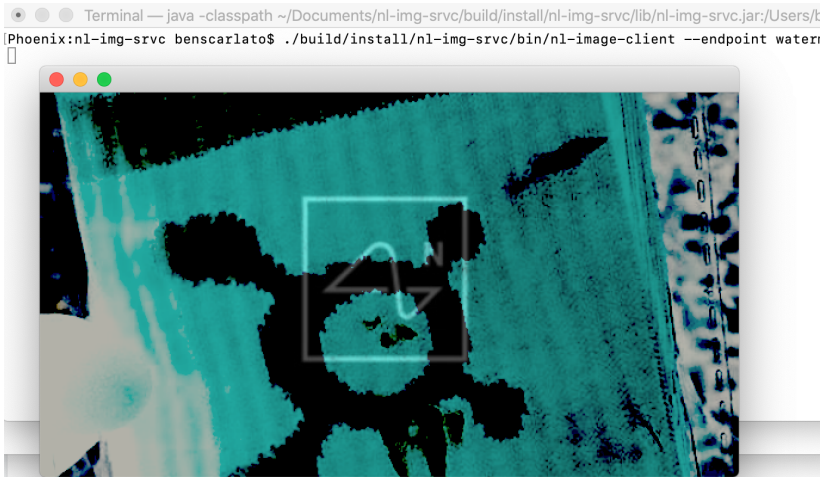
Neuralink Image Service

Intro

The Neuralink image service allows rotating an image or embedding a Neuralink logo watermark.

Example result:



Installation requirements

The service is designed to run on macOS Catalina. It requires an existing installation of Java, specifically it was tested with:

- *OS: macOS 10.15.5*
- *Java: openjdk 11*

Gradle is used for managing Java libraries and dependencies, although the Gradle wrapper is used so an installed binary should not be necessary.

Testing instructions

Perform the following to test the solution:

1. Verify the software listed in the Requirements section is installed
2. Unzip the project and navigate to the root source folder:

    cd neuralink-img-server/

3. Build the project

   ./gradlew installDist

4. Run the server

   ./build/install/nl-img-srvc/bin/nl-image-server

5. Run the client in a separate terminal window:
   a. Rotate an image file:

      ./build/install/examples/bin/nl-image-client --rotate 270

   b. Rotate an alternate file:

      ./build/install/examples/bin/nl-image-client --rotate 90 --filename sample-2.png

   c. Rotate a non-color image:

      ./build/install/examples/bin/nl-image-client --rotate 180 --filename grayscale.png --color false

   d. Watermark an image:

      ./build/install/nl-img-srvc/bin/nl-image-client --endpoint watermark

Issues and limitations

The current solution is designed to be the starting point for a production service but for running and expanding in a large production environment it should:

- **Security**: the client should be updated to no longer use a plaintext channel; based on the environment in which the service will be deployed we should also consider authenticating requests.
- **Error handling**: the server should check, for instance, for examples that are larger than it expects to be able to handle reasonably efficiently.
- **Integration and unit tests**: there should be a full set of tests.
- **Use of java.awt**: this creates unneeded windows on the server side and should be replaced
- **Flags**: a dedicated third party library should be used for managing command line flags.
- **Logging**: a more permanent log of users and requests should be kept.

Future work

There are a variety of related features that could be added to the image service; a starting point would be supporting arbitrary degrees of rotation and custom watermarks.