MBARI Power Buoy Classification Model for Monterey Bay

Goal

- Classify IFCB datastreams from Monterey Bay with enough fidelity to resolve broad changes in community structure.
- Resolve changes in functional groups of social interest, ie, Harmful Algal Bloom-forming organisms.
- Develop an operational framework where images are classified in near-real time.

Development

Jesse Lopez did the initial model development as a proof of concept. The convolutional neural network (CNN) was trained on a set of images curated by Alexis Fischer from IFCB data collected by the Kudela Lab at UCSC. The Xception model framework[1] was chosen based on classification performance metrics against other popular CNN frameworks.

The model is trained and tested on a GPU node on the Hummingbird cluster, an HPC, using Nvidia Telsa P100 GPU. Training logs can be saved and uploaded to TensorBoard for review and record keeping.

Training Dataset

Models are only as good as the data that trains them. To date, training data is all driven from IFCB104, and manual validation from members of the Kudela lab using the IFCB Annotate tools. Training sets are generated from the 'export_png_manual_fromROI.m' Matlab routine, which generates .png files from the IFCB datasets stored on a Synology NAS in the Kudela Lab. Manually classified images are then sorted into **Training (80%), Testing (10%), and Validation (10%).** Testing and training datasets are used to train the model. The validation dataset is used to evaluate model performance by quantifying robustness.

Saving Models

The trained model is saved as **saved_model.pb** with an **assets/** directory, which is used to make graphs of the model, and the **variables/** directory, where training checkpoints are saved.

Version Control

Currently: All images used for training and testing are compressed and stored as a .tar.gz file for archival. A plain text file (.csv) of all of the filenames for each class was

[1] https://arxiv.org/abs/1610.02357

created. This can be used to recreate the training set or to compare the difference with subsequent model generations.

Options: **D**ata **V**ersion **C**ontrol
DVC is a git-based open-source version control software for MLOps. But, training and testing data sets are on the order of 10s of gigabytes, so we need to develop a storage solution. We can implement this on *particle.shore* at MBARI, but this will limit who can access it. It is also inconvenient for current development since it mostly happens at UCSC on the Hummingbird Cluster. Since both systems are on internal networks getting them to connect is challenging.

One scalable solution would be to move all of the data storage and model storage to the cloud like an S3 bucket. There are some opportunities to apply for a small research grant for cloud compute credits (google, amazon) to prototype this.

Another solution would be to use a NAS at UCSC in the Kudela lab. This is less scalable and might limit outside collaboration but could be implemented relatively easily and quickly.

Embed metadata into .HDF for the CNN models

Option: Zenodo
Zenodo is an “open science” data repository that could be used for publishing and versioning models and compressed training sets.

Applying the Model

Model predictions (ie, classification) are performed hourly when data are retrieved from the MBARI Power Buoy IFCB. Classification is run on the *particle.shore.mbari* machine using the CPU. Each ROI is assigned a probability for each class in the model. The highest probability is selected as the “True” class. If the highest probability is below a threshold set at 0.75, then the “True” is determined to be “unidentified.” In future work, it will be worthwhile to perform some sensitivity analysis for the threshold to determine a value that best balances type I errors against type II errors.

Data are then summed up for the sample period for each class and divided by the sample volume to give a cell concentration value for each class for each sample period. These data are appended to a plain-text (.csv) file. Those are used to create plots each hour that is pushed to the CeNCOOS web server.

The approach of classification on demand reduces the computational requirements for the dataset at any given time by distributing them throughout data collection and allows for a near-realtime data analysis. This approach is unsuitable for classifying large amounts of data beyond a dozen days worth of data. Under those circumstances, it

would significantly more efficient to run the classification on a machine or cluster with ample memory storage and a GPU.

Technical Workflow:

1. Extract Manual Classification data from Kudela Lab NAS (Synology Server) using **export_png_manual_fromROI.m** routine from ifcb-analysis Matlab package.
2. Split manual classification data into roles using **sort-images-by-role.ipynb** notebook:
   Training (80%): These are used for training the model
   Validation (10%): These are how the model validates its accuracy
   Testing (10%): The model never sees these data, and they can be used to test the robustness
3. Move sorted images to the hummingbird cluster at UCSC using the **hbfeeder.ucsc.edu**
4. Place model training code (train-xception-hb.py) in the queue by running training-ifcb.slurm.
5. Run model validation code (test-xception-hb.py) in the queue by running test-ifcb.slurm.