GSOC 2021 Ideas list for INCF

(210225 current number of projects: 58, mentor groups: 27)

List of free mentors (that can take on a student-suggested project): see bottom of the document

1. High-order neuroimaging projects
    - 1.1 A Python toolbox for computing high-order information in neuroimaging
    - 1.2 Visualizing high-order interactions in neuroimaging data and beyond
2. ImageJ Active Segmentation projects
    - 2.1 Deep Learning using Geometric Features
    - 2.2 High content image feature and classification database
    - 2.3. Input/Out Workflows for Active Segmentation Platform
3. DevoWorm projects
    - 3.1 Upgrading DevoLearn
    - 3.2. Deep Learning for Diatom Non-neuronal Cognition
    - 3.3. Digital Microsphere
4. Unit test for brains
    - 4.1. Unit tests for brains
5. Neurodata Without Borders projects
    - 5.1 Visualization of Neurodata Without Borders data standard specifications
    - 5.2 Interactive visualization of neurophysiology data in NWB Widgets
    - 5.3 Enhancing web interface and automated processes of the Neurodata Extensions Catalog
6. GeNN projects
    - 6.1 Improving the PyNN interface to GeNN
    - 6.2 ISPC backends for GeNN
    - 6.3 Brian2GeNN
    - 6.4 Open-source biorobotics framework
7. The Virtual Brain (TVB) projects
    - 7.1 Reusable visualization tools for Jupyter
    - 7.2 Benchmark and Optimize TVB areas
    - 7.3 Computational modelling of FNIRS and EEG with The Virtual Brain and Kernel Flow
8 A Django Platform for comparing scientific methods for analyzing neural time series
    - 8.1 A Django Platform for comparing scientific methods for analyzing neural time series
9. sktime projects

Software engineering projects (9.1-9.4)
9.1 Time series regression
9.2 Time series clustering
9.3 Forecasting
9.4 Develop a new module
Applied neuroscience projects
Project 9.5: Classification using EEG
Project 9.6: Detecting Early Onset Dementia

10. XNAT connector for neuroimaging data integration cancelled

11. Measure Quality of CerebUnit Validation tests
11.1 Measure Quality of CerebUnit Validation tests

12. Open Source Brain projects
12.1 Open source, cross simulator, large scale cortical models in NeuroML and PyNN
12.2 Conversion of public neurophysiology datasets to NeuroData Without Borders format

13 MAXIMA
13.1 Maxima Demo and Documentation Tool

14 BIDS project
14.1 EEG/MEG-to-BIDS converter: a GUI application in Python
14.2 BCI2000-to-BIDS converter: a GUI application in Python/Matlab/C++

15 Turing Institute projects
15.1 The Turing Way: A how-to guide to data science
15.2 AutSPACEs: a co-created citizen science platform

16 BRIAN simulator projects
16.1 Supporting simulation based inference with the model fitting toolbox
16.2 Make Brian run in the web browser via Pyodide
16.3 Update documentation infrastructure
16.4 OpenMP for synaptic propagation

17 AnalySim project
17.1 AnalySim: a data sharing and notebook analysis platform written in Angular and ASP.net with C#

18 Decentralized Autonomous Organizations for Science project
18. 1 Securing Neuroimaging BIDS Datasets Stored on Decentralized File Sharing Networks with Ethereum Public Keys, Git Annex and IPFS
18. 2 Tokenizing and Deploying BIDS Apps on Distributed Confidential Cloud Resources for Anonymous Computation with Ocean Protocol and iExec RLC

19 DIPY projects

19.1 Population-based MRI Template Creation
19.2 Population-specific Tractography Bundle Atlas Creation
19.3 Extend DIPY Horizon workflow for Visualization
19.4 DIPY-Tract-or-Treat: DIPY DTI Post-Processing Pipeline

20 BrainBox projects
20.1 Brain Badges: Online certification for collaborative annotation and segmentation of brain imaging data
20.2 Brain Histology Metadata DB: Building an online database for storing, curating and sharing metadata from diverse collections of histological data
20.3 Improving end-to-end testing and code documentation for BrainBox

21 HDNet projects: Developing a Python-based codebase for analyzing stimulus prediction capabilities of neurons
21.1 Developing a Python-based codebase for analyzing stimulus prediction capabilities of neurons: improve the neural Maximum Entropy model
21.2 Developing a Python-based codebase for analyzing stimulus prediction capabilities of neurons: improve stimulus prediction

22 Mobile EEG Neurotechnology
22.1 Cognitive Neuroscience Experiments with Mobile EEG Neurotechnology

23 NeuroML project
23.1 Re-creating the Leech Heartbeat Network Model Tutorial using the Neuron Simulator in Python and NeuroML

24 lab.js projects
24.1 Collecting Behavioral and Neuroscientific Data with Shiny (in R) and lab.js
24.2 Laboratory Hardware Support for lab.js via the Lab Streaming Layer

25 LORIS open data platform for neuroscience research
25.1 Contribute to LORIS

26 Eye-tracker project
26. 1 Eye-tracker based on a convolutional neural network in Python + TensorFlow/Pytorch

27 Brain Dynamics Toolbox project
27.1 Brain Dynamics Toolbox for Matlab: Contribute new models to the collection

Available mentors for student-originated projects

---

1. High-order neuroimaging projects

1.1 A Python toolbox for computing high-order information in neuroimaging

The functioning of complex systems (i.e. the brain, and many others) depends on the interaction between different units; crucially, the resulting dynamics is different from the sum of the dynamics of the parts. In order to deepen our understanding of these systems, we need to make sense of these interdependencies. Several tools and frameworks have been developed to look at different statistical dependencies among multivariate datasets. Among these, information theory offers a powerful and versatile framework; notably, it allows detecting high-order interactions that determine the joint informational role of a group of variables.

The goal of this project is to collect and refine existing metrics of high-order interactions currently implemented in Matlab or Java, for example
https://github.com/brincolab/High-Order-interactions
https://github.com/danielemarinazzo/dynamic_O_information
https://github.com/jlizier/jidt
and integrate them in an unified python-based toolbox.

The main deliverable of the project is a toolbox, whose inputs are the measurements of different variables plus some parameters, and whose outputs are the measures of higher order interdependencies. Ideally, the toolbox will interface with visualisation & processing platforms of neuroimaging data, such as
https://mne.tools/stable/index.html
https://fmriprep.org/en/stable/
and become a docker container too. A parallel project would focus on the visualization of these higher order measures.

Experience in statistics and neuroimaging data analysis is a plus, but not a requirement.

Lead mentor: Daniele Marinazzo, Ghent University; [Neurostars] , [GitHub]
Co-mentor: Fernando Rosas, Imperial College London; [Neurostars],[GitHub]

Tags: neuroimaging, statistics, Python, Matlab, Java, MNE, fMRIprep

Discuss this project on Neurostars

1.2 Visualizing high-order interactions in neuroimaging data and beyond

The functioning of complex systems — such as the human brain, the climate, and many more — is characterised by emergent phenomena, which is a consequence of non-linear interactions between various of its constituent units. Crucially, the resulting dynamics

observed in such systems is qualitatively different from the sum of the dynamics of each part. A promising way to deepen our understanding of these systems is to leverage existing datasets of these complex systems; however, the analysis of these data requires specific analytic tools to make sense of their highly non-trivial interplay.

Several tools and frameworks have been developed to look at different statistical dependencies that can be observed in these fascinating multivariate datasets. Among these, information theory offers a powerful and versatile framework; notably it allows to detect and quantify high-order interactions: namely, the characteristic of the joint informational role of a group of variables. However, metrics of high-order interactions are challenging to visualize, and the lack of adequate visualization paradigms is hindering their potential impact in numerous research fields.

While visualization is easy and well explored for local measures and pairwise connectivity ones, it becomes trickier when it comes to higher order measures. In fact, a lot of the appeal and understandability of these metrics data analyses (in general, and in particular in neuroimaging) relies on their visual representation, with or without an anatomical overlay. The goal of this project is to conceptualize and implement strategies and tools for the visualization of high-order interactions, both for general datasets, and also specifically tailored to neuroimaging data analysis — mainly M/EEG and fMRI. The main deliverable of the project will be a toolbox that integrates the developed approaches. (A parallel project will focus on the implementation of these high-order metrics in a python-based toolbox.)
Any open source tool (starting with python and javascript) is welcome. Experience in statistics and neuroimaging data analysis is a plus, but not a requirement.

Lead mentor: Fernando Rosas, Imperial College London; [NeuroStars], [GitHub]
Co-mentor: Daniele Marinazzo, Ghent University; [NeuroStars], [GitHub]

Tags: neuroimaging, statistics, visualization, Python, Javascript, MEG, EEG

Discuss this project on Neurostars

2. ImageJ Active Segmentation projects

2.1 Deep Learning using Geometric Features

The Active Segmentation platform for ImageJ (ASP/IJ) was developed in the scope of GSoC 2016 - 2019. The plugin provides a general-purpose environment that allows biologists and other domain experts to use transparently state-of-the-art techniques in machine learning to achieve excellent image segmentation and classification. ImageJ is a public domain Java image processing program extensively used in life and material sciences. The program was designed with an open architecture that provides extensibility via plugins.

The project idea: The existing machine learning model of Active Segmentation is based on the Weka library. However, this is limited to traditional machine learning approaches. The objective of the project will be to incorporate deep-learning functionality into the platform. Deep neural nets are capable of record-breaking accuracy.

The project is a continuation of GSOC 2020. The project will start from an already available codebase implemented using Deeplearning4j. At present, there are 2 implemented architectures U-Net — an architecture for biomedical image segmentation, and SegNet — a deep learning semantic segmentation architecture.

The candidate will work out the GUI integration with the rest of the ASP/IJ platform.

Tasks:

- Fix existing issues and bugs
- Get familiar with Deeplearning4j
- GUI implementation and integration of the Deeplearning4j functionality

Minimal set of deliverables:

- Requirement specification - Prepared by the candidate after understanding the functionality.
- System Design - Detailed plan for the development of the plugin and test cases.
- Implementation and testing - Details of implementation and testing of the platform.

Desired skills: Java, Machine Learning

Mentors: Sumit Vohra, ZIB, Berlin, Germany; Dimiter Prodanov (dimiterpp@gmail.com), INCF Belgian Node (backup)

References:
ImageJ: https://imagej.nih.gov/
Weka https://www.cs.waikato.ac.nz/ml/weka/
Active Segmentation : https://github.com/sumit3203/ACTIVESEGMENTATION
Deeplearning4J: https://deeplearning4j.org/

Tags: ImageJ, segmentation, machine learning, deep learning, GUI

Discuss this project on Neurostars

2.2 High content image feature and classification database

The Active Segmentation platform for ImageJ (ASP/IJ) was developed in the scope of GSoC 2016 - 2019. The plugin provides a general-purpose environment that allows biologists and other domain experts to use transparently state-of-the-art techniques in machine learning to achieve excellent image segmentation and classification. ImageJ is a public domain Java image processing program extensively used in life and material sciences. The program was designed with an open architecture that provides extensibility via plugins computing different filters and region descriptors (i.e. image features).

The project idea: At present, the feature space and the classification results produced by the platform are stored in several files. The idea is that the image features and classification results would be stored in a SQL database for cross-comparisons between sessions. SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. The candidate is encouraged to use an SQLite database engine and a proof-of-concept implementation storing ImageJ measurements in order to integrate it with the GUI of ASP/IJ.

Tasks:

- Fix existing issues and bugs
- SQL database design
- GUI implementation and integration

Minimal set of deliverables:

- Requirement specification - Prepared by the candidate after understanding the functionality.
- System Design - Detailed plan for the development of the plugin and test cases.

- Implementation and testing - Details of implementation and testing of the platform.

Desired skills: Java, SQL

Mentors: Dimiter Prodanov (dimiterpp@gmail.com), INCF Belgian Node; (backup) Sumit Vohra, ZIB, Berlin, Germany

References

ImageJ: https://imagej.nih.gov/
Weka https://www.cs.waikato.ac.nz/ml/weka/
Active Segmentation : https://github.com/sumit3203/ACTIVESEGMENTATION
SQLite: https://www.sqlite.org/index.html

Tags: ImageJ, Java, SQL, classification

Discuss this project on Neurostars

2.3. Input/Out Workflows for Active Segmentation Platform

The platform for ImageJ (ASP/IJ) was developed in the scope of GSOC 2016 - 2019. The plugin provides a general-purpose environment that allows biologists and other domain experts to use transparently state-of-the-art techniques in machine learning to achieve excellent image segmentation and classification. ImageJ is a public domain Java image processing program extensively used in life and material sciences. The program was designed with an open architecture that provides extensibility via plugins.

The project idea: The existing active segmentation supports both traditional machine learning and deep learning approaches. However, the existing implementation supports a limited way to load the ground truth for learning i.e. region of interest (roi)file format. This forces users to convert their ground truth to specific format and which is not even suitable for multiple applications. The objective of the project will be to incorporate several ground truth formats into the platform e.g. image-based in which each pixel uniquely belongs to a particular class, partial ground truth format in which instead of whole image, several partial boxes in an image or stack are labeled, User need to customize metafile format in order to load such a partial ground truth. On the similar note, the candidate will also allow several output formats to store segmentation and classification results. For example, output only the user selected region of interest or partial stack.

Tasks

- Fix existing issues and bugs
- Get familiar with input/ out format i.e. image based, region of interest based, JSON.
- GUI implementation and integration

Minimal set of deliverables

- Requirement specification - Prepared by the candidate after understanding the functionality.
- System Design - Detailed plan for development of the plugin and test cases.
- Implementation and testing - Details of implementation and testing of the platform.

Desired skills: Java, Machine learning

Mentors: Sumit Vohra, ZIB, Berlin, Germany; Dimiter Prodanov (dimiterpp@gmail.com), INCF Belgian Node (backup)

References

1. ImageJ: https://imagej.nih.gov/
2. Weka https://www.cs.waikato.ac.nz/ml/weka/
3. Active Segmentation : https://github.com/sumit3203/ACTIVESEGMENTATION
4. Deeplearning4J: https://deeplearning4j.org/

Discuss this project on Neurostars

3. DevoWorm projects

3.1 Upgrading DevoLearn

Devolearn is a Python package that aims to automate the process of collecting metadata from videos/images of the C. elegans embryo with the help of deep learning models. It is specialized for the analysis of 2-D slices of C. elegans embryogenesis.

This project will focus on improving the performance of existing models and training/benchmarking new models on a broad range of data sets from different species relevant to developmental biology. Your goals will be to improve overall performance of the models in terms of accuracy and generalizability, training/adding new models and improve the library's usability.

There would be 4 key elements in this project:

- Improving the current models
- Training and adding more useful models
- Improving usability
- (optional) Interactive online demos.

Improving/adding new models
In order to get good performance from a deep-learning model, we should use techniques like:

- k-fold cross validation
- Ensembling/stacking
- Hyperparameter optimization
- Label smoothing
- Mixed precision inference
- Using more training data for re-training the pre-existing models for better accuracy

Improving usability
As the library grows, we should have a better place to host proper documentation on a static website. We also have to train devolearn's deep-learning models on commonly available data, the more common the use case, the more helpful it is to the community.

Online demos
Jupyter notebooks, however simple they may seem, are still a bit intimidating to people from non CS backgrounds, so there should be a focus on making interactive and easy to run online demos which showcase both our research and tutorials.

Pre-requisites
Proficiency in python with a good hold of tools like numpy, pandas and PyTorch will be required.

Resources

- DevoLearn source code and contribution guidelines
- Wormbook
- GSoC 2020 (parent project)

More information can be found at:
https://github.com/devoworm/Proposals-Public-Lectures/blob/master/GSoC/2021/upgrading-devolearn.md

Mentors: Bradly Alicea, Mayukh Deb

Tags: DevoWorm, Python, microscopy, deep learning, data science

Discuss this project on Neurostars

3.2. Deep Learning for Diatom Non-neuronal Cognition

The aim of this project will be to improve upon a Deep Learning model that extracts morphological features from microscopy images of *Bacillaria Paradoxa*. This will allow our organization to begin creating a database for the study of movement behavior and non-neuronal cognition in simple multicellular organisms. You will be improving upon the Digital _Bacillaria_ project, which was started in the Summer of 2019. This year our main aim is to enhance the existing deep learning model (implemented in TensorFlow), as well as integrate the model into our species-specific library of machine learning models (DevoWorm AI). You will be involved in pre-processing and analyzing microscopy videos from our database of _Bacillaria_ movement, along with tweaking the model for greater generalization. Integration of the model will involve adding functionality in the form of an interactive GUI, which will allow our community to analyze and display the data in terms of interesting behavioral variables. The successful applicant will be proficient in Python, C++, the basics of Machine Learning libraries and computer vision, HTML, and CSS.

More information can be found at:
https://github.com/devoworm/Proposals-Public-Lectures/blob/master/GSoC/2021/DL-for-Diatom-Non-neuronal-Cog.md

Mentors: Bradly Alicea, Mayukh Deb

Tags: DevoWorm, Python, deep learning, machine learning, TensorFlow, GUI, C++, HTML, CSS

Discuss this project on Neurostars

3.3. Digital Microsphere

To better understand the morphogenesis of the nervous system, it is necessary to develop novel techniques. This project will build upon the specialized microscopy techniques to develop a shell composed of projected microscopy images, arranged to represent the full external surface of a sphere. This will allow us to create an atlas of the embryo's outer surface, which in some species (e.g. Axolotl) enables us to have a novel perspective on neural development. The source data consists of nine images per sample from different points of view, which will be projected onto the surface of a virtual sphere. This will allow us to view the surface either continuously or in a montaged fashion. The student must be proficient in Computer Vision (C++, Python) and perhaps even Topological techniques, and it will be up to the student to propose a viable solution to this problem. The following references may be helpful to the student's application:

Gordon, R. (2009). Google Embryo for Building Quantitative Understanding of an Embryo As It Builds Itself. II. Progress Toward an Embryo Surface Microscope.
Biological Theory, 4(4), doi:10.1162/BIOT_a_00010

Crawford-Young, S., Dittapongpitch, S., Gordon, R., and Harrington, K. (2018). Acquisition and reconstruction of 4D surfaces of axolotl embryos with the flipping stage robotic microscope. Biosystems, 173, doi:10.1016/j.biosystems.2018.10.006

More information can be found at:
https://github.com/devoworm/Proposals-Public-Lectures/blob/master/GSoC/2021/digital-microsophere.md

Mentors: Bradly Alicea, Mayukh Deb

Tags: DevoWorm, Python, C++, microscopy, atlasing

Discuss this project on Neurostars

4. Unit test for brains

4.1. Unit tests for brains

Description: Unit tests are ubiquitous in software development (Google alone has 4.7 million running all the time!), but surprisingly sparse in science. The SciUnit (http://sciun.it) framework was developed to help researchers create unit tests for scientific models. SciUnit tests ask *how well* a scientific model does what it claims to do, by testing its predictions against specific experimental results. Performance on these tests is thus one measure of whether the model is a good (or at least useful) description of reality. SciUnit is being used in large projects in neuroscience, including in the Human Brain Project and OpenWorm, as well as in individual labs.

Aims: These unit tests can be used not only to assess developed models, but also to guide model development, including the tuning of model parameters to reproduce experimental data. Several challenges arise in multi-objective optimization of models against data, including parallelization, visualization, and reproducibility. We aim to solve some of these challenges in the optimization of models of neurons and neural circuits. In particular, we want to refine and deploy reproducible optimization workflows for models against diverse sets of neurophysiology data, and then share and compare these optimized models for subsequent use in larger research questions.

Mentor: Rick Gerkin (rgerkin@asu.edu), ICON Lab, Arizona State University, USA

Tags: Python, Git, Jupyter, Numpy

Discuss this project on Neurostars

5. Neurodata Without Borders projects

5.1 Visualization of Neurodata Without Borders data standard specifications

Neurodata Without Borders (NWB) is a data standard for neurophysiology, providing neuroscientists with a common standard to share, archive, use, and build analysis tools for neurophysiology data. The NWB standard defines over 60 data types and an organization of these types based on composition and inheritance, similar to object-oriented programming. The goal of this project is to enhance the Python-based documentation tools for NWB to support intuitive visualization of standard specifications and enhance the usability of NWB. Example visualizations of interest include the creation of Unified Modeling Language (UML) diagrams from specifications and the visualization of specifications of data tables. The first aim of this

project will be to create static visualizations for inclusion in online documentation. This will be done using open-source UML diagramming or graph-based visualization tools. If time permits, the second aim will be to create interactive visualization for specifications where users can step through type hierarchies and associated types.

Students should be familiar with Python and popular plotting libraries. Familiarity with UML, Sphinx, YAML, or neuroscience is also helpful.

Lead mentor: Oliver Ruebel

Backup mentor: Ryan Ly, Ben Dichter

Tags: NWB, Python, OOP, UML, YAML

Discuss this project on Neurostars

5.2 Interactive visualization of neurophysiology data in NWB Widgets

Neurodata Without Borders (NWB) is a data standard for neurophysiology, providing neuroscientists with a common standard to share, archive, use, and build analysis tools for neurophysiology data. The NWB standard covers over 60 data types. In tandem with the development of the data standard, we are also developing NWB Widgets (https://github.com/NeurodataWithoutBorders/nwb-jupyter-widgets), an extensible library of Jupyter widgets that allows researchers to navigate through the structure of NWB files and visualize specific data elements. Interactive visualization of these data allows researchers to explore the data, gain scientific insight, and check the data for errors and artifacts. The goal of this project is to develop visualizations for data types that do not yet have visualizations and to enhance usability and scalability of existing visualizations. The first aim is to develop visualizations for videos of neural activity acquired through a microscope. The second aim is to develop interactive tools that would allow a user to navigate easily through the stages of data processing.

Students should be proficient in Python and familiar with Jupyter, matplotlib, and plotly. Familiarity with Jupyter widgets or neuroscience is also helpful.

Lead mentor: Ben Dichter

Backup mentor: Ryan Ly, Oliver Ruebel

Tags: NWB, Python, Jupyter, ipywidgets, matplotlib, plotly

Discuss this project on Neurostars

5.3 Enhancing web interface and automated processes of the Neurodata Extensions Catalog

Neurodata Without Borders (NWB) is a data standard for neurophysiology, providing neuroscientists with a common standard to share, archive, use, and build analysis tools for neurophysiology data. The NWB standard can be extended to support new and specialized data types through the creation of user-defined neurodata extensions (NDX). The Neurodata Extensions Catalog (https://nwb-extensions.github.io/) provides a centralized listing of neurodata extensions for users to publish, share, find, and discuss extensions. The Catalog relies on a set of automated processes for testing, reviewing, and updating extensions, similar to the feedstock model used by conda-forge. The first aim of this project is to improve the robustness of the Neurodata Extensions Catalog by using authenticated GitHub API requests and caching responses. The second aim of this project is to support continuous integration for the Catalog so that unit tests and other processes are automatically run when new extensions are submitted, approved, and updated.

Students should be familiar with Python, APIs, and continuous integration (CI). Familiarity with the GitHub API, GitHub Pages, conda-forge, or neuroscience is also helpful.

Lead mentor: Ryan Ly

Backup mentor: Oliver Ruebel

Tags: NWB**,** Python, GitHub API, continuous integration, GitHub pages

Discuss this project on Neurostars

6. GeNN projects

6.1 Improving the PyNN interface to GeNN

PyNN is a Python based framework for describing neuronal network models.
It is an open standard, widely used in the computational neuroscience and neuromorphic computing communities. A PyNN interface for GeNN has already been developed (https://github.com/genn-team/pynn_genn) so that users of PyNN are able to benefit from accelerated GPU simulations with GeNN. However, there are several areas in which it could be improved:

- Using GeNN's new on-GPU spike recording system to reduce the overheads of spike recording
- Offloading initialisation of connectivity and initial state to the GPU.
- Replacing spike sources connected with one-to-one connectivity with current sources to reduce spike processing overheads

A final stretch goal would be doing some benchmarking of the performance of simulations implemented directly in GeNN and using PyNN GeNN.

Skills required: Python, PyNN, C/C++; experience with neuronal network simulations and SWIG would be helpful.

Mentors: Jamie Knight (J.C.Knight@sussex.ac.uk) and Thomas Nowotny (t.nowotny@sussex.ac.uk)

Tags: GeNN, Python, PyNN, C/C++, SWIG

Discuss this project on Neurostars

6.2 ISPC backends for GeNN

GeNN is a C++ library that generates code for efficiently simulating Spiking Neural Networks using GPUs. Currently, GeNN generates CUDA, OpenCL code meaning that it can target a wide range of GPUs . However, for simulating smaller models, focused targeting of SIMD CPUs may be advantageous. For this project you will develop a new GeNN code-generation backend for ISPC (http://ispc.github.io/), which targets the SIMD units in a wide range of modern CPUs using a CUDA-like programming model.

Skills required: C++, Single Instruction Multiple Thread (SIMT) programming

Mentors: Jamie Knight (J.C.Knight@sussex.ac.uk) and Thomas Nowotny (t.nowotny@sussex.ac.uk)

Tags: GeNN, CUDA, C++, SIMT

Discuss this project on Neurostars

6.3 Brian2GeNN

Brian2GeNN (https://github.com/brian-team/brian2genn) is an interface between the popular Brian 2 simulator for neuronal networks and GeNN, a software that supports efficient simulation of spiking neural networks on GPUs and similar backends. When Brian2GeNN was created, a number of Brian 2 and GeNN features could not be supported as corresponding mechanisms in the respective other software were not available or not easily translatable. With further development of both systems, many of these restrictions can now be removed. In this project you will remove unnecessary restrictions in Brian2GeNN, including but not limited to

- Heterogeneous synaptic delays

- Neuron and synapse initialisation on GPU

Skills required: Python, C++; experience with Brian 2, GeNN, or even Brian2GeNN would be highly beneficial

Mentors: Jamie Knight (J.C.Knight@sussex.ac.uk), Marcel Stimberg (marcel.stimberg@inserm.fr), and Thomas Nowotny (t.nowotny@sussex.ac.uk)

Tags: GeNN, Python, C++; Brian 2, GeNN, Brian2GeNN

Discuss this project on Neurostars

6.4 Open-source biorobotics framework

As researchers in biorobotics, we are fascinated with the remarkable navigational prowess of desert ants and are striving to produce resilient robots that are able to navigate just as dependably over challenging environments. As part of this, we are developing an open-source biorobotics framework in C++ (BoB robotics: https://github.com/BrainsOnBoard/bob_robotics) to enable collaboration on this vision.

Your task will be to work on the problem of sensor fusion using BoB robotics and the open-source robotics simulation software Gazebo, to investigate how best to combine information from multiple incoming sensory sources -- a problem faced by both ants and robots -- on a simulated bioinspired robotic platform.

This project can involve computer vision (we are particularly interested in visual sensors) and machine learning, depending on the student's interests and commitment level. Some prior experience with C++ is a must. Experience with the git workflow and using GitHub is also a plus, as this will be the primary way in which you interact with our team.

Mentors: Jamie Knight (J.C.Knight@sussex.ac.uk), Alex Dewar (alex.dewar90@gmail.com)

Tags: GeNN, biorobotics, robotics, gazebo, sensors

7. The Virtual Brain (TVB) projects

7.1 Reusable visualization tools for Jupyter

TVB's web-based UI provides several very useful visualization tools, which are setup for full screen use. As TVB is used in wider contexts (HBP collaboratory,

Jupyter notebooks), it is important to ensure the relevant visualization tools are present everywhere.

This project is to rewrite the widgets in TVB UI to become reusable components which can be employed from a Jupyter notebook for use in the HBP collaboratory, while maintaining compatibility with the existing TVB framework. Tools are to be refactored, choice up to the student, in order of priority:

- anatomical visualization (surface, connectivity)
- time series viewer
- the phase plane tool

Screenshots for inspiration:
https://github.com/the-virtual-brain/tvb-documentation/blob/master/manuals/UserGuide/screenshots/visualizer_dual_seeg.jpg
https://github.com/the-virtual-brain/tvb-documentation/blob/master/manuals/UserGuide/screenshots/simulator_phase_plane_interactive.jpg

Use of WebGL (in particular Python/notebook oriented GL tools) is encouraged, where numerous interesting opportunities for optimization are present, e.g. XTK for anatomy, vispy for time series.

Expected results: A set of classes usable within Jupyter notebook, for displaying common data objects via WebGL or WebGL-based libraries. Documentation and usage examples for all created classes.

Skills: Python, WebGL, IPyWidgets, Jupyter

Mentors: Lia Domide, Paula Popa

Tags: TVB, Python, WebGL, IPyWidgets, Jupyter

Discuss this project on Neurostars

7.2 Benchmark and Optimize TVB areas

TVB has become a complex tool, with some generic areas and few very specific critical parts. Due to these, some parts are slow now. The team has identified some area with great potential to improve:

1. Simulation wizard
2. H5 files writing
3. loading for visualizers from files

Starting from these critical parts (identifying others from the student tests would be much appreciated), writing or executing a flow, profiling the code usage and proposing improvements would be the goal of this project. We expect the student to start by accommodating with TVB documentation, understanding the main workflows, creating a work development setup, then use benchmark and profiling tools for identifying the critically slow areas in the code, as well as propose optimization solutions.

Expected results: A set of well described slow scenarios for benchmarking, profiling files with highlighted critical code areas, suggestions to improve (vectorize code, rewrite, use buffers, etc).

Skills: Python, Numpy, Numba, profiling tools

Mentors: Lia Domide, Paula Popa

Tags: TVB, Python, NumPy, Numba

Discuss this project on Neurostars

7.3 Computational modelling of FNIRS and EEG with The Virtual Brain and Kernel Flow

The Virtual Brain (TVB) is a widely-used software library for connectome-based modelling of large-scale neural dynamics and neuroimaging data. To date, TVB models have mainly focused on brain dynamics as measured by fMRI, EEG, and LFPs. Functional Near Infrared Spectroscopy (FNIRS) is another noninvasive neuroimaging modality, which like fMRI measures haemodynamic signals reflecting neural activity, that has major potential in cognitive and clinical neuroscience.

The aim of this GSoC project will be to build out the modelling and analysis capacity of TVB for simulations of whole-head, high-resolution FNIRS signals, as well as concurrently recorded EEG. This shall include writing code implementing a temporal forward model for FNIRS-specific haemodynamic signals, a spatial forward model for optical sensor projection, and running simulations exploring dynamics of concurrent FNIRS-EEG activity. In terms of hardware, these development activities will be primarily focused on the Kernel Flow FNIRS+EEG system, that we will have access to and be using as part of the project.

Candidates should have experience with Python for scientific computing, and a strong interest in computational neuroscience and neuroimaging. Experience with one or more of the following is desirable: FNIRS/fMRI/EEG data analyses, neural mass modelling, numerical simulations and numerical optimization / model fitting problems in neuroscience or other domains. The project will provide excellent experience and training for students interested in pursuing research in

human neuroimaging, theoretical/computational/cognitive/clinical neuroscience, and brain-computer interfaces.

Lead Mentor: John Griffiths
Co-Mentor: Randy McIntosh

Tags: Python, TVB, Computational Neuroscience, Neural mass modelling, Connectome, Biophysics, FNIRS, EEG, electrophysiology

Discuss this project on Neurostars

8 A Django Platform for comparing scientific methods for analyzing neural time series

8.1 A Django Platform for comparing scientific methods for analyzing neural time series

Time series are measured and analyzed across the scientific disciplines, with new methods for their analysis being developed regularly. This is particularly true for quantifying patterns in brain dynamics, for which hundreds of methods have been developed and applied. How can we distinguish a real advance from a new method that actually reproduces the behavior of an existing method? The need for comprehensive and systematic comparison is paramount in methodological literatures like time-series analysis, but is incredibly difficult to achieve practically. We have recently developed an online portal for comparing time-series data, CompEngine (https://www.comp-engine.org/), that allows scientists to drag-and-drop their data onto our portal to get an answer to the question: “what sorts of data from across science are similar to the data that I measure?” However, there is still no way for scientists to compare their methods to alternative methods from other disciplines. Achieving this would enable scientists to better work together across disciplinary boundaries, towards a unification of methods for time series. Such an endeavor would be transformative in facilitating the concentration of scientific effort towards meaningful interdisciplinary progress, and could become a template for similar efforts applied to other data types.

In this project, we will continue to develop and refine a Django online platform for comparing time-series analysis methods. As CompEngine does for time-series data, this would allow a scientist to upload their code, compute the results on a dataset, and search a library of existing features for the most similar methods. The scientist could then be given a ‘uniqueness’ score, and be able to visualize their method in a broader scientific context. We have begun to develop this at https://github.com/DynamicsAndNeuralSystems/Comp-Engine-Features. Interested

students should have interest and/or experience in web development, python, django, databases, and time series.

Mentors: Ben Fulcher (ben.fulcher@sydney.edu.au) and Oliver Cliff (oliver.cliff@sydney.edu.au).

Tags: time series, Django, Python, databases

Discuss this project on Neurostars

9. sktime projects

Why sktime? Time series data is ubiquitous in many applications. Examples include sensor readings from industrial processes, spectroscopy wavelength data from chemical samples, or bed-side monitor medical data from patients. Developing advanced time series analysis capabilities for researchers and practitioners is one of the major challenges of contemporary machine learning.

sktime is a new Python toolbox for machine learning with time series and, to the best of our knowledge, the first unified toolbox for time series. Our ambition is to provide for time series what scikit-learn provides for tabular data. This involves extending scikit-learn to the different time series learning tasks, such as time series classification, clustering, forecasting and anomaly detection. To find out more, check out our paper published at the Workshop on Systems for ML at NeurIPS 2019.

What we are looking for

We're actively looking for contributors and your help is extremely welcome. Therefore, if

- you are interested in time series, machine learning (ML), statistics, API design and software architecture,
- you like coding in Python,
- you are familiar with the basic data science ecosystem in Python, including numpy, pandas and scikit-learn,
- you enjoy working with a vibrant team of experienced ML scientists and software engineers,
- you always wanted to join an open-source community,

then GSoC with sktime is for you! You'll spend the summer working with our enthusiastic and open-minded team of developers who are creating one of the first comprehensive time series ML toolboxes out there.

Read more, including closer instructions to students, on the sktime Github GSoc page.

Mentors: Markus Löning, Tony Bagnall, Franz Király, Martina Vilas

Tags: sktime, time series, Python, machine learning (ML), statistics, API, citizen science

sktime projects

Please find below a list topics to help you get started. But please don't hesitate to propose your own topic to work on. Please check out our development roadmap and good first issues to get an idea of the topics we're working on.

Software engineering projects (9.1-9.4)

Title	Mentors	Description	Difficulty	Useful skills
9.1 Time series regression	@mloning	Refactoring classifiers into regressor	beginner	Regression with scikit-learn, Python
9.2 Time series clustering	@TonyBagnall	Implementing time series clustering algorithm using time series distances	beginner	Clustering with scikit-learn, Python
9.3 Forecasting	@mloning	Implementing algorithms for model selection, composition and reduction	medium	Familiarity with scikit-learn estimator API, Python
9.4 Develop a new module	@fkiraly	Annotation tasks, multi-sample forecasting tasks, event modelling tasks, probabilistic tasks; check out our enhancement proposal to find out more about the design process	ambitious :-)	Familiarity with typical user journeys for chosen tasks

Discuss these 4 projects on Neurostars

Applied neuroscience projects

Project 9.5: Classification using EEG

Mentor: @TonyBagnall

Electroencephalograms (EEG) and magnetoencephalography (MEG) are techniques for measuring, directly or indirectly, actual or relative changes in voltage throughout the body. This type of time series data is often related to classification problems. EEG and MEG are also central to human computer interface research. The question we wish to address on this project is, what is the best way to make predictions from ECG/EEG and MEG? We will treat the problem as time series classification. We have an existing database of publicly available labelled EEG/MEG datasets, tools in both Python for time series classification and collaborators who are experts in traditional techniques for analysing EEG/MEG. Our ultimate goal is an effective automated end-to-end pipeline for EEG/MEG classification using open source software such as sktime. This would encourage reproducible research in the field and facilitate more widespread analysis. This ten-week project will contribute to this goal and will involve three phases.

- The first will involve implementation of a pipeline benchmark approach of feature extraction and standard classification. This will require standard toolkits for features such as wavelets as soft dependencies in sktime and the evaluation of existing techniques with a range of transformation/scikit-learn classifier combinations.
- The second phase will involve the development of existing time series classification algorithms to work directly with EEG data.
- The final phase will be to help construct and manage an experimental scheme to compare the performance of a range of algorithms on problems of this kind.

Discuss this project on Neurostars

Project 9.6: Detecting Early Onset Dementia

Mentor: @TonyBagnall

Sea Hero Quest (SHQ) is a citizen science project which uses gamification as a means to collect large scale data for future use in preclinical dementia diagnosis. SHQ has to date collected data from 4.5 million healthy people worldwide. In addition, our collaborator Prof. Hornberger has patient cohort data on the game. The game itself collects basic demographic and level outcome measures, and trajectory data (position, heading direction) for each player in each level is collected every 500ms. This fully anonymised data can potentially be used to predict the risk of early onset of dementia through trajectory information. This project will involve

developing a direct interface to the SHQ game so that we can develop tools to process data and ultimately, attempt to make predictions.

Discuss this project on Neurostars

~~10. XNAT connector for neuroimaging data integration~~ cancelled

11. Measure Quality of CerebUnit Validation tests

11.1 Measure Quality of CerebUnit Validation tests

Quantitative analysis and thinking are essential in the scientific process. Models are computer implementation of mathematical system of equations built on one or more hypothesis. Scientific method requires testing the model against experimental observations: this is model validation - the term verification will be used to encompass software validation. The project revolves around CerebUnit: a SciUnit based validation module suite that deals with models of the cerebellum - the part of the brain below the cerebrum (larger) and behind the brainstem.

Validation tests compares the model prediction against experimental data. The model is then analyzed based on the results. To avoid getting overawed by the mystique of objectivity emanating from the numerical figures one should ask "How dependable is the test?"

The project will involve adding features in CerebUnit (CerebStats) that will try to address the question. In particular,

- ability to return the probability of type-I or type-II error for a validation test that has undergone hypothesis testing
    - which error type probability is returned will depend on the user's requirement
        - return type-I error probability for false positive
        - return type-II error probability for false negative

- ability to get measures of performance of a given validation test
    - sensitivity (true positive rate)
    - specificity (true negative rate)
    - prevalence

- positive predictive value
- negative predictive value

Skills: Python, statistics (basic principles of statistical testing)

Mentors: Lungsi Sharma (lead), Andrew Davison

Tags: CerebUnit, Python, statistics

Discuss this project on Neurostars

12. Open Source Brain projects

12.1 Open source, cross simulator, large scale cortical models in NeuroML and PyNN

An increasing number of studies are using large scale network models incorporating realistic connectivity to understand information processing in cortical structures. High performance computational resources are becoming more widely available to computational neuroscientists for this type of modelling and general purpose, well tested simulation environments such as NEURON and NEST are widely used. New, well annotated experimental data and detailed compartmental models are becoming available from the large scale brain initiatives. However, the majority of neuronal models which have been published over the past number of years are only available in simulator specific formats, illustrating a subset of features associated with the original studies.

This work will involve converting a number of published large scale network models into open, simulator independent formats such as NeuroML and PyNN and testing them across multiple simulator implementations. They will be made freely available to the community through the Open Source Brain repository (http://www.opensourcebrain.org) for reuse, modification and extension.

Skills required:

Python; XML; open source development; a background in computational/theoretical neuroscience and/or large scale modelling experience.

Aims:

1) Select a number of large scale cortical network models for the conversion & testing process (e.g. from ModelDB).

2) Convert network structure and cell/synaptic properties to NeuroML and/or PyNN. Where appropriate use the simulator independent specification in LEMS to specify cell/synapse dynamics & to allow mapping to simulators. Implementing extensions to PyNN, NeuroML or other tools may be required.

3) Make models available on the Open Source Brain repository, along with documentation and references.

Mentors: Padraig Gleeson (lead), Ankur Sinha

Tags: Python, XML, networks, modelling, simulation

Discuss this project on Neurostars

12.2 Conversion of public neurophysiology datasets to NeuroData Without Borders format

More and more of the experimental datasets behind publications in neuroscience are being publicly released, increasing transparency of the scientific process and allowing reuse of the data for new investigations. However, these datasets, which can include electrophysiology, 2D/3D imaging data and behavioural recordings, are not always in an accessible format, and it may take some effort for researchers to access and analyse the data before deciding to use it in their own research. The Neurodata Without Borders (NWB, https://www.nwb.org) initiative is developing a format for sharing data from neurophysiology experiments which, together with APIs for handling files in the format, promises to greatly facilitate the sharing and reuse of data in neuroscience.

This project will involve converting a number of publicly available datasets to NWB format, adding structured metadata to ensure maximal understandability and reusability of the data. The converted datasets will be made available through the new **NWB Explorer** on the Open Source Brain repository (http://nwbexplorer.opensourcebrain.org) which allows visualisation of the data as well as interactive analysis through an inbuilt Jupyter notebook.

Skills required:

Python; open source development; neuroscience (experimental or computational) background; data analysis.

Aims:

1) Select a number of publicly available datasets which require conversion to NWB format (see here for examples).

2) Read, understand and appreciate original publications related to data, convert datasets to NWB format, adding annotations and metadata to facilitate interpretability & reuse of the data by others. Document process to aid others.

3) Make data available via the NWB Explorer on the Open Source Brain repository

Mentors:

Padraig Gleeson (lead), Ankur Sinha

Tags: OSB, NWB, Python, HDF5, data analysis, open access.

Discuss this project on Neurostars

13 MAXIMA

13.1 Maxima Demo and Documentation Tool

Maxima is a system for the manipulation of symbolic and numerical expressions with more than 40-year history. The system is maintained by an active community of users and developers and is incorporated in systems, such as Sage. Maxima is an open source, and its functionality is on par with commercial systems, such as Maple and Mathematica. The main applications of the system are calculus, dynamical systems, and matrix algebra, which are used for example in the context of biophysical modeling.

Project idea: WxMaxima worksheet to Texinfo conversion. WxMaxima is a widely used user interface for Maxima. WxMaxima is a worksheet interface. The user creates a series of cells which contain inputs, outputs, plots, titles and subtitles, and plain text. Worksheets are stored as XML documents within a compressed archive. Texinfo is a documentation system that uses a single source file to produce both online information and printed output. It is primarily designed for writing software manuals. The reference manual of Maxima, which can be searched and displayed in a Maxima session, is composed in Texinfo format. The purpose of this project is to create functions which convert a wxMaxima worksheet to Texinfo. The functions to convert a worksheet will be written in Common Lisp, which is the implementation language for Maxima itself.

Tasks

1. Getting familiar with Maxima and wxMaxmia.
2. GUI implementation and integration

Minimal set of deliverables

- Requirement specification – Prepared by the candidate after understanding the functionality.

- System Design – Detailed plan for the development of the translator and test cases.

Desired skills: experience with Lisp or C++, experience with HTML or Latex
Mentors: Robert Dodier, Maxima project, Dimiter Prodanov,, INCF Belgian Node (backup)

References:
[1] http://maxima.sourceforge.net/

[2] https://www.gnu.org/software/texinfo/

[3] https://wxmaxima-developers.github.io/wxmaxima/

Tags: Maxima, GUI, Lisp, C++, HTML, LaTeX

Discuss this project on Neurostars

14 BIDS project

14.1 EEG/MEG-to-BIDS converter: a GUI application in Python

BIDS (Brain Imaging Data Structure; https://www.incf.org/sbp/brain-imaging-data-structure-bids) is a standard for organising neuroimaging and behavioural raw data. It enables interoperability by allowing multiple tools to analyse a given dataset without extra modifications, and reproducibility, by facilitating reanalysis of the same dataset by multiple research groups. Although originally developed for MRI data analysis, BIDS has also gained popularity for EEG/MEG data analysis, and with more analysis tools supporting BIDS, converting raw data to the BIDS standard has become an important first-step for EEG/MEG data analysis. Unfortunately, however, available tools are often too complicated to use for scientists with limited technical or scripting experience, a situation that is especially prevalent in EEG labs which are less likely to employ informatics support personnel.

One major barrier to user-friendly conversion of data to the BIDS standard is that most tools do not include a graphical interface. Moreover, the few general-purpose BIDS conversion tools that do have a GUI in addition to a command line interface, do not support conversion of EEG/MEG data. The proposed project aims to extend one of these existing BIDS conversion tools to add EEG/MEG support, resulting in a tool that is both easy to use, and as general as possible.

The converter will be cross-platform (Windows, Mac, Linux) for maximum applicability allowing a wide reach of EEG/MEG labs and users. Moreover, it will be integrated into AEDAPT/NeuroDesk (https://github.com/NeuroDesk/neurodesk), a Docker virtual machine incorporating a complete EEG/MEG (as well as MRI) analysis desktop environment. AEDAPT/NeuroDesk is an open-source project supported by the Australian Research Data Commons and actively developed by several universities in Australia.

Scope:

- Extending an existing GUI-based BIDS conversion tool to support EEG/MEG (see list at https://www.incf.org/blog/brain-imaging-data-structure-bids)
- Incorporating continuous integration to ensure backward compatibility
- Enhancing the tool so users can enter additional information, not recorded in the original EEG/MEG raw data files
- Redesigning the tool to allow either bulk or incremental conversion
- Improving support for multiple-session acquisitions of the same individual
- Adding automatic suggestions based on mappings provided by the community

Skill level needed:

- Python GUI programming - intermediate
- Text manipulation (regular expressions) in Python - intermediate
- Familiarity with EEG - advantage
- Familiarity with BIDS - advantage

Tech keywords:

- Python
- PyQt5, Tkinter, or wxPython
- JSON
- Circleci or Github actions
- YAML

Mentors: Oren Civier (Swinburne University of Technology), Steffen Bollmann (University of Queensland), Aswin Narayanan (University of Queensland), Tom Johnstone (Swinburne University of Technology)

Tags: BIDS, neuroimaging, GUI, EEG/MEG, NeuroDesk

Discuss this project on Neurostars

14.2 BCI2000-to-BIDS converter: a GUI application in Python/Matlab/C++

Acquisition of and real-time interaction with electrophysiological signals can be accomplished with many different bio-signal acquisition devices. The BCI2000 general-purpose software platform provides an open-source solution to abstracting a wide

range of bio-signal acquisition devices, synchronizing their signals to a variety of other devices that capture behavior (e.g., eye-trackers, data-gloves, etc.), and closed-loop experiments. BCI2000's large high-quality code base is documented by a comprehensive set of technical references that is available as a wiki at (http://doc.bci2000.org [1]). To date, BCI2000 has been provided to more than 6,000 users worldwide that have produced an extensive body of electrophysiological data.

The Brain Imaging Data Structure specifies a standard folder structure for many different electrophysiological signals, such as EEG, MEG and intracranial EEG, imaging data and is rapidly growing with ongoing proposals to extend BIDS. Metadata are both human- and machine-readable and their fields are prescribed to allow for automated processing. Moreover, there is a large community involved in the development of BIDS, including developers from many labs and analysis packages worldwide. A converter from BCI2000 data to BIDS will connect the many users of BCI2000 to the BIDS community and facilitate sharing of the unique BCI2000 experiments in BIDS.

The objective of this project is to develop the framework along with a graphical user interface (GUI) to convert BCI2000 data into the BIDS format. Achieving this goal will facilitate the dissemination of BCI2000-based data within the growing BIDS community and the open sharing of these data. We anticipate that this project will be structured into: 1) a framework for the conversion from BCI2000-based data format into BIDS data format; 2) implementation of a proof-of-concept command line tool in either Python, Matlab or C++; and 3) expansion of the corresponding command line tools with a graphical user interface to inspect and visualize the data and conversion process. The tools developed within this project will be open-source, developed on GitHub and hosted and maintained within the BCI2000 project and links will be added to the bids-starter-kit.

Skill level needed:

- Python | Matlab | C++ - command line - intermediate
- Python | Matlab | C++/Qt - GUI programming - intermediate
- Familiarity with data structures - advantage
- Familiarity with BIDS - advantage

Tech keywords:

- Python, PyQt5, JSON, Matlab, C++, Qt
- BCI2000, BIDS
- GitHub

Mentors : Peter Brunner (Washington University in St. Louis), Dora Hermes **@Dora_Hermes** (Mayo Clinic, Rochester)

Co-Mentor: Markus Adamek (Washington University in St. Louis), Max van den Boom (Mayo Clinic, Rochester)

Tags: BCI2000, BIDS, electrophysiology, data formats, GUI

15 Turing Institute projects

15.1 The Turing Way: A how-to guide to data science

The Turing Way is an open-source, community-led and collaboratively developed "book project" on making data research accessible for a wider research community (https://the-turing-way.netlify.com). We bring together individuals from diverse fields and expertise to develop practices and learning resources that can make data research accessible and easy to understand. Our community members are researchers, engineers, data librarians, industry professionals, and experts in various domains, at all levels of seniority, from all around the world. They collaborate in the project to develop chapters by compiling best practices, tools and recommendations used by the researchers and data science communities worldwide.

Technical details
All questions, comments, recommendations and discussions are facilitated through an online GitHub repository (https://github.com/alan-turing-institute/the-turing-way). The online book with multiple guides is hosted as a Jupyter Book (https://github.com/jupyter/jupyter-book/) at https://the-turing-way.netlify.com. Jupyter Book formats markdown files and Jupyter notebooks as static HTML making them easy to read. When a notebook is included in the book, the static page includes a link to an interactive version of the notebook via Binder (https://mybinder.readthedocs.io). Additional styling of the front end is possible by providing a CSS file that handles it across the entire book.

GSoC project plan
Since the project's launch in 2019, about 250 contributors have so far co-authored more than 140 subchapters and community documents on reproducible research, communication, collaboration, project design and ethics. As the number of chapters continues to increase, it is important for us to offer appropriate ways for our readers to discover relevant and desired content in the book based on their topics of interest and skill levels. A GSoC contributor will help us in developing such a feature in the project that provides an enhanced filtering or search functionality in the book. Based on their interest and availability, they will have the possibility to contribute to the development of Python scripts and GitHub actions to enhance the project workflow, chapter development, community engagement and the overall interactivity in the book. They will be provided with appropriate guidance and the opportunity to work in a positive working environment. They will be fairly acknowledged for their contributions to the project.

Expected outcome

1. An enhanced version of The Turing Way's online book by providing an advanced filtering or search feature that will allow our readers to navigate through the book and find the information they are looking for.
2. Upstream contribution of newly developed features to the Jupyter Book project will be a plus and supported by the mentors.

Required skills

1. Python programming skills (intermediate to advanced)
2. Basic web-development skill required to work with Jupyter Book
3. Experience working in distributed communities, using git and GitHub

Optional skills

- Interactive visualisation of small datasets
- Experience collaborating on data science or quantitative research projects at any level
- JavaScript skills (for front end development)

Possible mentors

- Lead mentor: Malvika Sharan (msharan@turing.ac.uk)
- Co-mentors: Batool Almarzouq and Martina Vilas

Tags: TuringWay, handbook, Javascript, Python, Jupyter

Discuss this project on Neurostars

15.2 AutSPACEs: a co-created citizen science platform

AutSPACEs is a co-created, participatory, citizen-science platform which will be used to investigate how sensory processing differences affect the daily experiences of autistic people. By collecting together diverse real-world experiences, contributed by autistic citizen scientists, we will build a dataset which can be used to make recommendations to improve autistic people's lives. The platform will also provide a safe and inclusive space to share stories, strategies, and build a community of contributors, which will be facilitated by a community-led, co-created moderation process and code of conduct. Supported by extensive research, and the collaborative input of autistic people and their families, we are currently building up a working prototype.

Integral to the platform is a fine-grained consent model which allows users to specify who they would like their data to be shared with, and for what purposes. The platform will also grant each user dynamic control of their data. This is supported by integration with a backend infrastructure provided by Open Humans. Open Humans is a foundation which enables research participants

to have awareness and understanding of, as well as an agency over, their own data. Supporting data agency and transparent data management is critical to the platform's design.

In summary, AutSPACEs enables rich, scalable, community-led research to be conducted in a way that is both transparent and encourages agency. Furthermore, as an openly licensed and publicly documented research tool, others can freely re-use, build on or adapt it to investigate other research questions.

Technical details

The platform is being built openly on GitHub. Data, documentation, details, and community recommendations, as well as supporting documents, protocols, and more, are all publicly available on our project management repository. The platform development takes place in a separate, linked repository: AutSPACEs using the Python/Django web development framework. Users will be able to share experiences through a website, which will then be deposited into a backend database supported by Open Humans. All stages of the project are being conducted in collaboration with autistic people, and we are committed to being open and transparent throughout.

GSoC project plan

We are seeking a GSOC mentee with web development skills to work on building the platform with a researcher and an experienced developer. This will involve elements of UX/UI design focussing on the needs and preferences of autistic collaborators. We are able to offer a unique opportunity for a GSOC mentee to grow their technical proficiency and collaborative skills in the context of innovative, impact-focussed research. We will provide our mentee with a firm grounding in the emerging best practices of open source development and research. They will also be welcomed into the The Turing Way community. Depending on their preferences and interests, they will be invited to join the *The Turing Way* community who are building an open source book to improve data science practices, to work with autistic collaborators on participatory design, and to help promote the platform to onboard new developers.

Expected outcome

Over the course of the 10 week period, we expect the mentee to progress the prototype through web development and design and to work with autistic community members to implement search and moderation processes.

Required skills

- Basic version control using GitHub
- Python
- Test-driven development

- Skills or interests in UI/UX design
- Basic web development: ability to design and develop new web links and pages
- Clear, concise, and responsive communication and documentation
- Respect for, and the ability to work with, diverse collaborators.

Optional skills

- Django
- Containers, docker
- Continuous integration
- Deployment and hosting
- Lived experience of autism

Possible mentors

- Lead mentor: Georgia Aitkenhead (gaitkenhead@turing.ac.uk)
- Co-mentor: James Kim (hyongsupkim@gmail.com)

Tags: GitHub, Python, Django, UI/UX design, web development

Discuss this project on Neurostars

16 BRIAN simulator projects

16.1 Supporting simulation based inference with the model fitting toolbox

The brian2modelfitting project provides a toolbox that enables users to conveniently fit their single-cell Brian models to experimental data. This toolbox leverages global optimization methods from the nevergrad and scikit-optimize libraries, as well as local, gradient-based methods from lmfit and scipy. Recently, simulation-based inference has been established as an alternative approach that improves over such methods in several ways. In particular, it does not only result in a single parameter set providing the "best fit", but instead provides an estimate of the full posterior distribution over parameters. The aim of the project is to support this kind of approach in the brian2modelfitting toolbox by linking it to the sbi package created by the Macke lab: https://www.mackelab.org/sbi/

Skills: Python programming, experience with optimization/machine learning/computational neuroscience modelling helpful

Mentors: Marcel Stimberg, Romain Brette

Tags: BRIAN, Python, optimization, machine learning

Discuss this project on Neurostars

16.2 Make Brian run in the web browser via Pyodide

Brian simulations can currently only run in a web browser if a server runs the code. Due to recent improvements in browser technology such as WebAssembly, browser are however capable of executing computationally complex code themselves, directly on the client machine.

The Pyodide project leverages this technology to run the Python scientific stack in the browser, and we have shown with a prototype project that this can be extended to support Brian. The specific aims of this project are to:

- create a robust Pyodide implementation of Brian2 that gets updated automatically with new releases of Brian 2
- create convenient methods to display simulation results in the browser
- investigate integrating this with the current Brian documentation and/or on a dedicated "showcase" website

Skills: Python programming and basic web development (HTML, JavaScript), experience with WebAssembly helpful

Mentors: Marcel Stimberg, Dan Goodman

Tags: BRIAN, Python, HTML, JavaScript

Discuss this project on Neurostars

16.3 Update documentation infrastructure

The Brian 2 simulator provides extensive documentation, examples, and tutorials. Examples and tutorials are provided in different formats: as websites, downloadable files, and jupyter notebooks which can also be executed on the binder infrastructure. Generating these formats is done partly manual, i.e. by running a script on the developer's local machine, which is inconvenient and error-prone. In addition, the examples and tutorials are not well integrated with the rest of the documentation, which could be improved by replacing some of the Brian-specific scripts by the solutions provided by the Python ecosystem.

Specifically, this project aims to:

- Automatize the generating of example/tutorial content (e.g. by using GitHub Actions)
- Run the examples/tutorials regularly to catch errors and incompatibilities introduced by changes in Brian
- Remove some of the Brian-specific scripts and classes by migrating to established packages such as napoleon or sphinx-gallery
- Improve the presentation of the examples/tutorials and their integration with the rest of the documentation (e.g. with sphinx-gallery)

Skills: Python programming, experience with sphinx and CI infrastructure such as GitHub Actions helpful
Mentors: Marcel Stimberg, Dan Goodman

Tags: BRIAN, Python, documentation, sphinx

Discuss this project on Neurostars

16.4 OpenMP for synaptic propagation

Brian can parallelize simulations over multiple processor cores by making use of the OpenMP framework. However, in its current state Brian does not yet make full use of the parallelization potential, in particular for synaptic propagation.

The aim of this project is to improve the OpenMP support, by:

- analyzing the connectivity structure and type of synaptic interaction to decide whether trivial parallelization is safely possible
- benchmarking and implementing parallelization approaches for non-trivial situations

Skills: C++ and Python programming, experience with OpenMP or other parallelization techniques helpful

Mentors: Marcel Stimberg, Dan Goodman

Tags: BRIAN, Python, C++

Discuss this project on Neurostars

17 AnalySim project

17.1 AnalySim: a data sharing and notebook analysis platform written in Angular and ASP.net with C#

AnalySim is a data sharing and analysis platform that seeks to simplify the visualization of datasets. With Analysim researchers can collaborate by hosting their data and publishing their analysis notebooks to the world, or browse through multiple user generated projects.

AnalySim aims to be a data sharing and hosting resource for crowdsourced analysis of a specific type of dataset: one where many parameter combinations need to be tested and measurements are recorded for each instance.These datasets are very useful in mathematical modeling of natural phenomena, such as in computational neuroscience. We provide easy sharing, analysis, visualization, and collaboration capabilities on these datasets.

Project is still in progress and a draft is available at: www.analysim.tech

Main Technologies: Angular, Typescript, Bootstrap, ASP.Net Core, C#, PostgreSQL,

Technologies for analysis notebooks: JavaScript, ObservableHQ, D3.js, Vega, Plotly

Tags: Angular, Typescript, Bootstrap, ASP.Net Core, C#, PostgreSQL,J avaScript, ObservableHQ, D3.js, Vega, Plotly

Lead Mentor: Cengiz Gunay, cgunay@ggc.edu

Co-Mentor: Anca Doloc-Mihu, adolocm@gmail.com

Discuss this project on Neurostars

18 Decentralized Autonomous Organizations for Science project

18. 1 Securing Neuroimaging BIDS Datasets Stored on Decentralized File Sharing Networks with Ethereum Public Keys, Git Annex and IPFS

Problem. A significant barrier to open science practice is the sharing and accessibility of neuroimaging datasets. The interplanetary file system (IPFS) addresses this barrier with peer-to-peer sharing of data and storage on distributed networks such as Bitorrent, Filecoin, and

Cloudflare. However, files stored on the distributed IPFS hash table are public by default, making it inappropriate for sharing protected health information, or confidential data.

Project. Ethereum public keys associated with identity following the decentralized identifier standard (DID) for web3.0 can be used to encrypt datasets such that only those with access privileges can access the data on the IPFS network. This project involves the creation of an open-source toolkit to encrypt datasets with Ethereum public keys prior to storage on IPFS. Contributors should have a working knowledge of Java, Python, Git, or Unix.

Project Lead + Mentor: Shady El Damaty, Ph.D.

This project will build off of the 2020 decentralized science brainhack project. More information can be found on the github repository.

Tags: federated identities, cloud computing, ethereum, blockchain, data security, Java, Python, Git, Unix

Discuss this project on Neurostars

18. 2 Tokenizing and Deploying BIDS Apps on Distributed Confidential Cloud Resources for Anonymous Computation with Ocean Protocol and iExec RLC

Problem. The analysis of massive neuroimaging datasets is made difficult due to 1) the accessibility of adequate storage and computational resources, 2) vast number of branch points in analysis pipelines for preprocessing and model training / validation, 3) maintaining the de-identification of data while building algorithms for predicting individual traits is a catch-22 that potentially challenges tenets set in GPDR and other regulations regarding privacy. Standardization of pipelines and dataset specifications via the open source BIDS specification has addressed issues of dataset reproducibility and sharing. However, issues regarding access to adequate cloud resources and privacy preservation of individuals remain difficult challenges.

Project. This project will address these remaining challenges by tokenizing BIDS-compliant datasets and apps on the Ethereum blockchain to tap into the vast storage of the peer-to-peer IPFS network and into anonymous computing using the Ocean protocol and iExec confidential distributed cloud computing resources. The summer scholar will work on a python/javascript application for converting a BIDS-app and associated BIDS-compliant dataset into a tokenized object compatible with the Ocean Protocol for exchange on data marketplaces and with iExec for anonymous cloud computing.

Project Lead + Mentor: Shady El Damaty, Ph.D.

This project will build off of the 2020 decentralized science brainhack project. More information can be found on the github repository.

Tags: federated identities, cloud computing, ethereum, blockchain, data security, Java, Python, Git, Unix

Discuss this project on Neurostars

19 DIPY projects

19.1 Population-based MRI Template Creation

Implement a method to create a population-based MRI template. Given an input of several subjects' MRI, create one standard template MRI for the population. The method will utilize the MRI registration framework available in DIPY.

Steps:

- Understand MRI data
- Implement template creation method
- Write DIPY workflow of the method
- Test it on different data sets

Difficulty: Intermediate

Skills required: Python, Image Registration, Image Processing.

Mentors: Bramsh Chandio, Shreyas Fadnavis, and Jong Sung Park

Tags: DIPY, neuroimaging, Python, Image processing

Discuss this project on Neurostars

19.2 Population-specific Tractography Bundle Atlas Creation

Description: Implement a method to create a population-specific Tractography Bundle Atlas. Given an input of several subjects' segmented white matter tracts, create one standard atlas of bundles for the population. The method will utilize the streamline-based registration framework available in DIPY.

Steps:

- Understand Diffusion Tensor Imaging and Tractography data

- Implement Bundle Atlas creation method
- Write DIPY workflow of the method
- Test it on different data sets

Difficulty: Hard

Skills required: Python, Registration.

Skills preferred: Experience with Diffusion Tensor Imaging

Mentors: Bramsh Chandio, Shreyas Fadnavis, and Jong Sung Park

Tags: DIPY, neuroimaging, Python, atlasing, tractography

Discuss this project on Neurostars

19.3 Extend DIPY Horizon workflow for Visualization

Description: Extend dipy_horizon workflow by adding more options for the visualization of the diffusion data. DIPY Horizon is a workflow that enables to visualize diffusion data such as dMRI, tractograms, white matter bundles, and more from the command line. This project requires student to add support for different types of file formats and visualizations in the horizon workflow.

Steps:

- Add support to visualize orientation distribution functions (odfs) generated from diffusion data
- Create an option in the horizon workflow to project anatomical measures such as functional anisotropy (FA), mean diffusivity (MD), etc on the white matter tracts and visualize them
- Add region-of-interest (ROI) capacity for streamline filtering in Horizon.
- Add Qt functionality in dipy_horizon workflow

Difficulty: Intermediate

Skills required: Python, VTK, Qt.

Mentors: Bramsh Qamar Chandio, Shreyas Fadnavis, and Jong Sung Park

Tags: DIPY, neuroimaging, Python, dMRI, VTK, Qt

Discuss this project on Neurostars

19.4 DIPY-Tract-or-Treat: DIPY DTI Post-Processing Pipeline

Description: DIPY has several methods for reconstruction, tractography, bundle extraction, and tratometry. The idea of this project is to combine them all into one command-line interface that does reconstruction, tractography, bundle extraction, and bundle analytics. Users will have the option to select among different methods and design their pipeline from the list of available options.

Steps:

- Understand DIPY and its workflows thoroughly
- Create a command-line interface (workflow) to create a pipeline of different existing methods.
- Test in on data

Difficulty: Intermediate

Skills required: Python

Skills preferred: Experience with Diffusion Tensor Imaging

Mentors: Bramsh Qamar Chandio, Shreyas Fadnavis, and Jong Sung Park

Tags: DIPY, neuroimaging, Python, CLI

Discuss this project on Neurostars

20 BrainBox projects

20.1 Brain Badges: Online certification for collaborative annotation and segmentation of brain imaging data

Problem: A wealth of open neuroimaging and histological data is publicly available, including data for subjects with different psychiatric disorders, different developmental stages, even different species. However, the data is not easy to find, is curated and analysed locally, in redundant, often wasteful and opaque ways.

Vision: Our vision is to create an interactive space where academic and citizen scientists can work together to curate public neuroimaging and histological data. The fruits of this common effort should be open, easy to find, access, and re-use. BrainBox and MicroDraw are our Web applications implementing this radical open science vision. BrainBox allows users to view, curate and annotate any neuroimaging dataset available on the Web, currently >13,000, and provides real-time interactive tools for manually segmenting and editing brain regions (3 min

video about BrainBox: https://www.youtube.com/watch?v=kwsLoVKnw24). MicroDraw provides similar functionalities for high-resolution histological data.

Aim: For researchers to embrace this vision we need a way to assess and develop contributor's expertise. Researchers leading a BrainBox or MicroDraw project, for example, need confidence about the quality of the results, and users need a clear path to begin contributing to a project.

We propose to develop interactive training and certification modules: the Brain Badges. Using the training module, researchers will be able to explain the tasks that are required. Users will be proposed examples to practice, and then the evaluation module will test them on new data. Users will earn a badge for their achievement, which will be added, for example, to their BrainBox or MicroDraw profiles. Through the Brain Badges researchers will be able to recruit collaborators, weigh their contributions based on expertise, improving their confidence in the results. Users will have a precise idea of what is required to contribute to a project. The Brain Badges will increase their reputation, motivating them to keep learning.

Scope: We will design and code Brain Badges based on the Open Badges standard (https://openbadges.org). We will build 3 sample training and certification modules: data quality annotation, manual segmentation, and automatic segmentation correction.

Skills needed: Javascript, jQuery, Vuejs, Mocha, Chai, Webpack, HTML and CSS, Nodejs, MongoDB.

Mentors: Katja Heuer & Roberto Toro.
Lead mentor: Katja Heuer.
Co-mentor: Roberto Toro

Keywords: Javascript, jQuery, Vuejs, Mocha, Chai, Webpack, HTML and CSS, Nodejs, MongoDB, BrainBox, MicroDraw

Tags: Javascript, jQuery, Vuejs, Mocha, Chai, Webpack, HTML and CSS, Nodejs, MongoDB, BrainBox, MicroDraw

Discuss this project on Neurostars

20.2 Brain Histology Metadata DB: Building an online database for storing, curating and sharing metadata from diverse collections of histological data

Keywords: database, metadata, RESTful API, GUI, Javascript, Vuejs, Nodejs, MongoDB, HTML, CSS

Problem: Brain histology provides unique information on the cellular structure of the brain. Histological data has been collected for more than a century, and is available for a wide variety of species and developmental stages. The physical slides require lots of effort for being digitised at high resolution, and thus, usually only selected slides, representing only specific regions and specimens within much larger collections have been scanned by individual researchers for their own projects. This has led to a wealth of digitised histological data being spread across several researchers and labs, with no coherent structure, common identifiers, or a method to see, access, and annotate the data.

Histological data is challenging to visualise, analyse and share, and often requires time intense manual curation and annotation. With the lack of a shared database, researchers find themselves redoing manual annotations redundantly, instead of being able to build an increasingly detailed picture of the brain microstructure together.

Vision: We would like to build an online database for storing, curating, sharing, accessing, and annotating metadata from several brain histology datasets, alongside with the real digitised data (images and text) whenever it is available. In a joint effort with anatomists worldwide we will build a searchable database for the primates of the Stephan and Zilles Collection, one of the most important histological data collections for comparative neuroanatomy. It represents the world's largest collection of whole brain histological slides from diverse primate species, as well as other animals. We would like to make this data available to the scientific community in a structured way in order to reduce redundant efforts, allow scientists to tackle analyses and challenges collaboratively in a coordinated fashion.

Aim: We have access to records from different histological collections. The aim of the project will be to organise the data, develop a database, a RESTful API to query the data programmatically, and a Web interface to allow researchers to browse and query the database. Whenever digitised histological data becomes available, it will be made available with a link to our tool MicroDraw to visualise it and segment it collaboratively.

Scope: Create a database, API and UI for metadata on brain histology collections.

Skills needed: Javascript, Nodejs, Vuejs, HTML, CSS, MongoDB, RESTful API.

Lead mentor: Alexandra de Sousa.
Co-mentors: Katja Heuer & Roberto Toro.

Tags: database, metadata, API, GUI, Javascript, Vuejs, Nodejs, MongoDB, HTML, CSS,RESTful API

Discuss this project on Neurostars

20.3 Improving end-to-end testing and code documentation for BrainBox

Neuroimaging data requires a large effort of curation and annotation.
These tasks are lengthy, and often performed independently by different research groups. BrainBox is a tool providing a real-time interactive platform to perform these tasks. As BrainBox increases in complexity, the necessity of a more extensive testing suit is required to ensure its robustness and reliability. A framework is already in place, with a battery of unit tests as well as end-to-end tests, however, many aspects of BrainBox are still not covered.

Challenge: Extend the coverage of the unit and end-to-end tests of BrainBox, integrate them in the continuous integration of the platform, and implement continuous deployment.
Vision. Every time a collaborator proposes a modification to BrainBox, the test suit should be able to determine whether the changes can be safely merged. The increased robustness of BrainBox should reassure future users on the reliability of the platform, and encourage our community to work distributedly and collaboratively.

Skills needed: DevOps, Mocha, Chai, Puppeteer, Websockets, CircleCI, Git, Github, Javascript, Vuejs, jQuery

Keywords: Javascript, HTML, CSS, Mocha, chai, puppeteer, CircleCI, Websocket, node.js and mongodb for server and database, BrainBox

Lead mentor: Roberto Toro
Co-mentor: Katja Heuer

Tags: Javascript, HTML, CSS, Mocha, chai, puppeteer, CircleCI, Websocket, node.js MongoDB, BrainBox, Vuejs, jQuery

Discuss this project on Neurostars

21 HDNet projects: Developing a Python-based codebase for analyzing stimulus prediction capabilities of neurons

21.1 Developing a Python-based codebase for analyzing stimulus prediction capabilities of neurons: improve the neural Maximum Entropy model

There are good theoretical reasons to believe that neurons learn to predict their input, but fewer experimental tests of this hypothesis. The mentor and co-mentor on this project have

developed new methods for assessing the predictive capabilities of a large number of neurons from data using both information theoretic and a Bayesian framework. These methods have not yet been optimized and integrated into an existing, ever-growing codebase that will, upon release, allow other groups to easily assess the predictive capabilities of their neural populations.

One of the metrics for assessing stimulus prediction is based on the fitting of a Maximum Entropy model that probabilistically describes stimulus activity and the corresponding neural response. The student on this project is expected to be familiar with Python and Matlab, and either familiar with or eager to learn new techniques for Maximum Entropy model fitting. He or she will improve the neural Maximum Entropy model fit to data in two steps:

- functions for Maximum Entropy model validation using pre-written Monte Carlo methods will be integrated into in the codebase
- hyperparameters of Minimum Probability Flow, a powerful and relatively new algorithm for Maximum Entropy model fitting, will be optimized so that the model validation is improved

The result of this project will be a state-of-the-art, computationally efficient approach to parameter fitting of Maximum Entropy models that will on its own be useful to neuroscientists. The result of this project may be therefore integrated into HDNet (https://github.com/team-hdnet/hdnet) in addition to being released as part of the stimulus prediction package.

Mentor: Sarah Marzen
Co-mentor: Joost le Feber

Tags: Python, MATLAB, parameter fitting, HDNet

Discuss this project on Neurostars

21.2 Developing a Python-based codebase for analyzing stimulus prediction capabilities of neurons: improve stimulus prediction

There are good theoretical reasons to believe that neurons learn to predict their input, but fewer experimental tests of this hypothesis. The mentor and co-mentor on this project have developed new methods for assessing the predictive capabilities of a large number of neurons from data using both information theoretic and a Bayesian framework. These methods have not yet been optimized and integrated into an existing, ever-growing codebase that will, upon release, allow other groups to easily assess the predictive capabilities of their neural populations.

One of the metrics for assessing stimulus prediction is based on the predictive information, the shared information between present neural response and future stimulus. The student on this

project is expected to be familiar with Python, Matlab, and TensorFlow/Keras, and either familiar with or eager to learn new techniques for mutual information estimation in the undersampled limit. He or she will implement existing state-of-the-art algorithms for such estimation and add such algorithms to the existing codebase.

The result of this project will be a state-of-the-art compilation of predictive information estimation methods.

Mentor: Sarah Marzen
Co-mentor: Joost le Feber
Discuss this project on Neurostars

22 Mobile EEG Neurotechnology

22.1 Cognitive Neuroscience Experiments with Mobile EEG Neurotechnology

EEG-Notebooks is a Python-based library developed by the NeuroTechX community for running cognitive neuroscience experiments with low-cost mobile EEG devices. It is intended as a tool for research, medical applications, and education - with the goal of making cognitive neuroscience and neurotechnology more accessible, affordable, and scalable. EEG-Notebooks is built around the standard Scientific Python and Neuroimaging-in-Python software stack (numpy, scipy, pandas, scikit-learn, MNE), and uses Psychopy for stimulus delivery and programming of experimental paradigms.

This GSoC project will focus on extending the experiment repertoire of eeg-notebooks, by porting the high-quality research paradigm implementations (face perception, auditory oddball, visual search, word pair judgment, flanker task) from the ERP-core platform. Additionally, the project will also develop novel statistical data analysis and machine learning analyses of various datasets shipped with the library.

Candidates should have experience with Python, data analysis, and EEG and/or behavioural/psychological experiments. Access to EEG hardware is not essential. The project will provide excellent experience and training for students interested in pursuing research in human neuroimaging, cognitive and clinical neuroscience, and brain-computer interfaces.

Lead Mentor: John Griffiths
Co-Mentor: Morgan Hough

Tags: Python, EEG, Neurotechnology, Data Analysis, Machine Learning, Cognitive Neuroscience, EEG-Notebooks, electrophysiology

Discuss this project on Neurostars

23 NeuroML project

23.1 Re-creating the Leech Heartbeat Network Model Tutorial using the Neuron Simulator in Python and NeuroML

The Calabrese Lab 8-cell Leech Tutorial that is described by Hill et al 2001 has been a staple for teaching computational neuroscience at Emory University for many years, and it has also been used in various summer courses. This tutorial is not only a fully constructed 8-cell circuit that can generate heart rhythms, but also a great teaching tool thanks to the visual interface where a student can turn on synaptic connections or change maximal ionic conductances. However, the original tutorial has been developed using the now obsolete Genesis simulator. Unfortunately, running the Genesis simulator nowadays requires complicated software set up that prevents many non-technical students from accessing the tutorial. We had previously started porting the tutorial to a more modern format that can be executed through a web browser (see 8-cell Leech Heartbeat Network Model Tutorial), increasing the accessibility of this classic tutorial for teaching and research purposes. At this summer's GSoC, we would like to finish this port and make the tutorial available. We selected the Neuron simulator language for running the model using NeuroML and Python as the description languages.
Neuron simulator, Python, NeuroML

Lead Mentor: Cengiz Gunay (cgunay@ggc.edu)
Backup Mentor: Padraig Gleeson (p.gleeson@ucl.ac.uk)
Tags: computational neuroscience, NeuroML, Python, NEURON

Discuss this project on Neurostars

24 lab.js projects

24.1 Collecting Behavioral and Neuroscientific Data with Shiny (in R) and lab.js

lab.js is an open-source JavaScript library and GUI for building experiments in the neuro- and behavioral sciences (https://lab.js.org). Researchers worldwide use it to construct and run

experiments online and in the laboratory, as well as for teaching research methods and experimental design.

In this project, we would like to extend the capabilities for data collection to the Shiny framework for web applications in R (https://shiny.rstudio.com/), which is a great, open-source library for building interactive, data-driven websites as well as collecting data, for example via questionnaires. Shiny also makes it easy for researchers to deploy websites on their own infrastructure as well as commercial providers such as shinyapps.io (https://shinyapps.io/). The goal we'd like to achieve with you will be to enable running behavioral experiments, and to collect data, via lab.js and Shiny.

Your task (as we envision it) will be to build a prototype lab.js study that runs via Shiny, document how it works, and work with us to automate the process of generating and deploying more studies like it. The end goal will be to provide a one-click study export to Shiny, similar to the deployment options that lab.js provides already (https://labjs.readthedocs.io/en/latest/learn/deploy/). Ideally, you're broadly familiar with R, have some interest or experience in web development (via Shiny), and are prepared to pick up some JavaScript to interact with lab.js. Of course, we would be thrilled to introduce you to the project, and support you in your work.

Please do not hesitate to get in touch – we would be glad to answer any questions you have, and help sketch out a formal proposal. You're warmly invited to say hello in our community (https://lab.js.org/resources/support/) – we'd love to hear from you!

Lead mentor: Felix Henninger (University of Mannheim / Helmholtz AI / LMU Munich)

Backup/Co-mentor: Yury Shevchenko (iScience group, University of Konstanz)

Skills: Intermediate R, basic familiarity with web development in JavaScript, ideally some React (or interest in picking it up)

Tags: R, lab.js, JavaScript, Shiny, GUI

Discuss this project on Neurostars

24.2 Laboratory Hardware Support for lab.js via the Lab Streaming Layer

lab.js is an open-source JavaScript library and GUI for building experiments in the neuro- and behavioral sciences (https://lab.js.org). Researchers worldwide use it to construct and run experiments online and in the laboratory, as well as for teaching research methods and experimental design.

In this project, our goal will be to expand the library's reach to laboratory equipment such as EEG and eye-tracking hardware, biosensors, and potentially neuroimaging systems. This will allow for the integration of further data sources in studies built with lab.js, and more types of research to benefit from the project. The technical conduit for this work will be the Lab Streaming Layer (LSL, https://github.com/sccn/labstreaminglayer), which provides a widely supported standard for communication between different instruments, and a framework for collecting time series from a multitude of sensors. Your task in this project (as we envision it) will be to establish communication between a study running in lab.js, and the LSL. We are open to your ideas and suggestions, but one way of achieving this would be to expose the LSL through Electron (https://www.electronjs.org/), a framework for creating desktop applications with JavaScript that provides wide-ranging capabilities for low-level access to the operating system.

As a stretch goal, we hope to benchmark the result with you, and establish the temporal resolution and alignment the system provides. To work with lab.js and Electron, ideally you're ready to develop in JavaScript, but we would be thrilled to introduce you to the project, and of course support you in your work.

Please do not hesitate to get in touch – we would be glad to answer any questions you have, and help sketch out a formal proposal. You're warmly invited to say hello in our community (https://lab.js.org/resources/support/) – we'd love to hear from you!

Lead mentor: Felix Henninger (University of Mannheim / Helmholtz AI / LMU Munich)

Backup/Co-mentor: Yury Shevchenko (iScience group, University of Konstanz)

Skills: Intermediate JavaScript, ideally some Electron (or interest in picking it up)

Tags: lab.js, JavaScript, Electron, GUI

Discuss this project on Neurostars

25 LORIS open data platform for neuroscience research

25.1 Contribute to LORIS

Interested in data platforms, open science, visualization, or neuroscience studies?

Projects can be proposed to match the applicant's interests, background and strengths, and developed with input from our team. Some examples from past INCF Google Summer of Code projects include: new module development, automated testing, API development.

Read on for important information about LORIS, try it at Demo.loris.ca, and look at our GitHub Issues github.com/aces/loris/issues for ideas.
Then email your CV and 2 tentative project ideas to **Lead Mentor** Christine Rogers <christine.rogers@mcgill.ca>.

About LORIS
LORIS (website: Loris.ca | github.com/aces/loris) is an open-source database for neuroscience research projects and Open Science initiatives, used by researchers in 22 countries. LORIS' open-stack web platform is used by scientists to collect and curate, analyze and share data - including brain scans, genetic data, psychological tests, wearables, and more. 3D visualization and advanced data tools provide a dynamic workflow environment for complex neuroinformatics research.

LORIS currently hosts datasets such as the UK Biobank and and the BigBrain 3D Atlas (bigbrain.loris.ca) - a high-resolution model of the human brain.
Visit LORIS.ca or github.com/aces/loris for more information - Try LORIS at Demo.loris.ca

Working with LORIS
LORIS runs on linux, mysql, javascript/React and php, with a RESTful API and NoSQL querying engine. LORIS developers work on this open stack with ubuntu and git, and we collaborate extensively on our team GitHub at github.com/aces/loris.

LORIS is made by a diverse team of 20 developers currently working remotely and based at the McGill Centre for Integrative Neuroscience at McGill University in Montreal, Canada.
We especially encourage those with background or interest in neuroscience, medicine or psychology research, bioinformatics, or image/signal processing to apply.

Tags: LORIS, JavaScript, React, REST, NoSQL, databasing, imaging, platform, open science, MRI, EEG

Discuss this project on Neurostars

26 Eye-tracker project

26. 1 Eye-tracker based on a convolutional neural network in Python + TensorFlow/Pytorch

Current eye-trackers generally rely on previous-generation computer-vision algorithms and the best ones are also expensive and closed-source. A recent publication from Google Research has shown that it is possible to obtain very good performance using a simple convolutional neural network (CNN) running off a simple mobile phone camera. The details of the CNN have been made available, but the actual implementation is not available. The goal of the project is to implement this algorithm in an open-source package and then explore various extensions including a) incorporating head-position estimation for eye-in-head measurements; and b) extending the algorithm to higher sampling-rates and incorporating filtered estimates of eye-position that take the time-series of previously estimated eye-positions into account.

Mentor: Suresh Krishna

Tags: eye-tracking, Python, PyTorch, TensorFlow, computer vision

Discuss this project on Neurostars

27 Brain Dynamics Toolbox project

27.1 Brain Dynamics Toolbox for Matlab: Contribute new models to the collection

Computational neuroscience seeks to understand the brain by modelling the behaviour of neurons (brain cells) using nonlinear dynamical systems. Likewise, computational cardiology uses nonlinear dynamics to understand the behaviour of cardiac myocytes (heart cells). In both cases, the nonlinear systems produce surprisingly complex behaviours that offer insights into the underlying electrophysiology.

The Brain Dynamics Toolbox (https://bdtoolbox.org) is an open-source toolbox for simulating nonlinear dynamical systems in Matlab. It allows user-defined models to be rapidly prototyped in an intuitive graphical application where interchangeable solvers and plotting tools can be applied with no additional programming effort. This frees the researcher to focus on the core concepts with minimal implementation burden.

The Brain Dynamics Toolbox currently ships with approximately 40 example models, ranging from point models of neurons to spatially-extended neural fields. These examples serve as both teaching tools and starting points for new models in research. Many of the examples are based on textbook models and published papers in computational neuroscience (e.g. Hodgkin-Huxley action potential; Wilson-Cowan neural masses). Others are based on classic models from dynamical systems (e.g. wave equation; the damped-and-driven pendulum; Brownian motion). See https://bdtoolbox.org/p/gallery for a preview.

The aim of this Google Summer of Code project is to contribute more models to that collection.

Students are free to implement any model of their choosing from computational neuroscience, computational cardiology or dynamical systems theory. These may be textbook examples or models from published papers. Guidance will be provided in choosing an appropriate model. It is feasible that students will be able to implement several models in the time available. In doing so they gain practical programming experience in Matlab as well as exposure to theoretical problems in their chosen field. The students can publish their finished models on the Zenodo digital archive (https://zenodo.org) where they will be credited with authorship and assigned a digital object identifier (doi) for their work.

Requirements: Basic programming experience with Matlab is recommended. Training will be provided for programming dynamical systems.

Lead Mentor: Stewart Heitmann, Creator of the Brain Dynamics Toolbox and Computational Scientist at the Victor Chang Cardiac Research Institute.

Co-Mentor: Adam Hill, Head of Computational Cardiology at the Victor Chang Cardiac Research Institute

Tags: Matlab, brain dynamics, BDToolbox, modeling, computational neuroscience computational cardiology, nonlinear dynamical systems

Discuss this project on Neurostars

Available mentors for student-originated projects

Anibal Solon Heinsfeld, Graduate Research Assistant
The University of Texas at Austin | Computer Science & Department of Diagnostic Medicine @ Dell Medical School

Skills: fMRI, deep learning, machine learning, cloud computing, python, javascript