CSCI 8360 Data Science Practicum
Department of Computer Science
University of Georgia

Project 4: Neuron Finding

DUE: Thursday, November 3 by 11:59:59pm

Out October 13, 2016

1 Overview

As they say in Monty Python–and now for something completely different!

In this final project, you're working on an open problem: the identification of neurons in a time-series image dataset. Here is everything you need to know:

http://neurofinder.codeneuro.org/

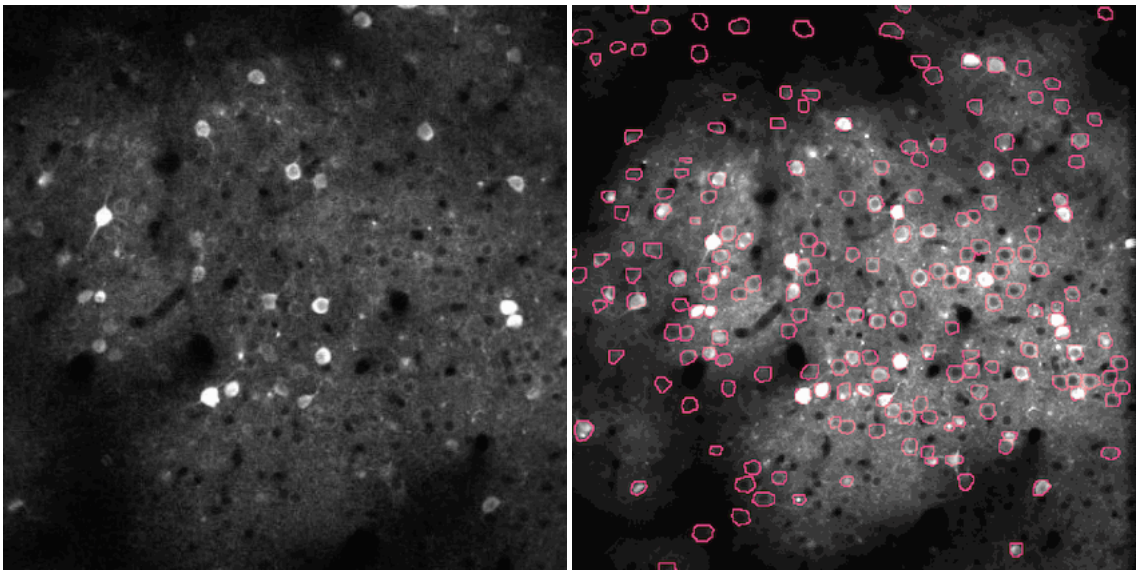
This is the trickiest project thus far, at least in terms of what you're trying to learn. This can best be described as object-finding or image segmentation, where your goal is to design a model whose output is the coordinates to regions of interest in an image. There's no discrete label; rather, your model needs to learn segments in a continuous two-dimensional plane.

Each folder of training and testing images is a single plane, and the images are numbered according to their temporal ordering. The neurons in the images will "flicker" on and off, as calcium (Ca^{2+}) is added, activating the action potential gates. You'll have to use this information in order to locate the neurons and segment them out from the surrounding image.

2 Data

Each file is a TIFF image, separated into folders, where each folder is a single sample. There are 19 training samples, and 9 testing samples.

Each folder contains a variable number of images; sample 00.00 contains 3,024 images, while sample 00.01 contains 3,048. The image files themselves are numbered, e.g. `image00000.tiff`, but all the images in a single folder represent the same sample, just taken at different times with different calcium levels. The training labels exist at the sample level, so you'll use *all the images in a single folder* to learn the locations of the neurons. Each folder will have a unique sample with unique numbers and positions of neurons. However, while time is a dimension to the data, you may not need to explicitly model it; you're just interested in finding the active neurons in space.



The image on the left represents more or less what you'll receive in the training and testing data. The image on the right is the goal of your learner: draw circles around the regions that contain neurons. The data are accessible at the CodeNeuro website linked above.

In addition to the images themselves, the training sets also contain ground-truth regions of interest (ROI) coordinates in the `regions` subfolder, formatted in a JSON file. Each ROI is represented using only two fields: an integer `"id"`, and an $N \times 2$ array of pixel coordinates named `"coordinates"`. This array lists the (i, j) coordinates in the image of the contour entirely surrounding the neuron (i.e. the pink borders in the right-hand image above).

Your goal is to develop an image segmentation pipeline that identifies as many of the

neurons present as possible, as accurately as possible. The ground-truth labels for the 9 test samples are stored in AutoLab (and the labels are not available publicly; ha!) and will be used to grade your submissions.

Like on the CodeNeuro website, we're using more than simple classification accuracy this time around to rank submissions. For the 9 testing sets, you'll get 9 different scores combining the four computed metrics. Each metric measures something slightly different:

1. **Recall**: Number of matched regions divided by the number of ground-truth regions
2. **Precision**: Number of matched regions divided by the number of *your* regions
3. **Combined**: $2 * (\texttt{recall} * \texttt{precision})/(\texttt{recall} + \texttt{precision})$
4. **Inclusion**: Number of intersecting pixels divided by the number of total pixels in the ground-truth regions
5. **Exclusion**: Number of intersecting pixels divided by the number of total pixels in *your* regions

3 Guidelines

The gloves are off; anything is fair game. Use whatever framework you want, distributed or not. If you're not sure where to start, I would highly recommend looking at the CodeNeuro website; existing submissions' code repositories are linked from the website. You are more than welcome to use these, too! Just fork the repositories and make your own changes to see if you can improve the scores further.

Create a GitHub repository for you and your team to work on the project. You are more than welcome to make it a private repository, but the development must take place on GitHub, as I will be using it and the activity you record on it as part of the grading process. In addition to placing the GitHub repository under the `eds-uga` GitHub organization (you can use the teams feature to keep your repo private from everyone else), I would highly recommend using the repository naming scheme of `team-name-project4` to help differentiate from other teams and future projects. Make sure your repository includes

1. a `README.md` giving an overview of your program, how to install and run it, and your design rationale (especially if you chose to use a framework other than Spark)
2. a `CONTRIBUTORS.md` file containing the names of everyone on your team and their respective contributions to the project,
3. good use of the GitHub tickets, milestones, and wiki for documentation, and

4. your code.

Document everything. I mean *everything*. Especially if you've decided to fork one of the existing codebases to give yourselves a head start: *document that your code was forked, and what changes you've made since.* Adopt a consistent coding style, document your code, use the GitHub ticketing system to identify milestones and flag bugs, and provide informative and frequent `git commit` comments. If your entire team is working on the code, make sure you've split up the work accordingly to take advantage of everyone's strengths.

Read over the CodeNeuro website! It has lots of great tips and pointers for how to get started, including a "random algorithm" submission and a "terrible algorithm" submission that serve as useful templates for how to create and format the output of your model. All the associated GitHub repositories are linked from this website. http://neurofinder.codeneuro.org/

Shut down your EC2 clusters when you are finished testing. AWS EC2 instances incur costs for every hour they are turned on, even if they are not actively running any jobs. I will not hesitate to revoke AWS credentials that are racking up disproportionate expenses. Here's the EC2 dashboard you can use to double-check if you have any resources allocated: https://quinngroup.signin.aws.amazon.com/console

Be careful to keep your AWS credentials out of GitHub. It's very easy to hard-code AWS credentials into configuration files or even the code itself, put these files under version control, and push them to the remote repositories. If this happens, take steps *immediately* to remove the commit history from GitHub (search StackOverflow for how to do this), as we have already had one instance of AWS credentials being picked up from public GitHub repositories and used fraudulently.

4 Submissions

All submissions will go to **AutoLab**.

You can access AutoLab at https://autolab.cs.uga.edu. Your team will first need to create an account on AutoLab before you can be added to the `csci8360-fa16` course and submit assignments.

You can make submissions on behalf of your team to the **Project 4** assignment that is open. When you do, you will submit one file: **the JSON file containing the coordinates that predict the locations of neurons in each of the testing examples**.

Your classification accuracy will be tabulated and compared against the true labels (hidden on the server) and should appear on the leaderboard.

If you think you can do better than what shows up–particularly to beat out everyone else–make another submission! There's no penalty for additional submissions, but keep in mind that swamping the server at the last minute may result in your submission being missed; AutoLab is programmed to close submissions *promptly* at 11:59pm on Nov 3, so give yourself plenty of time!

5 Grading

Given that this is a practicum, and that you're all talented graduate students, the grading for the projects is a little different, operating on a sliding scale with some stationary points that require extra effort to move beyond.

5.1 Baseline

For this project, the baseline starting grade is a **90**. To maintain this grade, you'll need at a minimum–

- Use an existing strategy, document it and how it works, adopt it for your own purposes, document the changes made and why, and obtain a comparable accuracy on AutoLab to that obtained by the original authors of the software.
- A `README.md` file that clearly explains the contents of your code, how to run it, and explains your design rationale.
- A `CONTRIBUTORS.md` file that lists your team members and what their role in the project was.
- GitHub tickets and corresponding milestones that show a coherent and cohesive overarching project design plan.
- Good wiki documentation showing the details of the inner workings of your code.
- Well-documented, well-designed, easy-to-read source code.

Meet all these criteria, and you and your team receive a **90**. If you're happy with this, excellent! If not, read on...

5.2 Improvements

Given that anything short of plagiarism (seriously: don't do it) is fair game for this project, the space for possible improvements is almost without limit. That said, extra points here will be rewarded for those with the **most concrete proposals and deliverables.** That means:

- You implemented, from scratch, a very specific algorithm or combination of algorithms from certain papers, and justified why you thought they would help for this particular problem
- You contacted the authors of the prior submitted work and/or proposed certain improvements to the prior work, implemented said improvements, and justified why you think they would help
- You designed a thorough grid hyperparameter scan to find the best possible model for a given model class
- You not only obtained the best numbers on AutoLab but even improved upon the submissions on the CodeNeuro website
- You have an exceptionally well-documented repository, with a clear development plan (milestones, tickets), a wiki highlighting the theory used and how it was implemented, and other documentation (e.g. install instructions, quickstart guide, dockerized examples, a LICENSE file, etc.)

Implementing some or all of these will push your team into the well-deserved **A** range *and possibly beyond*. I'm not enforcing an upper limit on grades for this project, so feel free to try things and see what works. I'll be grading based on effort, theoretical creativity, and development paper trails, not final accuracy. Good luck and **have fun!**