Design Docs

Fluffy is an interactive image analysis platform relying on deep learning models.

Goals and Non-Goals

1. Be simple. So simple that even a blind, peruvian avocado-farmer can use it while harvesting without any hassle.
2. Run reasonably fast and not roast the users computer in doing so.
3. Provide the essential image analysis add-ons to not rely on other platforms – but don't replace traditional softwares using standard algorithms.

Releases

v0.2.2, Released on: 2020-03-27

Main features:

- Overhauled logo & colors – add the newly designed logo as svg file and update color palette to match with global palette used above. Similarly make buttons and links the same colour for uniformity.
- Bug report / feature request using email templates – as many users will not have a GitHub account / I don't see the necessity for them to have one, it is simpler if they can submit their requests otherwise. To show some accountability (compared to a simple web form also requiring additional database management), I will use an email template defined in the HTML. As soon as users click on the link, their email will contain the body with a simple template comparable to the GitHub Issue request. Templates generated here. If the app develops further, it might be easier to use a form and submit rather than an email template.

Comments:

- None

v0.2.0/1, Released on: 2020-03-06

Second major release. Allows users to take images of various file types, but only single color channels and predict them using four models: Nucleus semantic / instances, stress granules, SunTag background. Predictions are then displayed with a before / after slider. The slider can be zoomed by scrolling and dragged around using the mouse. Instance images are displayed with the JET color palette for easy distinguishing. v0.2.1 fixed a minor bug in saving instance predictions.

v0.1, Released on: 2020-02-19

First major release. The project was still written in Streamlit. Discontinued due to extremely laggy performance and memory issues.

Design

Code design

The google style guide will be followed for javascript, html, and css files. The black formatter is used for python.

Backend

For simplicity, flask will be used to provide all website templates and API calls. Python is required to run all models. There have been a couple of design choices that are crucial for performance:

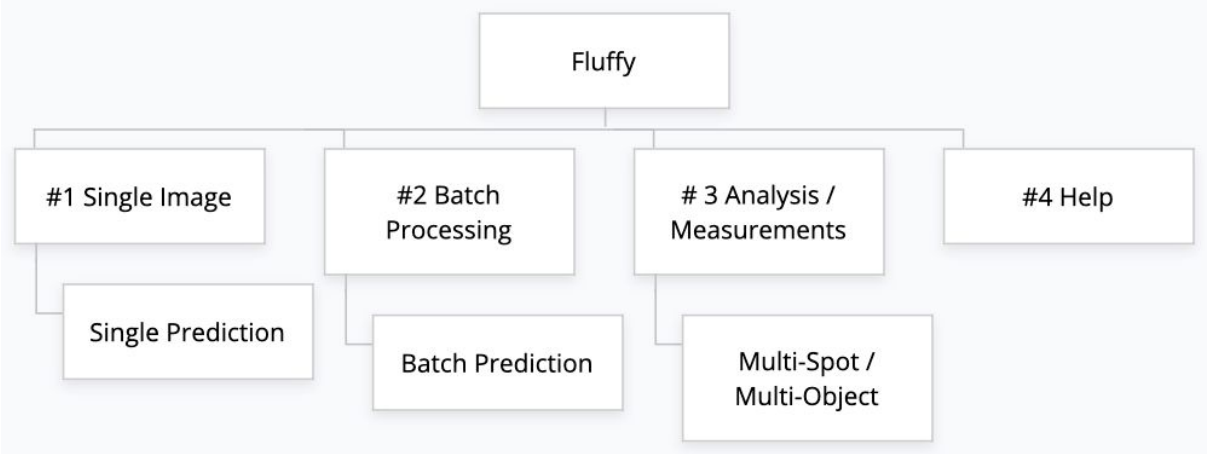
- Models are downloaded and loaded in memory before the first request. This allows a very fast experience but requires users to wait longer while booting. With few models (<5) this hasn't posed major slow-downs but must be kept in mind as more models are added. Possible ideas are loading the selected models only once a selection was made or running models using TensorflowJS and only performing image processing on the server side.
- Images are stored on the server depending on their status (raw, processed). Images used for display are stored in additional folders (display, instances). This separation of concerns uses more disk space but reduces front end complexity in image display.

If the website continues to grow and more features are added one has to think of re-writing the backend in django.
Currently there is no reason for databases. If the app is ever to be released to a web server, it is probably easier to have user logins to balance server load.

Frontend

To save time in learning and development, the main design elements are provided by Bootstrapv4. jQuery is required for these reasons. However, in all custom css, jQuery is omitted in most cases.
Similarly to flask, BS4 and jQuery aren't very supportive of growing applications. Therefore, alternatives have to be considered. When comparing performance, Vue, React, or VanillaJS sound like reasonable options. Furthermore, SCSS is probably simpler and Tailwind CSS will most likely replace BS4.

UX – Sitemap



Built with gloomaps.com.

#1 Single Image

This page is intended to allow the users to preview models which are subsequently applied to many images in bulk.

#2 Batch processing

Once a user previews the model(s), they can run the processing on as multiple images. I propose that it makes most sense if users are allowed to select multiple models either

#3 Analysis / Measurements

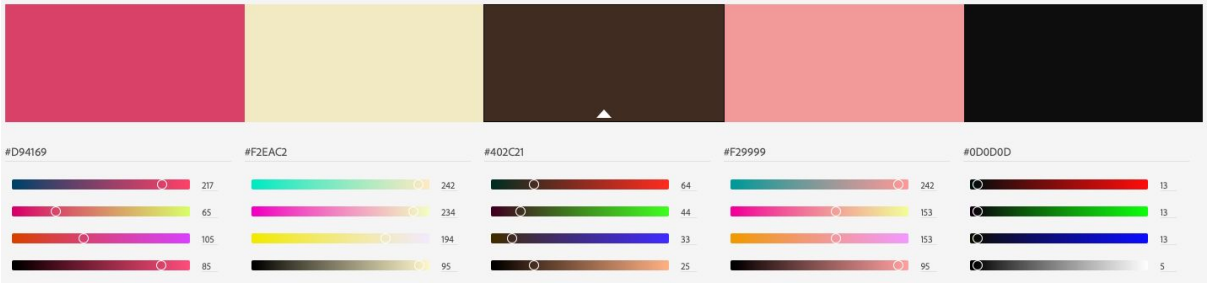
Page description to follow. Briefly, this page supplements the batch processing and allows users to perform measurements. The idea is to

#4 Help

Simple assistance page describing all do's and dont's of the website. There are no usage demos as the site should be intuitive enough but "invisible" information is displayed. This includes what file formats are supported, what models are currently available etc.

UX – Color palette

Trying to keep the Fluffy logo flair / slightly vibrant colors. Additionally, the default BS4 colors will be replaced by the below except for some signalling and buttons.



UX – Font families

There are three font families in use:

- Documentation (google suite): Roboto
- Logo: SignPainter HouseScript Regular
- Web interface: System dependent Bootstrap default (Segoe, Helvetica Neue, etc.)

UX – Page layouts

Simplicity implies a streamline user experience. Therefore, all sites will be single columns and guide users through the workflow without any required directionality changes.

Roadmap

v0.2.3, Proposed release date: tbd

Main features:

- ❏ Improved batch processing – typically one has multiple color channels per experiment and therefore needs multiple models. Each experiment, however, only consists of one image type (eg. Z-Stacks). In the future batch processing form, one will only be able to upload files and select the image type. Model selection is performed in the file lister. Similarly to delete/download buttons one can select models for all files or for each file individually.
- ❏ Add more image types – Z-Stacks are currently only max-projected. This is slightly lackluster. This new release will feature a selection of various preprocessing options keeping the max-projection as default.
- ❏ Graph showing data dependency in model training – a simple script to run to approximate model complexity. This allows indecisive users to plan out how many training images are required. Between 1 - labeled.max() images a short training will be performed. The test score is calculated on a fixed number of holdout images. The scores are then plotted with #images on the x axis and test score on the y axis. If need be a simple linear model can be fitted to predict at what #images the model will reach human level performance.

Comments:

- None

v0.3.0, Proposed release date: tbd

Main features:

- ❏ Spot model –

Comments:

- None

v0.4, Expected release date: tbd

Main features:

- ❏ Django –

Comments:

- None