noise does not converge toward a single value, suggesting that it plays a minor role in the optimization. This figure is also useful for visualizing the direction of the optimization process. By coloring each point based on the corresponding area value in the radar plot (see Figure 3.5), it becomes possible to distinguish between the best and worst-performing individuals. For certain parameters, a gradient of colors can reveal the trajectory of the optimization, indicating the parameter adjustments that are driving the process towards better results.

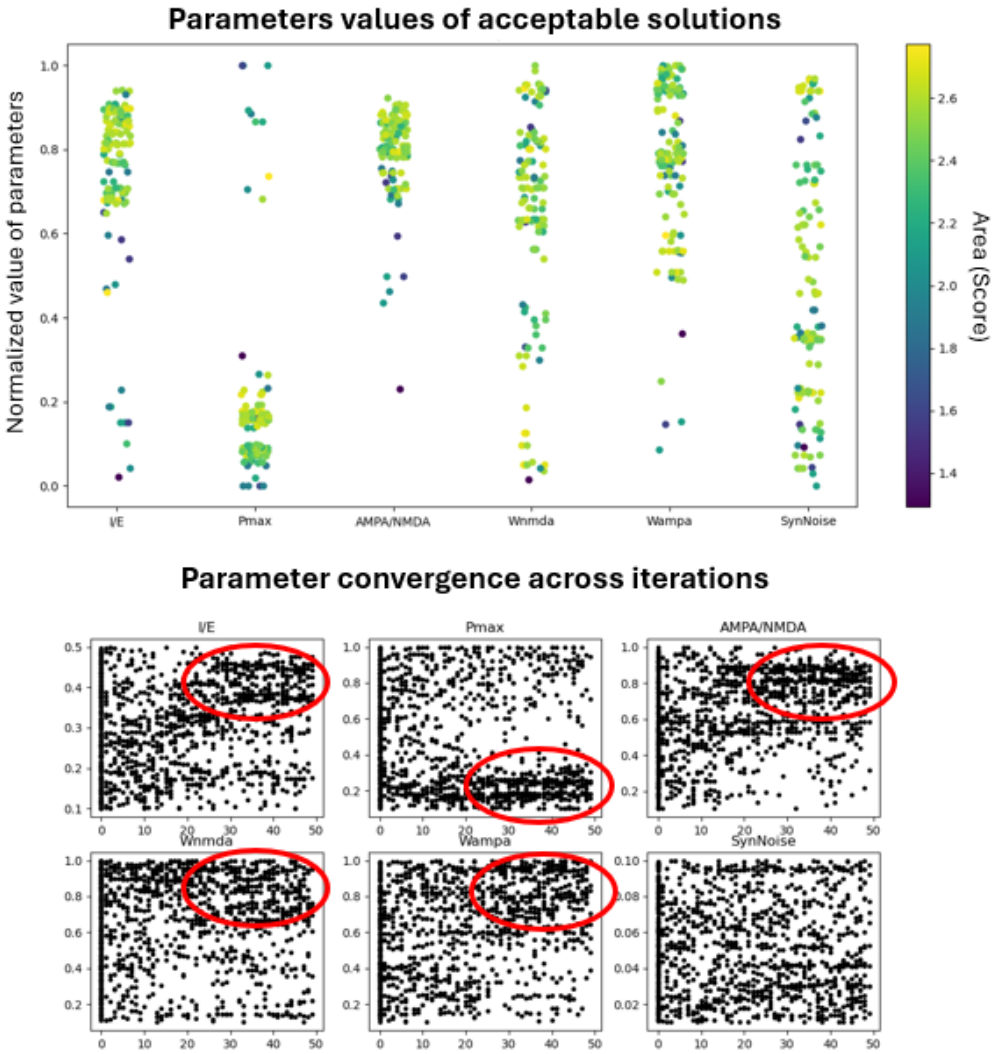

Figure 3.7: *(Top) Parameter values of the survived individuals at the last iteration. Only individuals that have less than 0.5 as DKL are considered as acceptable and shown. These values are normalized for the upper and lower bound of their range. Each combination of the six points represents the score according to the area of the corresponding triangle in the radar pot. We can see that, for some parameters, there is a gradient going from darker to lighter colors, possibly indicating the direction of the optimization. The mean value applied in the Ornstein-Uhlenbeck process to model synaptic noise does not converge in a precise cluster as other parameters, meaning that this value is not involved in the optimization process. (Down) Each graph has the whole set of parameters of the entire population at each iteration. The red circles indicate the most clustered parts of the graph. Even here, the synaptic noise have never had a convergence across iterations.*

Chapter 4

Discussion

Neuroprosthetic devices have received significant attention in recent years, raising critical questions about how to restore normal function in injured brain areas (Chiappalone, Cota, et al. 2022). The concept of using a stimulating device to reconnect damaged regions is based on the seminal work of Fetz and colleagues, who first demonstrated the ability to induce Hebbian plasticity by artificially linking the spiking activity of two cortical neurons in a monkey's brain (Jackson, Mavoori, and Fetz 2006).

Various strategies have since been explored, with some showing remarkable behavioral outcomes. Notably, in 2013, researchers at Kansas University Medical Center introduced a brain prosthesis utilizing a closed-loop architecture, marking a significant advancement in the field (Guggenmos et al. 2013). They successfully restored communication between the rostral forelimb area (RFA) and the sensorimotor (S1) area by "bypassing" a lesioned motor area that disrupted the connection between the two. This was achieved through a novel closed-loop technique called activity-dependent stimulation (ADS). The system worked by detecting neural activity in RFA and, each time a spike was generated, delivering a stimulus to the S1 area. As a result, they were able to restore the rats' reaching and grasping functions, which had been impaired by the lesion.

This type of stimulation is inherently 'neural-like' because the stimulation pattern is driven by another brain area. In this context, neuromorphic-based neuroprostheses, such as Spiking Neural Networks (SNNs), offer a promising approach to achieve this kind of neural stimulation. The work of Beaubois and colleagues in 2024 introduced a real-time FPGA-based Spiking Neural Network , which opens the door to conducting in vivo experiments (Beaubois et al.

2024). By interfacing this device with the animal's brain, it is possible to evaluate its ability to influence biological spontaneous activity. To do so, in this study, an analysis of the impact of the stimulation is provided, making a comparison between control group (SHAM), where no stimulation was provided, the group stimulated with the Zybo Z7-20 board (Z) and the ones stimulated with the novel Bioemus system implemented in the Kria KV260 board (K). Additionally, a method to automatize SNN parametrization through evolutionary algorithm is proposed, to obtain a SNN closer in mimicking its biological counterpart.

4.1 Effects of SNN-driven intracortial microstimulation

Following the same framework adopted in Kansas University Medical Center, the SNNs were interfaced with the S1 area of brain rats, to stimulate this in an open-loop fashion. Simultaneously, neural activity from the RFA was recorded to facilitate comparisons with prior experimental results obtained through ADS, as performed in similar acute experiments on healthy anesthetized rats (Averna, Pasquale, et al. 2020). The stimulation was provided for one hour.

Friedman's test was employed to evaluate differences between all the groups in S1. It revealed for all the evaluated metrics (MFR, LvR and area of PSTH), that their variations between pre- and post- stimulation phase did vary significantly across different groups in the S1 area. However, post hoc analysis failed to identify significant differences between paired groups, likely due to the low statistical power when groups were considered individually (7 animals per group). As seen in Figure 2.6, Figure 2.7, and Figure 2.8, the two stimulated groups appear to differ from the control group in all three metrics. Stimulation seems to reduce MFR levels in S1 compared to the control group, where it increases. For LvR, stimulation pushes the network towards more bursty activity in S1, while the effect in RFA is the opposite, with a direction toward a more random spiking pattern. Finally, regarding the PSTH area, which estimates tissue excitability, stimulation induces a more pronounced decrease in S1 compared to controls, while in RFA, the metric slightly decreases. Comparing these results to those obtained by Averna and colleagues in 2020, where activity-dependent stimulation (ADS) was employed, the outcomes appear partially consistent (Averna, Pasquale, et al. 2020). In that study, ADS similarly decreases both LvR and PSTH area after the first hour of stimulation.

The only contrasting result between the two studies lies in the network's firing rate: in this study, the firing rate remained stable, whereas in the previous work it increased. Since ADS has been proven to be an effective stimulation method for restoring lost functional capabilities, these findings suggest that SNN-driven stimulation methods could potentially achieve similar outcomes. Unfortunately, as the control group in the current study was not recorded in the RFA, a statistical comparison of stimulation effects relative to non-stimulated animals is not possible. Additionally, further experiments are necessary to improve the statistical power of these findings and allow for a more reliable comparison between the two methods. Another unexpected result was observed in the control group, where the firing rate in the S1 area appeared to increase spontaneously, despite the absence of stimulation. It is uncertain whether this effect is entirely physiological, as no additional data is available for comparison in this specific brain region of the rats. One potential explanation could be related to the variability in anesthesia effects, which may not have remained stable throughout the entire experiment. Further investigations are needed to clarify these aspects.

4.2 Differences between the two FPGA boards

In this study, two different FPGA platforms were used to stimulate the animals. The Zybo Z7-20 board implemented 100 Hodgkin-Huxley (HH) disconnected neurons, while the Kria KV260 board utilized Bioemus to implement 1024 HH interconnected neurons in a small-world network configuration. Both boards successfully engaged the network, demonstrating their potential to activate neural circuits effectively. Notably, the engagement, especially the one achieved with the Zybo board, was comparable to that obtained using ADS (Averna, Hayley, et al. 2021), suggesting that these devices could serve as viable neuroprosthetic tools for restoring neural function. Nevertheless, there are many differences in the stimulus train generated by the two boards:

1. The Zybo board delivered four times the number of stimuli compared to the Kria board.
2. The Kria board exhibited a significantly more bursty behavior than the Zybo board.

These differences led to varying levels of network engagement. The analysis of I/O correlation between the stimulus train and the evoked activity revealed statistically significant differences between the two boards, as confirmed by a two-way repeated measures ANOVA. The Zybo

board demonstrated a significantly greater ability to evoke responses compared to the Kria board.

Discrepancies in network engagement between the two likely stem from differences in their firing properties, raising questions about which stimulation pattern is more effective in eliciting stronger network responses. One potential solution is to align the electrophysiological parameters of the SNN with those of the biological system. This approach could enable a more personalized stimulation, potentially resulting in a more robust and appropriate neural response. To address this, the study introduces an optimization algorithm designed to mimic experimental outcomes using the spiking neural network. The Bioemus system provides flexibility in configuring the SNN, offering a broad range of network behaviors, further supporting this goal.

Another critical consideration is the selection of the stimulating neuron. In these experiments, where only a single electrode serves as the stimulation channel, just one neuron from the artificial network is responsible for delivering the stimulus train. A potential strategy for selecting this neuron could involve choosing the one whose firing rate most closely matches the median one of the biological network. Alternatively, selecting the neuron with the median firing rate of the artificial network could offer a representative choice for the interfacing SNN.

Further enhancements may be achievable by employing a "network" stimulation approach, where multiple channels are used to stimulate the animal's brain simultaneously. This would enable the stimulation from several neurons within the artificial SNN, potentially providing a more effective interaction with the biological system.

4.3 Evolutionary algorithm to fine-tune the biological and the spiking neural network

Designing an optimization algorithm to fine-tune a spiking neural networks, enabling a better mimicking of a biological one, presents a significant challenge, particularly in selecting the optimal strategy from the vast range of possibilities. One approach is to attempt a precise match of the firing patterns, a task that is nearly impossible due to the complexity of the networks. A more practical strategy, and the one adopted in this work, is to focus on matching broader electrophysiological parameters, specifically the Mean Firing Rate (MFR), the Mean

Bursting Rate (MBR) and the Inter-Spike Interval (ISI), which provide a more reasonable and achievable goal.

Furthermore, this match is not achieved across all metrics simultaneously, but rather by keeping the three metrics separate, thereby shifting the optimization towards a multi-objective approach. This allows for balancing the trade-offs between metrics, where improving one might not necessarily improve the others, and helps in finding a solution that provides a reasonable compromise across all objectives and adaptable basing on the specific needs. Indeed, due to its multi-objective nature, this algorithm produces a pool of optimal solutions rather than a single one, enabling the user to select the solution that most closely aligns with the desired properties.

The chosen algorithm is the Indicator-Based Evolutionary Algorithm (IBEA) (Zitzler and Künzli 2004), which has been previously employed in computational neuroscience to fine-tune parameters of mono- or multi-compartmental single neuron (Masoli et al. 2017). IBEA operates by generating a population of random individuals and iteratively converging toward an optimal set by recombining these initial population to produce other individuals and discarding the worst based on a fitness value, iteratively. Individuals that survive after a predetermined number of iterations are considered the "winners," whose distinct properties are further analyzed. After initial testing of the algorithm, the hyperparameters were fixed to achieve the fastest compromise for reaching the optimum. Two versions of the IBEA were implemented directly on the Kria board, using two different objective functions to quantify the distances between the goals and objectives: one based on the distance between median values, and the other using the Kullback-Leibler Divergence (DKL) between the distributions of the metrics (MFR, MBR, ISI). In each configuration, several key parameters were adjusted, focusing on those expected to have the most significant impact on the resulting network activity. An entire optimization process took about 8 hours.

To summarize the results of each run, a radar plot is presented to provide an overall view of the optimization process. As illustrated in Figure 3.5, several individuals can be considered as potential configurations that closely align with the biological counterpart, each maintaining different trade-offs in the obtained distributions. The configuration that maximizes the area within this plot is selected as the best compromise, facilitating a quick identification of the optimal solution across all three distance metrics. However, it is important to note that this

configuration should not be regarded as the absolute best individual. For instance, achieving a perfect match in the mean ISI distribution is significantly less critical than ensuring a strong alignment in the MFR distribution of the network. The mean ISI distribution serves primarily as a guideline to guarantee a physiologically relevant ISI and to avoid excessively bursty firing patterns on average, which could potentially disrupt interactions with the biological network. By comparing the results obtained with both medians or DKL, this algorithm is capable in complete its task, obtaining closer results when using the DKL.

Finally, this algorithm is proposed as a valuable tool for enhancing the Bioemus framework, shedding light on potential limitations and future improvements within the system. For instance, it highlights a set of conflicting metrics, prompting questions about why the spiking neural network (SNN) struggles to optimize some metrics simultaneously and offers insight on how these issues might be resolved.

4.4 Future improvements

The Indicator-Based Evolutionary Algorithm (IBEA) for Bioemus does not come without its challenges. The primary concern is the execution time, which is currently too onerous for routine use during biohybrid experiments. The algorithm was designed to be executed prior to the stimulation phase, allowing for a careful tuning of the spiking neural network with the biological one. However, this version of the algorithm struggles to meet this objective.

Several potential solutions could address this issue. One approach is to change the saving format; Bioemus currently saves configuration parameters in a large text file (approximately 8 MB, depending on the number of synapses). This file size presents a bottleneck in the system, so utilizing a more efficient format, such as a binary format, could enhance performance. Upgrading to a faster SSD card may also significantly reduce execution time.

Another strategy involves setting a threshold on the distances, alongside a specified number of generations. This would allow the system to terminate when a sufficiently good network configuration is produced. Conversely, if the algorithm fails to generate an acceptable solution, it would stop after the designated number of generations. While this approach carries some risk, primarily due to the ambiguity surrounding what constitutes a "sufficiently good" solution, especially in a multi-objective optimization context, it is noteworthy that the individual

maximizing the area in the radar plot often emerges many iterations before the algorithm completes. Ultimately, improving the speed of the IBEA will enable a broader exploration of the search space by allowing higher-dimensional configurations, increasing the number of parameters that can be adjusted in each setup. In addition, the inclusion of additional metrics is fundamental to enhance the system's ability to accurately represent the activity of its biological counterpart. By doing so, the optimization algorithm can yield more comprehensive insights, leading to more effective tuning of the spiking neural networks in conjunction with biological systems.

Conclusion

The work presented in this thesis can be divided in two parts: firstly, we tried to quantify the impact and the differences between two different SNN-driven intracortical microstimulations, secondly we developed an evolutionary algorithm able to finely tune the extracted experimental electrophysiological parameters of the animal with the ones of the SNN.

In addressing the first point, our findings partially align with previous studies, particularly one that investigated activity-dependent stimulation methods within a similar experimental framework (Averna, Pasquale, et al. 2020). We demonstrated that open-loop SNN-driven stimulation is capable of engaging the neural network and impacting its activity. To do so, we also devised a novel spike-detection algorithm, which is the adaptive version of a previously developed algorithm named SWTTEO (Lieb, Stark, and Thielemann 2017). In addition, the variations in firing properties across different SNNs led to differential network engagement, highlighting the need of a personalized SNN able to mimic the interfacing brain area. While these results are promising enough to propose SNN-driven stimulation as a potential brain prosthetic device, further research is essential.

To enhance the robustness of our conclusions, expanding the dataset is crucial to increase statistical power and validate our findings. Furthermore, testing this stimulation approach in injured animal models is necessary to assess its efficacy in restoring lost functional capabilities. To this end, it is necessary to conduct chronic experiments to evaluate the long-term effects of SNN-driven stimulation on brain activity.

In the second part of the thesis, we proposed an algorithm aimed at effectively matching experimental recordings with those from the SNN. We designed a Multi-Objective Indicator-Based Evolutionary Algorithm, following the framework outlined in Zitzler and Künzli 2004, which yielded promising results. However, there remains significant potential for improvement, particularly in reducing the algorithm's overall execution time and incorporating additional

metrics to enhance its precision in mimicking biological activity.

Future developments will focus on integrating a broader array of neuronal types and synaptic models to increase the diversity of potential solutions that the system can explore. This diversification may lead to more sophisticated and effective configurations of the SNN, further bridging the gap between artificial and biological neural networks.

The concept of neuromorphic neuroprosthetics represents a significant advancement in the field. Leveraging the biological plausibility of SNNs serves not merely an aesthetic purpose but has profound functional implications. Establishing a closed-loop system that facilitates communication between biological brains and artificial spiking neural networks is crucial for developing optimal neuroprosthesis. Such an interaction would enable the adaptive capabilities of SNNs to personalize stimulation based on real-time feedback from the biological system, marking a pivotal step in tailoring neuroprosthetic interventions. Ultimately, this approach aims to improve outcomes for individuals suffering from brain injuries or neurodegenerative diseases.

In summary, this thesis contributes to the growing field of neuroprosthetics by providing insights into the capabilities and limitations of SNN-driven microstimulations. While the initial results are encouraging, the journey toward developing effective neuromorphic neuroprosthetic devices necessitates continued exploration, validation, and refinement of these methods. This continued effort is essential for achieving meaningful therapeutic outcomes for individuals with neural impairments.

Appendix

Introducing a Novel Spike Detection Algorithm - Adaptive SWTTEO

The information exchange between neurons is based on rapid changes of the cell membrane potential called action potentials or spikes: the entire activity of a neuron can be represented as a delta-train, where each delta corresponds to a spike. However, in extracellular recordings, separating spikes from the surrounding noise is a challenging task, prompting the development of various detection strategies (Obeid and Wolf 2004).
In this framework, spike detection algorithms are generally classified into three main categories:

1. Simple Thresholding: This method focuses on the most distinguishable characteristic of spikes, their amplitude. It assumes that spike signals exhibit higher or more pronounced peak-to-peak values compared to background noise, allowing for detection through basic thresholding.

2. Template Based Correlation: This approach relies on the specific shape of spikes. Predefined template waveforms are compared to segments of the recorded signal. When the similarity between the template and the signal exceeds a set threshold, a spike is identified.

3. Transient Energy: This category focuses on the sudden changes in amplitude characteristic of spikes. These transients generate a distinctive frequency pattern that stands out from typical noise, facilitating their detection.

The Stationary Wavelet-based Teager Energy Operator (SWTTEO) is a recently developed method that belongs to the third category (Lieb, Stark, and Thielemann 2017).As can be

intended by the name, this method relies mainly on two operators: the Stationary Wavelet Transform (SWT) and the Teager Energy Operator (TEO). A wavelet transform is a signal processing technique that breaks down a signal into scaled and shifted versions of a "mother" wavelet, allowing the signal to be analyzed at multiple resolutions. It is particularly efficient with non-stationary signals, those whose frequency characteristics change over time—such as neural signals. The term "stationary" in Stationary Wavelet Transform (SWT) refers to its time-invariant property, meaning that the transform maintains consistency in time when analyzing the signal, unlike the Discrete Wavelet Transform (DWT). In DWT, the wavelet convolution is followed by downsampling (reducing the number of data points after each level), which can cause shifts in time and a loss of some details. With SWT, there is no downsampling, so the resolution remains constant across all scales, preserving the time information (Nason and Silverman 1995).

The Teager Energy Operator is defined as:

$$\Psi(x(t)) = \dot{x}^2(t) - x(t)\ddot{x}(t). \tag{4.1}$$

"Energy operator" is motivated by its analogy with the total energy of a classical harmonic oscillator (Kaiser 1993). These two operators are applied sequentially to the signal, followed by a global thresholding step to extract the spikes. In this adaptive version, the threshold is dynamically computed within a sliding window, which calculates the 99th percentile of the signal locally. Then another global threshold is set as n times the median signal value. When the transformed signal exceeds this local threshold, the segment is captured, and the position of its peak is identified and marked as a spike. This adaptive thresholding method ensures more accurate spike detection, especially in signals with varying characteristics over time (as slow-waves induced ON-OFF phases during sleep, Figure 2.3). The dimension of the window and the step-size of the slide is customizable, smaller windows and longer steps results in a faster computation.

Input-Output Correlation

The cross-correlation between spike trains is a fundamental metric, as it is capable to highlight synchronization phenomena at the network level. Different steps are necessary to find a *cross-correlation*.

1. Identify a reference train and a target train, denoted respectively as x and y.

2. Divide the two trains into equally spaced bins small enough to contain at most one spike. In this way a histogram that can tell how the spikes in the reference are related to the spikes in the target can be built. To do so, spikes of the target train are projected on the reference spike train. A time delay τ is applied as a variable shift between the two trains (going from $-T$ to T). The position of the projected spike becomes the zero of a new time axis, which has to be divided in bins again in order to count the spikes inside them.

3. At the end, an histogram is obtained, which has values between $-T$ and T on the x-axis and $C_{xy}(\tau)$ on the y-axis, that is the cross correlation between the spike trains Figure 4.1.

When $\tau = 0$, the reference spike train is synchronized with the target spike train: it means that the cross-correlation is high, hence a big central peak and very low lateral ones are expected (auto-correlation). From a mathematical point of view, the cross-correlation can be written as:

$$C_{xy}(\tau) = \frac{1}{\sqrt{N_x N_y}} \sum_{s=1}^{N_x} \sum_{t_i=\tau-\Delta\tau/2)}^{(\tau+\Delta\tau/2)} x(t_s) y(t_s - t_i), \tag{4.2}$$

with $1/\sqrt{N_x N_y}$ a normalization factor to keep values inside [0,1]. In the case of Input-Output (I/O) correlation, the reference train is always the stimulation train and the time lag is only positive, since the evoked-effects of the stimulation is the purpose. In this specific thesis, the τ goes from 0 to 50 ms and the bin size is 1 ms.

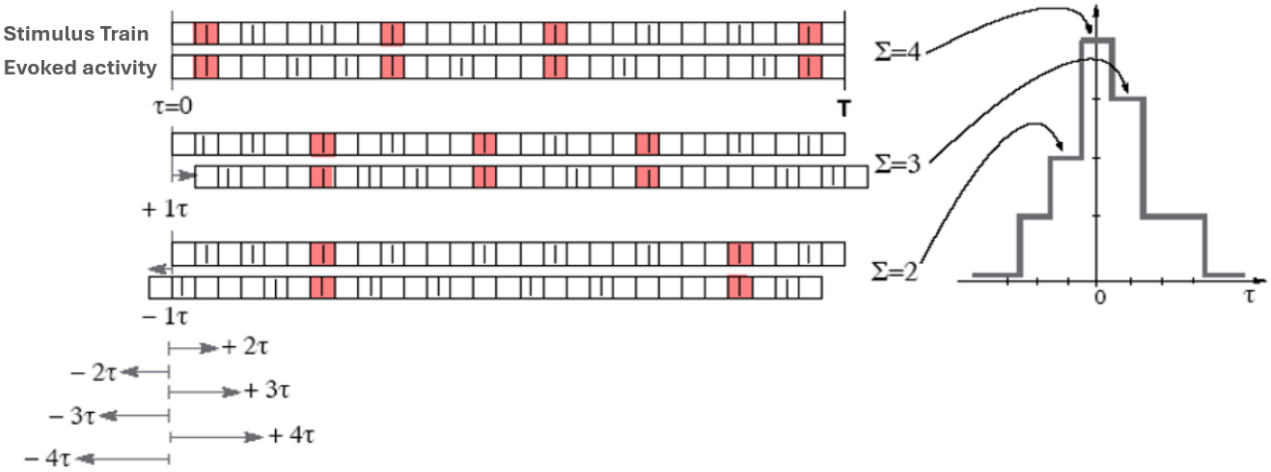

Figure 4.1: *Cross correlation algorithm representation. The final product is a vector representing the histogram on the right.*

Non-Local Derandomized Step-Size Control (CSA)

The mutation operator introduces ("small") variations by adding a point symmetric perturbation to the result of recombination, represented by a solution vector $x \in R^n$. This perturbation is sampled from a multivariate normal distribution $\mathcal{N}(0, C)$, where $C \in R^{n \times n}$ and it is the covariance matrix. We have $x + \mathcal{N}(0, C) \sim \mathcal{N}(x, C)$, meaning that x determines the expected value of the new offspring individual. For our purposes, we consider $C = \sigma$ as a diagonal matrix and is defined as the *mutation rate*, since it determine the distance of the new children from the parent. Controlling this parameters of the mutation operator is a key point in an evolution strategy: without a constant update of the mutation rate,the optimization process can get stuck in a cycle where newly generated children jumps from one side to the other of the solution. By iteratively adjusting this step, not only the algorithm should constantly better approximates the optimum, but also it works much faster, avoiding too big or too small mutations of the individuals. One possible solution to this is the *self-adaptation*
Control parameters can be stored at different "levels." Each individual can have its own specific step-size, or a single step-size can be shared and applied uniformly across the entire population. The same considerations can be applied to the adaptation strategy: either a single parameter can be used to update all individuals, or each individual can have its own. The following algorithm will adopt the first case, referencing to as *Non-Localized*. Another important aspect is that the *self-adaptation* should be derandomized. The adjustment of the

mutation rate should not be left to chance but should be informed by the ongoing progress of the search, tracking the evolutionary development to guide more effective optimization.
The Non-Local Derandomized Step-Size Control (CSA), is an algorithm that puts together these aspects: it uses a non-local strategy, in order to avoid that the single children can afflict the adaptation step, and it is derandomized, since it takes memory of the overall search progress to adjust the mutation rate. A schematic pseudo-code representation is written in the following: Given:

- $\circ$: the element-wise product
- $n \in N$: dimension of the single individual (and as a consequence of the mutation rate),
- λ : offspring size,
- μ : population size
- $c_\sigma \approx \sqrt{\mu/(n+\mu)}$
- $d \approx 1 + \sqrt{\mu/n}$
- $d_i \approx 3n$

Steps:

1. Initialize $\mathbf{x} \in R^n$,$\sigma \in R^n$, $\mathbf{s}_\sigma = 0$
2. while not happy
3. for $k \in \{1, \ldots, \lambda\}$ (generating children)
4. $\mathbf{z_k} = \mathcal{N}(\mathbf{0}, \mathbf{I})$
5. $\mathbf{x_k} = \mathbf{x} + \sigma \circ \mathbf{z_k}$
6. $P =$ Selection of the μ best individuals
7. Cross-over
8. $\mathbf{s}_\sigma \leftarrow (1 - c_\sigma)\mathbf{s}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)}\frac{\sqrt{\mu}}{\mu} \sum_{\mathbf{z_k} \in P} \mathbf{z_k}$
9. $\sigma \leftarrow \sigma \circ exp(\frac{1}{d_i}(\frac{|\mathbf{s}_\sigma|}{E|\mathcal{N}(0,1)|} - \mathbf{1})) \times exp(\frac{c_\sigma}{d}(\frac{\|\mathbf{s}_\sigma\|)}{E\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1))$

Here, $\mathbf{z_k}$ represents the local mutation steps applied to each individual. All $\mathbf{z_k}$ that survive the selection process are considered the best and are subsequently used to update the search path parameter, $\mathbf{s}_\sigma$. The search path encapsulates information about the interrelation between individual steps, which can significantly enhance both the adaptation and the overall search process (Hansen, Arnold, and Auger 2015; Ostermeier, Gawelczyk, and Hansen 1994).

References

Buzsáki, G. (Oct. 2006). "Rhythms of the Brain". In: Oxford University Press. DOI: 10.1093/acprof:oso/9780195301069.001.0001.

Mizusaki, B. E. and C. O'Donnell (Oct. 2021). "Neural Circuit Function Redundancy in Brain Disorders". In: *Current Opinion in Neurobiology* 70, pp. 74–80. DOI: 10.1016/j.conb.2021.07.008.

Chin, J. H. and N. Vora (July 2014). "The Global Burden of Neurologic Diseases". In: *Neurology* 83.4, pp. 349–351. DOI: 10.1212/WNL.0000000000000610.

Famm, K. et al. (Apr. 2013). "A Jump-Start for Electroceuticals". In: *Nature* 496.7444, pp. 159–161. DOI: 10.1038/496159a.

Reardon, S. (July 2014). "Electroceuticals Spark Interest". In: *Nature* 511.7507, pp. 18–18. DOI: 10.1038/511018a.

Wright, J. et al. (July 2016). "A Review of Control Strategies in Closed-Loop Neuroprosthetic Systems". In: *Frontiers in Neuroscience* 10. DOI: 10.3389/fnins.2016.00312.

Panuccio, G. et al. (2018). "Progress in Neuroengineering for Brain Repair: New Challenges and Open Issues". In: *Brain and Neuroscience Advances* 2, p. 2398212818776475. DOI: 10.1177/2398212818776475.

Guggenmos, D. J. et al. (Dec. 2013). "Restoration of Function after Brain Damage Using a Neural Prosthesis". In: *Proceedings of the National Academy of Sciences of the United States of America* 110.52, pp. 21177–21182. DOI: 10.1073/pnas.1316885110.

Berger, T. W. et al. (Aug. 2011). "A Cortical Neural Prosthesis for Restoring and Enhancing Memory". In: *Journal of neural engineering* 8.4, p. 046017. DOI: 10.1088/1741-2560/8/4/046017.

Chiappalone, M., V. R. Cota, et al. (Nov. 2022). "Neuromorphic-Based Neuroprostheses for Brain Rewiring: State-of-the-Art and Perspectives in Neuroengineering". In: *Brain Sciences* 12.11, p. 1578. DOI: 10.3390/brainsci12111578.

Beaubois, R. et al. (June 2024). "BiœmuS: A New Tool for Neurological Disorders Studies through Real-Time Emulation and Hybridization Using Biomimetic Spiking Neural Network". In: *Nature Communications* 15.1, p. 5142. DOI: 10.1038/s41467-024-48905-x.

Hines, M. L. and N. T. Carnevale (Apr. 2001). "NEURON: A Tool for Neuroscientists". In: *The Neuroscientist: A Review Journal Bringing Neurobiology, Neurology and Psychiatry* 7.2, pp. 123–135. DOI: 10.1177/107385840100700207.

Gewaltig, M.-O. and M. Diesmann (Apr. 2007). "NEST (NEural Simulation Tool)". In: *Scholarpedia* 2.4, p. 1430. DOI: 10.4249/scholarpedia.1430.

Brunel, N. and M. C. W. van Rossum (Dec. 2007). "Quantitative Investigations of Electrical Nerve Excitation Treated as Polarization". In: *Biological Cybernetics* 97.5, pp. 341–349. DOI: 10.1007/s00422-007-0189-6.

Hodgkin, A. L. and A. F. Huxley (Aug. 1952). "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve". In: *The Journal of Physiology* 117.4, pp. 500–544.

Computational Modeling Methods for Neuroscientists (2024). https://mitpress.mit.edu/9780262013277/co
modeling-methods-for-neuroscientists/.

Huguenard, J. R. and D. A. McCormick (Oct. 1992). "Simulation of the Currents Involved in Rhythmic Oscillations in Thalamic Relay Neurons". In: *Journal of Neurophysiology* 68.4, pp. 1373–1383. DOI: 10.1152/jn.1992.68.4.1373.

Sulaiman, N. et al. (Oct. 2009). "Design and Implementation of FPGA-Based Systems -A Review". In: *Australian Journal of Basic and Applied Sciences* 3.

Pospischil, M. et al. (Nov. 2008). "Minimal Hodgkin-Huxley Type Models for Different Classes of Cortical and Thalamic Neurons". In: *Biological Cybernetics* 99.4-5, pp. 427–441. DOI: 10.1007/s00422-008-0263-8.

Uhlenbeck, G. E. and L. S. Ornstein (Sept. 1930). "On the Theory of the Brownian Motion". In: *Physical Review* 36.5, pp. 823–841. DOI: 10.1103/PhysRev.36.823.

Destexhe, A., Z. Mainen, and T. Sejnowski (Jan. 1998). *Kinetic Models of Synaptic Transmission*. Vol. 2, p. 26.

Watts, D. J. and S. H. Strogatz (June 1998). "Collective Dynamics of 'Small-World' Networks". In: *Nature* 393.6684, pp. 440–442. DOI: 10.1038/30918.

Bassett, D. S. and E. T. Bullmore (Oct. 2017). "Small-World Brain Networks Revisited". In: *The Neuroscientist: A Review Journal Bringing Neurobiology, Neurology and Psychiatry* 23.5, pp. 499–516. DOI: 10.1177/1073858416667720.

Di Florio, M. et al. (July 2023). "Design of an Experimental Setup for Delivering Intracortical Microstimulation in Vivo via Spiking Neural Network". In: *2023 45th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 1–4. DOI: 10.1109/EMBC40787.2023.10340907.

Khoyratee, F. et al. (Apr. 2019). "Optimized Real-Time Biomimetic Neural Network on FPGA for Bio-hybridization". In: *Frontiers in Neuroscience* 13. DOI: 10.3389/fnins.2019.00377.

Kleim, J. A. et al. (Dec. 2003). "Motor Cortex Stimulation Enhances Motor Recovery and Reduces Peri-Infarct Dysfunction Following Ischemic Insult". In: *Neurological Research* 25.8, pp. 789–793. DOI: 10.1179/016164103771953862.

Negri, F. (Dec. 2023). "A Novel Stimulus Artifacts Rejection Algorithm Enables the Analysis of Evoked Activity in Vivo". PhD thesis. University of Genoa.

Lieb, F., H.-G. Stark, and C. Thielemann (June 2017). "A Stationary Wavelet Transform and a Time-Frequency Based Spike Detection Algorithm for Extracellular Recorded Data". In: *Journal of Neural Engineering* 14.3, p. 036013. DOI: 10.1088/1741-2552/aa654b.

Massimini, M. et al. (Aug. 2024). "Sleep-like Cortical Dynamics during Wakefulness and Their Network Effects Following Brain Injury". In: *Nature Communications* 15.1, p. 7207. DOI: 10.1038/s41467-024-51586-1.

Shinomoto, S. et al. (July 2009). "Relating Neuronal Firing Patterns to Functional Differentiation of Cerebral Cortex". In: *PLoS Computational Biology* 5.7, e1000433. DOI: 10.1371/journal.pcbi.1000433.

Averna, A., V. Pasquale, et al. (May 2020). "Differential Effects of Open- and Closed-Loop Intracortical Microstimulation on Firing Patterns of Neurons in Distant Cortical Areas". In: *Cerebral Cortex* 30.5, pp. 2879–2896. DOI: 10.1093/cercor/bhz281.

Chiappalone, M., A. Vato, et al. (May 2007). "Network Dynamics and Synchronous Activity in Cultured Cortical Neurons". In: *International journal of neural systems* 17, pp. 87–103. DOI: 10.1142/S0129065707000968.

Rieke, F. (1999). *Spikes, Exploring the Neural Code*. https://mitpress.mit.edu/9780262181747/spikes/.

Averna, A., P. Hayley, et al. (Nov. 2021). "Entrainment of Network Activity by Closed-Loop Microstimulation in Healthy Ambulatory Rats". In: *Cerebral Cortex* 31.11, pp. 5042–5055. DOI: 10.1093/cercor/bhab140.

Van Geit, W., E. De Schutter, and P. Achard (Nov. 2008). "Automated Neuron Model Optimization Techniques: A Review". In: *Biological Cybernetics* 99.4-5, pp. 241–251. DOI: 10.1007/s00422-008-0257-6.

Druckmann, S. et al. (Oct. 2007). "A Novel Multiple Objective Optimization Framework for Constraining Conductance-Based Neuron Models by Experimental Data". In: *Frontiers in Neuroscience* 1.1, pp. 7–18. DOI: 10.3389/neuro.01.1.1.001.2007.

Masoli, S. et al. (2017). "Single Neuron Optimization as a Basis for Accurate Biophysical Modeling: The Case of Cerebellar Granule Cells". In: *Frontiers in Cellular Neuroscience* 11, p. 71. DOI: 10.3389/fncel.2017.00071.

Eiben, A. E. and J. E. Smith (2015). "What Is an Evolutionary Algorithm?" In: *Introduction to Evolutionary Computing*. Ed. by A. Eiben and J. Smith. Berlin, Heidelberg: Springer, pp. 25–48. DOI: 10.1007/978-3-662-44874-8_3.

Holland, J. H. (Apr. 1992). "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence". In: The MIT Press. DOI: 10.7551/mitpress/1090.001.0001.

Rechenberg, I. (1989). "Evolution Strategy: Nature's Way of Optimization". In: *Optimization: Methods and Applications, Possibilities and Limitations*. Ed. by H. W. Bergmann. Berlin, Heidelberg: Springer, pp. 106–126. DOI: 10.1007/978-3-642-83814-9_6.

Hansen, N., D. V. Arnold, and A. Auger (2015). "Evolution Strategies". In: *Springer Handbook of Computational Intelligence*. Ed. by J. Kacprzyk and W. Pedrycz. Berlin, Heidelberg: Springer, pp. 871–898. DOI: 10.1007/978-3-662-43505-2_44.

Ostermeier, A., A. Gawelczyk, and N. Hansen (1994). "Step-Size Adaptation Based on Non-Local Use of Selection Information". In: *Parallel Problem Solving from Nature — PPSN III*.

Ed. by Y. Davidor, H.-P. Schwefel, and R. Männer. Berlin, Heidelberg: Springer, pp. 189–198. DOI: 10.1007/3-540-58484-6_263.

Deb, K. (1995). *Simulated Binary Crossover for Continuous Search Space by Kalyanmoy Deb and Ram Bhushan Agrawal*. https://www.complex-systems.com/abstracts/v09_i02_a02/.

Deb, K. and A. Kumar (1995). *Real-Coded Genetic Algorithms with Simulated Binary Crossover: Studies on Multimodal and Multiobjective Problems by Kalyanmoy Deb and Amarendra Kumar*. https://www.complex-systems.com/abstracts/v09_i06_a01/.

Falcón-Cardona, J. and C. Coello (Mar. 2020). "Indicator-Based Multi-Objective Evolutionary Algorithms: A Comprehensive Survey". In: *ACM Computing Surveys* 53. DOI: 10.1145/3376916.

Deb, K. (Jan. 2001). "Multiobjective Optimization Using Evolutionary Algorithms. Wiley, New York". In.

Zitzler, E. and S. Künzli (2004). "Indicator-Based Selection in Multiobjective Search". In: *Parallel Problem Solving from Nature - PPSN VIII*. Ed. by D. Hutchison et al. Vol. 3242. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 832–842. DOI: 10.1007/978-3-540-30217-9_84.

Kullback, S. and R. A. Leibler (Mar. 1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86. DOI: 10.1214/aoms/1177729694.

Destexhe, A., Z. F. Mainen, et al. (n.d.). "Kinetic Models of Synaptic Transmission". In: ().

Isaacson, J. S. and M. Scanziani (Oct. 2011). "How Inhibition Shapes Cortical Activity". In: *Neuron* 72.2, pp. 231–243. DOI: 10.1016/j.neuron.2011.09.027.

Myme, C. I. O. et al. (Aug. 2003). "The NMDA-to-AMPA Ratio at Synapses onto Layer 2/3 Pyramidal Neurons Is Conserved across Prefrontal and Visual Cortices". In: *Journal of Neurophysiology* 90.2, pp. 771–779. DOI: 10.1152/jn.00070.2003.

Jackson, A., J. Mavoori, and E. E. Fetz (Nov. 2006). "Long-Term Motor Cortex Plasticity Induced by an Electronic Neural Implant". In: *Nature* 444.7115, pp. 56–60. DOI: 10.1038/nature05226.

Obeid, I. and P. Wolf (June 2004). "Evaluation of Spike-Detection Algorithms Fora Brain-Machine Interface Application". In: *IEEE Transactions on Biomedical Engineering* 51.6, pp. 905–911. DOI: 10.1109/TBME.2004.826683.

Nason, G. P. and B. W. Silverman (1995). "The Stationary Wavelet Transform and Some Statistical Applications". In: *Wavelets and Statistics*. Ed. by A. Antoniadis and G. Oppenheim. New York, NY: Springer, pp. 281–299. DOI: 10.1007/978-1-4612-2544-7_17.

Kaiser, J. (1993). "Some Useful Properties of Teager's Energy Operators". In: *IEEE International Conference on Acoustics Speech and Signal Processing*, 149–152 vol.3. DOI: 10.1109/ICASSP.1993.319457.