GSOC 2024 Ideas list for INCF

Small projects (90 h): 0
Medium projects (175 h) 6
Large projects (350 h): 13
Flexible projects (175/350 h): 13
Total projects: 32

1. TVB projects

1.1 Structural Connectivity editor widget (175/350 h)

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 tailored for a 12-week time window, both full-time and part-time, as the features/pages can be built incrementally.

In the TVB ecosystem there is a new repository called tvb-widgets offering UI widgets for Jupyterlab environments. These widgets are compatible with TVB data formats and able to display them in different forms: either a 2D viewer for time series or a 3D viewer for brain surfaces. The purpose of this project is to implement a new widget which would allow users to edit the connectivity matrices involved in a TVB simulation. Necessary features for this widget: display connectivity matrix, normalize matrix, resect connections, resect nodes, change connection weights, save resulted connectivity. Of course, this new widget has to run in a Jupyterlab notebook as well.

Finally, it would be great to have all the widgets linked into the tvb-ext-xircuits repository which is a Jupyterlab extension based on React JS. At the moment, only the PhasePlaneWidget is linked there, but the rest could be added in a similar manner.

Examples of TVB data formats can be found on Zenodo. Connectivity matrices are available there as well.

Check out our Jupyter notebooks to play with the widgets we have available so far.

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

Skill level: Junior+/mid

Required skills: Python, IPywidgets, React JS, Jupyterlab, Jupyterlab extensions

Time commitment: Flexible (175/350 h)

Lead mentor: Lia Domide

Project website: https://req.thevirtualbrain.org/browse/TVB-2607

Backup mentors: Romina Baila

Tech keywords: Python, IPywidgets, React JS, Jupyterlab, Jupyterlab extensions

Discuss this project on Neurostars!

2. LORIS projects

2.1 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.

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.

Skill level: Intermediate/Advanced

Required 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.

Time commitment: Flexible (175/350 h, 350 preferred)

Lead mentor: Christine Rogers, Rolando Acosta

Project website: Loris.ca

Backup mentors: Laetitia Fesselier

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

Discuss this project on Neurostars!

3. Brian2GeNN projects

3.1 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.

Skill level: Advanced

Required skills: C++, Single Instruction Multiple Thread (SIMT) programming

Time commitment: Full-time (350 h)

Lead mentor: Jamie Knight, Thomas Nowotny

Project website: http://ispc.github.io/

Backup mentors: TBD

Tech keywords: C++, SIMT, GPU

Discuss this project on Neurostars!

4. OpenWorm projects

4.1 Graph Neural Networks (OpenWorm Foundation) (350 h)

Graph neural networks (GNNs) are a potentially powerful method for discovering connectivity in geometrically complex datasets. The DevoWorm group has developed an open-source GNN framework for embryogenetic data called DevoGraph. Developmental GNNs (D-GNNs) allow us to characterize a growing network that undergoes shape transformations along with increases in size. During GSoC 2022, we developed a roadmap for progress in this area, but were not able to develop full integration with our Deep Learning-based pre-trained model (DevoLearn). Ultimately, we aim to tie our D-GNN work into the group's work on embryo networks, developmental connectomes, and embryo differentiation.

During the project period, you will be involved in three activities: 1) refining a means to segment raw data and incorporate it into the DevoGraph pipeline, 2) refining our method for deriving graph embeddings, using techniques from topological data analysis and complex network theory, and 3) more tightly integrating DevoGraph as a network structure discovery module of DevoLearn. Achieving 1) will require refactoring CNN models and understanding biological training datasets. Activities 2) and 3) require the ability to work with mathematical models and associated algorithms. Knowledge of graph and/or network theory is helpful, but not required.

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.

OpenWorm Foundation: https://openworm.org/
DevoWorm website: https://devoworm.weebly.com/
DevoGraph (Github): https://github.com/DevoLearn/DevoGraph
DevoWorm AI: https://devoworm.github.io/DevoWormAi/

Skill level: Advanced

Required skills: All of our existing models are built for PyTorch, so experience with Python and PyTorch/Tensorflow workflows is preferred. The ability to work with datasets, such as segmenting video and generating graph visualizations is essential. An ability to build web interfaces, UI design, basic knowledge of biology, open-source practices, and applied mathematical tools will also be useful.

Time commitment: Full-time (350 hours)

Lead mentor: Bradly Alicea

Project website: https://devoworm.weebly.com/

Backup mentors: Jiahang Li

Tech keywords: GNNs, Computational Biology, Graph Theory, PyTorch

Discuss this project on Neurostars!

5. Orthogonal Research & Education Lab projects

5.1 Virtual Reality for Distributed Research (175 h)

Virtual and Augmented Reality (collectively known as XR) is a perpetually emerging area at the intersection of cognitive neuroscience and computing. A host of tools introduced in the past few years have improved the accessibility of this technology to a host of applications. This project will focus on building applications for research or science communication. The applicant will propose a project based on one of the following templates.

● Replicate a famous experimental paradigm (e.g. Morris Water Maze) where the subject can interact with both the environment and stimuli. This environment should allow for participant data to be collected and will exhibit a high degree of environmental realism.

● Replicate scientific knowledge or biological processes in a unique way. Possible applications include a navigable version of a brain atlas, creating an immersive version of the http://browser.openworm.org/, or an animation that displays a biological process from multiple points of view.

● The ability to visualize Agent-based Models (ABM) using both a first-person and third-person perspective. This involves integrating simulations from the NetLogo platform with a 3-D physics and graphics engine.

The outcome of your project should be an XR application published under an open-source license that runs in a widely available format (360 video, smartphone headset, or Meta Quest). This will allow learners and researchers from around the world to access and interact with these models. We would also prefer projects that allow for and measure haptic feedback and movement inputs.

What can I do before GSoC?

You can join the https://launchpass.com/orthogonal-research, as well as attend our Saturday Morning NeuroSim meetings. You will work within our Principles of Bits to Matter to Mind and PhASiC initiatives, but interactions with your colleagues across the organization 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.

Orthogonal Research and Education Lab: https://orthogonal-research.weebly.com/

Skill level: Beginner/intermediate

Required skills: Knowledge of either Unity or Blender is essential, in addition to an ability to convert models into formats such as 360 video or Meta Quest applications. A willingness to embrace open-source development practices and an interest in interdisciplinary research are essential.

Time commitment: Half-time (175 h)

Lead mentor: Bradly Alicea, Jesse Parent

Project website: https://orthogonal-research.weebly.com/

Backup mentors: TBA

Tech keywords: Virtual Reality, Simulation, Blender/Unity, 3-D Graphics

Discuss this project on Neurostars!

5.2 Open-source Community Sustainability (350 h)

As demonstrated by many organizations, open-source communities can do great things. But this is only true if the contributor community can maintain public goods such as the software codebase and institutional knowledge over time and despite contributor turnover. During GSoC 2023, our group of contributors used an adversarial approach to build several agent-based models (https://github.com/OREL-group/GSoC/tree/main/Open%20Source%20 Ethics). Taken

collectively, this codebase simulates open-source community interactions to encourage conditions leading to sustainable practice. These models have subsequently been incorporated into a web-based auditing tool (https://sustainabilityauditingtool.herokuapp.com/) that presents interactive evaluatory methods for community managers and other open-source project leaders.

For this project, you will maintain and develop the auditing tool backend, help to improve upon the underlying models, collect and analyze data where appropriate, and serve as a platform tester using multiple user personas. While the applicant can select their focus based on personal preference and/or strengths, they will be expected to engage in all aspects of maintenance and project improvement. Our goal is to develop one or more maintainers of the platform who are also capable of research software engineering (https://www.hpcwire.com/off-the-wire/ncsa-innovators-bridging-research-and-software-engineering/).

What can I do before GSoC?

You can join the https://launchpass.com/orthogonal-research, 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 our auditing tool (https://sustainabilityauditingtool.herokuapp.com/ and https://github.com/SAT-7), as well as the NetLogo platform for building agent-based models.

Orthogonal Research and Education Lab: https://orthogonal-research.weebly.com/

Skill level: Intermediate

Required skills: The following languages will be used extensively throughout the project: NetLogo, Python, and Javascript. This project will also involve web development, working with computational models, UI design, and open-source community-building, so experience in these areas is helpful but not required. Knowledge of open-source development practices and an interest in interdisciplinary research are a must.

Time commitment: Full-time (350 h)

Lead mentor: Bradly Alicea, Jesse Parent, Brian McCorkle

Project website: https://orthogonal-research.weebly.com/

Backup mentors: TBA

Tech keywords: Open Source Communities, Agent-based Models, NetLogo, Python, Javascript

Discuss this project on Neurostars!

6. Neuroptimus projects

6.1 Improving and extending the user interface of the Neuroptimus parameter optimization software tool (175/350 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 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.

The aim of the current project is to extend the existing user interfaces (GUI and command-line) of Neuroptimus with currently missing functionality that would make them more informative, flexible, and user-friendly. Examples of added functionality would include the following: (1) The ability of the user to save and load all the settings of the optimization at any point in the GUI. (2) The ability to monitor the progress of the optimization while it is running (solutions could range from a simple progress bar to a more sophisticated monitoring tool). (3) Saving the results of the optimization in formats that are more easily interpretable by either humans or computers. (4) Analysis and visualization of the final and intermediate results of the optimization in the space of optimized features and individual parameters.

Skills and effort: Intermediate

Required skills: At least intermediate coding skills. Comfortable with Python. Experience with visualization and user interface design would be beneficial.

Time commitment: Flexible (175/350 h)

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

Discuss this project on Neurostars!

6.2 Integration of simulation-based inference methods into the Neuroptimus parameter optimization software tool (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 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.
Simulation-based inference is a powerful emerging method for parameter estimation, which has been used successfully in a neuroscientific context, but is not currently available in Neuroptimus.
The aim of the current project is to extend the set of parameter estimation methods available in Neuroptimus (which aim to find a single best solution) by adding methods of simulation-based inference, which aim to determine the probability distribution of the parameters from the data, and enable the quantification of uncertainty and correlations in the estimated parameters. Methods would include Bayesian parameter inference based on the specification of the likelihood function and the prior distribution, solved via direct or approximate methods such as Markov-chain Monte Carlo (MCMC) or variational inference. Likelihood-free methods would also be included. Integration of these methods would rely on existing Python packages, and would leverage previous experience in the group with using these approaches.

Skills and effort: Intermediate

Required skills: At least intermediate coding skills. Comfortable with Python. Experience with probabilistic inference or basic AI/ML concepts and tools would be beneficial.

Time commitment: Full-time (350 h)

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

Tech keywords: Python, AI, ML

Discuss this project on Neurostars!

7. ImageJ projects

7.1 UI update for ASP/IJ (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 be applied directly to a dataset or called from your Java.

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, exposing 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.

Skill level: Intermediate

Required skills: Java, Machine learning.

Time commitment: Half-time (175 h)

Lead mentor: Dimiter Prodanov (dimiterpp@gmail.com), INCF Belgian Node

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

Backup mentors: Sumit Vohra, ZIB, Berlin, Germany (backup); Teodor Vakarelsky; IICT -BAS (backups)

Tech keywords: Image Segmentation

Discuss this project on Neurostars!

7.2 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 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 last GSOC 2023 project implemented a GUI integrating SQLite as a proof of concept. Computed image features and class memberships are stored in a general SQLite database. However, the present table structure is not optimal and can be improved.

The project idea: At present, the feature space and the classification results produced by the platform are stored in several separate files. The candidate should redesign the database and implement necessary changes at the UI level.

Tasks
- Fix existing issues and bugs
- SQL database design
- UI redesign

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.

Skill level: Intermediate

Required skills: Java, SQL

Time commitment: Full-time (350 h)

Lead mentor: Dimiter Prodanov (dimiterpp@gmail.com), INCF Belgian Node

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

Backup mentors: Sumit Vohra, ZIB, Berlin, Germany

Tech keywords: Image Segmentation

Discuss this project on Neurostars!

8. NetPyNE projects

8.1 Automated In-Silico Representation of Published Literature (175/350 h)

NetPyNE is a high-level graphical and python scripting interface to NEURON that describes multiscale models of brain neural circuits in a declarative, json-like syntax that can then be simulated in-silico via the NEURON simulator. NetPyNE's toolkit allows users to declare--within a single framework--interactions between molecules (reaction diffusion modeling), synapses, channels, morphologies, and networks exemplified in our published, publicly available large-scale, biologically realistic models of rodent motor (M1), rodent somatosensory (S1), and macaque auditory (A1) thalamocortical circuits. NetPyNE is used in over 100 published models developed over 40 institutions worldwide, from smaller networks that can be run on user machines to models encompassing 50 million synaptic interactions run on high powered

computing centers. NetPyNE additionally integrates with multiple popular neuroscience standards, toolkits and platforms including the NeuroML, SONATA formats, CoreNEURON, LFPykit, HNN, TVB, OSB, EBRAINS, and SciUnit.

NetPyNE's toolkit functionality is continuously extended based on use-cases in patho/physiological modeling and signal analysis tasks. To this end, automated tools for the aggregation of in-vivo/vitro experimental datasets and conversion of these datasets to appropriate parameters for in-silico modeling represent a research and development task for 2024. This includes development of data crawlers and domain specific language models to extract relevant information from existing datasets and conversion to a common standard and declarative syntax that can then be used to generate a corresponding in-silico model. For GSoC our project involves development of a proof of concept that converts Allen Brain datasets to parameters for comparison against NetPyNE validated M1/S1/A1. Of note, these aims represent larger project aims and can be tailored to the applicant's background expertise / time constraints/ interest.

Aims:

1. Design toolkit for automated extraction of parameters from Allen Brain datasets.
2. Design toolkit for conversion of extracted parameters to NetPyNE syntax.
3. Validation of extracted parameters via in-silico simulation.

Skill level: Flexible

Required skills: Python; open source software development. Other useful skills: background in computational/theoretical neuroscience and/or modeling experience; experience with NEURON/NMODL/hoc.

Time commitment: Flexible (175/350 h)

Lead mentor: Valery Bragin, Eugenio Urdapilleta, James Chen, Salvador Dura-Bernal

Project website: TBA (Discuss on Neurostars)

Backup mentors: TBD

Tech keywords: Python, open source, Data Science, Automation, NEURON, NetPyNE

Discuss this project on Neurostars!

8.2 Improving ML/AI Parameter Tuning/Optimization of Electrophysiological Models (175/350 h)

NetPyNE is a high-level graphical and python scripting interface to NEURON that describes multiscale models of brain neural circuits in a declarative, json-like syntax that can then be simulated in-silico via the NEURON simulator. NetPyNE's toolkit allows users to declare--within a single framework--interactions between molecules (reaction diffusion modeling), synapses, channels, morphologies, and networks exemplified in our published, publicly available large-scale, biologically realistic models of rodent motor (M1), rodent somatosensory (S1), and macaque auditory (A1) thalamocortical circuits. NetPyNE is used in over 100 published models developed over 40 institutions worldwide, from smaller networks that can be run on user machines to models encompassing 50 million synaptic interactions run on highly parallelized computing centers. NetPyNE additionally integrates with multiple popular neuroscience standards, toolkits and platforms including the NeuroML, SONATA formats, CoreNEURON, LFPykit, HNN, TVB, OSB, EBRAINS, and SciUnit.

To aid in model development, NetPyNE has developed multiple python tools for scalable model tuning within our "batch" subpackage. This subpackage is continually improved upon based on new hardware infrastructures and algorithms and based on end-user needs including model tuning for large multiscale models on increasingly available high powered computing platforms. To this end, part of 2024 will involve research and development into improving the "batch" subpackage and exploring the efficacy of already implemented/ to be implemented AI/ML algorithms in parameter optimization for large scale models, using parameter prediction against our M1, S1 and A1 circuits as algorithm testbenches. Of note, these aims represent larger project aims and can be tailored to the applicant's background expertise / time constraints/ interest.

Aims:

1. Improving fitness reporting/ data aggregation & visualization for batch automation
2. Validating new batch scripts on multiple hardware platforms
3. Testbench existing / to be implemented algorithms on tuning large scale models
4. Implement new AI/ML algorithms within NetPyNE batch subpackage.

Skill level: Flexible

Required skills: Python; open source software development. Other useful skills: background in computational/theoretical neuroscience and/or modeling experience; experience with NEURON/NMODL/hoc.

Time commitment: Flexible (X h)

Lead mentor: Valery Bragin, Eugenio Urdapilleta, James Chen, Salvador Dura-Bernal

Project website: TBA (Discuss on Neurostars)

Backup mentors: TBD

Tech keywords: Python, open source, Data Science, Automation, NEURON, NetPyNE

Discuss this project on Neurostars!

9. Eye-tracking projects

9.1 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.

In this AI for health project, that continues work from GSoC 2022 and GSoC 2023, we will address these limitations by developing a deep-learning based eye-tracker to test infant visual function by automatic monitoring of their eye-movements. Previous projects made progress towards developing an eye-tracker module that can work both with software-based and hardware-based eye-trackers and with developing a visual stimulus presentation module. Both these modules will be developed further in this year's GSoC so that a full proof of concept can be developed. The project involves a) the development of an application with a suite of visual stimuli and analytical procedures to probe multiple visual functions; b) testing and further developing a deep-learning based infant eye-tracker and c) developing a GUI and controller that holds the display, eye-tracking and analysis components together.

Skill level: Intermediate/advanced

Required skills: Comfortable with Python. Experience with image/video processing and using deep-learning based image-processing models. Ideally, also comfortable with Android/iOS app development and especially ARKit/equivalent, but not necessary.

Time commitment: Full-time (350 h, large project)

Lead mentor: Arvind Chandna (NeuroStars ID: @arvindchandna)

Project website: https://github.com/Shazam213/automated-preferential-looking

Backup mentor: Suresh Krishna (NeuroStars ID: @suresh.krishna), Soham Mulye (mulyesoham@gmail.com)

Tech keywords: Health AI, Infant vision, Image processing, App development, Python, health AI, IOS/Android, ML based eye-tracking, PyTorch, Deep Learning

Discuss this project on Neurostars!

10. Motion Capture projects

10.1 Using markerless motion capture to drive music generation and develop neuroscientific/psychological theories of music creativity and music-movement-dance interactions (350 h)

This is a new exploratory project that aims to use the recent AI-based advances in markerless motion capture software (e.g. MediaPipe from Google) to create a working framework whereby movements can be translated into sound with short-latency, allowing for example, gesture-driven new musical instruments and dancers who control their own music while dancing. The project will require familiarity with Python, and ability to interface with external packages like MediaPipe. Familiarity with low-latency sound generation, image processing, and audiovisual displays is an advantage, though not necessary. The development of such tools will facilitate both artistic creation, as well as scientific exploration of multiple areas, including for example - how people engage interactively with vision, sound, and movement and combine their respective latent creative spaces. Such a tool will also have therapeutic/rehabilitative applications in populations of people with limited ability to generate music and in whom agency and creativity in producing music have been shown to produce beneficial effects.

Skill level: Intermediate/advanced

Required skills: Comfortable with Python and modern AI tools. Experience with image/video processing and using deep-learning based image-processing models. Familiarity with C/C++ programming, low-latency sound generation, image processing, and audiovisual displays, as well as MediaPipe is an advantage, though not necessary.

Time commitment: Full-time (350 h, large project)

Lead mentor: Suresh Krishna (NeuroStars ID: @suresh.krishna)

Project website: github.com/m2b3/Gestures

Backup mentor: Yohai-Eliel Berreby

Tech keywords: Sound/music generation, Image processing, Python, MediaPipe, Wekinator, AI, Deep-learning

Discuss this project on Neurostars!

11. Open Science Publishing projects

11.1 A social-web tool to facilitate rating and commenting on research reports (350 h)

It is generally agreed that the 10 billion USD research/science publishing industry represents a bad way to expend scarce research funds. Funded by university budgets and grants to researchers, and based on the volunteerism of the research community, this market diverts substantial sums from the research activity itself, while putting up barriers to access to the research literature. Current open-access mandates merely transfer the costs from reader to authors and/or their funding/employing organizations. Research journals provide objective peer-review and an article rating and filtering mechanism. In last year's GSoC project, we leveraged modern internet-based social technology to create an open reviewing and quality-ranking web portal that, if adopted, will drastically improve research discourse, functioning and accessibility.

The project involves the further development of the portal and the addition of several features, including those based on LLMs and NLP AI. Familiarity with Django/React as well as Python will be ideal. The system will eventually be tested with the Aperture Neuro open access publishing

platform of the Organization for Human Brain Mapping (OHBM), in consultation with that community.

Skill level: Intermediate/advanced

Required skills: Comfortable with Python, Django, React (or close equivalents). Experience with web technology – React/equivalent Javascript library, front-end programming, back-end programming with databases.

Time commitment: Full-time (350 h, large project)

Lead mentor: JB Poline, Jyothiswaroop Bommareddy

Project website: https://www.scicommons.org/

Backup mentor: Suresh Krishna (NeuroStars ID: @suresh.krishna)

Tech keywords: Science publishing, social web, science portals, Natural language processing, large language models, AI.

Discuss this project on Neurostars!

12. Frites projects

12.1 Writing a R package for the computation of the O-information (350 h)

Real-world systems are often characterized by higher-order interactions (HOIs) within multiplets i.e. groups of three or more units (Battiston et al., 2021). In neuroscience, most pieces of evidence we have about brain networks come from the interactions between pairs of brain regions but little is known about what type of information remains hidden in the non-pairwise interactions (Luppi et al. 2023, Luppi et al. 2024). Interestingly, recent findings suggest that

HOIs might be a better neural marker of neurodegeneration than standard pairwise approaches (Herzog et al., 2022).

Several methods have been proposed to estimate HOIs, from popular fields like graph- and information-theory. The O-information (short name for “information about Organisational structure”) is an information-theoretical quantity to characterize statistical interdependencies within multiplets of three and more variables (Rosas et al., 2019). It allows us to not only quantify how much information multiplets of brain regions are carrying but also informs us on the nature of the information i.e. whether multiplets are carrying mainly redundant or synergistic information.

Estimating HOIs is computationally intensive. As an example, a cortical parcellation dividing the brain into 80 distinct regions involves estimating HOIs in 80.000 triplets, in 1.5 million quadruplets, in 24 million quintuplets, etc. The computational burden of the O-information only relies on simple quantities like entropies, which makes the O-information an ideal candidate to estimate HOIs in a reasonable time. Still, there is yet no neuroinformatic gold standard to estimate HOIs, in a decent amount of time and accessible to network enthusiasts encompassing experts and non-experts.

Currently an R implementation is missing, limiting the adoption to a relevant part of the community, in particular colleagues working with behavioral data and psychometrics.

Project aims and tasks

This project aims at building a R package, missing at the moment, for the computation of this quantity.

We divided this project into five main tasks:

1) Test current implementation in Matlab and Python
2) Build R functions to compute the Total Correlation and the Dual Total correlation
3) Implement and test statistical validation for the multiplets
4) Data simulation: add a function to simulate HOIs
5) Explore plotting solutions, in R or preparing the output for plotting with existing packages such as XGI
6) Explore interfaces with other R packages used in psychometrics (https://lavaan.ugent.be/ http://psychonetrics.org/ https://cran.r-project.org/web/packages/psychotools/index.html)
7) Prepare a package to be submitted to CRAN

Ultimately, this project could lead to the establishment of a gold standard to go beyond pairwise interactions by measuring HOIs, accessible to R experts such as to users with little programming knowledge.

Skill level: Intermediate/advanced

Required skills: R, some Python

Time commitment: Full-time (350 h)

Lead mentor: Daniele Marinazzo

Project website: https://github.com/brainets/frites, https://github.com/danielemarinazzo/HOI

Backup mentors: Fernando E. Rosas, Pedro Martinez Mediano

Tech keywords: R, Python

Discuss this project on Neurostars!

13. Open Source Brain projects

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

An increasing number of studies are using large scale network models incorporating realistic connectivity to understand information processing in the brain. 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 (from e.g. neocortex, hippocampus or cerebellum) 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 (https://v2.opensourcebrain.org) for reuse, modification and extension.

Aims/objectives:

1. Select a number of large scale 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.

Skill level: Mid+/High

Required skills: computational/theoretical neuroscience and large scale biophysically detailed modelling, Python, experience with NeuroML/NEURON, open source software development (git/GitHub)

Time commitment: Flexible (175/350 h)

Lead mentor: Padraig Gleeson (@pgleeson on GitHub)

Project website:

- https://neuroml.org
- https://github.com/NeuralEnsemble/PyNN/

Backup mentors: Ankur Sinha (@sanjayankur31 on GitHub)

Tech keywords: computational neuroscience, Python, NeuroML, PyNN, networks, modelling, simulation, NEURON

Discuss this project on Neurostars!

13.2 Implementation of SWC to NeuroML converter in PyNeuroML (175h/350h)

The SWC format is a standard used ubiquitously for the representation of neuronal morphologies reconstructed from experiments. NeuroML is a standard for the representation and simulation of biophysically detailed neuronal models. The conversion of SWC to NeuroML, however, is not trivial. A number of checks must be made to ensure that the generated NeuroML model is valid enough for use in simulation. This project will require the implementation of an interactive SWC to NeuroML converter in Python. It will read SWC, convert it into NeuroML,

asking the user various questions along the way while providing them with suggestions on how specific issues may be addressed. This will then be tested and validated against various repositories of neuronal reconstructions (like NeuroMorpho.org). See: https://docs.neuroml.org/Userdocs/ImportingMorphologyFiles.html

Aims/objectives:

1. Understand the SWC and NeuroML specifications (formats/representations) for biophysically detailed multi-compartmental neurons
2. Write a Python package/library that
    a. reads/imports SWC
    b. runs various tests on the SWC representation to check if it is valid for use in modelling
    c. includes suggestions/solutions for various test failures to allow the SWC representation to be used in modelling
    d. writes NeuroML
    e. Includes an interactive tool to allow users to use the functions of the library

Skill level: Low/Mid

Required skills: writing new packages/libraries in Python following best practices; knowledge of open source software development using Git/GitHub
Optional skills: some knowledge of computational neuroscience and modelling will be useful, but is not necessary.

Time commitment: Flexible (175/350 h)

Lead mentor: Ankur Sinha (@sanjayankur31 on GitHub)

Project website:
- https://neuroml.org

Backup mentors: Padraig Gleeson (@pgleeson on GitHub)

Tech keywords: Python, SWC, NeuroML, computational neuroscience

Discuss this project on Neurostars!

13.3 Incorporate new features into an advanced, cross-platform 3D viewer for NeuroML cells and networks (175h/350h)

NeuroML is a standard for the representation and simulation of biophysically detailed neuronal models. The primary Python package for using NeuroML, PyNeuroML (https://docs.neuroml.org/Userdocs/Software/pyNeuroML.html), recently incorporated a new Vispy (https://vispy.org) based viewer that allows interactive 3D visualisation of NeuroML networks and detailed cells. This project will extend and improve this 3D viewer to make it more interactive to allow users to gain information about the model that they are visualising, similar to features implemented in neuroConstruct (http://www.neuroconstruct.org/) and Open Source Brain v1 (https://v1.opensourcebrain.org).

Aims/objectives:

1. Extend current Vispy PyNeuroML viewer to make it more interactive
    a. Show information about model features “on click” or “on selection” of entities in the visualization
    b. Give easy access to views of cells showing ion channel distributions, segment groups, etc.
    c. Include other information about network models, such as connectivity
2. Ensure performance with Vispy viewer to allow visualisation of large scale networks.
3. Embed Vispy viewer into NeuroMLLite-GUI (https://docs.neuroml.org/Userdocs/Software/NeuroMLlite.html)
4. Investigate options for use of Vispy viewer inside Jupyter notebooks/JupyterLab

Skill level: Mid

Required skills: writing packages/libraries in Python following best practices; knowledge of open source software development using Git/GitHub; ability to understand APIs using documentation and examples (PyNeuroML/Vispy)
Optional skills: some knowledge of computational neuroscience and modelling will be useful, but is not necessary.

Time commitment: Flexible (175/350 h)

Lead mentor: Ankur Sinha (@sanjayankur31 on GitHub)

Project website:

- https://neuroml.org
- https://docs.neuroml.org/Userdocs/VisualisingNeuroMLModels.html

Backup mentors: Padraig Gleeson (@pgleeson on GitHub)

Tech keywords: Python, NeuroML, Vispy, PyQT

[Discuss this project on Neurostars!]

14. INCF projects

14.1 INCF Impact Visualization Portal with ML and Data Analytics (175 h)

The International Neuroinformatics Coordinating Facility (INCF) is a non-profit organization dedicated to advancing neuroscience and neuroinformatics. Their mission involves promoting open, fair, and easily citable standards while providing support to researchers worldwide. INCF's projects encompass a wide range of areas, from predicting Alzheimer's and detecting cancer cells to mapping the brain in 3D, developing connectomes, and analyzing EEG data. Given this extensive and impactful work, there is a pressing need for an online portal to effectively track INCF's progress and showcase its global contributions using simple language. This portal is thoughtfully designed to make INCF's achievements accessible to a broad audience, regardless of their familiarity with technical terminology. Leveraging the capabilities of large language models and Langchain, a comprehensive dashboard is created. This user-friendly, interactive, and visually appealing platform serves to highlight research progress and the real-world impact on people's lives. It aims at ultimately raising awareness about the field of neuroinformatics and INCF's pivotal role in advancing neuroscience.

Proposed Ideas

Collaborative Filtering for Project Recommendations:

- Implement a recommendation system using collaborative filtering to suggest projects to users based on their interests and past interactions.
- **Implementation**: Use collaborative filtering algorithms like Singular Value Decomposition (SVD). Implement recommendation systems using scikit-surprise.
- **Tools**: Surprise library, scikit-surprise.

Sentiment Analysis with User Feedback:

- Create a user feedback system and integrate sentiment analysis to analyze user comments and feedback on projects and organizations. This can provide valuable insights into the community's perception and satisfaction. Natural Language Processing (NLP) techniques can be applied to understand the sentiment behind user reviews and comments.

- **Techniques**: Utilize Natural Language Processing (NLP) techniques, specifically sentiment analysis.
- **Tools**: NLTK or spaCy for NLP preprocessing, and libraries like TextBlob or transformers (Hugging Face) for sentiment analysis.

Dynamic Visualization of Impact Metrics:

- Implement dynamic visualizations using data science libraries (e.g., Matplotlib, Plotly) to showcase impact metrics over time. Allow users to interact with the visualizations to explore trends and patterns in the impact of different projects or organizations.
- **Techniques**: Time-series analysis using methods like moving averages, trend analysis, and anomaly detection.
- **Tools**: Matplotlib and Seaborn for static visualizations, Plotly or Bokeh for interactive visualizations.

Semantic search:

- Enhance the search functionality by allowing users to search using natural language queries instead of just keywords. This would utilize NLP to understand the intent behind the query and retrieve relevant results even if the exact keywords aren't used.
- **Implementation**: Making use of models like BERT for vector representations. Further use of algorithms like cosine similarity for promoting the search of relevant projects.
- **Tools**: Transformers Library (Hugging Face),Annoy or Faiss.

Multilingual support:

- Extend the website's capabilities by using multiple languages, making the portal more accessible to a wider audience.
- **Implementation**: Add a user interface element, like a language switcher, that allows users to select their preferred language.
- **Tools**: Google Speech_recognition, Microsoft Translator, or DeepL.

Any new ideas that can improve the overall system are welcome as well as this project is yet in POC stage.

Skill level: Medium

Project Length: Medium (175 h)

Lead Mentor: Atharva Purohit

Backup Mentor: Arnab Banerjee (arnab1896@gmail.com)

Tech Keywords: Python, Machine learning, Deep Learning, Django

Project URL(GitHub): https://github.com/atharva434/INCF-Impact-visualization-Portal

Discuss this project on Neurostars!

15. AnalySim projects

15.1 Implementing new features for project dashboard, notebook management, and commenting. Improving security of the user registration system, usability of CSV data browser and querying components. (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. In this GSoC iteration, we are improving on features developed in the summer of 2023.

Project is still in progress and a demo site is available at: https://analysim.tech

Source code: https://github.com/soft-eng-practicum/AnalySim

Main Technologies: Angular (Typescript), HTML/CSS/Bootstrap, ASP.Net Core (C#), PostgreSQL

Technologies for analysis notebooks: JavaScript (ObservableHQ, D3.js, Vega, Plotly) and Python (Jupyter)

Skill level: intermediate/advanced preferable

Time commitment: Full-time (350 h)

Lead mentor: Cengiz Gunay (cengique@gmail.com)

Project website: https://analysim.tech

Backup mentors: Anca Doloc-Mihu (adolocm@gmail.com)

Mentor break: *Planned break for both mentors due to CNS meeting: June 28 - July 24 still available, but with limited email contact.*

Discuss this project on Neurostars!

15.2 Working on a more consistent visual user interface style and improved user experience. Design an optimal dashboard layout along with other necessary pages. (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. In this GSoC iteration, we are improving on features developed in the summer of 2023.

Project is still in progress and a demo site is available at: https://analysim.tech

Source code: https://github.com/soft-eng-practicum/AnalySim

Main Technologies: Angular (Typescript), HTML/CSS/Bootstrap, ASP.Net Core (C#), PostgreSQL

Technologies for analysis notebooks: JavaScript (ObservableHQ, D3.js, Vega, Plotly) and Python (Jupyter)

Skill level: intermediate/advanced preferable

Time commitment: Half-time (175 h)

Lead mentor: Anca Doloc-Mihu (adolocm@gmail.com)

Project website: https://analysim.tech

Backup mentors: Cengiz Gunay (cengique@gmail.com)

Mentor break: *Planned break for both mentors due to CNS meeting: June 28 - July 24 still available, but with limited email contact.*

Discuss this project on Neurostars!

16. HNN-core projects

16.1 Implement batch simulation and optimization routines (350 h)

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.

IRC channel: https://gitter.im/jonescompneurolab/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
- HNN-SBI preprint

Goal

HNN-core currently lacks the ability to simulate large batches of simulations which is critical for parameter optimization. The goal is to add this functionality and develop tutorials on optimization techniques that leverage batch simulations, namely simulation based inference (SBI), a deep learning based Bayesian inference method.

Subgoals

- Develop batch simulation functionality to facilitate parameter sweeps over a range of model parameters
- Create tutorials demonstrating how to use simulation based inference (SBI) with HNN-core

- Consolidate the different optimization functions as much as possible, producing a clear API with minimal redundancies. This should also allow the user to constrain parameter ranges and run simple parameter sweeps by specifying or eliminating the optimization cost function.
- Create a function for visualizing the parameter changes pre-to-post optimization.
- Document the optimization routines with examples and develop tests for each function.

Related issues: https://github.com/jonescompneurolab/hnn-core/issues/140
https://github.com/jonescompneurolab/hnn-core/issues/176
https://github.com/jonescompneurolab/hnn-core/issues/423
https://github.com/jonescompneurolab/hnn-core/issues/567

Skill level: Intermediate

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

Time commitment: Full-time (350 hours)

Lead mentor: Nicholas Tolley

Project website: https://hnn.brown.edu/

Backup mentors: Ryan Thorpe, Mainak Jas

Tech keywords: Python, networks, modeling, simulation

Discuss this project on Neurostars!

17. Physiopy projects

17.1 Semi-Automated Workflows for Physiological Signals (175 h)

Physiopy is an international community formed around developing solutions to acquire, process, and utilize physiological files (e.g. cardiac, respiratory, etc) in neuroimaging contexts (e.g functional magnetic resonance imaging, fMRI). For the past five years, Physiopy has been developing several physiology-oriented modular toolboxes to this end, including 1) *phys2bids*, a toolbox to standardize physiological files in BIDS format 2) *peakdet*, a toolbox for automatic detection and manual correction of peaks in physiological data and 3) *phys2denoise*, a toolbox to prepare derivatives of physiological data for use in fMRI denoising. Currently, we have no complete workflows

encompassing all steps of physiological data processing and model estimations. The goal of this project is to update toolboxes and facilitate a unified workflow across these for semi-automated physiological signal processing.

Tasks:

- Update and upgrade the current codebase of *peakdet* and *phys2denoise*, including better harmonization between toolboxes
- Create a semi-automated workflow based on *peakdet* and *phys2denoise* to process respiratory and cardiac signals and obtain regressors to model physiological signal variance for neuroimaging analysis (e.g. respiratory volume per time, heart rate variability, etc)
- Improve accessibility of toolboxes through detailed documentation

Minimal set of deliverables:

- Update and implement a single object class shared between *peakdet* and *phys2denoise*
- Create a command line interface (CLI) for the workflows
- Update documentation/tutorials for the toolboxes on Read the Docs

Optional aims:

- Create a graphical report of the workflows output
- BIDS-App-lify the workflows: we want to create an entry point to transform the workflow into a BIDS Application
- Pydra-ify the libraries: to learn the strength of Pydra as a workflow manager, we can create a version of the workflow using Pydra.
- Add support for eye-tracking and skin conductance data
- Contribute to specification of BIDS standards for physiological data derivatives

For this project, experience working with physiological and/or neuroimaging data are helpful but not necessary. We follow the all-contributors specification to report contributions, and adopt physiopy's contributors guide and code of conduct.

What can I do before GSoC?

You can join the Physiopy mailing list and Slack by emailing physiopy.community@gmail.com, as well as attend our monthly community meetings. You can become familiar with the existing Physiopy toolboxes and/or the use of physiological data in brain imaging by following some of our tutorials.

Expected results: A workflow for processing physiological signals with up-to-date documentation/tutorials and unit test coverage.

Skill level: Intermediate

Required skills: Python

Time commitment: Half-time (175 h)

Lead mentor: Mary Miedema, Marie-Eve Picard, Stefano Moia

Project website: https://github.com/physiopy/

Backup mentors:

Tech keywords: Python, open source, data analysis, signal processing, biomedical

Discuss this project on Neurostars!

18. Brian Simulator projects

18.1 Brian2Wasm: simulations in the browser (175h/350h)

Brian is a clock-driven spiking neural network simulator which is easy to learn and use, highly flexible and easy to extend. It is written in Python and allows describing and running arbitrary neural and synaptic models without having to write code in any other programming language. It is built on a code-generation framework that transforms the model description into efficient low-level code. WebAssembly is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web.

Brian's code generation pipeline can generate C++ code, and the emscripten toolchain can compile this code into WebAssembly. This makes it possible to run simulations directly in a web browser, requiring no installation on the client side. We have recently created a proof-of-concept, brian2wasm (https://github.com/brian-team/brian2wasm), showing the feasibility of this approach.

The aim of this project is to turn this proof-of-concept into a convenient tool for the wider community. In particular, the aims of this project are to:

- Improve the general workflow (e.g. detection/configuration of emscripten toolchain)
- Implement efficient data transfer between the WebAssembly simulation and the JavaScript code that displays the results
- Provide convenient tools for website customization and plot display
- Document the package and its usage

Additional goals for a 350-h project:

- Set up an automatic test suite for brian2wasm

- Provide convenient tools for setting parameters at the start of the simulation (big project)
- Provide components for real-time interaction with Brian scripts, e.g. providing microphone or camera input to the Brian simulation and returning spikes or other data from the Brian simulation to the website (big project)

Skill level: Novice/Intermediate

Required skills: HTML, JavaScript, basic C++ and Python

Time commitment: Flexible (175/350 h)

Lead mentor: Marcel Stimberg

Project website: https://github.com/brian-team/brian2wasm

Backup mentors: Dan Goodman

Tech keywords: HTML, JavaScript, WebAssembly, emscripten, Python, C++

Discuss this project on Neurostars!

18.2 Improve Brian's markdown exporter (175 h)

Brian is a clock-driven spiking neural network simulator which is easy to learn and use, highly flexible and easy to extend. It is written in Python and allows describing and running arbitrary neural and synaptic models without having to write code in any other programming language. It is built on a code-generation framework that transforms the model description into efficient low-level code. While this code generation is commonly used to generate code to run a simulation, it can also be used to generate a description of a simulation. This approach is implemented in the Markdown exporter that is part of the brian2tools package (https://brian2tools.readthedocs.io). The aim of this project is to extend this exporter in the following ways:

- add easy customization of the output via templates (including the necessary code reorganisation and refactoring)
- integrate with pandoc to transform the markdown output into other formats such as LaTeX or HTML
- add support for Brian's multi-compartment neuron models

Skill level: Novice/Intermediate

Required skills: Markdown, Python

Time commitment: Medium (175 h)

Lead mentor: Ben Evans

Project website: https://github.com/brian-team/brian2tools

Backup mentors: Marcel Stimberg

Tech keywords: markdown, python, pandoc

Discuss this project on Neurostars!

18.3 Replace Brian's just-in-time compilation mechanism (350 h)

Brian is a clock-driven spiking neural network simulator which is easy to learn and use, highly flexible and easy to extend. It is written in Python and allows describing and running arbitrary neural and synaptic models without having to write code in any other programming language. It is built on a code-generation framework that transforms the model description into efficient low-level code. In Brian's "runtime mode", the generated code can interact with Python code by accessing the same data structures in memory. To make this work seamlessly, Brian makes use of Cython. This approach comes with two major disadvantages: 1) Cython compilation is slow (it generates a lot of code for error checking, etc.). For Cython, this is not a big downside, since it is commonly used to compile libraries once, but for Brian it matters since it needs to compile dynamically generated code frequently 2) We need to maintain a third code generation target besides Python and C++, with small but non-trivial differences to both of them. The aim of this project is to:

- Replace the Python data structures that are currently used from within Cython code (dynamic arrays and the "spike queue") by C++ equivalents
- Research solutions to call the compiled C++ code from Python and make it directly access the memory in the shared data structures storing the simulation variables. This could build upon existing just-in-time compilation technologies such as numba, or a package such as scipy-weave.
- Implement the above solution, and refactor the current code to make use of it.

Skill level: Intermediate/Advanced

Required skills: Python, C++

Time commitment: Full-time (350 h)

Lead mentor: Marcel Stimberg

Project website: https://github.com/brian-team/brian2

Backup mentors: Dan Goodman

Tech keywords: Python, C++, compilation, JIT

Discuss this project on Neurostars!

18.4 Package Brian's unit system (175h/350h)

Brian is a clock-driven spiking neural network simulator which is easy to learn and use, highly flexible and easy to extend. It is written in Python and allows describing and running arbitrary neural and synaptic models without having to write code in any other programming language. One of Brian's features is that it comes with a system to express quantities with physical dimensions, and will automatically check the consistency of these dimensions when such quantities are combined. While there are a number of other Python packages with similar features, Brian's unit system has a few unique features and could therefore be useful to the wider community. The aim of this project is to turn Brian's unit support into a library that is distributed separately from Brian. This is an interesting opportunity to learn about Open Source development, and the infrastructure around package development (documentation, tests, packaging, …). The specific aims of this project are:

- Package the code from brian2.units into an independent package
- Set up the framework for tests and documentation, adapting the existing tests and docs
- Revisit/refactor the current package structure if necessary, and decide upon reasonable defaults for the provided units
- Systematically test the performance for common use cases

Additional goals for a 350h project:

- Implement useful features from other unit packages, e.g.:
    - integration with matplotlib's unit system (pint)
    - string formatting (pint, Quantiphy)
    - type annotations (astropy units)
    - integration with pandas (pint-pandas)

Skill level: Novice/Intermediate

Required skills: Python

Time commitment: Part-time or full-time

Lead mentor: Dan Goodman

Project website: https://github.com/brian-team/brian2/

Backup mentors: Marcel Stimberg

Tech keywords: Python, packaging

Discuss this project on Neurostars!

19. Neurobagel projects

19.1 An LLM-assisted service for annotating research data with machine-understandable, semantic data dictionaries (175/350 h)

Neurobagel builds tools to provide a way for researchers and other data users to define and find cohorts of individuals across a federated ecosystem of data nodes. These tools are developed with the goal of making annotation, integration, and searching of datasets easier. With the increasing popularity of standardized vocabularies for dataset harmonization, dataset variables can be annotated with machine-understandable tags based on a globally accessible standard. Unfortunately, this annotation process needs human experts who know the data well and can be technically challenging and tedious if done entirely manually. We hope to improve the user experience by assisting human experts in the data standardization process with LLM-based recommendations.

Neurobagel nodes use a graph database to store harmonized datasets for query federation. To participate in the Neurobagel federation, datasets must conform to Neurobagel's data model. Currently, Neurobagel provides a web-based annotation tool that features a graphical interface enabling users to manually upload a tabular data file to annotate a dataset for inclusion in the graph. This manual annotation can become cumbersome, especially with large datasets or cohorts.

The goal of this project is to develop an LLM-driven agent that creates an initial first-pass annotation for the user to check and correct. The agent will interface both with the ingested dataset as well as the employed standardized vocabularies to identify the most likely coding of the provided data. This assistant will be capable of interpreting uploaded files, offering recommendations for mapping dataset columns to variables modeled by Neurobagel, applying suitable heuristics, and identifying any missing values. The ability to audit the decision process

will facilitate continuous improvements to the service. The user will then be able to override the initial coding as necessary to accurately reflect the data.

The tasks involved in this project include:

- Becoming acquainted with the annotation tool's codebase
- Exploring LLMs and relevant libraries, such as LangChain, Ollama
- Embedding the automated assistant within the annotation tool for a more efficient process. Given the flexible time commitment, this task would only be part of the project for a contributor who would like to spend the full 350 h with us

What can I do before GSoC?

Check out Neurobagel's website and GitHub organization to familiarize yourself with the relevant tools and codebases. Please feel free to reach out to one of the mentors through email (Brent and Arman) to raise questions/discussions that you may have about the project.

Skill level: Beginner / Intermediate

Required skills: Python or JavaScript/TypeScript

Helpful skills: Basic understanding of Linux command line, Git, Docker, network requests / API calls via HTTP

Time commitment: Flexible (175/350 h)

Lead mentors:

- Brent McPherson (bcmcpher@gmail.com)
- Arman Jahanpour (armanjahanpour7@gmail.com)
- Sebastian Urchs (sebastian.urchs@gmail.com)
- Alyssa Dai (alyssa.ydai@gmail.com)

Project website: https://neurobagel.org/

Backup mentors: Members of the Neurobagel team and the Origami Laboratory at McGill

Tech keywords: Python, JavaScript, TypeScript, React, Large Language Models, Artificial Intelligence, Knowledge Graph

Discuss this project on Neurostars!

19.2 A natural language interface for querying federated research data (175h/350h)

Neurobagel is a federated data ecosystem that allows researchers and other data users to find and consume research data that has to remain at their original institute for data governance reasons. To make this possible, Neurobagel provides tools that make annotation, integration, and searching of data easier, and maintains a common data model that allows for federating data queries. However, the scope of query options can be daunting for a user, and obtaining the desired results often requires iteration. Making the search process more accessible and conversational would motivate people to use and contribute back to the federated harmonization ecosystem of Neurobagel, ultimately benefiting all users.

The Neurobagel cohort query workflow allows users to search for cohorts of individual research participants across federated data nodes hosted at each participating institute. Each Neurobagel node consists of a graph database for data storage and an API that exposes specific query parameters and controls what results a user can see. Currently, Neurobagel provides a graphical web query interface that communicates with the node APIs on behalf of the user, making complex queries easier to formulate. We hope to improve the user query experience further by providing a LLM chatbot-style interface to populate queries and elaborate on search results.

Leveraging the existing Neurobagel cohort query workflow, this project aims to create a chatbot using existing large language models (LLMs) for parsing user-provided text into accurate queries and reliably summarizing the results to the user. At a high level, this chatbot should be capable of receiving and understanding user prompts in natural language, initiating the corresponding API calls using predefined Neurobagel parameters (minimum age, maximum age, sex, etc.), interpreting the results, and conveying that information to the user. Ideally, open tools and models can be selected to provide flexible hosting options.

The tasks involved in this project include:

- Getting familiar with the codebase of existing tools, including the API and cohort query tool
- Exploring LLMs and relevant libraries, such as LangChain, Ollama
- Identify a model and sequence of prompts that can generate accurate API calls for the project
- Developing a simple user interface for the agent. Given the flexible time commitment, this task would only be part of the project for a contributor who would like to spend the full 350 h with us

What can I do before GSoC?

Check out Neurobagel's website and GitHub organization to familiarize yourself with the relevant tools and codebases. Please feel free to reach out to one of the mentors through email (Brent and Arman) to raise questions/discussions that you may have about the project.

Skill level: Beginner / Intermediate

Required skills: Python or JavaScript/TypeScript

Helpful skills: Basic understanding of Linux command line, Git, Docker, network requests / API calls via HTTP

Time commitment: Flexible (175/350 h)

Lead mentors:

- Brent McPherson (bcmcpher@gmail.com)
- Arman Jahanpour (armanjahanpour7@gmail.com)
- Sebastian Urchs (sebastian.urchs@gmail.com)
- Alyssa Dai (alyssa.ydai@gmail.com)

Project website: https://neurobagel.org/

Backup mentors: Members of the Neurobagel team and the Origami Laboratory at McGill

Tech keywords: Python, JavaScript, TypeScript, React, Large Language Models, Artificial Intelligence, Knowledge Graph

Discuss this project on Neurostars!

20. Brainlife projects

20.1 Python interfaces for brainlife.io (175/350 h)

Brainlife.io is the major platform for free and open neuroscience data analysis and visualization (see Hayashi, Caron, Heinsfeld, et al. in Press Nature Methods, see here https://arxiv.org/abs/2306.02183). Brainlife.io allows researchers to use easy graphical interfaces to standardize, process, visualize and analyze brain imaging data. Currently, brainlife.io handles magnetic resonance imaging data, but the project is growing quickly with

new data modalities becoming compatible with the platform. The platform is developed using a combination of JavaScript, and Python. The future is to integrate brainlife.io with a set of Python-based interfaces.

This project will aim at developing basic functionality that will control brainlife.io from within a Jupyter notebook. The goal is to contribute to the existing github.com/brainlife/pybrainlife Python library to enhance the ability of the library to make API and CLI calls to instances of brainlife.io so as to allow users to process, visualize and analyze data from within local Jupyter Notebooks.

The concrete goals of this projects (details also depending on whether selected as a 175h or 350h project):

- Develop functions in Python that allow to control the CLI and API functionality of brainlife.io
- Develop functions in Python to allow controlling Large Language Models
- Develop Python-based calls to the Network Visualizer https://brainlife.io/ui/nnview/.

Skill level: Intermediate
Required skills: JavaScript/TypeScript, JSON, Python

Time commitment: Flexible, 175h or 350h depending on scope

Lead mentor: Franco Pestilli (@frakkopesto on neurostars), Anibal S. Heinsfeld

Discuss this project on Neurostars!

21. HarmonyHub projects

21.1 HarmonyHub: A Web-Based Platform for Learning Variable-Pitch Musical Instruments (350 h)

Music education faces unique challenges, especially when teaching instruments with variable pitch such as wind instruments and bowed strings. Achieving mastery over these instruments requires personalized guidance and deliberate practice strategies tailored to each student's level and progress pace. The HarmonyHub project aims to address this challenge in music education by developing a comprehensive web-based platform that facilitates the creation of

customized exercises by music teachers for their students. Such an innovative platform will allow for the intuitive composition of exercises, adaptable to students' varying levels, enhancing the learning experience and fostering a more effective teacher-student interaction.

Outcome: HarmonyHub seeks to empower music educators with the tools to create more engaging and personalized learning experiences, bridging the gap between traditional music education and modern technological capabilities. By fostering a more interactive and personalized learning environment, the project aims to significantly improve students' musical skills and motivation, making the journey of learning musical instruments more intuitive and rewarding

Objectives:

- Develop a user-friendly interface that enables music teachers to compose and customize music exercises.
- Integrate API management for musical score generation and sound generation features (e.g., metronome beats and MIDI sounds).
- Implement audio analysis capabilities for real-time feedback to the music student, using audio processing libraries like Essentia.
- Create personalized learning pathways for students, considering factors like age, technical level, and desired progression speed.
- Enhance student learning experiences and promote deliberate practice through digitalization in music education.

Skill level: Intermediate
Skills Required: HTML, JavaScript, and Python. Experience in digital signal processing and musical knowledge is preferred but not mandatory.
Time Commitment: Full-time (350 hours)
Lead Mentor: Alberto Acquilino (alberto.acquilino@mail.mcgill.ca)
Backup mentor: Mirko D'Andrea (mirko.dandrea@gmail.com)
Project Affiliation: This project is part of the research initiatives at McGill University, aiming to improve music education through digital means.
Technical Keywords: Open Source Music Education, Digital Signal Processing, JavaScript, Python, HTML, MIDI.

Discuss this project on Neurostars!

Questions: If you have general questions about the GSoC 2024 applications for INCF, please contact our org-admins - Arnab (arnab1896@gmail.com) and Greg (greg@incf.org) for a quick response. Do not hesitate to reach out. We are happy to help you!
