GSOC 2022 Ideas list for INCF

Full-time projects (350 h): 22
Flexible projects (175/350 h): 16
Half-time projects (175 h): 13
Total: 51

Unattached mentors open to take on student-proposed projects:

Salman Khan, India
past INCF GSoC student in GSoC 2020 (see project report)

> My experience and skills revolve around web development, automation , time series data, data-analytics and CLI tools. I hold good command over Python, JavaScript, NodeJS, ReactJS, Firebase, SQL and GIT and I would be happy to support any project that requires these skill sets.

One of our long-time GSoC mentors, Bradley Alicea, is applying as a separate organization this year. Please check out his project ideas on Neurostars:

Orthogonal Research Lab

- A Collective Cognition Model for AI Ethics (350 h)

OpenWorm

- GNNs as Developmental Networks (175 h)
- Digital Microsphere (175 h)

---

1. Disimpy projects

Disimpy is a GPU-accelerated diffusion-weighted magnetic resonance simulator that is useful in the development and validation of neuroimaging methods. Disimpy is written in Python, making the source code very approachable to researchers with little or no prior experience in GPU-computing.

There are several possible projects depending on the skill level, time commitment, and interests of the student, such as (but not limited to):

1.1 Improving GPU-accelerated Monte Carlo simulations in Disimpy: Adding new features (175/350 h)

Adding new features such as new substrates (e.g., analytical geometries or triangular meshes) or new dynamics (e.g., permeability or flow). Skill level: Novice to advanced.

Required skills: Python, NumPy.

Nice-to-have skills: 3D modelling for advanced projects.

Time commitment: half-time or full-time.

Lead mentor: Leevi Kerkelä

Backup mentors: Rafael Neto Henriques, Marco Palombo

Project website: https://disimpy.readthedocs.io/

Tech keywords: Python, NumPy, Numba, CUDA, 3D modelling

Please send an email to leevi.kerkela.17@ucl.ac.uk if you are interested in this project, and we can discuss more!

Discuss this project on Neurostars!

1.2 Improving GPU-accelerated Monte Carlo simulations in Disimpy: Code refactoring and optimization (350 h)

Code refactoring and optimization to reduce simulation runtimes and GPU memory usage.

Skill level: Advanced

Required skills: Python, Numba, CUDA.

Time commitment: Full-time

Lead mentor: Leevi Kerkelä

Project website: https://disimpy.readthedocs.io/

Backup mentors: Rafael Neto Henriques, Marco Palombo

Tech keywords: Python, NumPy, Numba, CUDA, 3D modelling

Please send an email to leevi.kerkela.17@ucl.ac.uk if you are interested in this project, and we can discuss more!

Discuss this project on Neurostars!

2. BIDS Computational Modeling projects

2.1 Developing a tool to easily store data and study details for computational neuroscience research (350 h)

The values of reproducibility and data sharing gain more and more importance in the research community. To make neuroscience studies reproducible and allow for easy data sharing, the Brain Imaging Data Structure (**BIDS**, https://bids.neuroimaging.io/) standard has been developed. It allows researchers to store their neuroimaging data in a standardized way, speeding up data sharing efforts. Since the development of this BIDS standard, the advantages of using the proposed data structure have been growing steadily: multiple bids-apps have been developed for common data processing pipelines used in the neuroscience community, allowing users to directly load or generate their BIDS-conform data set. Among those apps, is the bids-validator that easily checks the correctness and completeness of all files. BIDS extension

proposals (**BEP**s) have followed for various other modalities used in neuroscience research, such as EEG or PET.

We recently proposed a new standard for storing the necessary data to reproduce a computational neuroscience modeling study. We exemplified our standard by putting a first data set for one of our computational modeling studies using the neuroinformatics platform The Virtual Brain (thevirtualbrain.org/) into the proposed data structure (Triebkorn et al., 2020). We have a first set of scripts allowing the conversion from the modeling and imaging data to the proposed standard but they are so far limited to the specifics of that particular study.

The goal of this project is to develop a user-friendly tool to store study details and data for full reproducibility of computational neuroscience modeling publications. We want to incorporate the existing tools for MRI data, like e.g. bidscoin, into our tool and possibly tools from other modalities.

As an end goal, it would be ideal to have **a couple of diverse examples for using the developed tool in the form of computational modeling study publications prepared for sharing in this BIDS-conform standard**. To be more user-friendly, the conversion tool should also offer a **graphical user interface (Figure 1)** where researchers can still **add their user knowledge** to the conversion tool. One could also aim for **developing an extension of the bids-validator app** for the newly developed standard to be able to easily check the validity and completeness of files.

For this project, we will need to stay in close communication with the research community. The GSOC contributor will be involved in the ongoing efforts of the neuroscience community to standardize the storage of data, code and simulation details. This will allow the GSOC contributor to **build a large network involving different experts for the different modalities of neuroscience research.** If successful, this project will represent the **groundwork for future computational neuroscience studies offering an easy way to store the accompanying data and code in a standardized way enabling replication.**

Lead mentor: Dr. Jil Meier (Charité Universitätsmedizin Berlin) - CET time zone
Co-mentors:
Dr. Michael Schirner (Charité Universitätsmedizin Berlin, CET time zone),
Lia Domide (Codemart, software development company based in Romania, EET time zone),
Prof. Daniele Marinazzo (Ghent University, CET time zone),
Prof. Petra Ritter (Charité Universitätsmedizin Berlin, CET time zone)

Planned effort: 350 hours

Intended skill level: all levels welcome, project can be adapted to the contributor's level

Pre-requisite skills: basic programming knowledge

Tech keywords: Python, docker, BIDS

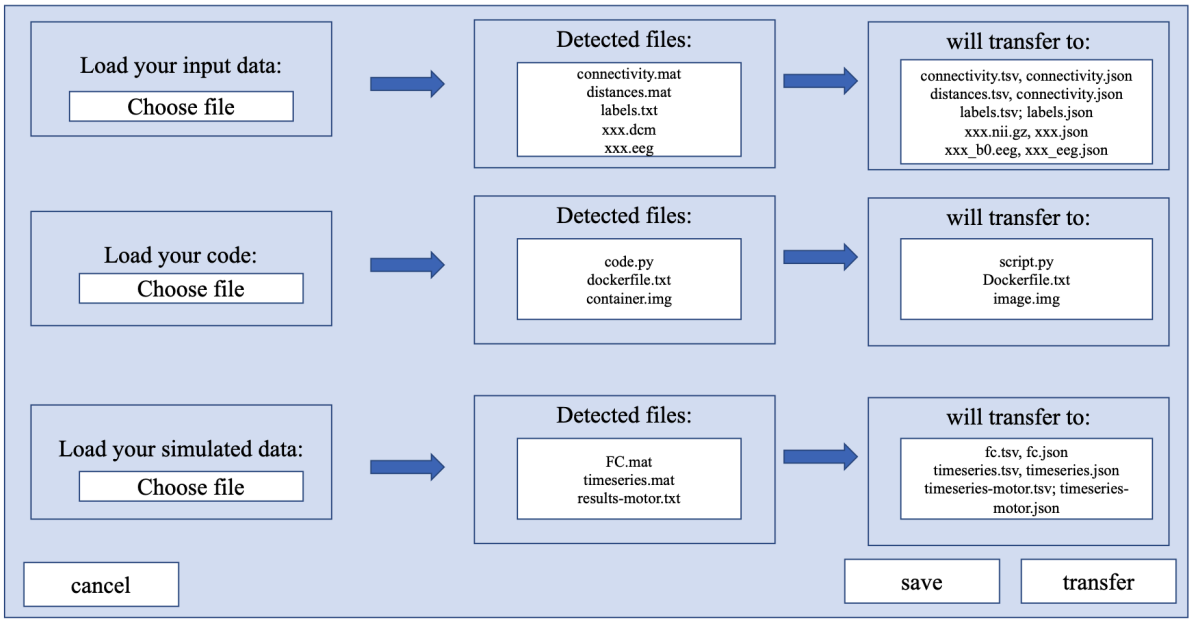


Figure 1: Schematic layout of the final conversion tool. Researchers can load their neuroscience data from different modalities as well as derived data such as connectomes, region labels, simulation results etc.. The tool then detects the file types and suggests files and filenames that it will generate in the standardized folder structure. At this stage in the GUI, the researcher can add his/her user knowledge and adjust file names as well as metadata. Afterwards, the conversion tool transfers the input, code and simulation results into the new data structure with accompanying metadata files (in *.json format).

Further references:

Poldrack, R.A., … , Ritter, P. and Rogers, T.T., 2019. The importance of standards for sharing of computational models and data. *Computational brain & behavior*, 2(3), pp.229-232, doi.org/10.1007/s42113-019-00062-x

bids-validator: https://github.com/bids-standard/bids-validator

Pull request for computational modeling BEP: https://github.com/bids-standard/bids-specification/pull/850

Triebkorn et al. (2020): https://www.biorxiv.org/content/10.1101/2020.03.26.009795v1.abstract

Bidscoin: https://github.com/Donders-Institute/bidscoin

Discuss this project on Neurostars!

3. Turing Way project

3.1 The Turing Way: A how-to guide to data science (175/350 h)

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.

Background: Since the project's launch in 2019, more than 300 contributors have so far co-authored more than 200 subchapters and community documents on reproducible research, communication, collaboration, project design and ethics. As the number of chapters continues to increase, it becomes 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. Over the last 3 months, software engineers at the Turing have enhanced the user interface (UI) of the book (developed a modular Python package) that made it possible for us to create multiple entry points for different user groups, who can start reading the book by exploring a curated set of chapters, rather than browsing the entire book (See details: https://github.com/alan-turing-institute/bio-Turing-Way/blob/malvikasharan-readme/pathways/documentation/README.md).

GSoC project plan and expected outcome: A GSoC contributor will help us integrate this newly developed package to The Turing Way book and further enhance this feature through user experience (UX) design. They will be supported in setting up community/user feedback processes and conducting interviews/focus groups to understand how our readers and contributors use the book and how this UX/UI enhancement adds to their experience. Based on their interest and availability, they will have the possibility to contribute to the development of Python scripts and GitHub actions to improve 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.

Required skills: 1) Python programming, 2) basic web-development skill required to work with Jupyter Book, 3) experience working in distributed communities, using git and GitHub.

Optional skills: Experience collaborating on data science or quantitative research projects at any level, JavaScript skills (front end development), interactive visualisation of small datasets.
Possible mentors:

- Lead mentor: Malvika Sharan (msharan@turing.ac.uk)
- Co-mentors: Turing Way community

Tech keywords: Python, Jupyter, git, JavaScript

Discuss this project on Neurostars!

4. GeNN simulator projects

4.1 ISPC backends for GeNN (350 h)

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)

Tech keywords: C++, SIMT, GPU

Discuss this project on Neurostars!

4.2 Brian2GeNN (350 h)

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)
Tech keywords: Python, C++, GPU

Discuss this project on Neurostars!

4.3 Creating benchmark datasets for object recognition with event-based cameras (175 h)

In our group we have a setup for generating event-based camera recordings of objects with ground truth position (pose) and semantic segmentation data from a separate 3D tracking system. In this project we will augment our recordings in different ways to create challenging segmentation and pose estimation benchmarks.

Perlin noise augmentation: We will generate 2D Perlin noise and transform it into visual event streams that are added to our recordings of moving objects.

Movie background augmentation: We will use off-the-shelf and/or recorded videos and transform them into visual events by implementing a simulation of an event-based camera and add the resulting event stream to our recordings of moving objects.
As a stretch target, we may train a classification network using a spike-based learning rule such as eProp or eventProp to recognise objects in the newly created benchmark datasets.

Skills required: Python. Experience with event-based cameras (and (Py)GeNN for the stretch target) would be beneficial.

Mentors: James Turner (J.P.Turner@sussex.ac.uk) and Thomas Nowotny (t.nowotny@sussex.ac.uk)

Tech keywords: Python

Discuss this project on Neurostars!

4.4 A GPU-accelerated model of the mouse primary visual cortex (175 h)

The Allen Institute has produced a data-driven model of the mouse primary visual cortex (https://portal.brain-map.org/explore/models/mv1-all-layers). Using GeNN (http://genn-team.github.io/genn/) - our GPU-accelerated spiking neural network simulator - you will reproduce the point neuron version of this model. As modern GPUs are capable of running models of this scale in real-time, ambitious students could consider connecting this model to live input from a webcam.

Mentors: This project will be supervised by Dr James Knight and Prof Thomas Nowoty

Skills: Python programming, Maths, Computational Neuroscience

Tech keywords: Python

Discuss this project on Neurostars!

5. AnalySim projects

5.1 Improving graphical interface and user experience for AnalySim (350 h)

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: https://analysim-dev.herokuapp.com/home

This project will involve improving the graphical interface and user experience for AnalySim. The developer would focus on the visual elements and would be good to be able to understand the AnalySim architecture.

Main Technologies: Angular, Typescript, Bootstrap, ASP.Net Core, C#, PostgreSQL, Inkscape/Adobe Illustrator, Gimp/Adobe Photoshop, CSS, HTML

Technologies for analysis notebooks: JavaScript, ObservableHQ, D3.js, Vega, Plotly, potential for Jupyter add-on

Skill level: intermediate/advanced preferable

Tech keywords: Angular, Typescript, Bootstrap, ASP.Net Core, C#, PostgreSQL, JavaScript, ObservableHQ, D3.js, Vega, Plotly

Lead mentor: Anca Doloc-Mihu (adolocm@gmail.com)
Back-up mentor: Cengiz Gunay (cengique@users.sf.net)

Discuss this project on Neurostars!

5. 2 Implementing CSV data browser and querying component for AnalySim (175 h)

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: https://analysim-dev.herokuapp.com/home

This project will involve implementing a CSV file data browser and querying component for AnalySim. The developer would have to modify both the backend API service in C# and frontend components in Angular. An intuitive user interaction workflow would need to be designed as well.

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

Technologies for analysis notebooks: JavaScript, ObservableHQ, D3.js, Vega, Plotly, potential for Jupyter add-on

Skill level: intermediate/advanced preferable

Tech keywords: Angular, Typescript, Bootstrap, ASP.Net Core, C#, PostgreSQL, JavaScript, ObservableHQ, D3.js, Vega, Plotly

Lead Mentor: Cengiz Gunay (cengique@users.sf.net)
Back-up mentor: Anca Doloc-Mihu (adolocm@gmail.com)

Discuss this project on Neurostars!

6. Eye-tracking projects

6.1 Eye-tracker using deep learning with Python + TensorFlow/Pytorch (350 h)

Current eye-trackers generally rely on previous-generation computer-vision algorithms and the best ones are also expensive and closed-source. Recent work has reported that it is possible to obtain very good performance using simple convolutional neural networks (CNN) running off a mobile phone camera. This project primarily involves implementing and extending these CNNs to evaluate and improve their performance. Potential extensions include: connecting the CNN to

a mobile-phone app's camera stream, incorporating head-position (pose) estimates for eye-in-head measurements, and incorporating time-series filtering to improve the eye-position estimation.

Planned effort: 350 hours.

Skill level: Intermediate, Advanced.

Pre-requisite skills: At least comfortable with Pytorch/Tensorflow. Have experience with image/video processing and computer vision. Alternatively, comfortable with Android/iOS app development.

Lead mentor: Suresh Krishna.

Co-mentor: Dinesh Sathia Raj, Vineet Gandhi

Tech keywords: Python, PyTorch, TensorFlow, Android, iOS

Discuss this project on Neurostars!

6.2 Efficient app-based measurement of visual functions in infants and young children (350 h)

Accurate and efficient measurement of visual function is difficult in infants and young children because of limited cooperation, inability to provide cognitive verbal responses and lack of efficient behavioural methods. This is important in the clinical and research context where detection and treatment of eye conditions in infancy is dependent on measurement of visual function. Visual deprivation in infants disrupts normal visual development and affects multiple visual functions that are important in visually guided behaviors in everyday life such as contrast sensitivity, motion perception, contour integration, and face recognition. At present there are no reliable automated objective methods for measuring visual functions in infants and young children below the age of 3 years.

This new project will address these limitations. It involves the development of an application with a suite of visual stimuli to probe multiple visual functions. The application will employ an adaptive staircase with a preferential looking behavioral paradigm and eye tracking. The application will measure the sensory threshold of each visual function and the response path to the threshold, such as uncertainty, providing important additional indicators creating an individual and disease specific profile of visual loss. This application will extend to establish visual function norms, profile visual loss allowing targeted intervention therapy and monitor the effects of treatment.

Planned effort: 350 hours

Skill level: Intermediate/Advanced

Pre-requisite skills: Comfortable with 1 language (e.g. Python). Experience with app development and basic image/video processing. Ideally, comfortable with Android/iOS app development.

Mentors: Arvind Chandna (lead) and Suresh Krishna (co-mentor)

Tech keywords: App development, iOS/Android, Python/equivalent

Discuss this project on Neurostars!

7. CAMH EEG projects

7.1 Computational modelling of FNIRS-EEG in Python with The Virtual Brain and Kernel Flow (350 h)

Lead Mentor: Dr. John Griffiths (CAMH, University of Toronto)

Co-Mentor: Dr. Davide Momi (CAMH)

Commitment Level: 350 Hours

Skill Level: Intermediate

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, and fNIRS data analysis libraries. 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. The data analysis-based parts of the project will also build upon and extend our kftools Python library.

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.

Tech keywords: Python, TVB, Computational Neuroscience, Neural mass modelling, Connectome, Biophysics, FNIRS, EEG

Discuss this project on Neurostars!

7.2 Mobile EEG Cognitive Neuroscience Experiments with the EEG-Notebooks Python Library (175/350 h)

Lead Mentor: Dr. John Griffiths (CAMH, University of Toronto)

Co-Mentor: Dr. Davide Momi (CAMH)

Commitment Level: 350 OR 150 Hours

Skill Level: Intermediate / Beginner

EEG-Notebooks is an open-source, Python-based library developed by the NeuroTechX community for running cognitive neuroscience experiments with low-cost mobile EEG devices. By combining standardized and established research protocols with the simple-to-use structure of the library, EEG-Notebooks intends to bring high standards implemented in research practice to a larger audience, especially outside of academia. The applications of EEG Notebooks range from research, medicine, outreach, and education. By developing this toolkit, we aim to make 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. The library is also designed to be intuitively implementable without extensive knowledge and experience in psychology and neuroimaging research.

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.

Tech keywords: Python, EEG, BCIs, Neurotechnology, Data Analysis, Machine Learning, Cognitive Neuroscience

Discuss this project on Neurostars!

8. TVB projects

There are several modeling studies using brain network models which incorporate biologically realistic macroscopic connectivity (the so-called connectome) to understand the global dynamics observed in the healthy and diseased brain measured by different neuroimaging modalities such as fMRI, EEG and MEG.

For this particular modelling approach in Computational Neuroscience, open source frameworks enabling the collaboration between researchers with different backgrounds are not widely available. The Virtual Brain is, so far, the only neuroinformatics project filling that place.

All projects below can be either full time or part-time, as the features/pages can be built incrementally.

8.1 Integrate TVB with Zenodo (175/350 h)

TVB (https://www.thevirtualbrain.org/) has a demo dataset currently published on Zenodo (https://zenodo.org/record/4263723#.Ydi-dX1Bzx4). We use it by manually downloading it, unzip then use inside tvb code and web GUI. We intend to use Zenodo API instead.

The task is mainly for part-time, if only the above feature will be used, but it can be extended if needed with other external data sources for full time applicants.
More details here: https://req.thevirtualbrain.org/browse/TVB-2607

Expected results: A set of classes , with demo Jupyter notebook, and unittests.

Skills: Python, Zenodo, Jupyter

Mentors: Lia Domide, Paula Prodan, Romina Baila

Tech keywords: Python, Jupyter, API

Discuss this project on Neurostars!

8.2 Separate tool to upload data into TVB (175/350 h)

TVB (https://www.thevirtualbrain.org/) has become a complex tool. It exposes REST services also. We are interested to build, from scratch, a new module, to remotely call TVB web service for importing data into TVB. This tool could also be enhanced into encrypting data before upload, validate it, check BIDS compatibility, etc.

Expected results: A new tvb-* package, ready for Pypi, to be able and call TVB rest services for uploading data into a running tvb-distribution. Unit-tests.

Skills: Python, REST, encryption, BIDS

Mentors: Paula Prodan, Romina Baila, Lia Domide

Tech keywords: Python, REST, BIDS

Discuss this project on Neurostars!

8.3 TVB Web page redesign (175/350 h)

TVB (https://www.thevirtualbrain.org/) has become a complex tool, with a large web front-end. We have few very complex pages which could benefit from a redesign. One of such pages in the Simulator page. We expect from this project to first understand the current web design and logic of the application, then improve one or multiple pages in TVB through CSS/HTML. https://www.thevirtualbrain.org/tvb/zwei/image-zoom/122368/26?stage=brainsimulator

Expected results: A static HTML with CSS containing the new design, respecting TVB logic.

Skills: HTML, CSS

Mentors: Romina Baila, Lia Domide, Paula Prodan

Tech keywords: HTML, CSS

Discuss this project on Neurostars!

9. Open Source Brain projects

9.1 Open source, cross simulator, large scale cortical models in NeuroML and PyNN (175/350 h)

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.

Note: this project is suitable for a half-time or full-time commitment by the GSoC contributor, with the scope of the model conversion scaled as appropriate.

Mentors: Padraig Gleeson (lead), Ankur Sinha

Tech keywords: Python, XML, networks, modelling, simulation

Discuss this project on Neurostars!

9.2 Conversion of public neurophysiology datasets to Neurodata Without Borders (NWB) format (175/350 h)

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

Note: this project is suitable for a half-time or full-time commitment by the GSoC contributor, with the scope of the data conversion scaled as appropriate.

Mentors: Padraig Gleeson (lead), Ankur Sinha

Tech keywords: Python, HDF5, data analysis, open access.

Discuss this project on Neurostars!

10. Brian simulator projects

10.1 Make Brian run in the web browser via Pyodide (175/350 h)

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
- [for a 6 months project]: create a mechanism to specify model parameters that a user should be able to interactively modify when running a simulation, and expose convenient access to these controls from the HTML interface.

Planned effort: 175h or 350h (350h preferred)

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

Skill level: intermediate

Mentors: Marcel Stimberg, Dan Goodman

Tech keywords: Brian, Python, HTML, JavaScript

Discuss this project on Neurostars!

10.2 Make Brian run in the web browser via emscripten (175/350 h)

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 emscripten project is a compiler toolchain that compiles C++ code into WebAssembly, which can then be executed in the browser. It is therefore possible to use Brian's existing C++ compilation framework to run a Brian simulation in the browser, but there are still a number of obstacles to overcome before this can be useful in practice. The specific aims of this project are:

- Adapt Brian's C++ standalone mode so that it can compile to WebAssembly. In particular, this requires to replace the current mechanism that relies on files on disk to initialize values and return the results of a simulation.
- create convenient methods to display simulation results in the browser
- investigate integrating this with the current Brian examples and/or on a dedicated "showcase" website
- [for a 6 months project]: create a mechanism to specify model parameters that a user should be able to interactively modify when running a simulation, and expose convenient access to these controls from the HTML interface.
- [for a 6 months project (alternative)]: enhance the display methods so that they can display simulation results continuously, instead of requiring for the simulation to end.

Planned effort: 175h or 350h (350h preferred)

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

Skill level: intermediate

Mentors: Marcel Stimberg, Dan Goodman

Tech keywords: Brian, Python, HTML, JavaScript

Discuss this project on Neurostars!

10.3 Update Brian documentation infrastructure (175 h)

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)

Planned effort: 175h

Skills: Python programming, experience with sphinx and CI infrastructure such as GitHub Actions helpful

Skill level: novice

Mentors: Marcel Stimberg, Dan Goodman

Tech keywords: Brian, Python, documentation, sphinx

Discuss this project on Neurostars!

10.4 Improved OpenMP support (synaptic propagation) (175/350 h)

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
- [for 6 months project:] identify other parts of a simulation (e.g. creation of synapses, "summed variable" mechanism) that could benefit of parallelization and implement these approaches.

Planned effort: 175h or 350h

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

Skill level: advanced

Mentors: Marcel Stimberg, Dan Goodman

Tech keywords: Brian, Python, C++

Discuss this project on Neurostars!

10.5 Improved parser for model descriptions (175 h)

Computational research in biology commonly consists of describing a model system and its parameters, simulating the system with specialized software, and then analyzing the results. Model descriptions often make use of a domain-specific language that is parsed and interpreted by the simulation software. Such languages have the advantage that they are more accessible to researchers than code written in a general-purpose programming language; they also make it easier to discuss and share models with other researchers.

Parsing such languages is mostly straightforward with standard techniques, but these techniques regularly have at least two shortcomings: 1) For syntactically incorrect descriptions, e.g. a missing parenthesis, error messages are typically not very helpful and of the form "unexpected symbol at position x", and 2) annotations via comments are usually simply ignored, instead of being used to enrich a model description.

Both of these shortcomings are currently present in the Brian simulator, an open-source simulator for biological spiking neural networks written in Python, developed in our research group and used by researchers world-wide. The Brian simulator describes models with a domain-specific language that uses mathematical notation with additional annotations, e.g. to assure the consistency of physical dimensions.

The goal of this internship is to rewrite the Brian simulator's parsing code to give clear and helpful error messages for incorrect model descriptions, as well as treating comments in the model descriptions as annotations that are stored for future usage.

Planned effort: 175h

Skills: Python programming, parser/compiler techniques

Skill level: intermediate

Mentors: Marcel Stimberg, Dan Goodman

Tech keywords: Brian, Python, parser

Discuss this project on Neurostars!

11. NIDM and PyNN projects

11.1 Extending NIDM to include electrophysiology experiments (350 h)

The aim is to extend the Neuroimaging Data Model (NIDM) standard to cover the provenance of electrophysiology data sets. To develop these extensions, our team is in an unique position, as we are responsible for or involved with development of other tools in this domain, notably openMINDS-ephys (extension of the open Metadata Initiative for Neuroscience Data Structures (openMINDS) for electrophysiology data), Neo (importing and exporting electrophysiology data in Python), and the BIDS extension proposal for animal electrophysiology data.

Mentor: Peyman Najafi, co-mentor Andrew Davison.

Effort: 350 hours.

Tech keywords: Python, BIDS

Discuss this project on Neurostars!

11.2 Type hints for the PyNN model description API (175 h)

PyNN is a simulator-independent language for building neuronal network models, implemented as a Python API. Type hinting is a way to indicate the type of a value within Python code, introduced in Python 3.5. By applying type hinting to the PyNN code base, we will make the code more self-describing, make it easier for simulator developers to implement the API for their own simulator, and make it possible to develop automated checks for PyNN-compatibility.

Mentor: Andrew Davison, co-mentor Chitaranjan Mahapatra.

Effort: 175 hours.

Tech keywords: Python, API

Discuss this project on Neurostars!

12. HNN-core projects

Human Neocortical Neurosolver (HNN) is a software for interpreting the neural origin of macroscale magneto-/electro-encephalography (MEG/EEG) data using

biophysically-detailed microcircuit simulations. HNN can be run through a user-friendly graphical user interface or through a Python interface HNN-core.

Mailing list(s): https://groups.google.com/g/hnnsolver

- Overview of HNN Utility
- HNN-GUI tutorials
- HNN-core tutorials and examples
- Contributing guide

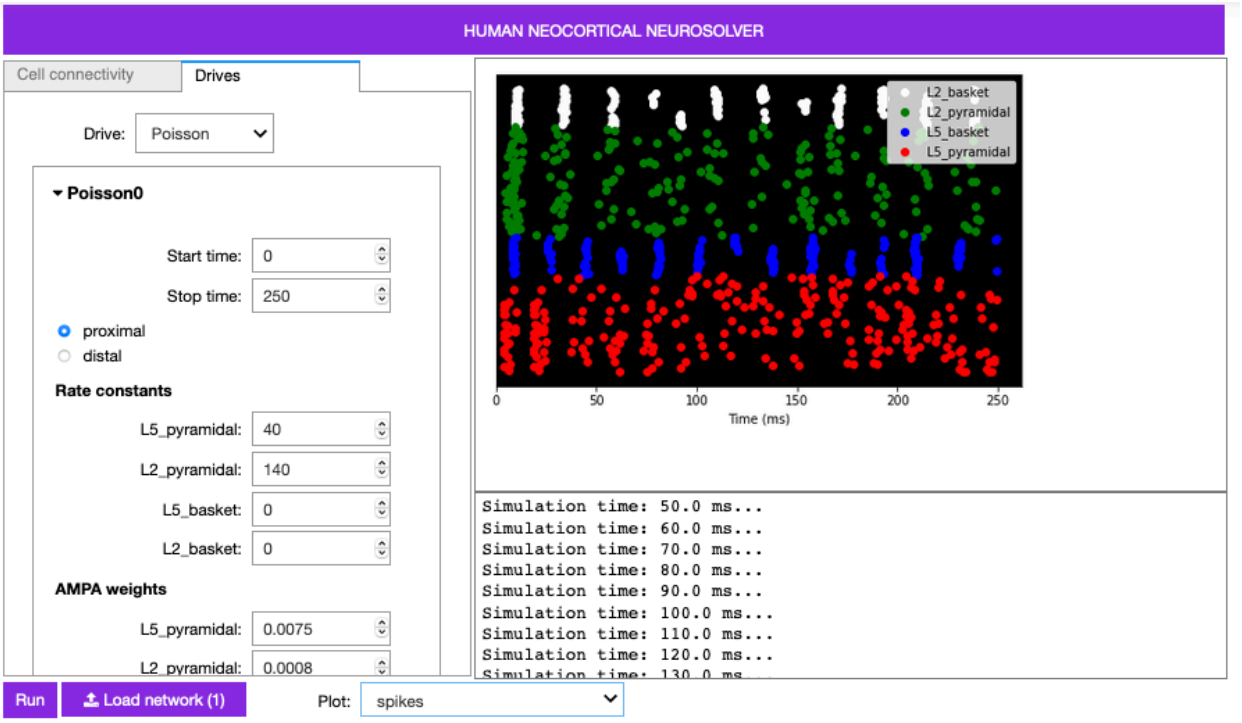
12.1 Build new GUI using ipywidgets (350 h)

Goal: The aim of this project is to build a new GUI with the same functionality as the current HNN, using the HNN-core API and following best practices in open source software design.

Subgoals:

Start from: https://github.com/jonescompneurolab/hnn-core/pull/76

- Run HNN-GUI tutorials with old as well as new GUI to identify and implement functionality that missing
- Build new GUI components based on identified weaknesses in new GUI. This includes implementing dialog boxes for optimization, displaying waveforms from previous simulations
- Write tests for the GUI using "fake clicks" and incorporate them in continuous integration
- Extra credit: Test GUI over remote cluster connection, incorporate new API elements in GUI such as viewing connectivity matrices and modifying connectivity



Difficulty: Medium

Duration: 350 hours (full time)

Skills: Some experience in GUI development such as matplotlib, ipywidgets or PySimpleGUI may be helpful but not necessary. Experience in neuroscience is also helpful but not required.

Possible mentors: Mainak Jas, Stephanie Jones

Tech keywords: Python, GUI

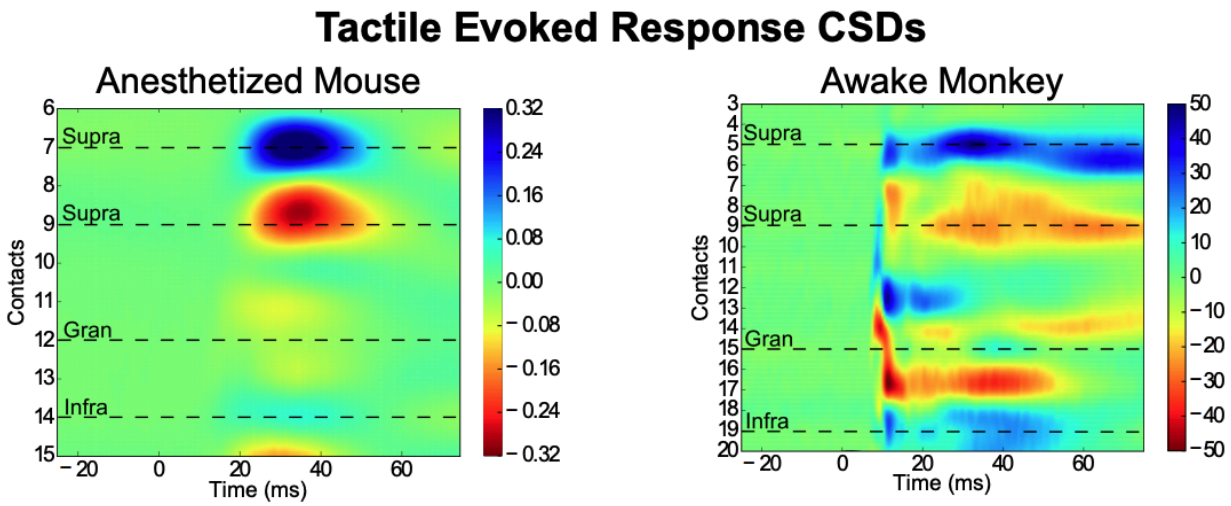
Discuss this project on Neurostars!

12.2 Implement methods to calculate and visualize current source density signals (350 h)

Goal: The aim of this project is to enhance HNN functionality with tools to simulate and visualize current source density (CSD) signals from the HNN neocortical model, beginning with the HNN-core API for simulating local field potential signals (LFPs).

Subgoals
Related issue: https://github.com/jonescompneurolab/hnn-core/issues/68

- Understand the LFP example and code
- Decide on CSD methods and API. For example: standard difference of LFP, spline, step etc. See the iCSD package
- Develop the code for documenting CSD tools following the current examples for simulation event relate potentials and low frequency rhythms
- Write tests for the CSD functionality. For example, by computing the CSD on artificial "sinusoidal" LFP pattern, or checking that the peaks are roughly the same with different methods.
- Bonus: Build local fields potential and CSD visualization into the next generation HNN GUI components in https://github.com/jonescompneurolab/hnn-core/pull/76. Begin implementation of comparison of simulated LFP/CSD to empirically recorded data.



Difficulty: Medium

Duration: 350 hours (full time)

Skills: Some experience in neuroscience and Python. Experience in simulating neural activity with NEURON is helpful but not required.

Possible mentors: Nicholas Tolley, Stephanie Jones, Mainak Jas, Ryan Thorpe

Tech keywords: Python

Discuss this project on Neurostars!

12.3 Develop IO routines for HNN-core outputs (350 h)

Goal:
The current IO routines in HNN-core are fragmented as they were adapted from HNN-GUI. The goal is to develop IO routines adapted from HNN-core objects while maintaining backwards compatibility with HNN-GUI.

Subgoals:

- Develop a method to write cell_response object and read from it. It should be able to handle multiple trials and be able to plot rasters after reading from a saved file.
- Develop a function to write and read from Dipole and ExtracellularArray. It should be able to handle multiple trials. The format should be standardized between the two objects as much as possible.
- Develop a function to write and read Network object. It should be based on hdf5 and use the h5io library.
- Document each of the IO formats in an rst document and develop tests for each function.
- Bonus: Develop a function to write Network object to NeuroML format and test that it can be loaded in NetPyne

Difficulty: Medium

Duration: 350 hours (full time)

Skills: Python, some experience in neuroscience data analysis may be helpful

Possible mentors: Ryan Thorpe, Nicholas Tolley

Tech keywords: Python

Discuss this project on Neurostars!

13. Neuroptimus project

13.1 Integration of automated model testing and parameter fitting tools for neuroscience applications (350 h)

Biologically detailed models are useful tools in neuroscience, and automated methods are now routinely applied to construct and validate such models based on the relevant experimental data. The open-source parameter fitting software Neuroptimus (formerly Optimizer) was developed to enable the straightforward application of advanced parameter optimization methods (such as evolutionary algorithms and swarm intelligence) to various problems in

neuronal modeling. Neuroptimus includes a graphical user interface, and works on various platforms including PCs and supercomputers. Neuroptimus currently uses various built-in cost functions and those implemented by the eFEL feature extraction library to compare the behavior of the models to the (experimental) target data. However, this approach severely limits the range of neuronal behaviors that can be targeted by the optimization. On the other hand, the popular model-testing framework SciUnit allows the implementation of tests that quantitatively evaluate arbitrary model behaviors.

The aim of the current project is to extend the open-source neural parameter optimization tool Neuroptimus so that it is able to use test scores from the SciUnit framework as the cost function during optimization. Direct applications would include the construction of detailed biophysical models of hippocampal neurons using a combination of Neuroptimus and HippoUnit, an open-source neuronal test suite based on SciUnit. All of these tools are implemented in Python.

Skills and effort: The task would probably require a full-time effort during GSoC (350h), and at least intermediate coding skills.

Mentors: The project would be supervised by members of the Computational Neuroscience laboratory at the Institute of Experimental Medicine (Budapest, Hungary), including Sára Sáray (the developer of HippoUnit) and Szabolcs Káli (head of the laboratory), with contributions from Máté Mohácsi (the current lead developer of Neuroptimus).

Tech keywords: Python

Discuss this project on Neurostars!

14. FAIR data informatics lab projects

14.1 InterLex (175/350 h)

InterLex (https://interlex.org), which supports INCF's KnowledgeSpace, is a dynamic lexicon of biomedical terms. Unlike an encyclopedia, a lexicon provides the meaning of a term, and not all there is to know about it. InterLex is being constructed to help improve the way that biomedical scientists communicate about their data, so that information systems like NIF and dkNET can find data more easily and provide more powerful means of integrating that data across distributed resources.

Mentors: The primary mentors for these projects would be Tom Gillespie and Fahim Imam (mainainters of the NIF ontologies and SCKAN), Jeffrey Grethe (PI for InterLex associated projects), Monique Surles-Ziegler (SPARC anatomical ontology working group).

Skill level: intermediate/experienced

Aim: The projects will focus on adding specific components to InterLex to enhance the semantic descriptions of terms and integrate select additional information from other knowledge sources (e.g. from SCKAN below)

Tech keywords: Ontology, SPARQL, Python, GUI, PHP

Discuss this project on Neurostars!

14.2 SPARC Connectivity Knowledge Base of the Autonomic Nervous System (175/350 h)

SPARC Connectivity Knowledge Base of the Autonomic Nervous System (SCKAN; https://sparc.science/resources/6eg3VpJbwQR4B84CjrvmyD) is part of the NIH Common Fund's Stimulating Peripheral Activity to Relieve Conditions (SPARC) program. SCKAN is a semantic store housing a comprehensive knowledge base of autonomic nervous system (ANS) nerve to end organ connectivity. Connectivity information is derived from SPARC experts, SPARC data, literature and textbooks.

Mentors: The primary mentors for these projects would be Tom Gillespie and Fahim Imam (mainainters of the NIF ontologies and SCKAN).

Skill level: intermediate/experienced

Aim: The projects will focus on the development of user interface components that can be embedded within a website for the visualization of connectivity information from the knowledgebase.

Tech keywords: Knowledge Graph, SPARQL, Neo4J, Python, Graph UI

Discuss this project on Neurostars!

14.3 Open Data Commons for Spinal Cord Injury (175/350 h)

Open Data Commons for Spinal Cord Injury (ODC-SCI; https://odc-sci.org) is a cloud-based community-driven repository to store, share, and publish spinal cord injury research data. The ODC-SCI also aims to increase transparency with individual-level data, enhance collaboration,

facilitate advanced analytics, and conform to increasing mandates by funders and publishers to make data accessible.

Mentors: The primary mentors for these projects would be Jeffrey Grethe (mPI for the ODC) and Michael Chiu (lead developer of the ODC).

Skill level: intermediate/experienced

Aim: The project will focus on 1) providing the ability to add analytics/analyses to the ODC through the use of Jupyture notebooks and UI frameworks for these notebooks (i.e. Appyters); 2) Further development of the ODC API to support Frictionless Data standards and provide functionality to allow integration with analytic environments (i.e. R, Python, Matlab, etc.)

Tech keywords: Data Science, Python, R, Jupyter, Appyter

Discuss this project on Neurostars!

15. ancpBIDS projects

ancpBIDS is a lightweight Python library to read/query/validate/write BIDS datasets. It is a tool which enables researchers to embed their BIDS datasets of any kind or modality into their workflows or analysis pipelines. Its implementation is based on the BIDS schema and allows it to evolve with the BIDS specification in a generic way.

You can find the project on GitHub here : https://github.com/ANCPLabOldenburg/ancp-bids

15.1 Interactive graph visualization of the BIDS schema (175 h)

The ancpBIDS implementation is based on the machine-readable schema files in the BIDS specification Github repository. Interested contributors are expected to implement a graph visualization for users to interactively navigate through transitions and nodes of interest of the schema. In contrast to reading the human-readable BIDS specification, this approach allows BIDS users to more quickly gain knowledge about formalized aspects of the specification.

Skills: Contributors should be familiar with Python and the specification of the Brain Imaging Data Structure to successfully work within this project.

Tasks/Goals:

- Analyze problem domain - What is the challenge?
- Design possible solutions
- Implement and test at least one of the solutions
- Report status on a bi-weekly basis to a broader audience (lab intern, online)

Mentors: Erdal Karaca (lead mentor), Aaron Reer (co-mentor)

Tech keywords: Python, BIDS

Discuss this project on Neurostars!

15.2 Validation engine for BIDS datasets (350 h)

The ancpBIDS implementation allows to plug-in additional functionality at API level. Interested contributors are expected to use the plug-in mechanism to extend the validation module of ancpBIDS to allow for more complex validation rules derivable from the BIDS specification (human-readable) and/or the BIDS schema (machine-readable).

Skills: Contributors should have an intermediate level in Python and should know about basic principles of Software Engineering in order to get the most out of this project.

Tasks/Goals:

- Analyze problem domain - What is the challenge?
- Design possible solutions
- Implement and test at least one of the solutions
- Report status on a bi-weekly basis to a broader audience (lab intern, online)

Mentors: Erdal Karaca (lead mentor), Aaron Reer (co-mentor)

Tech keywords: Python, BIDS

Discuss this project on Neurostars!

16. DIPY projects

16.1 GPU parallelization of DIPY algorithms (350 h)

We have multiple versions of GPU parallelized algorithms. The project will be bringing those together in a common framework and adding new methods as well. For example, we currently have cudipy and GPU streamlines. A unified framework is required. In addition, we will need to parallelize some algorithms such as those used for probabilistic tractography and nonrigid registration.

Difficulty: Intermediate
Time : Full time

Skills required : CUDA, C/C++, Python

Mentors: Shreyas Fadnavis, Jongsung Park, Bramsh Qamar Chandio

Tech keywords: CUDA, C/C++, Python, GPU

Discuss this project on Neurostars!

16.2 A pythonic implementation of topup (350 h)

A popular request is to bring topup [1] in a simple pythonic implementation that can be easily extended. The student will work on implementing topup (or similar function) from the ground up using Python, Cython and Numpy.

Difficulty: Hard

Time: Full time

Skills required : Strong familiarity with diffusion MRI, Python, Numpy, Cython or C/C++

Mentors: Jongsung Park, Shreyas Fadnavis, Bramsh Qamar Chandio

Tech keywords: Python, C/C++

[1] J.L.R. Andersson, S. Skare, J. Ashburner. How to correct susceptibility distortions in spin-echo echo-planar images: application to diffusion tensor imaging. NeuroImage, 20(2):870-888, 2003.

Discuss this project on Neurostars!

16.3 Add more options on DIPY Horizon (350 h)

Extend dipy_horizon workflow by adding more options for the visualization and cleaning of tractograms generated from diffusion MRI 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 students to add support for different types of file formats and visualizations in the horizon workflow. Students will work on adding multiple features to help manual cleaning of streamlines such as select and remove streamlines from a bundle, cut some parts of the bundle and so on.

Steps:

- Add region-of-interest (ROI) capacity for streamline filtering in Horizon.
- Create bundle diagnostics. Eg: given bundle and FA file as input. It should generate bundle profiles, information about total streamline count, average length of the bundle and visualize all this.

Difficulty: Intermediate

Time: Full time

Skills required: Python, OpenGL, VTK

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

Tech keywords: Python, OPenGL, VTK

Discuss this project on Neurostars!

16.4 Add mutual information for non-rigid registration (175 h)

Currently mutual information (MI) similarity metric is available only for affine registration and not nonrigid registration. The student will need to implement, compare and test MI for nonrigid registration. The existing implementation provides examples for SSD and Expectation Maximization metrics. The MI method is expected to work slightly better than our existing EM method for multimodal images. This project will focus on multimodal images. If time is permitted the student can also investigate a tensor based metric for registration.

Time: Half time

Difficulty: Beginner

Skills required: Familiarity with registration algorithms, Python, Cython or C/C++.

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

Tech keywords: Python, Cython, C/C++

Discuss this project on Neurostars!

17. Pydra projects

Pydra is a new lightweight dataflow engine written in Python. The package is a part of the second generation of the Nipype ecosystem — an open-source framework that provides a uniform interface to existing neuroimaging software and facilitates interaction between different software components. The Nipype project was born in the neuroimaging community, and has been helping scientists build workflows for a decade, providing a uniform interface to such neuroimaging packages as FSL, ANTs, AFNI, FreeSurfer and SPM. This flexibility has made it an ideal basis for popular preprocessing tools, such as fMRIPrep and C-PAC. The second generation of Nipype ecosystem is meant to provide additional flexibility and is being developed with reproducibility, ease of use, and scalability in mind.

Pydra is developed as an open-source project in the neuroimaging community, but it is designed as a general-purpose dataflow engine to support any scientific domain. Scientific workflows often require sophisticated analyses that encompass a large collection of algorithms. The algorithms that were originally not necessarily designed to work together, and were written by different authors. Some may be written in Python, while others might require calling external programs. It is a common practice to create semi-manual workflows that require the scientists to handle the files and interact with partial results from algorithms and external tools. This approach is conceptually simple and easy to implement, but the resulting workflow is often time consuming, error-prone and difficult to share with others. Consistency, reproducibility and scalability demand scientific workflows to be organized into fully automated pipelines. This was the motivation behind Pydra.

17.1 Adding BIDS Prov to new dataflow engine written in Python: Pydra (175 h)

Provenance captures the relationship of data objects to the processes that generated them and the inputs to those processes, enabling scientific results to be interpreted and compared based on their generating processes. To be useful, the provenance must be comprehensive, understandable, easily communicated, and captured automatically in machine accessible form.

In the neuroimaging context, there is an ongoing effort (BIDS-Prov) to create a specification for the provenance of BIDS datasets, built on the W3C PROV standard.

There is a natural correspondence between a workflow definition, which describes the flow of data objects, and the provenance of the results of the workflow. We would like to build provenance tracking capabilities directly into Pydra, to dynamically construct BIDS-Prov compatible metadata that can be saved alongside any results. This work will be done in close collaboration with the BIDS-Prov working group.

Planned effort: 175 hours

Skills:

- Python 3: novice +
- Bash: novice +
- Semantic web, RDF, JSON schema: novice +
- Data workflows: beginner +

Mentors: Dorota Jarecka, Michael Dayan, Satra Ghosh; collaboration with BIDS-Prov working group

Tech keywords: Provenance, BIDS, Workflow, Python, Pydra, Nipype

Discuss this project on Neurostars!

17.2 Adding new workers and resource management to new dataflow engine written in Python: Pydra (175 h)

Pydra workflows are intended to be written independent of considerations of the computational resources that will ultimately execute them. Pydra "workers" are classes that describe how to submit nodes in an execution graph to a computational resource, for instance a process pool on a local machine or a high-performance computing cluster. Currently Pydra has support for local multiprocessing, Slurm, Dask, and Oracle/Sun Grid Engine (SGE). We would like to expand the range of systems that Pydra can manage, as well as improve utilization of features for existing workers. One goal of particular interest to us is to track resource (CPU, GPU, RAM) allocation to allow scheduling that makes efficient use of those resources. The specific task can depend on the participant's interest and experience.

Planned effort: 175 hours

Skills:

- Programming, OOP: intermediate +
- Python 3: novice +
- Bash/Shell: novice +
- HPC and schedulers: beginner +

Mentors: Dorota Jarecka, Chris Markiewicz, Satra Ghosh

Tech keywords: HPC, schedulers, Python, Pydra, Nipype

Discuss this project on Neurostars!

17.3 Converting existing scientific workflows to new dataflow engine written in Python: Pydra (350 h)

There are many scientific workflows, written using a variety of languages and frameworks. In neuroimaging, many use Nipype 1 or bash. Pydra is intended to be a generic dataflow engine, and we would like to demonstrate its utility by converting existing workflows from different scientific domains. There is flexibility according to participant interest and experience to select the specific workflows. This project will require the creation of new Pydra Task classes that wrap the necessary tools. Depending on the tools to be wrapped, it may be possible to automatically generate classes from a pre-existing specification, or they may be written manually as needed. Likewise, there may be opportunities to convert specifications of entire workflows into Pydra workflows. Generated workflows will be made accessible through the Niflows framework (or similar) and submitted for reuse to workflow hubs such as workflowhub.eu and dockstore.org.

Planned effort: 350 hours

Skills:
- Python 3: novice +
- Bash: novice +
- Scientific software (e.g. AFNI, SPM, ITK, SpikeInterface, CalmAn): intermediate +
- Creating data workflows: beginner +
- Data file format, e.g. NWB, NIfTI, OME-TIFF, PLINK, HDF5: beginner +

Mentors: Dorota Jarecka, Chris Markiewicz, Hao-Ting Wang, Satra Ghosh, collaboration with groups responsible for specific workflows

Keywords: Workflow, Python, Pydra, Nipype, CWL

Discuss this project on Neurostars!

18. ImageJ projects

18.1 Deep Learning using Geometric Features (175 h)

The Active Segmentation platform for ImageJ (ASP/IJ) was developed in the scope of GSOC 2016 - 2021. 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.

Weka (Waikato Environment for Knowledge Analysis) is a collection of machine learning algorithms for data mining tasks. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. The algorithms can either be applied directly to a dataset or called from your Java itself.

The project idea: The Weka library offers some advanced visualization and analysis functionality. The student will develop a new visualization and reporting panel within ASP/IJ which will expose the Weka visualization and analysis functions.

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 development of the plugin and test cases.
- Implementation and testing - Details of implementation and testing of the platform.

Desired skills: Java, Machine learning

Effort: 175 h

Mentors: Dimiter Prodanov (dimiterpp@gmail.com), INCF Belgian Node; Sumit Vohra, ZIB, Berlin, Germany, Purva Chaudhari, Vishwakarma Institute Technology (backups)

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

Tech keywords: Java

Discuss this project on Neurostars!

18.2 High content image feature and classification database (350 h)

The Active Segmentation platform for ImageJ (ASP/IJ) was developed in the scope of GSOC 2016 - 2021. 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).

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 and is available on many platforms.

The project idea: At present, the feature space and the classification results produced by the platform are stored in several separate files. The idea is that the types and values of image features and classification outcomes would be stored in an SQLite database for cross-comparisons between sessions. The candidate is required to use the SQLite database engine 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 development of the plugin and test cases.
- Implementation and testing - Details of implementation and testing of the platform.

Desired skills: Java, SQL

Effort: 350 h

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

Tech keywords: Java, SQL

References

1. ImageJ: https://imagej.nih.gov/
2. Active Segmentation : https://github.com/sumit3203/ACTIVESEGMENTATION
3. SQLite: https://www.sqlite.org/index.html

Discuss this project on Neurostars!

19. NBDT Journal

19.1 Automatic reviewer matching using Natural Language Processing: Infrastructure for NBDT Journal (350 h)

Abstract
The peer-review process is one of the most crucial processes in science. It allows scientists who work in a similar research area to assess and review the publication before publishing. However, as the research community grows, it gets harder to find relevant reviewers. This issue not only puts the burden on editors who need to search for relevant reviewers but also on publications that may get reviewed slower. Moreover, a selection process may introduce bias. An automatic selection approach promises to fasten the review process and reduce bias during an assignment. Here, we propose an automatic pipeline for paper-reviewer assignments for the Neurons, Behavior, Data analysis, and Theory (NBDT) platinum open access overlay journal which allows editors to search, create, find, and assign reviewers in the field facilitated by NLP and machine learning. We envision that the open-source repository can be used in other journals or conferences in the future.

Planned effort: Approximately 300-350 hours

Intended skill level: Intermediate, Advanced

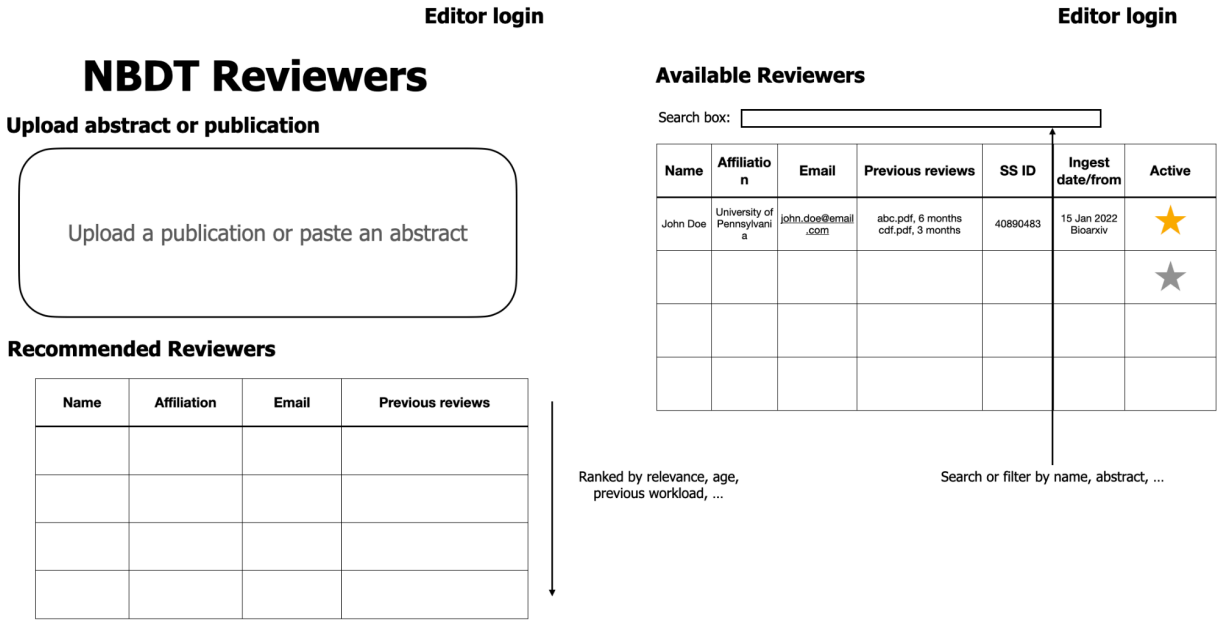
Project effort: Preferred full-time

Pre-requisite skills: Python, REST, Javascript

Lead mentor: Titipat Achakulvisut (Mahidol University) – BKK timezone
Co-mentors: Daniele Marinazzo (Ghent University), Konrad Kording (University of Pennsylvania) – EST timezone

Project aims and tasks:
We describe the project structure and approximate work as follows

- Create and design a reviewer database structure for the NBDT journal. This may include name, email, institution, research interests, published papers, relevant papers, Semantic Scholar ID (50 hours)
- Create and design a web application with log-in using ReactJS that allows editors to store and add information about the reviewers. This should store active reviewers, reviews they have done, and timestamp of the reviews (100 hours)
- Write a Python script to parse reviewers from publications from Arxiv, PsycArxiv, BioArxiv, and relevant neuroscience journals from the MEDLINE database. (50 hours)
- Create an API using FastAI for a web application that allows an editor to put in a given abstract and then find relevant reviewers using Natural Language Processing and deep learning techniques (50 hours)
- Optimizing the assignment function to maximize relevance and reduce potential biases (50 hours)



NBDT Reviewer Application. An application allows editors or authors to upload an abstract or a publication to search for available reviewers. Reviewers are ingested automatically from NBDT, preprints, or relevant journals.

Tech keywords: Python, FastAPI, SQL, REST, ReactJS

Discuss this project on Neurostars!

19.2 Developing a Latex to XML pipeline and exploring a standalone platform for NBDT Journal (175 h)

We launched a platinum open access overlay neuroscience journal (Neurons, Behavior, Data Analysis, and Theory – NBDT). After 30 articles accepted, we are now ready for inclusion in Pubmed Central (PMC). This requires providing the full text of articles in an XML format that conforms to an acceptable journal article DTD (Document Type Definition). The following pages provide detailed information about PMC's technical requirements:

https://www.ncbi.nlm.nih.gov/pmc/pub/addjournal/#tech-eval.
This section details all of our data sample requirements.

See the PMC file specifications at: http://www.ncbi.nlm.nih.gov/pmc/pub/filespec/.
This page details the XML and image file requirements, and describes how these files must be named and packaged for delivery.

The minimum criteria for evaluation of sample data can be found at:
http://www.ncbi.nlm.nih.gov/pmc/pub/min_requirements/.

At the moment papers are submitted to NBDT in latex format.
NBDT is hosted by Scholastica, and they have already something in place for archiving in XML: https://blog.scholasticahq.com/post/why-archiving-essential-for-open-access-journals-how-to-get-started/. We can certainly start from there, but it would be good to have an independent streamline in case we decide to change platform.

Planned effort: Approximately 175 hours

Intended skill level: Intermediate, Advanced

Project effort: Half-time

Pre-requisite skills: Latex, XML, Python, Html

Lead mentor: Daniele Marinazzo (Ghent University) – CET timezone
Co-mentor: Konrad Kording (University of Pennsylvania) – EST timezone

Project aims and tasks:
We describe the project structure and approximate work as follows

- Create a pipeline to convert latex papers in xml, this can be a good start https://jblevins.org/log/xml-tools (45 hours)
- Adapt the final format to PMC requirements (see above) (30 hours)
- Integrate with the current Scholastica platform (30 hours)
- Explore and prototype a standalone platform (70 hours)

Tech keywords: Latex, XML, Python, Html, DTD (Document Type Definition)

Discuss this project on Neurostars!

20. LORIS open data platform for neuroscience research

20.1 Contribute to LORIS (175/350 h)

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, imaging pipelines, automated testing, API development.

How to apply: If you might be interested, send a quick email to our Lead Mentor < christine.rogers [at] mcgill.ca > to introduce yourself and share your CV. If our projects could be a good match, we'll walk you through the GSOC proposal process.

Planned effort: 350 (preferred) or 175 hours

Intended skill level: Intermediate, Advanced

Project effort: Full-time (preferred) or Half-time; extended half-time project also possible.

Pre-requisite skills: strong coding foundation, typically in python or javascript or unix-based experience. Some knowledge of psychology/science research, databasing or imaging is an asset.

About LORIS github.com/aces/loris
LORIS is an open-source database for neuroscience research projects and Open Science initiatives.

- Website: Loris.ca
- Try LORIS at: Demo.loris.ca

LORIS databases are used by research projects 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, electrophysiology and more. 3D visualization and advanced data tools provide a dynamic workflow environment for complex neuroinformatics research.

For a few examples, LORIS currently hosts datasets such as the UK Biobank and the BigBrain 3D Atlas (bigbrain.loris.ca) - a high-resolution model of the human brain.

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, and we collectively test and review code at github.com/aces/loris.

We'd love to have you join our diverse team of 20 developers affiliated with the Montreal Neurological Institute and McGill University in Montreal, Canada – We especially encourage those from underrepresented backgrounds and/or interest in neuroscience, medicine or psychology research, bioinformatics, or image/signal processing to apply.
Tech keywords: LORIS, JavaScript, React, REST, NoSQL, databasing, imaging, platform, open science, MRI, EEG

Discuss this project on Neurostars!

21. Orthogonal Research Lab

21.1 A Collective Cognition Model for AI Ethics (350 h)

The need for automated evaluation of real-time data is important in a number of sociotechnical contexts. Our group is looking to develop an auditing system and simulation of collective cognition that will improve open-source community sustainability. This interdisciplinary approach to AI ethics will involve both the development of a homeostatic system that encourages cooperative and altruistic interactions, and using simulated data generated through an agent-based model of open-source behaviors and interactions.

In taking a cybernetic approach, the candidate will build an analytical model that incorporates features such as general feedback loops (recurrent relationships) and causal loops (reciprocal causality). This might be in the form of a traditional boxes and arrows (input-output) model, or something more exotic such as Reinforcement Learning. Applicants might take inspiration from Mick Ashby's ethical regulator (Ethical regulator - Wikipedia 1).

The broader goal is to build a model of cultural evolution that will encourage desired behaviors. The first part of this project will involve building a computational system to model the resources, activities, and interrelationships of an open-source community. The second part of the project will involve simulating this community using an agent-based model, which will provide the candidate with output data necessary to train and benchmark the cybernetic model.

What can I do before GSoC?

You can join the Orthogonal Lab Slack and Github, as well as attend our Saturday Morning NeuroSim meetings. You will work with our Ethics, Society, and Technology group, and interactions with your colleagues is key. You will also want to become familiar with a scientific programming approach (such as Python or Julia) to construct your cybernetic model, as well as the NetLogo platform for building agent-based models.

Requirements

Expertise or the ability to learn Python, Julia, or Kotlin (for the cybernetic model) and Scala and Java (for the agent-based model). The ability to extract model representations from complex systems is helpful. Knowledge of open-source development practices and an interest in interdisciplinary research are a must.

Planned Effort

350 hours.

Mentors: Bradly Alicea (bradly.alicea@outlook.com), Jesse Parent (jtparent2018@gmail.com).

Discuss this project on Neurostars!

22. OpenWorm

22.1 GNNs as Developmental Networks (175 h)

Biological development features many different types of networks: neural connectomes, gene regulatory networks, interactome networks, and anatomical networks. Using cell tracking and high-resolution microscopy, we can reconstruct the origins of these networks in the early embryo. Building on our group's past work in deep learning and pre-trained models, we look to apply graph neural networks (GNNs) to developmental biological analysis.

We seek to create graph embeddings that resemble actual biological networks found throughout development. Potential activities include growing graph embeddings using biological rules, differentiation of nodes in the network, and GNNs that generate different types of movement output based on movement seen in microscopy movies. The goal is to create a library of GNNs that can simulate developmental processes by analyzing time-series microscopy data.

DevoWorm is an interdisciplinary group engaged in both computational and biological data analysis. We have weekly meetings on Jit.si, and are a part of the OpenWorm Foundation. You may also have the chance to work with our DevoLearn (open-source pre-trained deep learning) software, in addition to adding your contributions to the DevoWorm AI library.

What can I do before GSoC?

You can ask one of the mentors to direct you to the data source and you can start working on it. Please feel free to join the OpenWorm Slack or attend our meetings to raise questions/discussions regarding your approach to the problem.

DevoWorm website: link 2

DevoLearn (preprint): link 1

DevoWorm AI: link 4

Skills/requirements

PyTorch/Tensorflow (PyTorch will be preferred because all our other models are on that framework already) Wrangling with video data Building a simple GUI on top of the model to run it on local systems (on Linux/windows/macOS). Basic knowledge of biology and complex networks theory would be helpful.

Planned Effort: 175 hours

Mentors: Bradly Alicea (balicea@illinois.edu), TBA

Discuss this project on Neurostars!

22.2 Digital Microsphere (175 h)

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. You will build a computational tool that allows us to visualize 4D data derived from the surface of an Axolotl embryo.

What can I do before GSoC?

Build basic prototypes for this project and discuss about them with the mentors, then read these papers:

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, 396–412.

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, 214-220.

Skills/requirements

Handling higher dimensional microscopy data (preferably also creating an API to load them as tensors for computation on the GPU). Building an intuitive GUI (or a web interface). Feature extraction (canny edges/thresholding/denoising).

Planned Effort

175 hours

Mentors: Bradly Alicea (balicea@openworm.org), Susan Crawford-Young (susan.crawfordyoung@gmail.com).

Discuss this project on Neurostars!

23. Global Brain Consortium (GBC)

23.1 GBC EEG Open Platform (350 h)

The Global Brain Consortium (GBC) aims to develop an open platform for freely spreading up-to-date methods and certified data among a large community of EEG researcher all around

the world. This platform will also facilitate the standardization of methodologies among researchers and will be a step towards the reproducible research. The present project consists in developing tools to be populated in the platform. They will include programming algorithms for:

- standardization of data storage formats, from the original format stored by the different EEG devices to the newly standard BIDS format.
- 3D interactive visualization of brain topographies as well as connectivity-tensors at the scalp and the cortex, over time or frequency, which can be shown on web browsers.
- Dockerization of tools into existing open-source repositories, like CBRAIN, EBRAIN, LORIS, etc., and dynamic design of interfaces.
- Programming basic code to create multinational norms of EEG cross-spectrum and brain microstates.
- Brain dynamics using neural network programming.
- Developing a multilingual translator for the CBRAIN.
- Programming new EEG inverse methodologies.
- Standardization of pipelines for EEG signal cleaning and artifact rejection.
- Development of platforms for collaborative work in EEG research with different levels of data sharing: a) raw EEG and MRI; b) cross-spectra and Lead Fields, c) only sufficient statistics.

Planned effort: 350 hours.

Skill level: Intermediate, Advanced.

Pre-requisite skills (at least 3 of them): Comfortable with MATLAB, Python, Jupiter Notebooks, Windows, and Linux. Experience with image/video processing and data visualization. Knowledge of JavaScript, VS-coding, and Node-JS. Mathematics, deep learning, differential equations, neural networks.

Lead mentor: Pedro Valdes-Sosa.

Co-mentor: Jorge Bosch-Bayard, Deirel Paz-Linares, Eduardo Aubert-Vazquez, Arturo Orellana, Héctor González

Tech keywords: Matlab, Python, Javascript, Node-JS, Windows/Linux, brain dynamics.

Discuss this project on Neurostars!