

FREIE UNIVERSITÄT BERLIN

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
INSTITUTE OF COMPUTER SCIENCE

**Development of Machine Learning
strategies for programming *in vitro*
Biological Neural Networks**

MASTER THESIS
submitted for the degree of Master of Science in Computational Sciences

by

ISHITA SINGH

Berlin, October 28, 2024

1st examiner Dr. Mohsen SADEGHI
2nd examiner Prof. Dr. Felix. HÖFLING

Declaration of Authorship

I hereby declare to Freie Universität Berlin that I have completed the following thesis on *Development of Machine Learning strategies for programming in vitro Biological Neural Networks* independently and without the use of sources and aids other than those cited.

I declare that the present work is free of plagiarism. Any statements that have been taken from other writings, whether directly or indirectly, have been clearly marked as such.

Further, I declare that this work has not been submitted to any other university as part of an examination attempt, either in identical or similar form, nor has it been published elsewhere.

October 28, 2024

Date

Signature

ACKNOWLEDGEMENT

To be able to successfully complete this work of research after a long gap of 10 years from academics has been quite the journey, which therefore calls for special acknowledgments.

First and foremost, my heartfelt gratitude goes to my supervisor and first examiner Dr. Mohsen Sadeghi for giving me a chance to be part of this project. I would like to thank him for his enthusiasm for the project, for his encouragement and constant support. Under his guidance, I was able to compass a huge learning curve and make my contribution to this interesting field of study.

I would like to thank Prof. Dr. Felix Höfling for volunteering to be my second examiner and sharing his useful insights on the analysis of output data.

To Jean-Marc Comby, FinalSpark, thank you for answering my queries related to Neuro-platform and for replacing the organoids promptly when needed.

To my friends at Freie Universität, Berlin, thank you for all the laughter and long-lasting memories. I am equally grateful my lifelong friends - for their unwavering support and a constant push to go forward.

To Shrikant and Xiaofan, thank you for rigorously reading my thesis and finding all the petty mistakes.

Finally, my deepest gratitude goes to my family, who are my foundation, for believing in me. Your support and encouragement have kept me motivated even during the most challenging times.

Thank you very much!

ABSTRACT

Neuroplatform is a research platform created by FinalSpark which offers biological ‘*mini-brains*’ or neurospheres built on top of multi-electrode-arrays to be used for Wetware Computing. This thesis explores strategies for reinforcement learning on a mini-brain using which it ‘learns’ to adapt itself towards a target.

This work uses data analytics methods to study neural activity captured from a neurosphere and investigates how the spontaneous activity varies with and without external stimuli. Experimental results reveal the displacements in Centre of Activity of neurospheres under the influence of an external stimuli. A target for a neurosphere is calculated using computation methods such as Principle Component Analysis, Correlations and Jensen-Shannon distance from historical activity data of the neurosphere. A closed-loop algorithm for reinforcement learning is executed on the neurosphere which provides positive or negative feedback to it based on the comparisons with the target value.

Several iterations of the algorithm reveal the learning ability of a neurosphere which adapted its output gradually towards the target. This research contributes to the field of wetware computing by offering an algorithm using which an organoid is able to closely reproduce its historical activity, suggesting that it has potential to be trained as computational system. The fact that the organoid responded well to the training algorithm corroborates the nascent jump towards the implementation of silicon-computing alternatives.

CONTENTS

1	Introduction	1
1.1	Motivation	1
1.2	Objective	2
1.3	Scope	2
2	Fundamentals	3
2.1	Wetware computing	3
2.1.1	Energy efficiency	3
2.1.2	In vitro neural networks	3
2.1.3	Learning	4
2.2	The Neuroplatform	5
2.2.1	Platform setup	5
2.2.2	MEA	6
2.2.3	Stimulation experiments	6
2.2.4	Neural activity response	7
2.3	Machine learning	8
2.3.1	Supervised learning	8
2.3.2	Unsupervised learning	8
2.3.3	Reinforcement learning	9
2.3.4	Closed-loop feedback protocol	9
2.4	Data analysis and defining targets	10
2.4.1	Correlation in time series data	10
2.4.2	Principle component analysis	11
3	Objective	13
4	Computational Methods and implementation of training algorithm	15
4.1	Initial Experiments	15
4.1.1	Centre of activity	16
4.2	Stimulation protocols	16
4.2.1	Shuffled background stimulation SBS	17
4.2.2	Controlled stimulation	17
4.2.3	Controlled Experiment Results	18
4.3	Output data analysis	21
4.3.1	Capturing neural activity	21
4.3.2	Correlation between channel activities	22
4.3.3	PCA	25
4.4	PCA in learning algorithm	31
4.4.1	Concepts of probabilities	31
4.4.2	Comparisons between datasets	31
4.4.3	Identification of a target	34
4.5	Closed-loop feedback algorithm	34

4.5.1	Choose a reference distribution for comparison	34
4.5.2	Stimulation protocol and choosing electrodes	35
4.5.3	Stimulation and distance calculation	35
4.5.4	Comparison with target and feedback	36
4.5.5	The learning algorithm combined	37
5	Training results and discussion	39
6	Conclusion	45
A	Appendix	47
A.1	Software Development and Integration	47
A.2	Technology Stack	47
A.3	Modules	47

LIST OF FIGURES

2.1	Stimulation and data recording setup at FinalSpark	6
2.2	MEA composition with electrodes/channels	7
2.3	Classic feedback loop of Reinforcement Learning	10
3.1	Representation of reinforcement learning	13
4.1	Spikes activity capture in neurosphere	15
4.2	Stimulation response on channels	17
4.3	Stimulation protocols on channels	18
4.4	Controlled stimulations results	20
4.5	Time-correlation of channels	22
4.6	Correlation Coefficient matrix (CCM)	24
4.7	PCA calculations	27
4.8	Training of unique PCA object	28
4.9	PCA transformed data	30
4.10	Histogram plots for PC1 components	33
4.11	Identification of target distance for neurosphere	34
4.12	Closed-loop feedback algorithm	38
5.1	Result 1	40
5.2	Result 2	41
5.3	Result 3	42
5.4	Result 4	43
A.1	Stimulation example	48
A.2	Get live data example	49

LIST OF TABLES

4.1	Sample saved activity data for a channel	21
4.2	Sample restructured data for PCA	25
4.3	Selection probabilities of electrodes	35
5.1	Summary of the 4 results	44
A.1	Jupyter notebooks list	49
A.2	Numbered electrode in each sub/neurosphere of an MEA	49

INTRODUCTION

In recent years, Artificial Intelligence(AI) has emerged as a key enabler of scientific discovery by helping scientists accelerate their research¹. With latest systems like chat-bots, relying heavily on Artificial Neural Network (**ANN**) to understand and effectively generate natural language, the need to explore the underlying behaviour of Biological Neural Network (**BNN**) is unprecedented. Moreover, the advancement of technology with AI are facing serious challenges of enormous power consumptions and are steering the world towards an unforeseen energy crisis. The computational cost of smart chatbots such as ChatGPT²¹ is immensely high. Therefore, there is a need to search for fast-computing silicon-alternatives.

The human brain offers high level computation and pattern recognition within fraction of seconds by using the least amount of energy². As such, one of the most important questions posing in front of the research community is about our abilities to create artificial biological systems that can harness the human-brain-like capabilities of high-speed computing with the least amount of energy consumption. Can we create BNNs which can ‘learn’ and reproduce outputs while potentially lowering the computation costs? The pertinent question is - are we ready to replace the ‘silicon’ processors by ‘bio processors’ in future?

A BNN can be created using a network of **brain organoids** which are miniature, simplified versions of brain, cultured artificially from biological cells and tissues. Wetware Computing refers to the concept of using these BNN systems together with Organoid Intelligence²⁵, a concept to impart AI to these living systems, so that they can ‘learn’ to perform some computational tasks. These fields are highly interdisciplinary, combining neuroscience, stem cell biology, bio-engineering, and computer science and are currently under research with numerous ongoing projects.

1.1 Motivation

BNN has been a dynamic field of research for understanding neural activity and cognition. While brain organoids are being used for modelling neurological diseases⁷, together with Machine Learning, datasets from neural activity can help in drug screening as well⁸, by identifying patterns and classifying neural states. Synthetic cognition techniques, integration with AI, modeling neural dynamics are few of many fresh research streams which have been slowly gaining momentum.

For the past couple of years FinalSpark¹¹, a biotechnology research company based in Vevey, Switzerland, has paved way for the futuristic vision of closing gap between biological and silicon digital computation. The company has set up a research platform to investi-

gate and train systems of Biological Neural Network also known as neurosphere⁶. They have developed a hardware and software system called ‘neuroplatform’⁵ that allows for electrophysiological research on these neurospheres. Experiments can be performed to interact with biological systems using electrical signals through Multi-Electrode Arrays (**MEA**), the responses of which are saved and accessed over time. Once charted, the path from Digital processor to Bioprocessor can unleash unbound, energy effective and scalable computational abilities. This work motivates from the exciting opportunity that the Neuroplatform offers to apply concepts of Machine Learning for the training *in vitro* BNNs.

1.2 Objective

The objective of this work is to lay grounds for development of Machine learning (**ML**) techniques for neurospheres. The project aims to investigate stimulation protocols which can cause a desired change in the activity of a neurosphere. The data collected from neurosphere is analyzed using various methods like correlation, to extract a ‘target’ information such as a pattern. This ‘target’ can then used by a ‘controller’ created for reinforcement learning. The controller tries to train a neurosphere in iterations by giving feedback in each iteration in order to generate more reproducible results. This work also aims for creating a reusable interface and computational package to facilitate future continuation of the research.

1.3 Scope

This research work has been carried out under the following scope :

- to understand the core FinalSpark system⁵, stimulation methods, input-parameters and output activity data.
- to carry out stimulations using different protocols in order to observe the neural activity changes and to identify metrics to explain the neural activity.
- to explore data analytics techniques like correlation in order to correctly quantify the changes in neural activity.
- to implement a closed-loop feedback algorithm which prepares the system for reinforcement learning.
- to develop a modular code package with the functions to support future work.

FUNDAMENTALS

This introductory chapter showcases the main concepts underlying this work including the basics of wetware computing and machine learning. It exhibits references from the past and ongoing work in the subject field and also covers the fundamentals of the FinalSpark’s neuroplatform setup for research⁵.

2.1 Wetware computing

Wetware computing is a field of study intersecting biology, neuroscience and computer science which makes use of biological systems to perform computational tasks. Instead of silicon based processors(hardware) and traditional programming(software), *wetware* systems consisting of living cells and tissues are leveraged to solve a computational task¹⁴. Wetware has the *hardware* characteristics of electronic devices and computational characteristics of a *software*. The cells and tissue are rich in neurons which act as the building block of network through which the information flows in the wetware. Wetware computing is the field that attempts to harness the computation capabilities of neurons in network in order to surpass the traditional computational systems.

2.1.1 Energy efficiency

A human brain consumes 20W power for its 86 billion neurons and their activity¹⁷. On the contrary world’s current fastest supercomputer Frontier, built with around 40,000 GPUs, consumes around 21 MW of power¹⁶. OpenAI ChatGPT’s²¹ single query consumes an estimated 2.9 Wh which is 10 times that of a google search query. In a recent study, the energy requirements of an Artificial Super-intelligence-computer (**ASI**) is estimated as 2.7 GW¹⁸. It is estimated with the help of an “Energy Requirement of Artificial SuperIntelligence equation (ERASI)” which makes use of brain’s energy use, ASI’s computational power, AI superiority as few of many variables. Even though a working silicon-brain replicating the human-brain capabilities is still far from physical reality, this comparison correctly shows the alarming energy demands in near future. The same work on energy requirements also concludes biological computing to be about 9×10^8 times more energy efficient than silicon computing¹⁸.

2.1.2 In vitro neural networks

Wetware computing works on the idea of creating a organic brain on a chip, where tens of thousands of neurons are put in a dish and micro-electrodes are added to act as an interface for

sending and receiving signals to and from the neurons. These neurons grow over time and form a dense neural network inside the chip. They can be trained using simple, repetitive signals which gets reinforced by getting updates through a closed-loop feedback cycle. There are many ongoing works in the field of Wetware computing currently, of which all are well-knit in the interdisciplinary fields of biology, neurobiology, physics, computer science, cognitive learning etc.

One of the recent works in creation of this organic brain is DishBrain¹⁹, a system to harness the adaptive computation of neurons in a structured environment. In vitro neural networks were ‘grown’ on MEA in silico computing chips using neural culture, and were subjected to a stimulated game world of ‘Pong’. Closed-loop structured feedback was provided by tapping into electrophysiological contact between the BNN and the silicon system. In response to the feedback, the cultures were able to self-organise their activity to display ‘intelligence’ over time and learn according to the goal-oriented feedback. This paper also conveyed that a feedback was essential for learning by the BNN, without which the responses produce was much random.

Another ongoing work²⁰ investigates the ability of neurons to learn to control a F22 Raptor aircraft simulation in real-time feedback system. Two out of 60 electrodes on a planar MEA with rat neurons are selected to ‘pitch’ and ‘roll’ the aircraft through a high frequency stimulation. The acquired response data is then used to update neural ‘weights’ which control the ‘pitch’ and ‘roll’ for the aircraft. It is observed that after several minutes of flight, the network begins to correct errors in the simulated flight path to a levelled flight path, thus showing that the neurons were able to transmit electrical activity through their network as well as adjust their response to be able to reach desired weights. This work is ongoing with attempts to use more than two electrodes on broader scale and to solve the problem of overfitting arising with longer flight duration.

2.1.3 Learning

Neuroplasticity is defined as the “*ability of the nervous system to change it’s activity in response to intrinsic or extrinsic stimuli by re-organising it’s structure, functions or connections*”²². Inducing neuronal plasticity into a BNN to achieve Organoid Intelligence²⁵ requires elements such as a training pattern to induce *memory*, a feedback loop to accomplish *training*, ML and statistical algorithms to quantify organoid function changes, and analysis of responses to determine a desired feedback.

Electrical stimulus has been used extensively over the years to achieve neuroplasticity. In their paper, Bakkum *et al* 2008⁴ designed an adaptive training algorithm to train neuroplas-

ticity and improve an animat's²³ performance; the animat being a motor output configured to move in a 2-dimensional plane controlled by a cortical neural network MEA. The movement of the motor output was mapped with the help of a population coding, termed as *Centre of Activity (CoA)* and defined as vector summation of responses on each electrode weighted by the location of the electrode.

A closed-loop-training consisted of capturing neural activity, movement of the motor output, assessment of performance by computer, to adjusted electrical stimulation, and back to capturing neural activity²⁴. As mentioned, movement of the motor was mapped with CoA of probe-electrodes. The next stimuli were adjusted according to the behavior of the motor output; with a random shuffled electrical activity to keep the same output behavior, or with a more controlled activity to change the output behavior. They were able to shift the neural activity towards the target with training of the neural plasticity of the animat.

2.2 The Neuroplatform

This section is an introduction to the basics of neuroplatform⁵, a remote platform developed by FinalSpark for research in Wetware computing. It enables remote electrophysiological experiments on neural organoids and capturing of the neural responses. The system can be accessed via an Application Programming Interface (**API**), and a published library provides methods to evaluate data and interact with the system through experiments.

The biological network used are spheroids called 'Neurospheres' or 'Forebrain Organoids' (**FOs**). Experiments can be performed with configurable parameters such as amplitude, and the output data is stored with metadata such as timestamp and spikes per minute. Also, spontaneous activity from FOs can be visualized live on their website¹⁰. The platform can be accessed programmatically through Datalore server⁹ on which Jupyter notebooks can be used to trigger experiments and access data with the help of classes defined in their `neuroplatform.py` interface.

2.2.1 Platform setup

A simple illustration of the neuroplatform setup is provided in Fig. 2.1.

- The neurospheres on a multi-electrode-array are held in an Incubator.
- The MEA is controlled by an Intan Controller for sending and receiving electrical signals.
- With the help of python notebooks, triggers are generated in a Raspberry Pi to send input electrical signals to the neurosphere.

- An Intan Software continuously collects out response read at electrodes and writes it to an InfluxDB database.

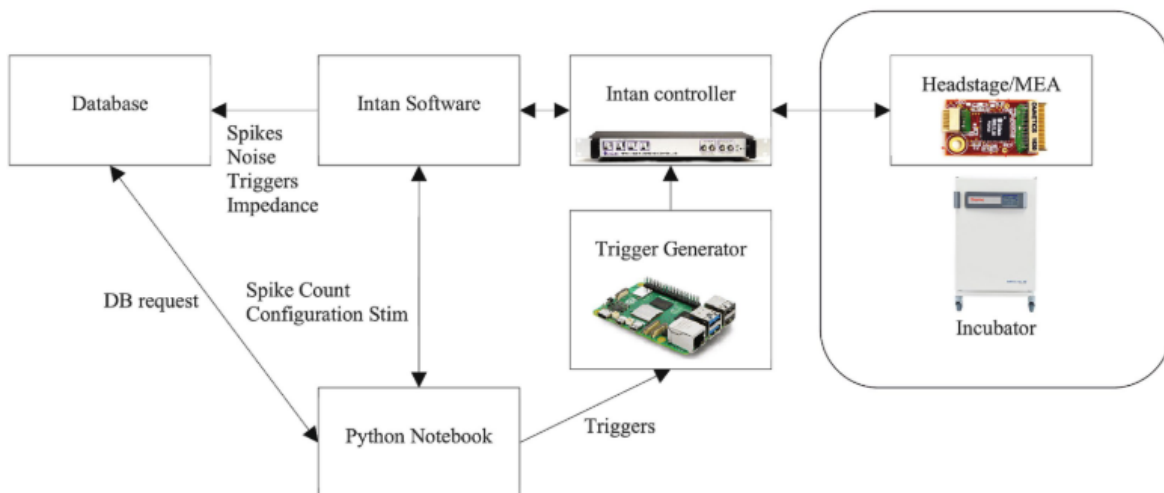


Figure 2.1: Stimulation and data recording setup at FinalSpark, source¹¹

2.2.2 MEA

An MEA is a device made up of multiple closely spaced electrodes, which can interact with the organoids and measure electrical signals from tissues like neurons or muscle cells. The ‘electrodes’ are also referred to as ‘channels’ interchangeably. The neuroplatform consists of four MEAs. Each MEA has four FOs (neurospheres) and each FO is connected with eight electrodes as shown in fig. 2.2a. Each neurosphere is also referred to as ‘**sub**’ in this work for ease of identification. During this project, the FU research group is given access to one MEA at a given time through a secure access token, which is unique. This ensures concurrency of experiments by avoiding conflicts among research groups. Electrodes of an MEA are arranged in a numbered position as shown in fig. 2.2b. The numbered channels help to visualize the electrical activity in a 2-dimensional pictorial way. Even for a changed organoid, the electrode numbering stays the same. The pre-numbered channels in each *sub* are also tabulated in A.2 for reference.

2.2.3 Stimulation experiments

Stimulation experiments can be done by sending electrical signals to electrodes on MEA with the help of a Raspberry Pi. The neuroplatform interface exposes methods with some parameters to conduct experiments on the neurosphere.

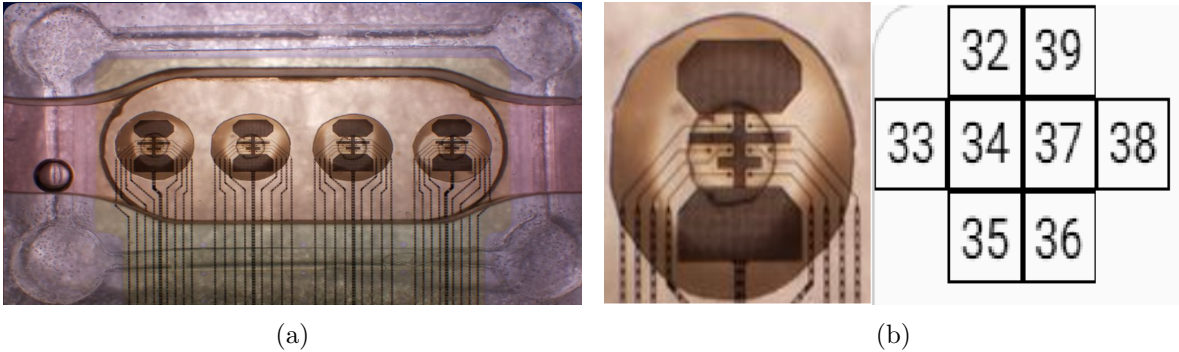


Figure 2.2: (a) MEA-2 comprising of 32 electrodes/channels (b) One neurosphere/Forebrain Organoid in MEA-2. Dots inside the inner sphere are electrodes/channels, pre-numbered and arranged in same positions, source-¹⁰

- **StimParam** class sets the parameters for stimulations such as *stimshape* (biphasic or triphasic), *intensity* (0.1uA to 20uA) and electrode number.
- **Trigger** class sets the number of triggers such as *single-start* for all electrodes, or a *pulse-train* to stimulate electrodes one after the other. There can be a maximum of 16 triggers set at one time
- **IntanSoftware** class sends the triggers to the electrodes with **StimParam** object with the help of the Raspberry Pi. The neurons in neurosphere receive the electrical pulse through electrodes and can react in the form of response electrical pulse which is captured by the **IntanSoftware** class.

2.2.4 Neural activity response

An Air-Liquid-Interface (**ALI**) system captures the activity on the channels by measuring the action potential, which is a rapid sequence of changes in voltage across a membrane⁵. The **IntanSoftware** continuously records the electrical activity from all electrodes in an MEA at a sampling rate of 30KHz and saves them in an InfluxDB Database. Only significant events which cross a certain threshold T are recorded in the database. A ‘**spike**’ is a ‘threshold crossing’ which gets captured when the voltage at an electrode crosses T . At every 3s, T is also updated to keep the thresholds updated with the neural activity.

Database class is used to access the recorded activity data for a timestamp or for a specific channel(electrode) and trigger information. The data stored in the system is recorded in the following format:

Spike: Time, Channel, Amplitude

Spike count: Time, Channel, Spike count per minute

In section 4, the stimulation protocols and analysis methods used on neural activity data used in this work is covered in detail. Experiments on the Neuroplatform MEA can be performed by booking a time slot to stimulate neurosphere. The response data is available to be 24×7 from the platform.

2.3 Machine learning

ML²⁶ is the branch of Artificial Intelligence which focuses on development of algorithms and statistical models which enables digital systems to learn from available data without being explicitly programmed. ML for BNN focuses on development of algorithms to achieve Organoid Intelligence²⁵. With recent breakthroughs in Deep learning²⁷ and Reinforcement learning, ML is being employed in numerous sectors such as technology, automobile, cyber-security, business and finance, healthcare and medicine to achieve AI in their function. There are several types of ML paradigms depending on the ‘learner’ machine, the training data and goal of the learning problem; many of them often overlap with one-another in terms of implementation and popularity.

2.3.1 Supervised learning

Supervised learning is the process of using prior experiences to gain knowledge. The learner is trained with the help of *labelled* data, and set of correct actions (mapping from input to output) acting as a ‘teacher’. The learner reads from this mapping and then predicts output of new data. It is a most commonly used type of ‘task-driven’ ML in order to predict values for new data.

2.3.2 Unsupervised learning

Under unsupervised learning, the learner has to create an objective function by identifying patterns, structures or relationships from *unlabelled* training data. There is no mapping provided as a ‘teacher’. These methods are ‘data-driven’ and are much essential for making sense of large complex datasets without any prior knowledge of relationship in them.

2.3.3 Reinforcement learning

Reinforcement learning(RL) is a ML technique in which a machine learns through a trial-and-error learning process by directly interacting with it's surroundings. The machine is not taught on which response to send, instead it is given feedback and gets rewarded on every response. The closer the response is to the target, the higher is the reward. The response is used to 'reinforce' the behaviour of the machine and thus, it learns by maximizing the reward.

Reinforcement Learning is different from Supervised Learning as instead of returning the correct actions from an available mapping, the learner uses rewards and punishments for determining the correct response. Reinforcement Learning is different from unsupervised learning in terms of goals. Goal in unsupervised learning is to find patterns in data, while in Reinforcement Learning, the goal is to maximize the total reward.

In RL algorithms, the machine is referred to as an 'agent' which tries to learn and create knowledge. An 'environment' is the surrounding of the agent, with which it interacts. All agents should have concrete goals, therefore Reinforcement Learning has a much larger integration with statistics, optimizations, and mathematics which help in creating these goals by dividing them into sub-problems and solving each of the sub-problems individually. There are several techniques which have contributed to computation of these goals such as Dynamic Programming, Monte Carlo methods, temporal-difference learning, etc.²⁸.

RL in biological neural networks

Animals, including humans, learn from their environment through trial and error and by adjusting their behaviour to maximize rewards and minimize punishments. The nervous system of humans and many other species have influenced many aspects of reinforcement learning algorithms. While Artificial Neural Network rely heavily on methods such as Deep Learning to understand the intrinsic manifolds of the network's activity²⁷, it is quite logical to apply Reinforcement Learning concepts to train BNNs to a goal-oriented tasks. These concepts are being widely used to process large datasets of neural activity to help identify patterns, create policies to provide rewards or punishments for training the agent.

2.3.4 Closed-loop feedback protocol

An important concept in RL is to iteratively provide feedback to the agent in terms of rewards which make the agent learn to produce correct results. This is achieved by creating a *closed-loop* in which feedback is sent back to the agent and the agent adjusts the actions accordingly. A simple example of a closed-loop feedback system design is shown in figure.2.3. The Agent performs an action A_t on the Environment which moves to a new state S_t . Depending on

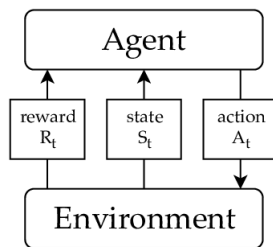


Figure 2.3: Classic feedback loop of Reinforcement Learning²⁹

the correctness of the state S_t , the environment provides a feedback reward R_t to the agent. Based on the positive/negative reward R_t , the agent is encouraged/discouraged to take actions similar to A_t in upcoming steps. This feedback process is implemented for a number of iterations, so that the agent learns to maximize its positive rewards iteratively.

A feedback cycle can also be *open-loop*, in which there is no impact of output on the agent's next action. These require manual intervention and are not automated, but are useful in data analysis and providing insights on agents functionality. The iterative open-loop is executed and large amounts of output data can be collected and recorded for subsequent uses, without having any impact of outputs on inputs in next cycle.

2.4 Data analysis and defining targets

One of the prominent tasks in RL is to define target/goal using a training data (recorded using *open-loop* setup). In most cases of learning algorithms based on unlabelled data, the environment is model-free, i.e. not much information is known about it. The only information available is in the form of large previously recorded dataset. In BNN, the data available is the neural activity which has been recorded over a longer period of time. The exact computation which happens inside the network, is not known. Therefore, it is imperative to use some data analysis techniques to identify patterns in data and define targets for the learning algorithm.

2.4.1 Correlation in time series data

Correlation is the measure of strength of a linear relationship between two variables in a dataset. The most common way of correlating multi-variate data is by calculating a correlation coefficient between its variables. The absolute value of the correlation coefficient gives the strength of the relationship between these variables; a larger coefficient represents a stronger relationship. Out of several coefficients available in statistics, the most widely used are Pearson's correlation coefficient ' r ' and Spearman's rank coefficient. In this work, we use

Pearson's ' r ' or $\text{corr}(X, Y)$ ³³ defined in eq.2.1 with $\text{Cov}(X, Y)$ is the covariance between X and Y ; $\sigma(X)$ and $\sigma(Y)$ are the standard deviation of X, Y as two vectors to be correlated.

$$\text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma(X) \times \sigma(Y)} \quad (2.1)$$

The value of the correlation coefficient r ranges between -1 and 1. If $r > 0$, X and Y are positively correlated i.e. increase in X , increases Y and vice-versa, for $r < 0$, X and Y are negatively correlated and $r = 0$, X and Y are not correlated, i.e. they do not have any relationship with each-another.

Two time series data can also be compared using Pearson's r correlation coefficient to measure how similar they are with each other. In a cross-correlation of time series, they are shifted onto each other with different lag times. In the graph of cross-correlation, a prominent peak occurs at the lag-time at which there is maximum correlation (similarity) between the compared series. The existence of this prominent peak suggests existence of a relationship between two time series. This work uses time-correlation to compare neural activity from different channels in order to find level of similarity between them.

2.4.2 Principle component analysis

Principle component analysis³⁰ is a statistical method of dimension reduction in data, in which information comprising of several variables are combined into fewer variables known as Principle Components (**PCs**)³². In a multivariate dataset, the PCs are calculated using total variance of original variables and represent the underlying interpretation of the dataset.

The PCs represent the directions of the data which explain a maximum amount of variance. They are constructed as linear combinations functions of initial variables. There exists ' n ' PCs for a ' n ' dimensional dataset. However the first Principle Components is the one which captures maximum variance of the dataset. The steps to calculate PCs for a given dataset are:

- Standardization of dataset with variables, so that each of them contributes equally to the analysis. Mathematically, this is achieved by removing the mean and dividing by the standard deviation for each value of each variable.

$$X_{standard} = \frac{X - \mu_X}{\sigma(X)} \quad (2.2)$$

where μ_X = mean of X , $\sigma(X)$ = standard deviation.

- Covariance ($\text{Cov}(X, Y)$) of each pair of n variables is computed and saved in a $n \times n$

Covariance Matrix to observe interdependence of variables. Covariance is calculated using mean and count of values.

$$\text{Cov}(X, Y) = \frac{\sum(X - \mu_X)(Y - \mu_Y)}{k - 1} \quad (2.3)$$

A sample covariance matrix for 3-dimensional dataset is shown below. The entries in the matrix are symmetric with respect to the main diagonal.

$$\begin{bmatrix} \text{Cov}(X, X) & \text{Cov}(X, Y) & \text{Cov}(X, Z) \\ \text{Cov}(Y, X) & \text{Cov}(Y, Y) & \text{Cov}(Y, Z) \\ \text{Cov}(Z, X) & \text{Cov}(Z, Y) & \text{Cov}(Z, Z) \end{bmatrix}$$

- The eigenvectors of the Covariance matrix gives direction of most variance and their corresponding eigenvalues form the Principal Components. By ranking the eigenvalues, from highest to lowest, the principle components are found in the order of significance.
- In general a principle component is a linear combination of variables. For example, principle components of a 2 variable (x,y) dataset are $pc_1 = a_{1x} * x + a_{1y} * y$ where a_{1x} and a_{1y} are coefficients of contributions of x and y respectively; and $pc_2 = a_{2x} * x + a_{2y} * y$ where a_{2x} and a_{2y} are coefficients of contributions of x and y respectively;
- The percentage contribution of PCs in information are calculated by dividing eigenvalue of component by sum of all eigenvalues. The most contributing PCs are chosen to create feature vector which has p dimension for p chosen components out of n .

$$\text{Feature vector} = [\lambda_1, \lambda_2.. \lambda_p]^T \quad (2.4)$$

- The standardized dataset can be re-casted along these p features and thus reducing the dimensions of the initial data.

$$\text{Transformed dataset} = \text{Dataset} \times \text{Feature vector} \quad (2.5)$$

With the neural activity captured in BNNs on a MEA, PCA can be utilized to identify the contribution of electrodes in the activity and help in creating target patterns for learning algorithm. The electrodes/channels can be treated as variables for the dataset, and the 8-dimensional dataset can be reduced to lower dimensions.

OBJECTIVE

With the fundamental concepts in Chapter 2, the objective of this work is to lay grounds for a reinforcement learning technique with a closed-loop-feedback algorithm for the neurosphere. The aim is to investigate strategies with which a learning algorithm can be developed. The algorithm begins with an input to a FO and trains it to furnish a desired output with the help of a success/failure feedback. A target pattern is identified using historical activity of the FO, and the FO should be trained to return the same pattern by adjusting the stimulation parameters in every training loop. A basic diagram representing the objective of this work with closed-loop feedback learning algorithm is provided in figure. 3.1.

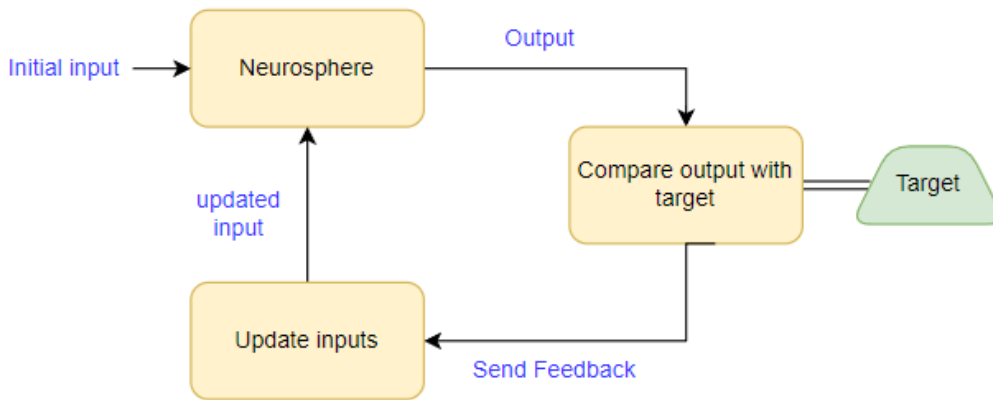


Figure 3.1: Representation of the closed-loop feedback algorithm for a neurosphere.

The **input protocol** and parameters for stimulations should be identified by performing different experiments and observing the subsequent output behaviour. The protocol needs to be configurable so that it can be re-used across different neurospheres as biologically all organoids are different and in the current platform, they can last up to 100 days⁵.

The **output data** (2.2.4) being captured in the system should be analyzed for underlying patterns in neural activity. Each neurosphere has activity surrounded across the eight attached channels in MEA and it is also capable of preserving information over time. Hence, techniques of dimensionality reduction are required to study multivariate dataset and its behaviour over time. This work specifically investigates the use of Principle Component Analysis in detail.

A **quantifiable target** should be calculated which can be compared with the results of the stimulation in each learning loop. Either the average of neural activity over time, or value of contributions of each channel, etc, can give a suitable target.

A **controller** should validate the outcome in iteration and based on deviation from the target value, should adjust the stimulation input for the next cycle as a **feedback**.

The iterations should be repeated till a convergence in target is obtained. The feedback cycle should be designed in a way that it supports multiple neurosphere in MEAs of the neuroplatform. The interfaces should be made modular and easily extensible.

COMPUTATIONAL METHODS AND IMPLEMENTATION OF TRAINING ALGORITHM

This chapter illustrates the different steps which were performed during the course of research work. It begins with describing the initial experiments, followed by analysis of neural activity data, identification of targets for the learning algorithm and finally, the set-up for the closed-feedback-loop algorithm. Every section includes the implementation details and highlights the important observations which aid in achieving the objective of this work. The results from each section supported in deciding the investigation strategy for subsequent steps.

4.1 Initial Experiments

At the start of the research, initial experiments were performed on the neurosphere which were allocated to the team by FinalSpark, in order to observe the behaviour of the platform and get familiarity with the input parameters, simulation protocols, and the behaviour of organoids towards electrical signals through response spikes. Fig. 4.1 shows the neural activity data on channels [0-7] during a 10-minute slot after sending electrical pulses of $1\mu A$ on channel numbered (0,1). Fig. 4.1a shows the amplitudes of events in course of the experiment (accessible via `db.get_spike_event` method), and Fig. 4.1b shows the number of spikes captured per minute (accessible via `db.get_spike_count` method).

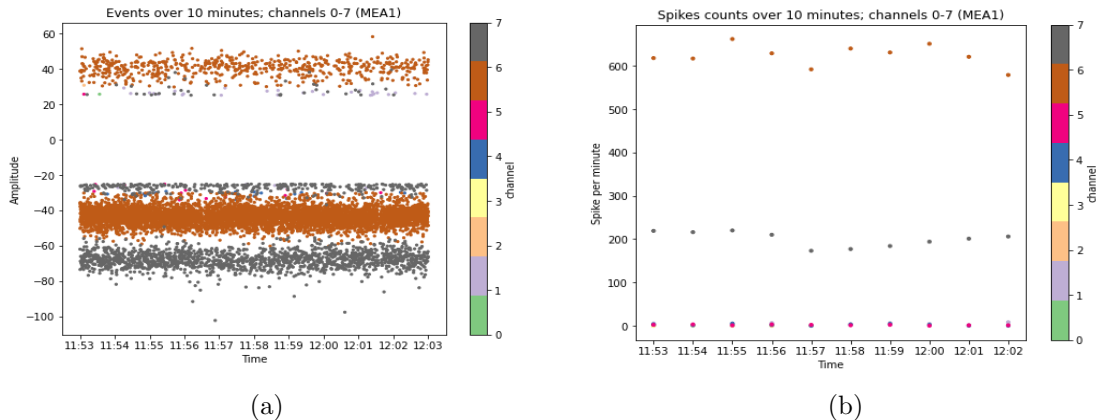


Figure 4.1: (a) Spike event with amplitude (b) Spikes per minute for channels 0-7 in MEA-1

Over the course of research, the MEA provided to FU Berlin for experiments, have been changed and updated with new Forebrain Organoid/neurospheres, along with the accesstoken for the platform. This is the reason why examples and results captured in this report are not restricted to one MEA but have results from

both MEA1 and MEA2. However, these updates do not impact the implementation and algorithms adversely as the computation code is kept modular, extensible and MEA-independent.

4.1.1 Centre of activity

The electrodes in each MEA are arranged in prescribed order and are attached to a specific region of a given Organoid. Hence, the concept of examining the activity behavior with the weighted average of spikes according to electrode position, can naturally be an important metric for quantifying the organoid behaviour. The metric is known as the Centre of Activity (CoA) of the neurosphere. Also used in defining the motor-mapping in 2008 stimulation of cortical networks for goal-oriented task⁴, **CoA** is calculated with electrode positions and activity spikes to observe the movement of activity inside the neurosphere.

The electrodes are positioned on a 2D grid with fixed coordinates. CoA is calculated based the weighted average of position coordinates and total number of spikes observed in each channel for each trigger in the given duration of the experiment.

$$\text{CoA} = \frac{\sum_{n=i}^j s_n \cdot v_n}{\sum_{n=i}^j s_n} \quad (4.1)$$

In eq 4.1, s_n is the spike count for nth channel, v_n is the vector representing 2D position of the channel, whereas i and j are the starting and ending index for the channels attached to the neurosphere such as (0,7), (8,15), (16,23), (24,31) for the 4 neurospheres in MEA-1.

An example of shifts in CoA with in response to stimulation is shown in Fig. 4.2. Since the spike count were very high for channel 5,6 & 7, the centre of activity tends to concentrate near these channels, with minor shifts when other electrodes were stimulated. In Fig. 4.2b, there were greater shifts when channels 5,6,7 were stimulated.

4.2 Stimulation protocols

After the initial experiments and establishment of CoA as a metric, the approach to analyze the behaviour of electrodes under electrical pulses is motivated by the work of Bakkum et al. [4] on how to stimulate a neural network towards a goal oriented task. The goal of this section is to move CoA more towards the stimulated channels, so that a complementary relationship between the ‘stimulated’ channels and the most active ‘channels’ in response data can be established. Due to plasticity of neurons, it is quite possible that the neural activity lingers in the neural network for a longer period of time after a stimulation is completed. In order to flush out the effect of a previous stimulation, a shuffled stimulation is run on the neurosphere,

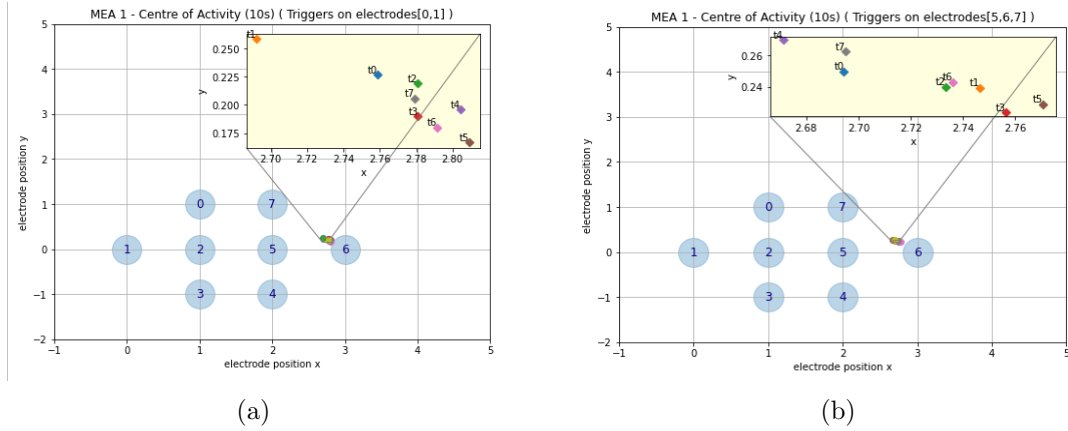


Figure 4.2: Stimulations on channels [0-7] in MEA-1 (a) 10-second shifts in CoA during stimulation of channels (0,1) (a) 10-second shifts in CoA when channel(5,6,7) are stimulated.

before each controlled (defined electrode) stimulation.

4.2.1 Shuffled background stimulation SBS

A random set of ‘n’ electrodes in the selected MEA is stimulated for predefined duration. The triggers are randomly set on the selected electrodes with a fixed number of pulses. This is an attempt to clear out any residual memory from the neural network by shuffling the activity, hence the name shuffled background stimulation (**SBS**). An example of SBS can be :

1. randomly select 2 channels from MEA-2 (channels 32-39) as [33, 37]
2. send triggers in combinations of [33, 37], [33, 33], [37, 33] etc. for 10 minutes.

An example of such a simulation with change in Centre of Activity during the experiment duration is shown in Fig. 4.3a. Through continuous shuffling the calculated CoA are scattered across the entire network area.

4.2.2 Controlled stimulation

After having flushed out the memory through an SBS stimulation, a controlled stimulation is carried out by using specific set of electrodes. The trigger sequence is kept constant, unlike the shuffled sequences in SBS. The `StimParam` object controls the electrodes to be triggered and signal properties such as number of pulses and amplitude. An example of a Controlled Stimulation is shown in Fig. 4.3b. Electrode 34 is triggered continuously with amplitude $10\mu A$ for a span of 10 minutes. A quick observation result is the concentration of CoA near electrode 34 as a result of continuous stimulations on it.

While the controlled stimulation is expected to move the centre of activity towards the controlled channel, it is not always a prominent case. In some cases, a few channels in MEA are found to be so active that even after a SBS stimulation, they continue to show the highest level of activity. In these cases, a controlled stimulation is unable to move the CoA by a substantial amount.

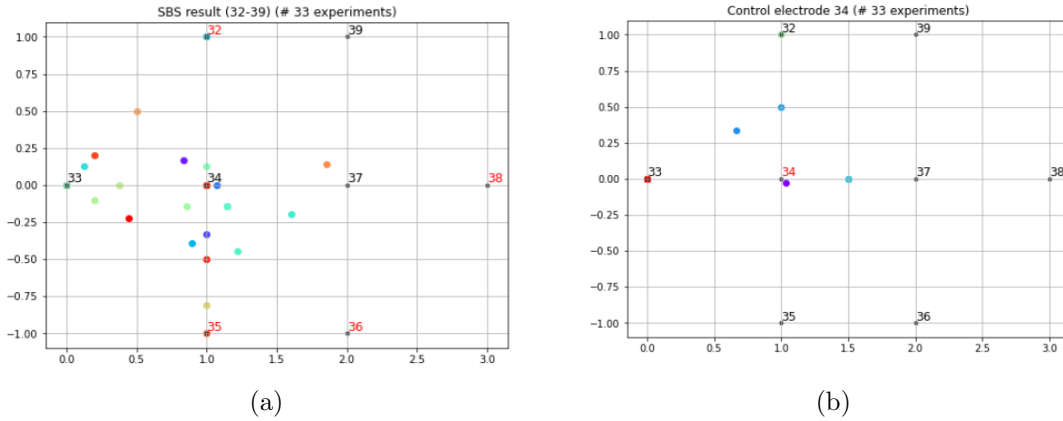


Figure 4.3: (a) CoA shifts observed in SBS for 10 min on MEA 2. (b) CoA shifts observed in Controlled stimulation on channel 34 for 10 min on MEA 2. (The numbered dots are channels; Channels numbered in ‘red’ are the stimulated ones; The scatter plot is for CoA for different timestamps.)

4.2.3 Controlled Experiment Results

Controlled stimulations were run for different combinations of channels in an MEA; each experiment was performed with the same `StimParam` values for the electrical pulse. Fig. 4.4 shows shift in CoA for some combinations of channels after being stimulated. Before each of these experiments, a SBS was also run to flush out any existing memory from previous stimulations.

- The most important observation of the controlled experiments are shifts in the Centre of Activity. The channels being stimulated is near the centre of highest activity in most cases.
- Channel 37 in this experiment set took over most of the activity. However, the shifts in CoA can be observed with few other stimulations.
 - In Fig.4.4a and 4.4b, it is seen that some activity moves towards to channel 33 when stimulated.

- In Fig.4.4c, channel 34 pulls CoA towards it when stimulated.
 - In Fig.4.4e, channel 35 pulls some activity towards it when stimulated in combination with channel 36. While in Fig.4.4f, channel 33 being stimulated alone, pulls the CoA towards itself in a more linear transition.
- SBS is helpful to ease out the activity and spread it across the channels.

Observation

The results of the control stimulations show that the centre of activity can be moved with time. Thus, a controlled experiment with ‘selected’ set of electrodes can be used as an efficient **input protocol** in our closed feedback loop to train a neurosphere towards a target goal. The selection of electrodes in each cycle will depend on the feedback provided by the controller which compares the output with the target. Since the electrodes are numbered in the platform, this input protocol can easily be created and updated, along with other stimulation parameters such as amplitude of a pulse and number of pulses in a trigger.

4. COMPUTATIONAL METHODS

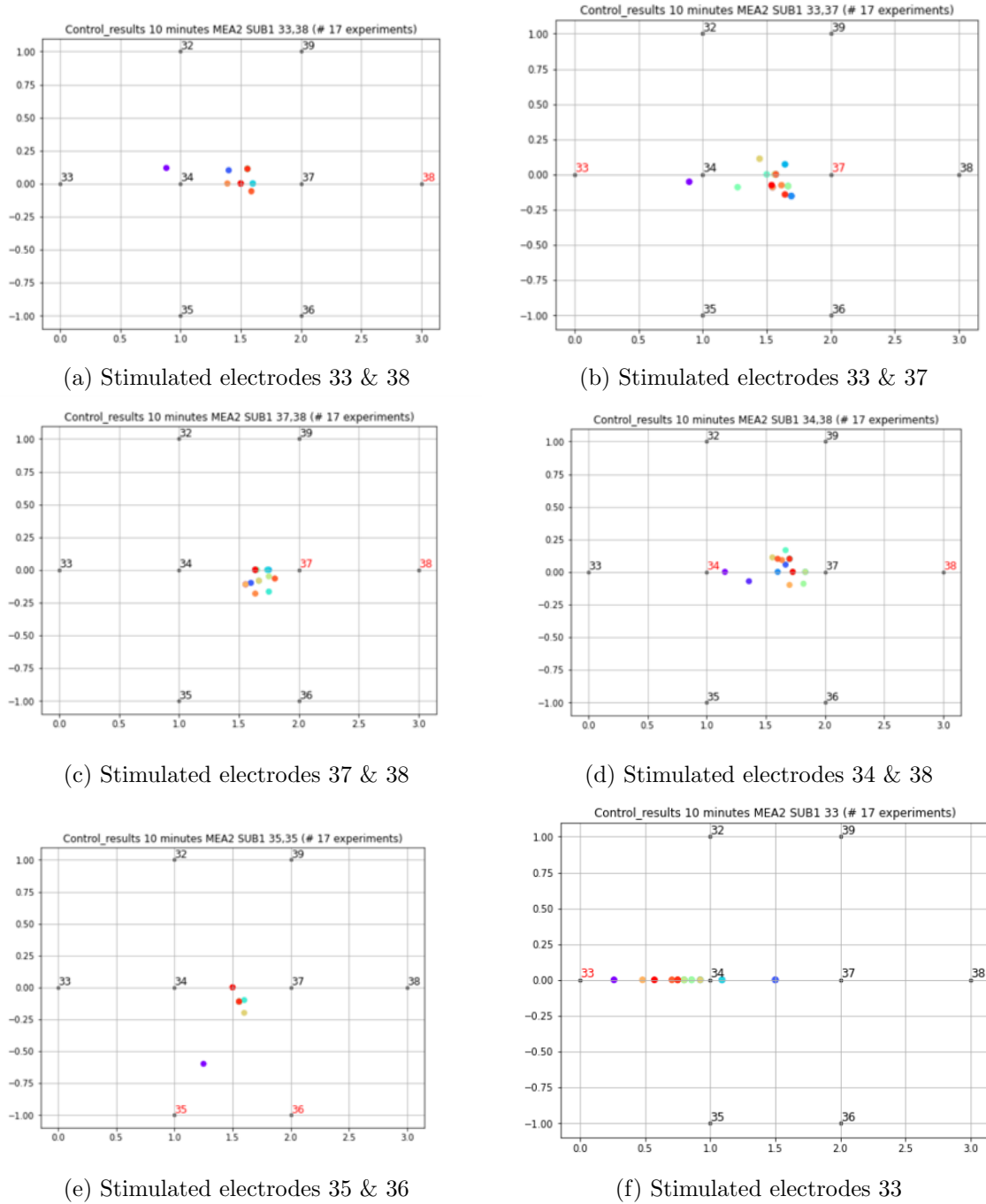


Figure 4.4: CoA shifts in Controlled stimulation on channels [32,39] in MEA 2; (The numbered dots are channels; Channels numbered in ‘red’ are the stimulated ones; The scatter plot is for CoA for different timestamps.)

4.3 Output data analysis

It is though challenging to observe concrete pattern shifts in CoA as some of the channels (37 in Fig. 4.4) are very dominant in terms of number of spikes. Similar conditions are observed with the same stimulation setup in other neurospheres of MEA; where highly active channels dampen the stimulation effects on other channels. An organoid replacement in the system affects the activity on the electrodes as well; which is expected as each organoid behaves differently than the other. Thus, in order to design a numeric and quantifiable target for the learning algorithm, further analysis of neural activity data is needed to identify metrics other than the CoA. This section highlights different methods used to analyse the neural activity data.

4.3.1 Capturing neural activity

Getting the large neural activity data (2.2.4) from neuroplatform for a long duration of time synchronously while attempting to study the nature of correlation between them resulted in a very high computation time. Hence, the activity data from FOs for a time period are recorded and saved in Datalore environment, so that they are available for instant computation. Stimulation ‘spike events’ are collected through an open-loop stimulation process, where different electrodes are triggered in SBS or controlled stimulation, and the response activity is saved for analysis. Spontaneous or unstimulated ‘spike events’ are captured when there are no triggers to the FO. Total ‘spike events’ for each channel in a 60 seconds window is calculated using the neuroplatform endpoints. This activity data can be captured for 1-10 hours and is saved separately for each day, to be read easily in data processing. Table 4.1 shows a sample of activity saved in Datalore.

Time	spikes	channel
2024-09-17 13:10:00	472	86
2024-09-17 13:11:00	632	86
2024-09-17 13:12:00	676	86
..

Table 4.1: Sample saved activity data for a channel(86)

4.3.2 Correlation between channel activities

One of the statistical methods used to measure relationship between variables in a multi-variate dataset is correlation analysis. Since our output data consists of activity from multiple channels, the foremost step was to investigate the relationship between different channels with respect to their neural activity. We use the Pearson’s correlation coefficient mentioned in 2.1 to correlate different channel time-series data.

Temporal correlation between time series

When each of X and Y are variables realized in a time-series, then the correlations of X with Y are called time-correlations or *temporal* correlations. This concept is frequently used in statistics and signal as a measure of similarity between two time series or signals. A time series can be called as ‘signal’. When a ‘signal’ is cross-correlated or slid onto the other with a certain lag time, the shifts in the behaviour of the signal with each other can be observed. A signal with itself will always have the maximum auto-correlation at zero lag time, with further peaks hinting at propagation of information in time or repetition of a pattern.

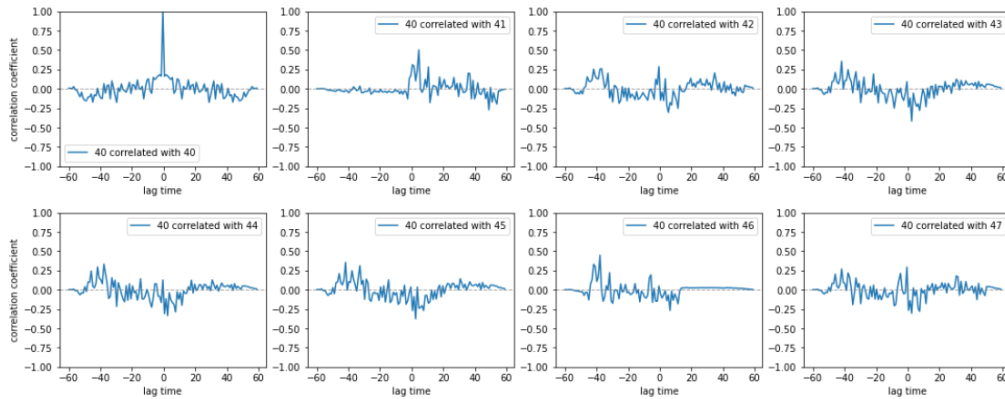


Figure 4.5: Time-correlation of Channel 40 with other channels in MEA2 over 1 hour of neural activity

Since a time series data for channel activity is available, using Pearson correlation the time-correlation of activity between two channels can be checked to see similarity between them. Fig. 4.5 shows the time-correlation of 1-hour activity in MEA-2 between channel 40 and others. As expected, a single peak at can be observed at $Corr = 1.0$ at lag-time=0 when activity in channel 40 is correlated with itself. In a time-correlation graph, a prominent peak occurs at the lag-time at which there is maximum correlation (similarity) between the compared components. However, in all these comparisons of channel activities, there is no

prominent peak obtained, and the values of the correlation coefficients are also low. This was the case observed across the 4 neurospheres in the MEA-2 suggesting that activity in one of the channels may not have a substantial impact on the other.

Correlation coefficient matrix(CCM)

Correlation matrices are $n \times n$ matrices used to numerically display the correlation coefficients calculated pairwise between n variables. The absolute value of coefficients denote the strength of the relationship and the $+/-$ sign denote the direction of the correlation. The value in the diagonal is always 1.0 , representing the relationship of variable with itself. The correlation coefficient matrix is used to compare the relationship between channels. Also, the plots of correlation matrix during an unstimulated period and a controlled stimulation (4.2.2) was useful to analyze the impact of a controlled stimulation.

Fig. 4.6 shows the CCM calculated in a neurosphere in MEA-2 which undergoes a controlled stimulation. Fig 4.6a shows CCM containing values from channel 40 and 41 which are the only 2 electrodes having activity in the neurosphere. At the start of a controlled stimulation, the first activity is observed in channel 42 (Fig. 4.6b), which is the closest to channels 40 and 41 as shown in Fig. 4.6e. During the stimulation period, the activity gets propagated to other remaining channels, which show low correlation with [40,41] but a high correlation of 1.0 with each other (Fig. 4.6c). Occurrence of concurrent activity on these electrodes can be the cause for high correlation values. Five minutes after the experiment, the activities subside in other channels, while [40,41,42] continue to be active but with low correlation values.

Observations

- The channels activities are not **highly** correlated due to the absence of a clear spike in the time correlations plots. The values of correlation coefficients calculated are very low.
- However, in multiple cases of stimulations, electrode 42 was the first one to record activity when the nearest electrodes 40 and 41 are stimulated, followed by other electrodes. The farther the electrodes the later they respond, which shows a flow of activity in the neurosphere that loses coherency as it moves forward.

These results prove an existence of neural activity flow across channels, however the relationship between the channels is still not quantifiable. As one of the objectives of this work is

to create a quantifiable target for the learning algorithm, which the neurosphere can aim to achieve, the neural activity output data is further investigated through PCA.

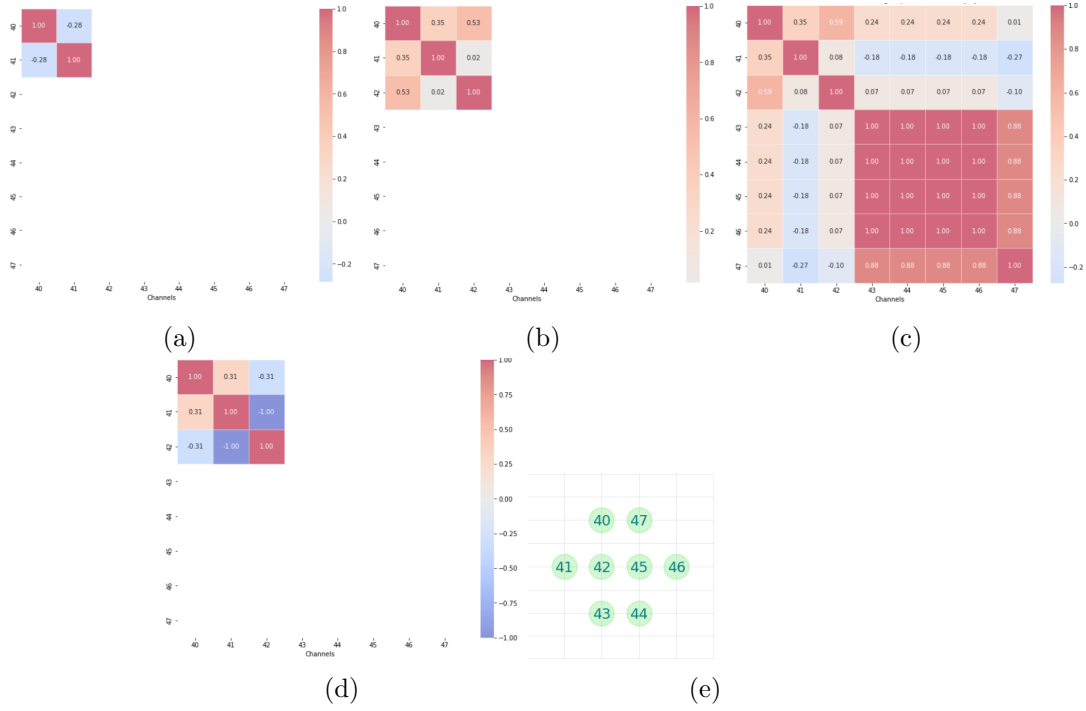


Figure 4.6: Temporal Correlation Coefficient Matrix of channels 40-47 in MEA-2. (a) CCM right before stimulation start, activity is present only on channels 40 and 41 with low negative correlation. (b) CCM at the beginning of controlled stimulation on electrodes [40,41] (c) CCM right after the end of 5-min stimulation (d) CCM during course of 15 mins including stimulation period. (e) Representation of physical positions for electrodes 40-47

4.3.3 PCA

As detailed in 2.4.2, Principle Component Analysis is a dimensionality reduction technique which helps in identifying underlying relationships between variables in a dataset. In order to study the relationship of channels inside neurosphere in details, PCA is employed on neural activity output data. The objective here is to be able to use PCA to find components which can help quantify a desired target for the Reinforcement Learning algorithm.

Restructuring of input data

The analysis is done by taking pre-recorded neural activity data for a substantial duration from the neuroplatform. This data includes stimulated time periods as well as un-stimulated activity from electrodes. The data is standardized with variables by reconstructing it as shown in table 4.2. Since overall activity in neurosphere is a combination of individual electrodes, the 8 electrodes attached to the neurosphere form the continuous *variable set* or '*features*' for the data. Principle components are new variables which are calculated as a linear combination of these 8 initial variables, in such a way that most of the information contained in these initial variables are compressed into the first components of PCA.

channel_1	channel_2	channel_3	channel_4	channel_5	channel_6	channel_7	channel_8
5	823	6	0	897	90	361	0
1	810	0	11	171	24	870	2
..

Table 4.2: Sample restructured data for principle component analysis

Calculation of principle components

The scikit³⁴ library provides a PCA³⁶ module which can be used to calculate the components easily. `pca.fit()` method calculates the covariance matrix of the features and then finds the principle components by calculating the eigenvalues and eigenvectors of the covariance matrix. The 8 features (one for each channel) in the dataset create 8 principle components as linear combination of these features and the first components contain the maximum information explaining the dataset. This can be verified by plotting the eigenvalues of the covariance matrix in descending order.

As an example, Fig. 4.7a shows covariance matrix obtained by applying PCA on a dataset of channels 40-47 on MEA-2. The eigenvalues plot in Fig.4.7b shows that the first principle

component, henceforth named as PC1, has the maximum contribution in explaining the data. PC2, the second principle component shows relatively less contribution as compared to PC1. Since the first two eigenvalues are much larger than the rest, they contain enough information about the 8-dimensional signal. Hence, keeping just 2 PCA components is justified for dimensionality reduction.

Also, the amount of variance explained by each of the selected components is shown by `explained variance`; which is available as a part of the PCA object created using `pca.fit()`. Fig. 4.7 show calculation of PC1 and PC2 from activity data. In this example, the calculated values for PC1 and PC2 are :

```
explained variance = (123.46101434, 20.58236248)
```

```
explained-variance-ratio = (0.83034538, 0.13842807)
```

Since the sum of `explained-variance-ratio` for both components is 0.968 close to unity, it is justified again to use the two highest principle components.

Transform original data using principle components

After choosing the most important Principle Component, PC1 and PC2 in this case, dimensionality reduction is applied to the dataset with `pca.transform()` method, i.e. the data is projected on the PC1 and PC2. This is similar to creating one condensed series for each Principle Component. The ' n -dimensional' data is reduced to '2-dimensions'. Fig. 4.7d shows a condensed signal created after applying dimensionality reduction on 8-channel data.

Observations

When compared with the spikes count data in Fig. 4.7c, the compressed series with PC1 in Fig. 4.7d has the closest resemblance to spikes count in Channel 42, which is reasonable as Channel 42 shows the highest spikes count. This is synonymous to the behaviour observed in the case of CoA, that the neural activity average was centred around the most active electrode. Hence, using principle components to continue investigation for a quantifiable target for learning algorithm is justified. Also, since CoA is calculated in a 2-dimensional space, using 2 principle components can also give grounds for comparison between the changes in CoA and the Principle Components.

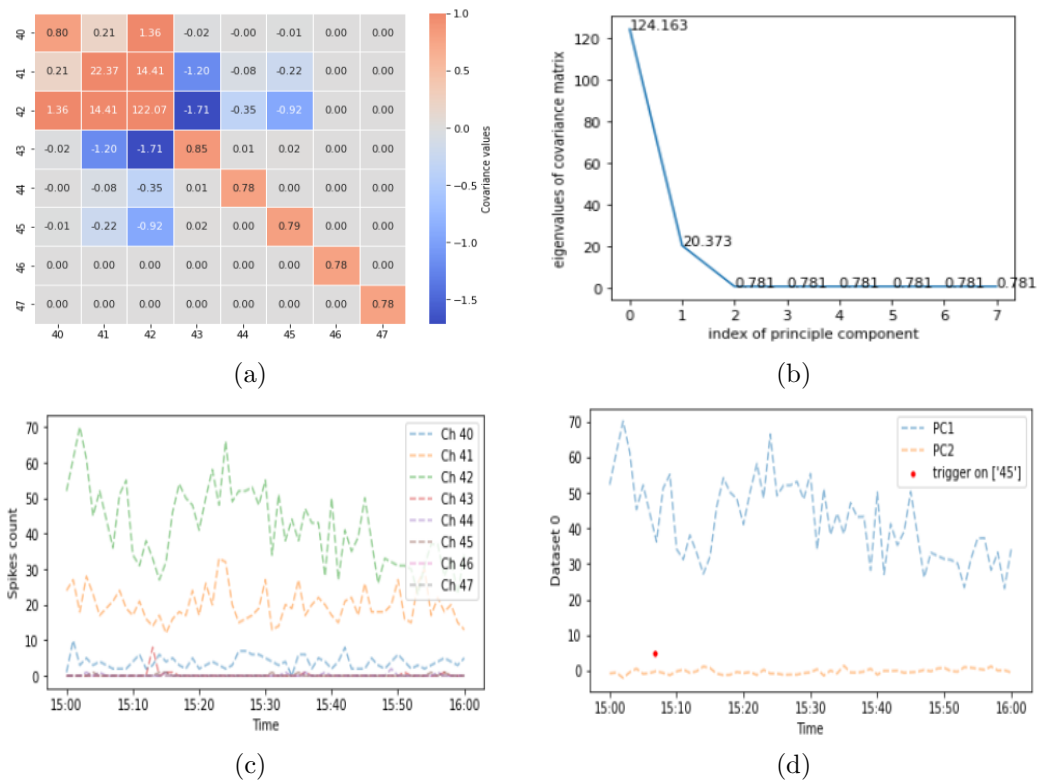


Figure 4.7: PCA on 1-hour neural spikes data for Channels 40-48 (MEA2) on 08.08.2024 (a) Covariance matrix of structured data. (b) Eigenvalues of covariance matrix, the largest eigenvalue corresponds to PC1 and has maximum explained variance. (c) 8-channel spikes count time-series for observed duration (d) Transformed time-series after dimensionality reduction using PC1 and PC2.

Uniquely trained PCA object

For any data analysis method, the analytics object is first trained on subset of data known as 'training' dataset, with which it defines the parameters and components. This object then makes use of information it contains to predict results in several other 'test' dataset. Similarly for a neurosphere, we need a PCA object which is initially trained on historical activity data using combined results from stimulated and unstimulated periods. It has to be ensured that the principle components stay the same for one organoid, and it captures the essence of the neural activity in that organoid. This process is pictorially shown in Fig. 4.8, and has been implemented with help of a Datalore notebook.

- Neural activity data from different days, including stimulated and unstimulated activity is recorded in Datalore.
- PCA is performed on this data, the resulting components are verified for **explained variance ratio**. In the case when **explained variance** is small or the eigenvalues of first 2 components are not far apart, more data needs to be collected and the object should be trained again.
- The verified object is now saved globally in Datalore workspace files. Every neurosphere inside a MEA gets it's own PCA trained object.
- For any future PCA analysis, this object is read from Datalore workspace files. Since each neurosphere is unique, this trained object becomes invalid when the organoid is refreshed. In this case, the PCA trained object should to be created again with data from new organoid.

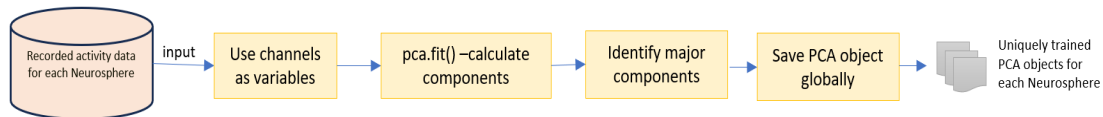


Figure 4.8: Training of unique PCA object for each neurosphere.

PCA on stimulated vs unstimulated data

After establishing that 2 principle components can be helpful in explaining the neural activity data, different datasets of unstimulated and stimulated activities for one neurosphere are collected and transformed using PCA object. In the transformed 2-dimensional data, the PC1 and PC2 components are visualized using a 2D histogram for spatial distribution. A 2D histogram takes 2-dimensional values X and Y and divides the plot area into hexagonal bins. Each bin is coloured differently based on the number of (X,Y) items that fall into the bin area of the plot. It is similar to a 1D histogram showing a distribution by count of bins along x-axis, but instead of using one axis, it divides the plot area into bins.

On comparing PC1 and PC2 components of data for large time intervals, it was observed that the histogram plots were sparser when data is collected during stimulation as compared to a unstimulated activity. The PC1 and PC2 components tend to loose relationship as PC1 is explains the neural activity more closely and gets more affected by the stimulation of electrodes than PC2. An example is shown in Fig. 4.9. Counts of both components are relatively concentrated before stimulation. The stimulations on channel 50 disrupts the counts and the plot becomes sparse before going back to a concentrated blob. These changes are observed in an hourly time-duration. Similar instances are observed in other neurospheres, with 2D histogram turning sparse whenever there are stimulations of electrodes, with maximum changes observed in PC1.

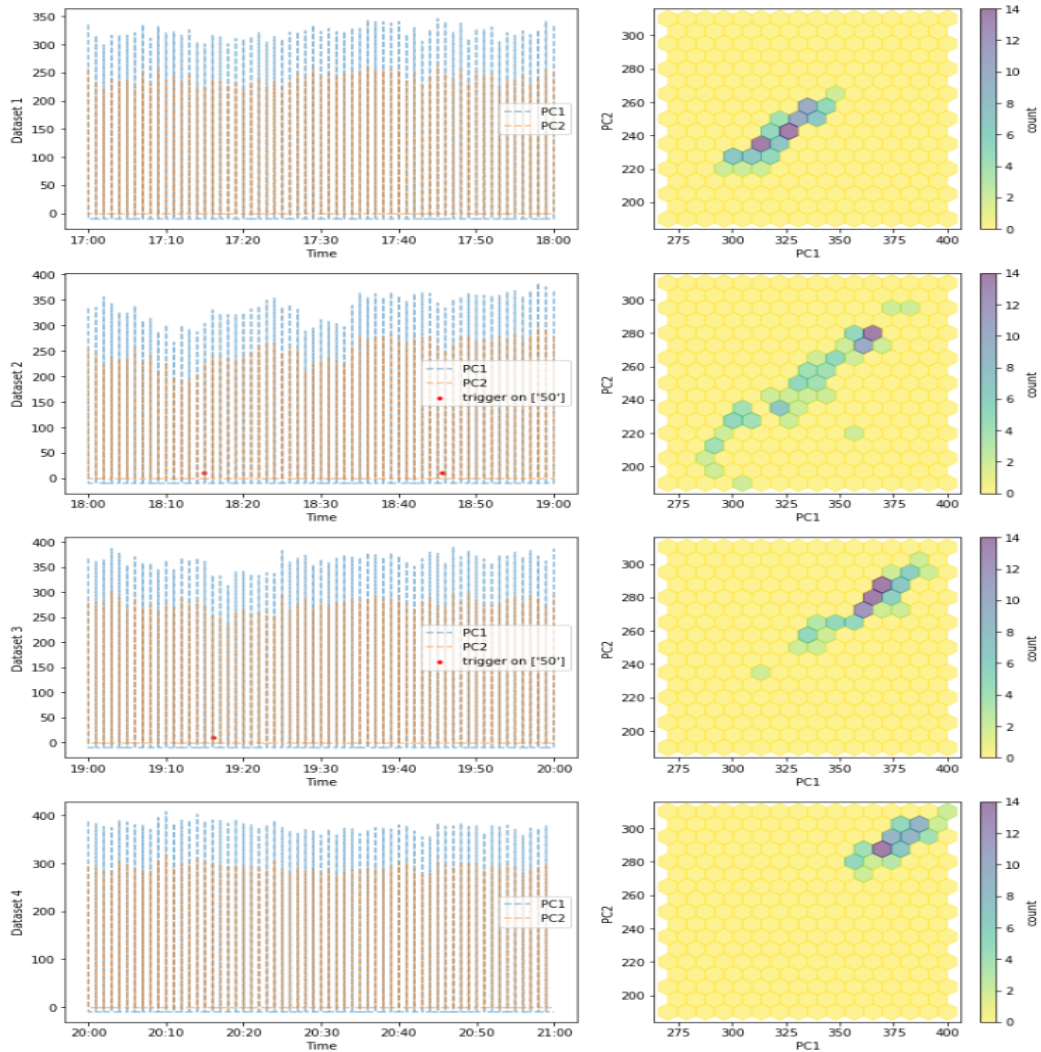


Figure 4.9: Transformed PC1 and PC2 for channels 48-55 on MEA-2. Each row shows transformed components for one hour, on the left is the transformed signal PC1 and PC2 against time, on right is the 2D histogram plot for PC1 vs PC2. Shifts in histogram are observed when trigger is sent on channel 50.

4.4 PCA in learning algorithm

4.4.1 Concepts of probabilities

A histogram plot is a statistical graph to represent continuous dataset³⁷. It converts the dataset into categories known as ‘bins’ and plot the number of data points that occur in a category with a bar on the chart, where height of a bar represents the frequency of data point within that category. Histogram plots are used to get probability density function of the underlying variable in dataset. For each bin in a histogram, the probability of that category is the number of data points in the bin divided by the total number of data points in the histogram. In order to get through probability distributions for each compared dataset, a 1-dimensional histogram is plotted for PC1. Here PC1 is chosen as standard for calculating probability distributions since it has the maximum **explained variance ratio** 4.3.3. The PC1 are plotted together on a histogram for each duration in dataset and the probability distributions from histogram plots are calculated.

Since each neurosphere is unique, the components in PCA are expected to vary a lot when an organoid is changed. The input stimulation data can be controlled, but it’s response in neural activity is quite unpredictable and varies as much. Therefore in cases like these where output model and environment are quite varying, statistical distances are extremely useful measures to quantify the gap between two distributions. There are several statistical distances which are commonly used as part of ML algorithms. The Jensen-Shannon distance from SciPy³⁸ is used to calculate a measure of the distance between the probability distributions provided by histograms.

4.4.2 Comparisons between datasets

The Jensen-Shannon divergence is a method of measuring similarity between two probability distributions³⁹. It is the symmetrized and smoothed version of Kullback-Leibler divergence. The square root of Jensen-Shannon divergence is a metric called Jensen-Shannon distance. The similarity between the distributions is higher when the distance is closer to zero. The Jensen-Shannon distance (**JSD**) between two probability distribution p and q is given by (4.2) where m is the pointwise mean of p and q , and D is the Kullback-Leibler divergence between the two distributions $D_{KL}(p||q) = \sum p_i \cdot \log \frac{p_i}{q_i}$.

$$JSD(p||q) = \sqrt{\frac{D(p||m) + D(q||m)}{2}} \quad (4.2)$$

By using histograms, the transformed PC1 components on a dataset of *unstimulated* activity is taken as a *reference distribution*. It is then compared with other probability distributions using the Jensen-Shannon distance. It is observed that the JSD between 2 unstimulated periods are comparable, however JSD between an unstimulated period and a stimulated period can vary. Fig. 4.10 shows the probability distributions obtained from histograms and a general comparison of Jensen-Shannon distance calculated between them, observed across different neurospheres in MEA-2. The yellow encircled point is the slot which is used as reference dataset, hence value of distance is zero from itself. For example, in Fig. 4.10b the distance from selected unstimulated slot index **3** to stimulated slots 0 and 1 are comparable, and the distance to other unstimulated slots 2 and 4 are comparable.

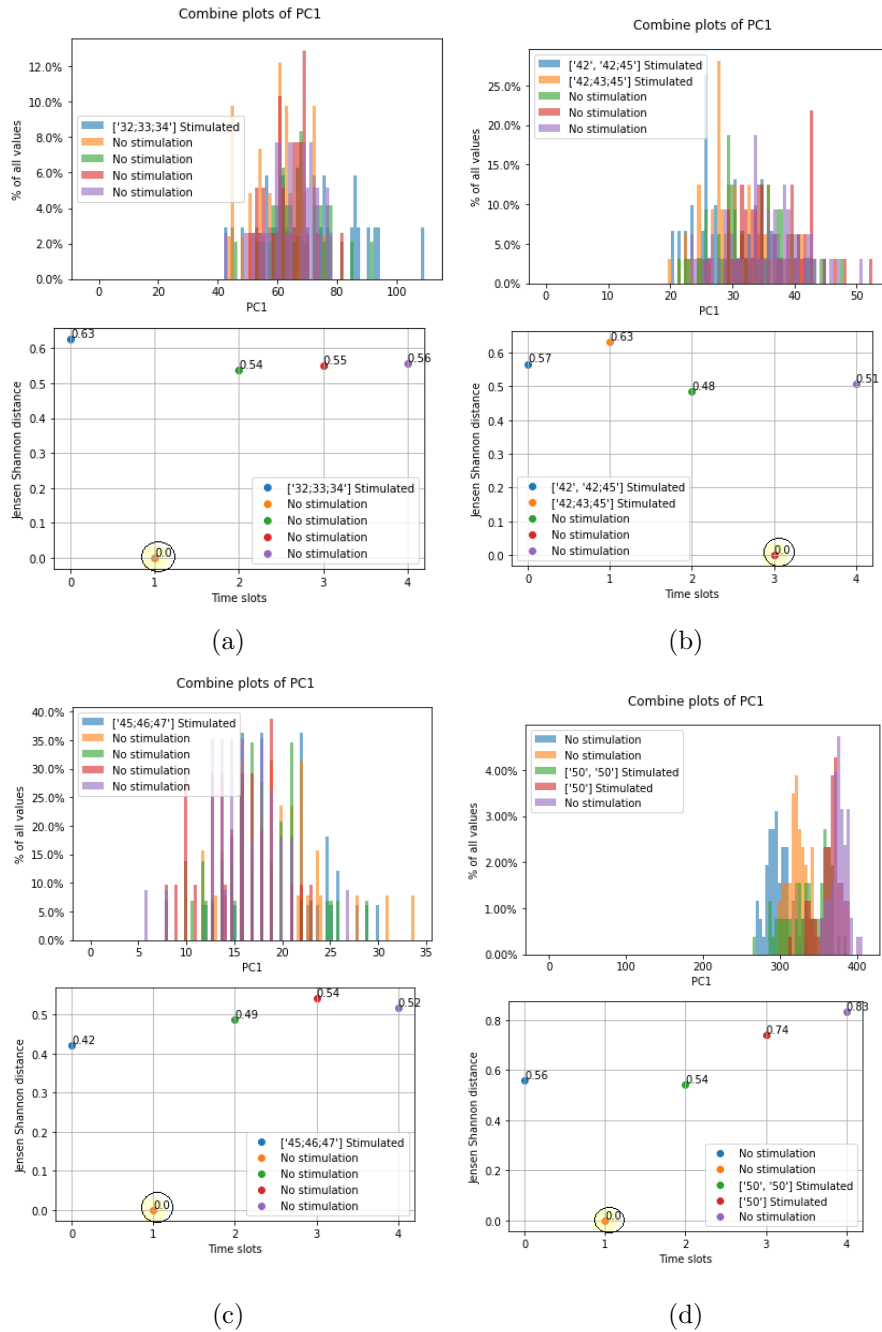


Figure 4.10: Identification of target distances in different neurospheres. Histogram plots transformed PC1 components and are used for acquiring the probability distributions. The distance between probabilities distributions of unstimulated and stimulated neural activity is plotted in lower graphs. Additionally, the distance between an unstimulated set and a stimulated set is higher as compared to two unstimulated sets.

4.4.3 Identification of a target

In Fig. 4.10, the selected distribution for comparison known as ‘reference distribution’ is taken from an unstimulated set and a distance to a stimulated set probability distribution is calculated. For any given neurosphere, this *distance between unstimulated set and stimulated set* can be selected as **the target distance**, d_{target} , for the learning algorithm, thus fulfilling an important objective 3 of the data analysis section in this work. The deviation of a stimulation cycle from this target value, can imply whether a stimulation cycle is favourable or not. If deviation is high, it is unfavourable and if the deviation is low, then the stimulation is favourable and is preferred in upcoming cycles. Fig. 4.11 summarizes the steps in obtaining target distances for each neurosphere.

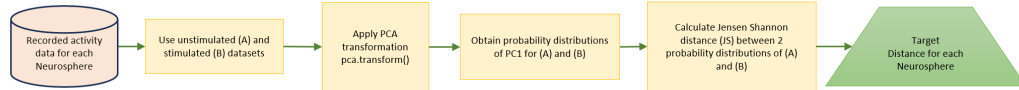


Figure 4.11: Identification of a target distance for each neurosphere.

4.5 Closed-loop feedback algorithm

In order to train the FOs towards a *targetted* goal, a *closed-feedback loop* is implemented with the help of a *controller*. A controller is designed to execute the entire learning process in cycles. Each cycle stimulates a neurosphere and processes the neural activity during the timestep Δt . The controller takes care of setting up the ‘reference distribution’ for comparison, choosing electrodes for stimulation, comparing current cycle transformed data and reference distribution, and providing feedback to the FO for the next cycle. This section explains different functions of a controller and different steps in the learning algorithm. Fig.4.12 pictorially depicts the closed-loop-feedback algorithm utilizing all the methods discussed in the implementation section.

4.5.1 Choose a reference distribution for comparison

As the first step in the learning process, the controller creates a ‘reference distribution’ for statistical distance comparison by processing an unstimulated set of neural activity using principle components as described in 4.4.1. The data for a time-period Δt can be fetched from historical data of the FO, or from the current *unstimulated* activity at the start ($t=0$) of a learning programme. An *unstimulated* activity data is read because a comparison between unstimulated and stimulated activity was made while creating target for the FO. Thus, in

subsequent learning cycles, the distance between this 'reference' unstimulated probability distribution and the current stimulated response probability distribution is compared with target distance for sending feedback.

4.5.2 Stimulation protocol and choosing electrodes

At the start of a learning cycle, the controller selects electrode(s) from the neurosphere for stimulation. Inclusion of any kind of bias in the training procedure should be avoided, so that each electrode gets a fair chance to be selected for stimulation. Therefore, it fits well to start by assigning a *probability*(\mathbf{p}) to select electrodes for stimulation. The controller uses *cumulative probabilities*(\mathbf{cp}) to select one electrode e_k from a set of eight electrodes($e_1 - e_8$) with assigned probabilities ($p_1 - p_8$).

The cumulative probability (cp_i) of an electrode (e_i) is the probability of selecting the electrode plus the probability of selecting all the electrodes before it, i.e. $cp_i = \sum_{j=0}^i p_j$. The controller rolls a dice to pick a random number $r \in [0, 1]$. The random number r is compared with each electrode's cumulative probability and the first electrode cp_i is selected where $cp_{i-1} \leq r \leq cp_i$. An example of cumulative probabilities at $t=0$ is shown in table 4.3. At $t = 0$ all eight electrodes are assigned equal probabilities $p_i = 1/8 = 0.125$, the cp_i are calculated by adding probabilities of all previous electrodes to e_i . For example, if the random number is picked up as $r = 0.58$, then electrode e_5 is chosen as $0.50 \leq 0.58 \leq 0.625$.

Assigning probabilities to electrodes, makes the provision of feedback in the learning cycle easier. At the completion of each cycle, with positive and negative feedback, the selection probabilities can be adjusted for the next cycle as explained in upcoming section 4.5.4.

Electrode e_i	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
$p_i(t=0)$	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
$cp_i(t=0)$	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.0

Table 4.3: Probability p_i and cumulative probability cp_i of selection of each electrode e_i , at time $t=0$. If a random number $r \in [0, 1]$ is picked as 0.58, then electrode e_5 is chosen as $0.50 \leq 0.58 \leq 0.625$.

4.5.3 Stimulation and distance calculation

The learning algorithm is executed in iterations with each iteration executing for time period Δt . To avoid bias, all eight electrodes are set with same probability initially for being chosen. After choosing a *reference distribution*, the controller randomly chooses an electrode e_k for stimulation. The trigger for stimulation is sent to the neurosphere for a controlled stimulation

on electrode e_k . The controller waits for a span of Δt before reading the neural activity from the corresponding stimulation period. This then becomes the current activity and probability distribution of PC1 is calculated using PCA transformations explained in 4.4.1. Then $d_{current}$ is calculated via Jensen-Shannon distance 4.4.2 between stimulated current and reference probability distributions by using eq.4.2.

4.5.4 Comparison with target and feedback

As defined in 4.4.3, d_{target} is the target distance between stimulated and unstimulated set. The aim of a stimulation cycle is for $d_{current}$ to be as close as possible to d_{target} . The closer these distances are, the more likely is the neurosphere to give an expected stimulated outcome. The deviation Δd from target is calculated as $\Delta d = |d_{target} - d_{current}|$. The smaller is the deviation Δd , the more favourable the stimulation is in achieving the target. If Δd is less*, the stimulation is favourable and controller sends a *positive* feedback. If Δd is more*, the stimulation is unfavourable and controller sends a *negative* feedback. The quantification of less* and more* is a permissible range Δd_{perm} for Δd calculated based on an average value of deviations from multiple learning cycles.

The feedback to the neurosphere is provided by adjusting the probability of stimulated electrode e_k for the next cycle. In case of positive feedback, the probability of selecting e_k increases while probabilities of selecting other electrodes for the next cycle is decreases. For negative feedback, the probability of selecting e_k decreases while the probabilities of selecting other electrodes is increases. The increment and decrement in probabilities is handled by using equations 4.3 and 4.4.

$$Positive\ feedback\ on\ e_k = \begin{cases} p_k(t+1) = \frac{1.5p_k(t)}{1 + 0.5p_k(t)}, & k = selected\ electrode \\ p_i(t+1) = \frac{0.5p_i(t)}{1 + 0.5p_i(t)}, & i \neq k \end{cases} \quad (4.3)$$

$$Negative\ feedback\ on\ e_k = \begin{cases} p_k(t+1) = \frac{0.5p_k(t)}{1 - 0.5p_k(t)}, & k = selected\ electrode \\ p_i(t+1) = \frac{1.5p_i(t)}{1 - 0.5p_i(t)}, & i \neq k \end{cases} \quad (4.4)$$

4.5.5 The learning algorithm combined

As shown in Fig.4.12, the closed-loop learning algorithm can be summarized as below:

1. Read unstimulated data for time period Δt , transforms it using PCA, and saves the probability distribution of PC1 as *reference distribution*.
2. Select e_k electrode when each has been assigned equal probabilities initially, and performs a controlled stimulation on e_k .
3. Read the stimulated neural activity response, transforms the dataset using PCA and converts to PC1 probability distribution after Δt time.
4. Calculate Jensen-Shannon distance between unstimulated reference distribution and stimulated distribution as $d_{current}$.
5. Calculate $\Delta d = |d_{target} - d_{current}|$. If Δd lies within permissible values Δd_{perm} , positive feedback is sent, otherwise negative feedback.
6. Sends positive/negative feedback on e_k is sent by increasing/decreasing it's probability for next cycle while subsequently decreasing/increasing probabilities of other electrodes using Eq. 4.3.
7. Repeat the cycle from step 2 with a new e_k till convergence of $d_{current}$ in d_{target} .

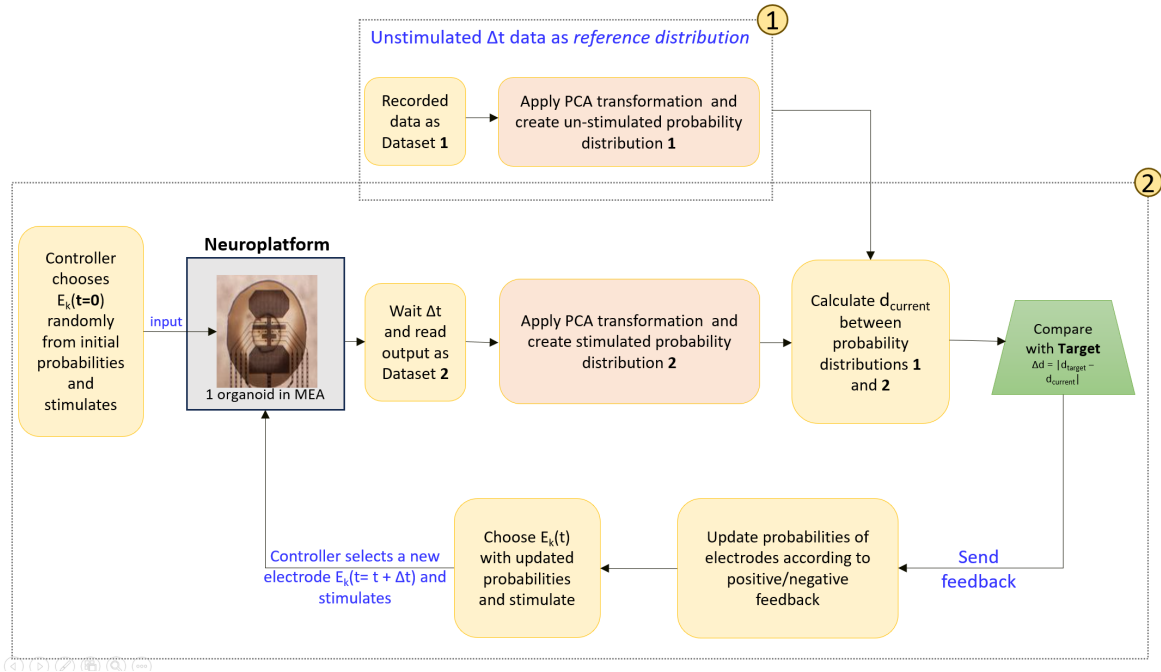


Figure 4.12: Implementation of a closed-loop feedback algorithm for reinforcement learning. (1) Retrieval of unstimulated dataset and conversion using PCA into a probability distribution of PC1 as ‘reference distribution’. (2) The closed-loop algorithm starts with the controller selecting 1 of 8 electrodes with equal probabilities. The algorithm reads the stimulated data for Δt after trigger of electrode and converts it to probability distribution of PC1. The Jensen-Shannon distance between stimulated and unstimulated probability distributions is compared with d_{target} to provide positive or negative feedback. The feedback is given by updating the probability of selection of each electrode for the next cycle. This feedback-loop runs till convergence in distance is obtained.

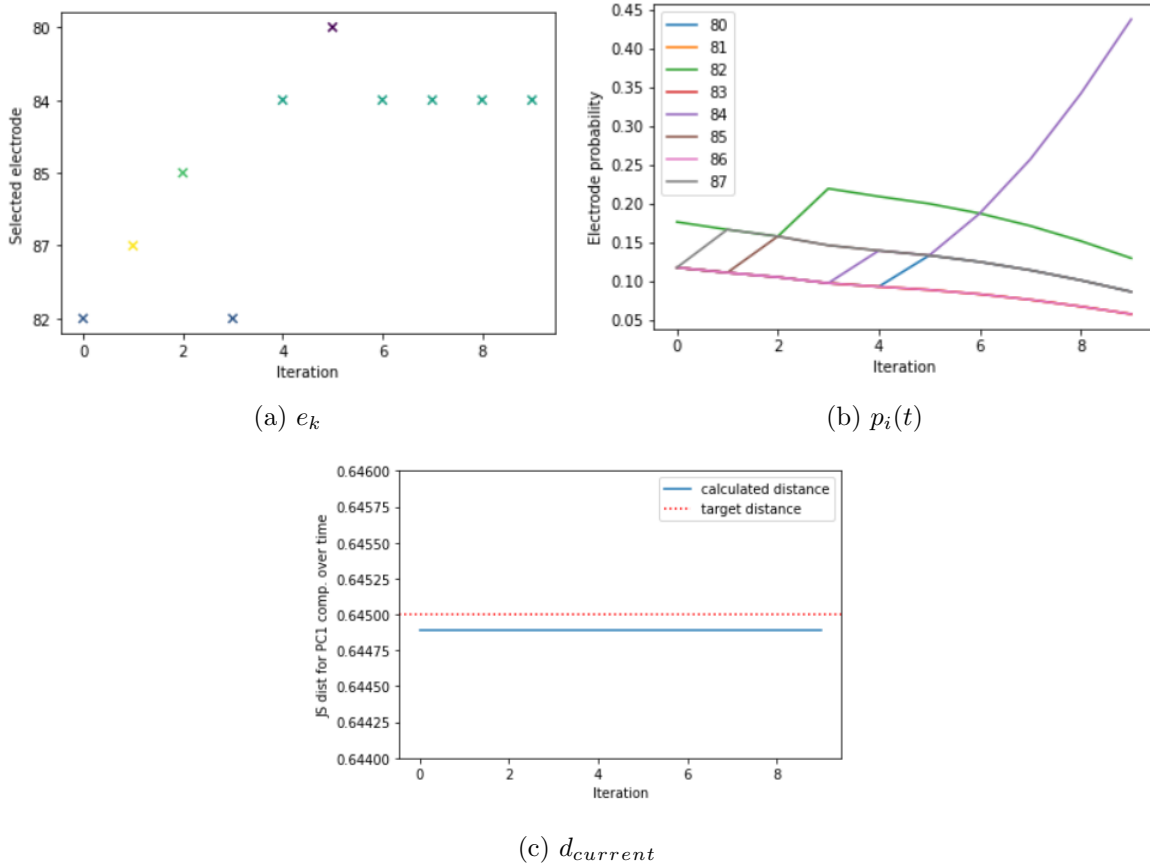
TRAINING RESULTS AND DISCUSSION

The results of each computational step used in creation of reinforcement learning closed-loop algorithm has been shown in Chapter 4; each successive step was based on the observation of the previous one. The proposed algorithm was then run multiple times on different neurospheres of an MEA on neuroplatform. The goal was to achieve convergence between d_{target} and $d_{current}$ with a permissible margin Δd_{perm} for deviation Δd .

This chapter showcases results from a few iterations to highlight the main outcomes of the training algorithm shown in 4.5.5. Fig.5.1-5.4 in next pages show selected electrode e_k , probabilities of channels over different iterations $p_i(t)$, and Jensen-Shannon distances between activity data after each stimulation and initial unstimulated activity data across different runs of the learning algorithm.

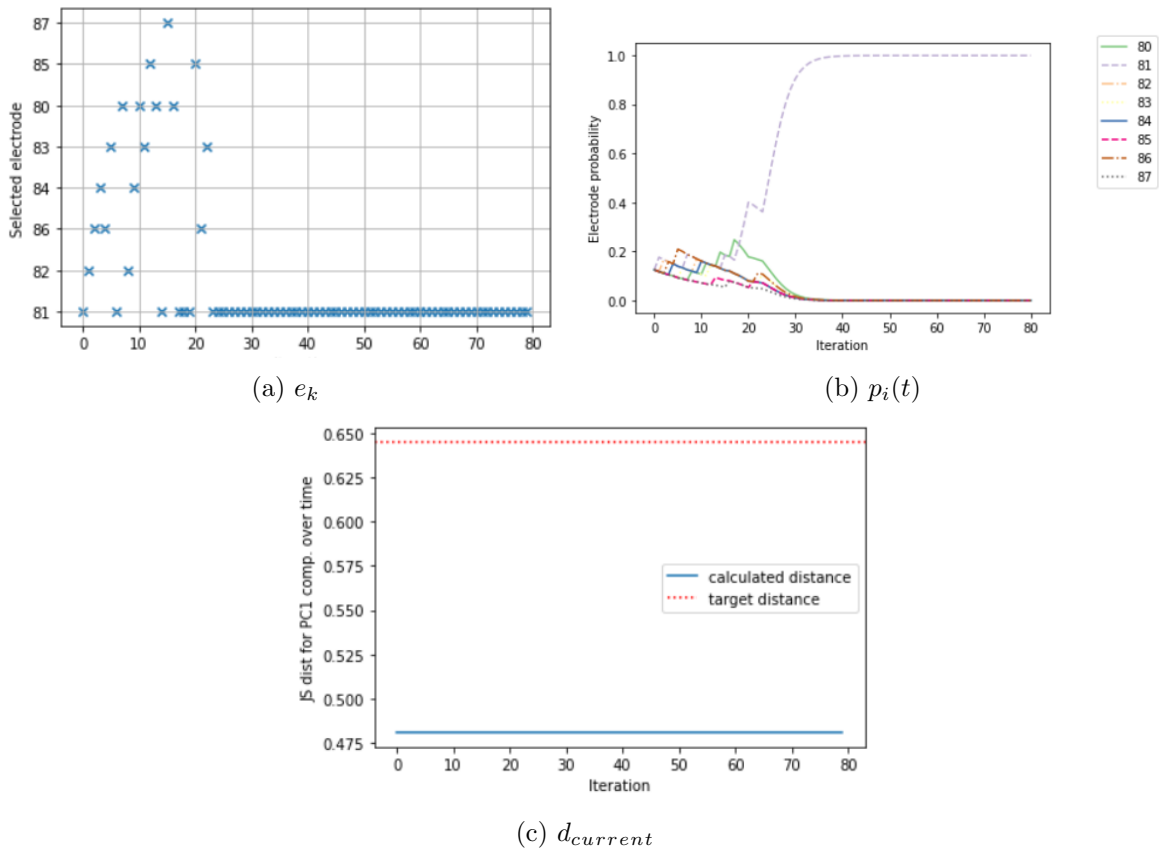
Result 1 (Fig.5.1)

As one of initial runs, 10 iterations of closed-loop algorithm are performed on channels 80-87 of MEA-3 with $d_{target} = 0.645$ and $\Delta d_{perm} = \pm 0.005$. In each iteration, selected electrode e_k was stimulated with a current of 2uA and the time for fetching data after stimulation Δt was kept as two minutes. The deviation in Jensen-Shannon distance between PC1 transformed component $d_{current}$ and d_{target} was smaller than Δd_{perm} , which resulted in positive feedback and increase in the probability p_k of selected electrode, while other p_i decreased. At the end of 10 runs, the channel 84 became the most probable. However, since $d_{current}$ remained constant, 10 iterations might not be enough to determine whether this behaviour was consistent and is capable of converging to d_{target} .

Figure 5.1: **Result 1**, 80-87 in MEA-3

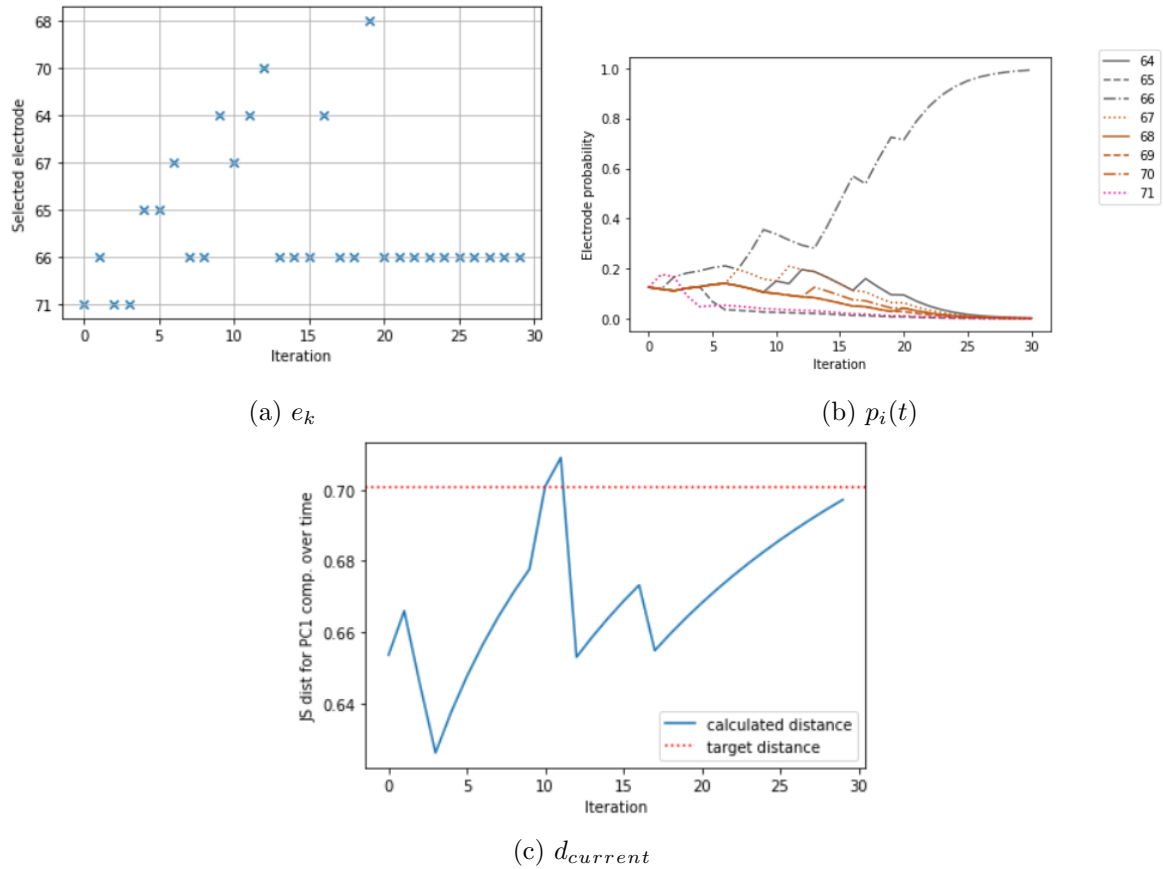
Result 2 (Fig.5.2)

The number of iterations here was increased to 80, and the time duration Δt was increased to three minutes. In this case, electrode 81 was the most selected electrode e_k after about 25 iterations. However, $d_{current}$, the calculated distance from the stimulated data to the unstimulated ‘reference distribution’ still remained constant in each iteration suggesting that, the stimulations had minimal impact on altering the behaviour of the neurosphere.

Figure 5.2: **Result 2**, 80-87 in MEA-3

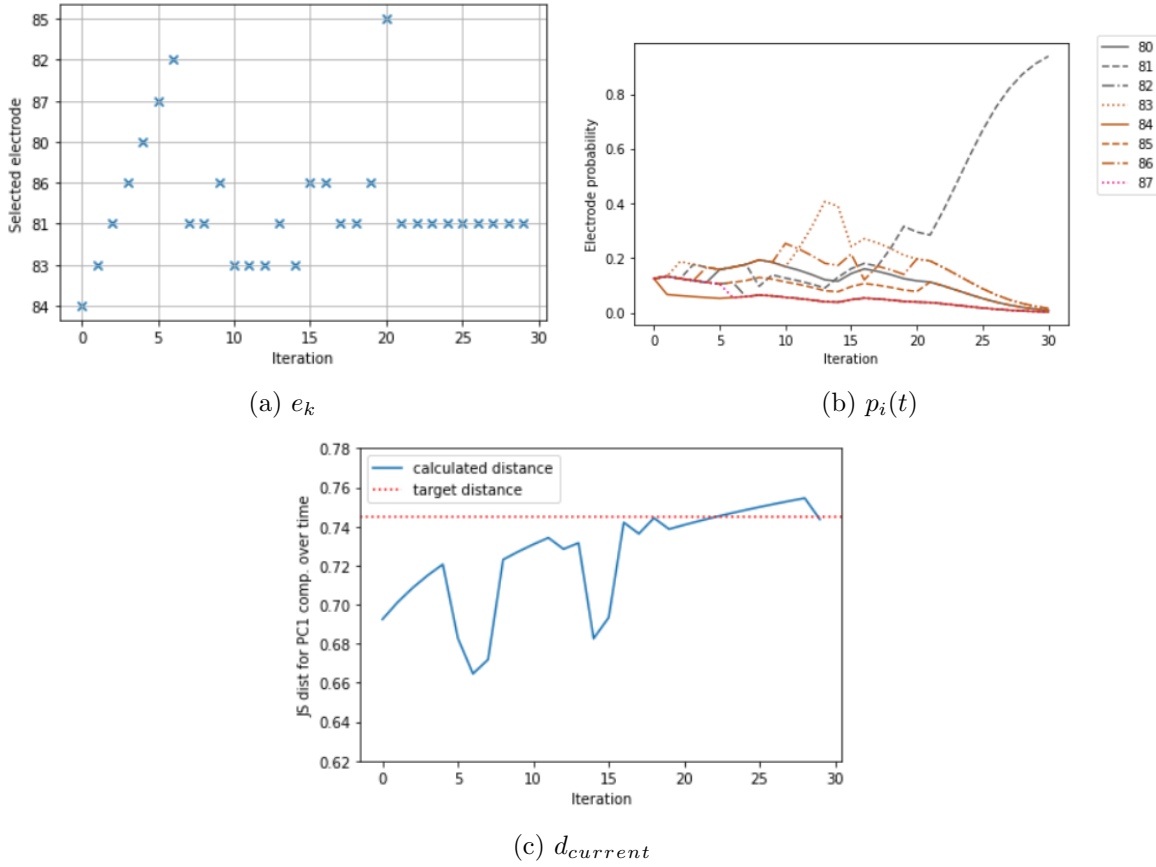
Result 3 (Fig.5.3)

The Δt was set as 15 minutes to increase the lag time between stimulations. A total of 30 iterations were performed on channels 64-71 in MEA-3 with d_{target} as 0.7006 based on historical data of these channels. After 20 iterations, the probability of electrode 66 increased steadily accompanied with a steady reduction in the deviation from d_{target} .

Figure 5.3: **Result 3**, 64-71 in MEA-3

Result 4 (Fig.5.4)

The same approach was repeated on channels 80-87 in MEA-3 with 30 iterations. The d_{target} was set to 0.7446 based on historical channel data. Similar to the previous neurosphere, after about 20 iterations, the probability of electrode 81 increased, and the deviation from d_{target} steadily decreased in the subsequent cycles.

Figure 5.4: **Result 4**, 80-87 in MEA-3

Different combinations of two electrodes rather than one single electrode for were also used for controlled stimulations in the learning algorithm. $\binom{8}{2}$ combinations of electrodes were configured with the same initial probability and stimulated with equal current. The observed behaviour was similar to that of a single electrode, as the likelihood of choosing a pair increased steadily after a certain number of iterations. However, convergence in d_{target} was not achieved when Δt was set to two or three minutes.

General observations

The general behaviour observed in the above four results have been summarized in table 5.1 and explained in detail below.

- The closed-loop feedback learning algorithm responds well to stimulations, and is able to update probabilities of electrodes according to deviation of distances $d_{current}$ between stimulated and unstimulated responses. In cases where $d_{current}$ stays the same, the neurosphere tries to keep its current behaviour intact.
- An iteration for shorter duration is unable to influence the $d_{current}$. Deviations in distances are more prominent with larger Δt values like *15 minutes*. The proper Δt for iteration needs to be set so that the learning algorithm can be computationally faster. However, since the neural channel activities are not highly-correlated when compared in smaller units of time (as observed in 4.3.2), it is advised to use a larger Δt such as 15 minutes for learning of the neurosphere.
- Moreover, at least 30 iterations of algorithm are required to develop reasonable learning, as e_k and $d_{current}$ are observed to move towards convergence only after 20 or more iterations.
- When performed for more than one electrode, the controlled stimulation for a random pair of electrodes showed similar behaviour to that observed when single electrode was selected.

Sn.	channels	iterations	d_{target}	Δd_{perm}	Δt (minutes)	Convergence	
						$d_{current}$	electrode
1	80-87	10	0.645	± 0.005	2	No	Yes
2	80-87	80	0.645	± 0.005	3	No	Yes
3	64-71	30	0.7006	± 0.005	15	Yes	Yes
4	80-87	30	0.7446	± 0.005	15	Yes	Yes

Table 5.1: Summary of the 4 results

CONCLUSION

The primary goal of this work is to lay grounds for a reinforcement learning algorithm using which a neurosphere can be trained to furnish a desired target output. The results of several iterations converging towards target indicated the organoid’s strong responsiveness to the feedback provided, and that it had learned to mimic a target response pattern.

Initial stimulation experiments resulted in displacements of Centre of Activity. While the CoA helped to observe changes in neurosphere over short periods of time, Pearson’s time correlation ruled out any immediate substantial correlation with channels. For neural activity data taken over longer periods of time, Principle Component Analysis was more successful to identify pattern shifts between the two identified principle components. Indeed, PCA transformations done with Jensen-Shannon distance fulfilled the objective of setting a quantified target for the algorithm.

When put to training, the neurosphere showed signs of learning and adapting according to the feedback received on it’s activity. The selection probabilities of favourable channels incremented during the learning. At the end of all iterations, the neurosphere showed reproducible behaviour since the similarity achieved between the target and the output value was substantial. These findings support the potential of an organoid to be a learnable computational element, thus getting a step closer to the idea of training a bio-computing on *in vitro* networks.

Challenges

The biggest challenge so far is the life span of a neurosphere. Even though the forebrain organoids are expected to last around 100 days⁵, a constant rework and training of neurospheres was inevitable during the course of the project with some organoids being replaced as often as a fortnight. Each new “bio-processor” needs new training starting from data recording and PCA to calculation of d_{target} and estimation of Δt .

Moreover, since the learning algorithm required a large Δt which results in longer experiments, one must plan activities in advance to prevent potential delays due to limited experiment slot availability on the neuroplatform for stimulations.

Lastly, for organoids with intense ‘channels’, where the maximum activity is concentrated around one electrode, it is very difficult to dampen the neural response even with SBS stimulations. This leads to an ineffective controlled stimulation in learning iterations, disabling the algorithm to bring a substantial change in the neurosphere’s behaviour. Similarly, organoids with very less activity have an adverse impact on the data analysis step, due to higher instances of zeroes in the data. In some cases, it has been observed that the stimulations on

electrodes decrease the response activity of neurosphere instead of increasing them.

Future work

Some ideas which can assist to take the research forward are:

- Computing the relationship between two highest principle components with the two-dimensional Centre of Activity to identify the flow of activity in neural network.
- In neurospheres there are regions pertaining to “intense” electrodes which are highly active. Currently, the learning algorithm starts by assigning equal selection probability to each channel in first iteration. Adjusting these initial probabilities to promote or demote highly active electrodes based on the activity in each channel, could lead to a different trend for selected electrodes $e_k(s)$ and convergence of d_{target} .
- Defining a learning rate from existing variables to quantify the effect algorithm has on the neurosphere, since many runs of the learning cycle are needed to fine-tune the algorithm.
- It will be beneficial to compare the effect of reinforcement learning on a freshly prepared FO with one approaching end of life, since the spontaneous activity of the organoid can vary overtime⁴².
- Including Time Independent Component Analysis (**TICA**)⁴⁰ might help checking lag times at which the transformed components (PC1 in this case) have the maximum auto-correlation, This is because neural responses change instantly after a stimulation, but effects of stimulation lingers much longer through the BNN. The results of iterations show the learning cycles with larger Δt being more convergent than those with smaller values. TICA as a dynamical version of PCA, can help to find the Δt , that is the most suited for learning cycles.

APPENDIX

A.1 Software Development and Integration

The interaction with neuroplatform is made available through a Datalore server, on which a project is setup for accessing neurospheres with the help of Jupyter notebooks⁹. A python library *computation.py* is provided which exposes APIs for running experiments and fetching the neural activity data. This is covered in details in section 2.2.

One of the goals of this work is to produce a computational package which can form a basis for a learning algorithm. A computational package has been created to consume the APIs of neuroplatform and provide methods for easy experimental runs, setting experimental parameters, recording activity data from neurospheres, and perform analysis of data. This chapter gives an overview of the computation class created for FU Berlin research group.

A.2 Technology Stack

computation.py is a python script which contains the code for different implementation steps in this work. *computation.py* is uploaded on Datalore environment under neuroplatform Workspace files. All edits and versions to the file are maintained under FU Berlin GitLab repository⁴¹ and users can request access to it. There is however no continuous integration (CI) available presently between GitLab and Datalore. The code in *computation.py* is modular and easily extensible to support additional functionalities. The package is divided into 3 main modules(classes) to support the implementation of the work.

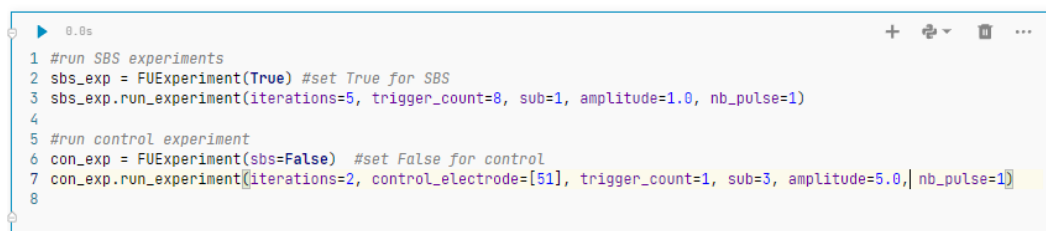
A.3 Modules

FUBase

FUBase is the base class which defines primary properties of the neuroplatform. The accesstoken shared by the FinalSpark team is unique for each MEA, and provides readwrite access to all 32 electrodes in the MEA. Accesstoken is maintained as property of **FUBase** class. Once **FUBase** class is initialized, the constructor sets up electrodes positions on a 2D lattice for each organoid, these positions are used in the calculation of Centre of Activity and for visualisation of activity. **FUBase** also provides methods for reading files that are saved in Datalore environment.

FUExperiment

This class inherits from `FUBase` class and provides methods to setup and run experiments. Both the protocols SBS and Control stimulations 4.2 can be set and experiments can be run using this class. The class takes care of setting up type(SBS/Control), triggers, electrodes, amplitude(uA) and number of pulses per signal for an experiment. `FUExperiment` performs stimulations by calling the Intan software api of neuroplatform.py interface. The experiment details are also saved on Datalore in the path `/data/workspace_files/experiments` as csv files for future references. The configuration details for stimulations performed by learning algorithm in each iteration is read from `/data/workspace_files/exp_setup`.



```

0.0s
1 #run SBS experiments
2 sbs_exp = FUExperiment(True) #set True for SBS
3 sbs_exp.run_experiment(iterations=5, trigger_count=8, sub=1, amplitude=1.0, nb_pulse=1)
4
5 #run control experiment
6 con_exp = FUExperiment(sbs=False) #set False for control
7 con_exp.run_experiment(iterations=2, control_electrode=[51], trigger_count=1, sub=3, amplitude=5.0, nb_pulse=1)
8

```

Figure A.1: Stimulation example

FUComputation

Inherited from `FUBase`, this class is responsible for all computation methods being performed throughout this work. Important functionalities include getting experiment summary for a given day, getting channel(s) activity for given time period, calculating centre of activity of a neurosphere for a given time period, saving large response dataset for training. This class implements the PCA Analysis and calculation of target distances. The Intan Software continuously records the electrical activity from all electrodes and `FUComputation` provides a method to visualize current activity in an MEA as shown in Fig. A.2. This performs a sanity check on neural response in organoids in MEA, and helps to visualize outcomes of an ongoing experiment in real time.

`FUComputation` class contains methods for saving neural activity data for large periods of time. The recorded data is saved date-wise as csv files under the path `/data/workspace_files/data`, and is used to initialize and fit a Principle Component Analysis `sklearn.decomposition` object. A separate object for each sub in an MEA is saved globally under the path `/data/workspace_files/pca-trained` labelled with channel numbers. The same trained PCA object is then used to apply transformation during the learning iteration and during identification of targets.

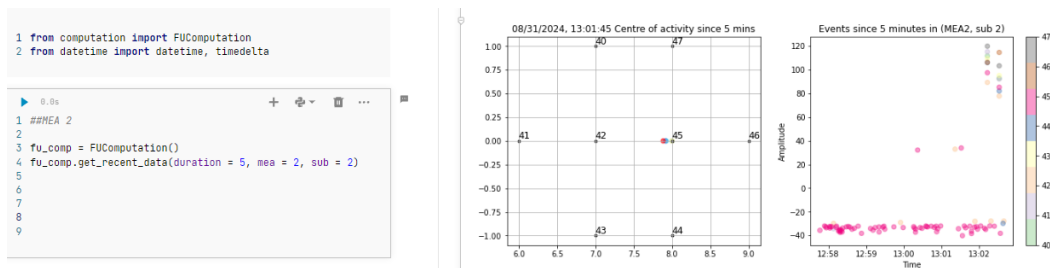


Figure A.2: Get live data example

FUController

`FUController` class is primarily associated with implementation of closed-loop feedback algorithm. It selects electrode(s) for stimulation, sends trigger to neurosphere, and transforms response activity using PCA. It compares result with target and sends feedback to neurosphere by adjusting the probabilities of electrode selection for the next iteration.

Notebook	Description
SBS & Control Experiment	Runs stimulations on neurospheres by setting SBS and control configurations
View live data	Gets live activity from neurospheres by plotting centre of activities
Activity Data Recording	Records neural activity for hours and saves to Datalore
Time-shift correlations	Correlates channel activity data using Pearson's correlation coefficient
Trained PCA object	Trains PCA objects and save to Datalore location
Channel Observations - create Target	Calculates d_{target} for each neurosphere by using PCA
Closed Feedback Loop	Executes learning algorithm on neurospheres and process the results.

Table A.1: List of Jupyter notebooks on Datalore

MEA	Sub with electrode numbers			
	1	2	3	4
1	0-7	8-15	16-23	24-31
2	32-39	40-47	48-55	56-63
3	64-71	72-79	80-87	88-95
4	96-103	104-111	112-119	120-127

Table A.2: Numbered electrode in each sub/neurosphere of an MEA

ABBREVIATIONS

AI	Artificial Intelligence. 1, 8,
ANN	Artificial Neural Network. 1, 9,
API	Application Programming Interface. 47,
ASI	Artificial Super-intelligence. 3,
BNN	Biological Neural Network. 1, 2, 4, 8–10, 12, 46,
CCM	Correlation Coefficient Matrix.
CoA	Centre of Activity. 5, 16, 21, 26, 45–47,
ERASI	Energy Requirement of Artificial SuperIntelligence equation. 3,
FO	Forebrain Organoid. 5–7, 13, 15, 21, 34, 46,
GPU	Graphics Processing Unit. 3,
JSD	Jensen-Shannon distance. iii, 31, 32, 39,
MEA	Multi-Electrode Arrays. 2, 4–6, 12, 28,
ML	Machine Learning. 1, 2, 8, 9,
OI	Organoid Intelligence. 8,
PC	Principle Component. 11, 12, 26,
PC1	First Principle Component. 26, 36,
PC2	Second Principle Component. 26,
PCA	Principle Component Analysis. iii, 13, 24, 25, 28, 36, 45,
RL	Reinforcement Learning. vi, 9, 10,
TICA	Time Independent Component Analysis. 46,

GLOSSARY

channel	electrodes attached to the Neurospheres..
exaflop	10^{18} floating point operations(flop).
forebrain organoid	minibrain/neurosphere. 45
graphics processing unit	a specialized electronic circuit initially designed for digital image processing and to accelerate computer graphics.
multi-variate dataset	Dataset which is dependent on more than one variable.
neuroplatform	computation platform offered to interact with Neurospheres. 2, 3, 5, 6, 14, 21, 25, 39, 45, 47
neurosphere	in vitro organoid mini-brain created using biological cells/ forebrain organoids. 2, 21, 24, 32–35, 39, 41, 45–47
reinforcement learning	Learning through trial and error with feedback. 2
spike	A ‘threshold crossing’ event recorded when voltage at an electrode crossed a threshold..

REFERENCES

1. Wang, H. *et al.* Scientific discovery in the age of artificial intelligence. *Nature* **620**, 47–60 (2023).
2. Balasubramanian, V. Brain power. *Proceedings of the National Academy of Sciences of the United States of America* **118**. ISSN: 1091-6490 (32 Aug. 2021).
3. Wang, H., Li, X., You, X. & Zhao, G. Harnessing the power of artificial intelligence for human living organoid research. *Bioactive Materials* **42**, 140–164. ISSN: 2452-199X. <https://www.sciencedirect.com/science/article/pii/S2452199X24003657> (2024).
4. Bakkum, D. J., Chao, Z. C. & Potter, S. M. Spatio-temporal electrical stimuli shape behavior of an embodied cortical network in a goal-directed learning task. *Journal of Neural Engineering* **5**, 310. <https://dx.doi.org/10.1088/1741-2560/5/3/004> (Aug. 2008).
5. Jordan, F. D., Kutter, M., Comby, J.-M., Brozzi, F. & Kurtys, E. Open and remotely accessible Neuroplatform for research in wetware computing. *Frontiers in Artificial Intelligence*. <https://doi.org/10.3389/frai.2024.1376042> (2024).
6. Ciarpella, F. *et al.* Generation of mouse hippocampal brain organoids from primary embryonic neural stem cells. *STAR Protocols* **4**, 102413. ISSN: 2666-1667. <https://www.sciencedirect.com/science/article/pii/S2666166723003805> (2023).
7. Wang, H. Modeling Neurological Diseases With Human Brain Organoids. *Frontiers in Synaptic Neuroscience* **10**. ISSN: 1663-3563. <https://www.frontiersin.org/journals/synaptic-neuroscience/articles/10.3389/fnsyn.2018.00015> (2018).
8. Hartley, B. J. & Brennand, K. J. Neural organoids for disease phenotyping, drug screening and developmental biology studies. *Neurochemistry international* **106**, 85–93 (2017).
9. Datalore. *Datalore Documentation* <https://www.jetbrains.com/help/datalore/datalore-quickstart.html>.
10. Jordan, F. D., Kutter, M., Comby, J.-M., Brozzi, F. & Kurtys, E. *Finalspark live* <https://finalspark.com/live>.
11. Jordan, F. D., Kutter, M., Comby, J.-M., Brozzi, F. & Kurtys, E. *FinalSpark* <https://finalspark.com/>.
12. Feng, E. Brain on a Chip: Exploring the Potential of Wetware Computing. *researchgate.net* (2009).
13. Habibollahi, F., Khajehnejad, M., Gaurav, A. & Kagan, B. J. *Biological Neurons vs Deep Reinforcement Learning: Sample efficiency in a simulated game-world* in *NeurIPS 2022 Workshop on Learning Meaningful Representations of Life* (2022).
14. contributors, W. *Wetware computer* Online; accessed 24-October-2024. https://en.wikipedia.org/wiki/Wetware_computer.
15. Bray, D. *Wetware: a computer in every living cell* (Yale University Press, 2009).
16. contributors, W. *Frontier (supercomputer)* Online; accessed 3-September-2024. [https://en.wikipedia.org/w/index.php?title=Frontier_\(supercomputer\)&oldid=1242375879](https://en.wikipedia.org/w/index.php?title=Frontier_(supercomputer)&oldid=1242375879).

17. Clarke PhD, D. D. & Sokoloff, L. Circulation and energy metabolism in the brain/Donald D. Clarke and Louis Sokoloff. *Chemistry Faculty Publications*. 81 (1999).
18. Stiefel, K. M. & Coggan, J. S. The energy challenges of artificial superintelligence. *Frontiers in Artificial Intelligence* **6**, 1240653 (2023).
19. Kagan, B. J. *et al.* In vitro neurons learn and exhibit sentience when embodied in a simulated game-world. *Neuron* **110**, 3952–3969.e8. ISSN: 10974199 (23 Dec. 2022).
20. Demarse, T. B. & Dockendorf, K. P. Adaptive Flight Control With Living Neuronal Networks on Microelectrode Arrays. *IEEE* (2005).
21. OpenAI. *ChatGPT and OpenAI* <https://openai.com/index/chatgpt/>.
22. Mateos-Aparicio, P. & Rodríguez-Moreno, A. The Impact of Studying Brain Plasticity. *Frontiers in Cellular Neuroscience* **13**. ISSN: 1662-5102. <https://www.frontiersin.org/journals/cellular-neuroscience/articles/10.3389/fncel.2019.00066> (2019).
23. Meyer, J. & Wilson, S. W. Animat. *Scholarpedia* **4**. revision #90966, 1533 (2009).
24. Chao, Z. C., Bakkum, D. J. & Potter, S. M. Shaping Embodied Neural Networks for Adaptive Goal-directed Behavior. *PLOS Computational Biology* **4**, 1–17. <https://doi.org/10.1371/journal.pcbi.1000042> (Mar. 2008).
25. Smirnova, L. *et al.* Organoid intelligence (OI): the new frontier in biocomputing and intelligence-in-a-dish. *Frontiers in Science* **1** (Feb. 2023).
26. Shalev-Shwartz, S. & Ben-David, S. *Understanding machine learning: From theory to algorithms* (Cambridge university press, 2014).
27. Lecun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444. ISSN: 14764687 (7553 May 2015).
28. Sutton, R. S. & Barto, A. *Reinforcement learning : an introduction* ISBN: 978-0-262-03924-6 (The MIT Press, 2020).
29. Hildebrand, M., Andersen, R. & Bøgh, S. Deep Reinforcement Learning for Robot Batch-ing Optimization and Flow Control. *Procedia Manufacturing* **51**, 1462–1468 (Nov. 2020).
30. Greenacre, M. *et al.* Principal component analysis. *Nature Reviews Methods Primers* **2**. ISSN: 26628449 (1 Dec. 2022).
31. Jaadi, Z. *Principal Component Analysis (PCA): A Step-by-Step Explanation* <https://builtin.com/data-science/step-step-explanation-principal-component-analysis>.
32. Ringnér, M. What is principal component analysis? *Nature biotechnology* **26**, 303–304 (2008).
33. Kader, G. D. & Franklin, C. A. The Evolution of Pearson’s Correlation Coefficient. *The Mathematics Teacher* **102**, 292–299. ISSN: 00255769. <http://www.jstor.org/stable/20876349> (2024) (2008).
34. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* **12**, 2825–2830 (2011).

-
35. Garreta, R. & Moncecchi, G. *Learning scikit-learn: machine learning in python* (Packt Publishing Birmingham, 2013).
 36. Scikit-learn developers. *sklearn.decomposition - PCA* <https://scikit-learn.org/dev/modules/generated/sklearn.decomposition.PCA.html>.
 37. Jaspersoft. *What is a Histogram Chart* online, last accessed : 18 October 2024. <https://www.jaspersoft.com/articles/what-is-a-histogram-chart>.
 38. Virtanen, P. *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17**, 261–272 (2020).
 39. contributors, W. *Jensen–Shannon divergence* Online; accessed 10-September-2024. https://en.wikipedia.org/wiki/Jensen%E2%80%93Shannon_divergence.
 40. Schultze, S. & Grubmüller, H. Time-Lagged Independent Component Analysis of Random Walks and Protein Dynamics. *Journal of Chemical Theory and Computation* **17**. PMID: 34449229, 5766–5776. eprint: <https://doi.org/10.1021/acs.jctc.1c00273>. <https://doi.org/10.1021/acs.jctc.1c00273> (2021).
 41. Berlin, F. *FU Berlin Gitlab* Online; accessed 18-October-2024. <https://git.imp.fu-berlin.de/ishitaS/finalspark/neurospheres>.
 42. Wagenaar, D., Pine, J. & Potter, S. An extremely rich repertoire of bursting patterns during the development of cortical cultures. *BMC Neurosci*, **7**, 11. <https://doi.org/10.1186/1471-2202-7-11> (2006).