Phytoplankton detection using machine learning and a mobile application.

Jordan Janakievski
Bellarmine Preparatory School

Abstract

Artificial intelligence and its implementations have become more established over the past decade. Connecting the tedious task of phytoplankton quantification and artificial intelligence, the time needed to perform the task can be greatly minimized. By using machine learning to develop an object detection model identifying basic shapes of phytoplankton and deploying it onto a mobile application, brings plankton research closer to automated quantification for the masses. The use of image classification and object detection were both tested and performed as expected, however, there was a clear advantage with object detection as it could classify multiple shapes at once and was not confined to the entire image frame. The object detection model was developed alongside a mobile application to help pave the future of remote lab classification with mobile devices. The application and the model are open source, allowing anyone interested to continue development.

Introduction

As technology becomes more widespread and user friendly, artificial intelligence (AI) will also become more common. Researchers can benefit from the introduction of this technology into data analysis. Currently, AI and its counterpart machine learning (ML), are working to improve processes that are labor intensive. While AI has an end goal of performing cognitive processes previously only possible by humans, ML uses algorithms to learn and improve upon a preprogrammed task. ML proves to be a vital partner for a researcher.

Gathering quantitative data with a microscopic technique is a time-consuming process. Apart from the length of time, the ability to properly identify the object usually requires years of experience. Focusing on phytoplankton, many researchers have developed sophisticated neural networks all around the world. The benefits from researching automated technologies to aid in the quantification of plankton comes down to the experience required to properly identify a species. Embleton et al. (2003) created a neural network to identify four species of phytoplankton in Northern Ireland in years predating easily accessible ML aids such as Tensorflow and PyTorch. After creating a sophisticated neural network and fine-tuning the training process for maximum results, researchers in California show that around 4000 images per class in a dataset bring the training data just short of the limiting number of 89% accurate (Ellen et al., 2015). Ellen et al. also concluded that the size of training data had a greater impact on the successful identification than the neural network algorithm. These findings are supported with Faillettaz et al.'s (2016) study from the fact that the training data played a large role in the success of the identification process. It was discussed that having clear, easily identifiable images allows the neural network to properly learn the plankton and would therefore improve the accuracy (Faillettaz et al. 2016). There are two areas that clearly can be improved. One limitation present in all previous studies was lack of an open-source mindset when creating the ML models. This results in each group having to recreate training data whereas sharing the data can allow improvement upon previous research more frequent. Another factor not present is an attempt to use a mobile device as the camera for the classification process. This ability can make gathering data easier while in the field and allow for more people to have the ability to study plankton with an automated process.

This study set out to create another neural network, programmed in Python, to identify phytoplankton on the characteristics of shape. The intention of the neural network is to aid the first-year students in Bellarmine Preparatory School's Marine Chemistry program. The shapes chosen to be identified are circles, chains, and boxes; the same shapes used for classification in Marine Chemistry's 30+ year history. Another part of the study, coinciding with the neural network, is the development of an open-source, mobile application to utilize the ML model and identify phytoplankton using a mobile device camera.

Methods

2.1 Plankton Collection

Titlow Beach Marine Preserve was used as the collection site. A plankton net was moved in a linear motion three times about 3 cm under the water to collect a 3 mL sample of plankton. Two drops of iodine were used to preserve the sample and the sample was given 12 hours to settle. This allowed all the preserved plankton to fall into the bottom 1 mL of the sample. After sitting for 12 hours, the top 2 mL of the sample was pipetted away and discarded. This left the bottom 1 mL with the preserved plankton. The final millimeter was collected and analyzed drop by drop at a magnification of 100x under a microscope with a counting slide. This process was used to capture 168 images of phytoplankton.

2.2 Dataset Preprocessing

Using the images collected in 2.1, two datasets were created: one for image classification and one for object detection. Data augmentation was used to increase the size of the datasets by a factor of 3. Each image was flipped along the horizontal axis and rotated clockwise 90 degrees. The total size of the dataset became 504 images.

The image classification dataset consisted of 300 images classified into one of the three classes (circle, chain, box). Some images of the full dataset needed to be removed due to the presence of multiple types of plankton. Classifying the images created the training data.

The object detection dataset consisted of all 504 images. LabelImg, an open-source, graphical interface was used to categorize and characterize the images based on the plankton shapes present in each. LabelImg saved the image files as XML files that will be used by the object detection code for training.

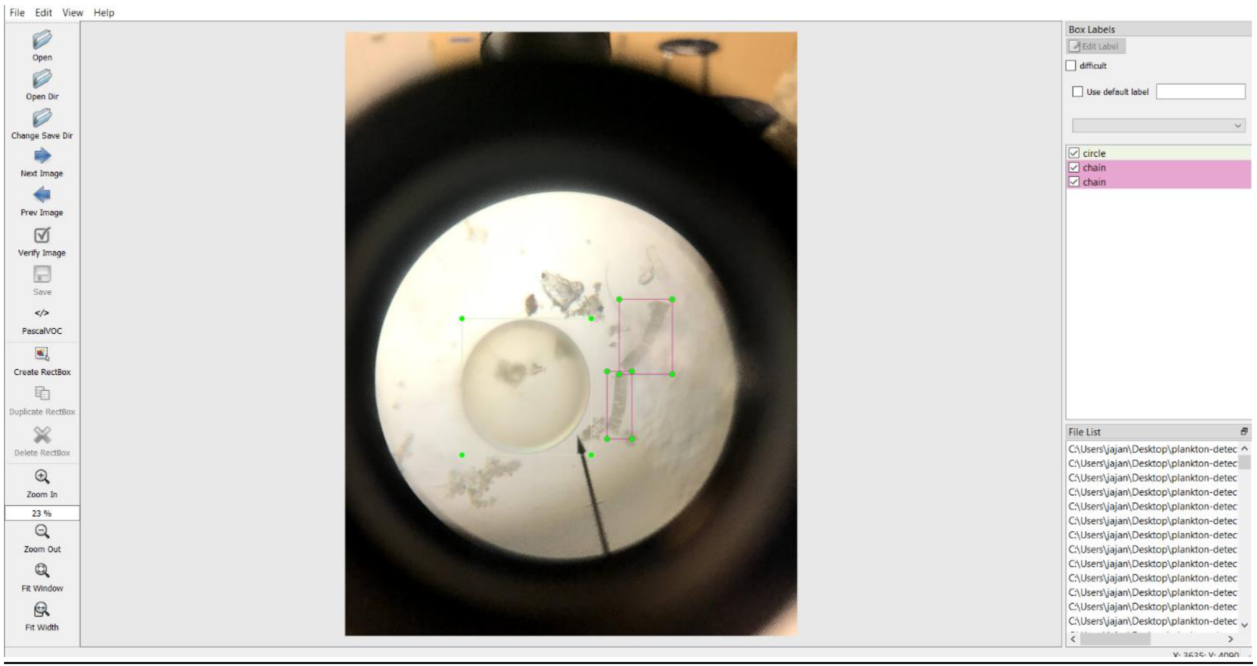


Figure 1 shows the completion of classification using LabelImg, as the phytoplankton have boxes around them and labels based on shape.

2.3 Image Classification Training

All the preprocessing of the training data was performed in Python. With the use of different libraries (often containing prewritten code that allows certain functions to be completed more easily) such as OpenCV, Matplotlib, and NumPy, all the images are converted into grayscale and reduced to a pixel ratio of 100:100 instead of the default 300:300. This is to reduce the size of the files and make the training easier for a simple ML model. Each class of the training data had a corresponding array index ranging from 0 to 2. After all the images were preprocessed for training, the convolutional neural network (CNN) is designed with TensorFlow (an open-source ML platform developed by Google—the Python library was used in this section).

A 3-layer CNN was designed and trained. There should be a positive correlation between the amount of training and the accuracy of the model if set up correctly.

2.4 Object Detection Training

The Python files available in the Tensorflow Object Detection API (Application Programming Interface) were used to prepare the training data for training. They were used to retrain the final layer of the SSD Mobilenet V1 Pets model, setting the batch size to 12 images, and tailoring it to the 3 classes of circle, chain, and box instead of its original 37 classes. For training, the images are compressed from their original pixel ratio to 300x300.

2.5 Mobile Application

Making the ML model available to all both for further development and use was achieved with a mobile application (app). The app needed to have a simple user experience so there would be no confusion with how it worked. The user interface (UI) needed to fit in with modern apps so it would have a familiar feel to any user. The architecture of the app was designed with Flutter (a UI toolkit created by Google. Designs the layout of the mobile app) and programmed in Dart (a programming language developed by Google that is specifically focused on app development). The trained model from 2.4 was added to the app, allowing object detection to function and identify phytoplankton.

The mobile app works in a way that holding the phone or tablet’s camera to the microscope’s lens allows for the object detection model to classify the organisms. It features a live video feed from the camera, drawing a box around the organism it is identifying and labeling it along the perimeter of the box.

Results

The training of both ML models was successful. The measured parameter from the training was loss. The loss value in its most basic state is a difference between the predicted value (the output of the ML model), and the actual value (assigned during preprocessing).

$$Loss = \sum_{i=1}^{n} \left| y_{actual_i} - y_{predicted_i} \right|$$

A loss value is calculated at each epoch of the image classification training and each step of the object detection training. An epoch represents a full iteration through the dataset whereas a step is a specified batch size.

The figures show the data gathered from the training. Both models achieved desired loss values of ≤ 1. Figure 2 shows a loss graph comparing the training loss with the validation loss which is logged from the model. The validation loss is used to check the model against images it has not been trained on.

With object detection, Figures 3-5 show the different losses; total, classification, and localization. The total loss is an average loss calculated from a combination of parameters, most notably the classification and localization. The classification loss is gauge for the successful identification performance of the model. The localization loss is a gauge for the successful mapping of the object. The value is found by finding the difference between the predicted bounding box and actual bounding box set up in 2.2 Dataset Preprocessing. By using the Tensorflow API and its prewritten code, there is no validation loss values for object detection.

The performance of the mobile application was as expected and the ability to continue processing the live detection is hardware dependent.

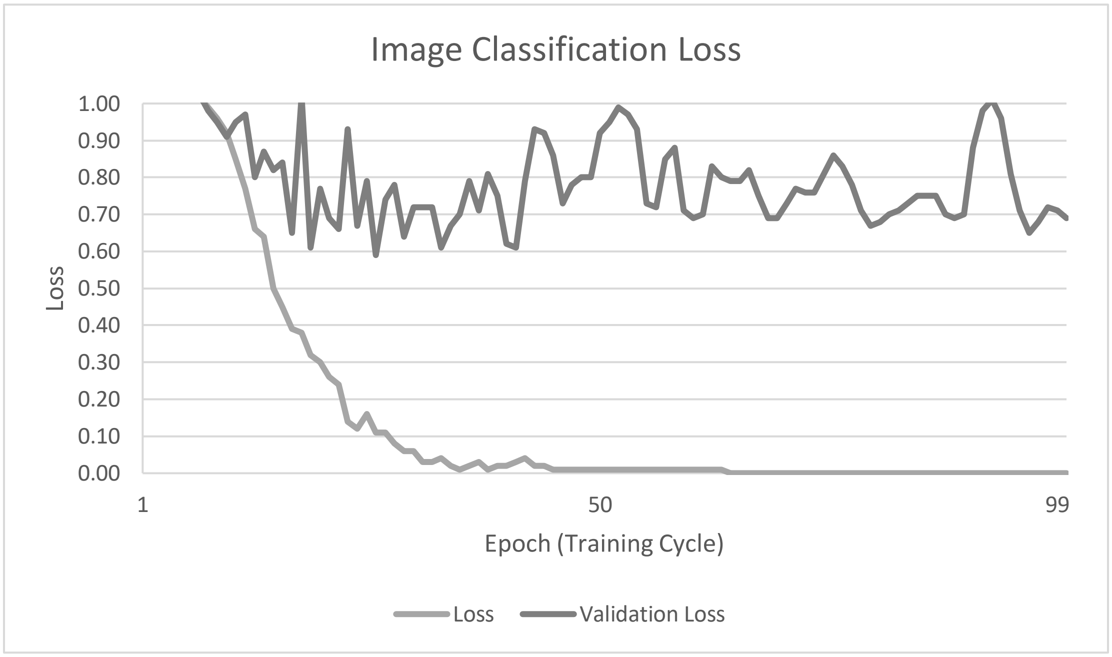


Figure 2 shows the comparison between the loss and the validation loss of each training cycle from 1 to 100.

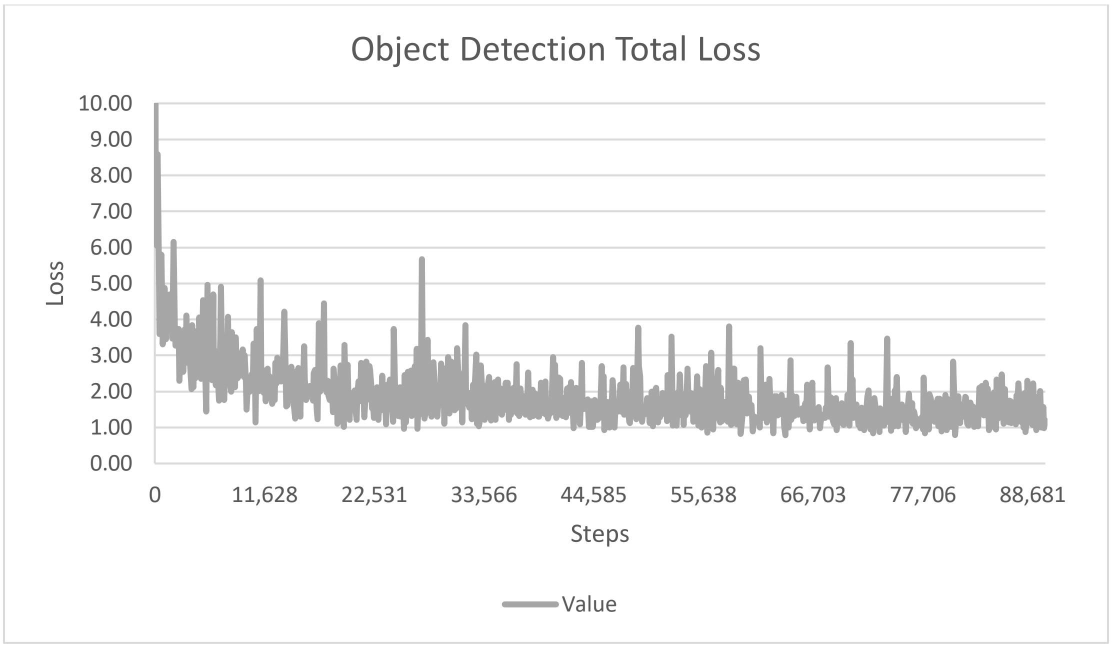


Figure 3 shows the total loss of the object detection model. Beginning with a value of 22.82 and ending with oscillation around 1 after 90,000 steps.

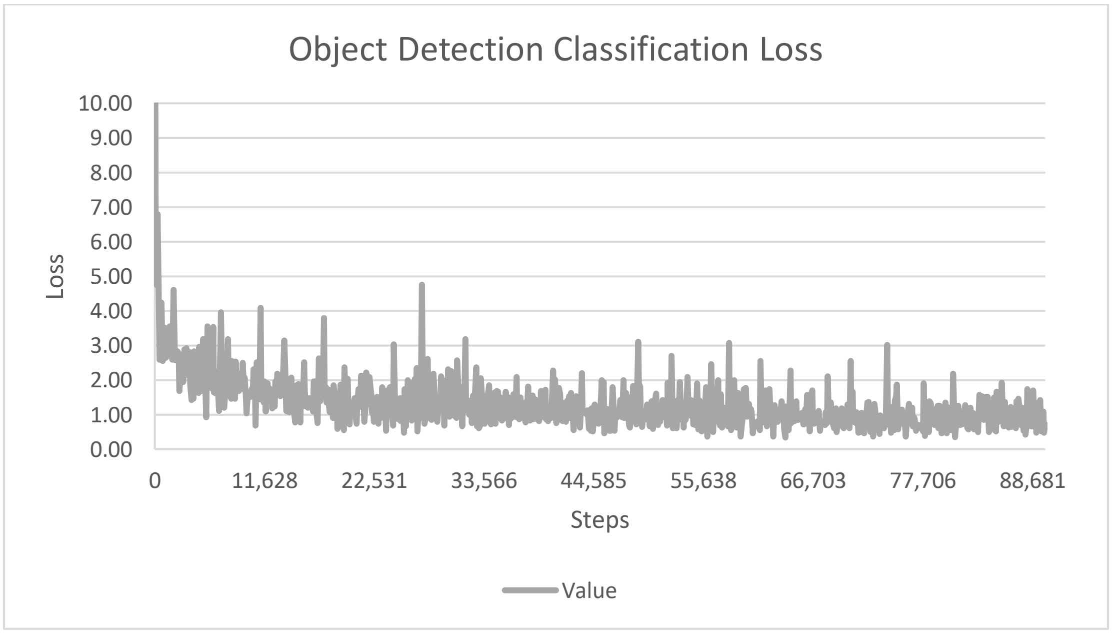


Figure 4 shows the classification loss of the object detection model. Beginning with a value of 19.39 and ending with oscillation around .7 after 90,000 steps.

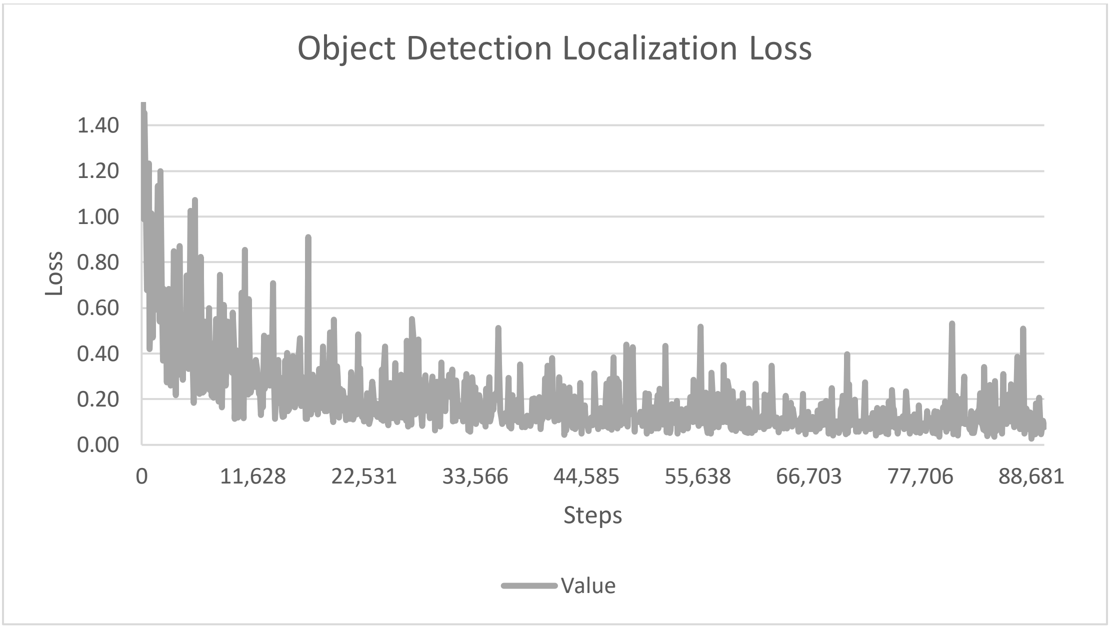
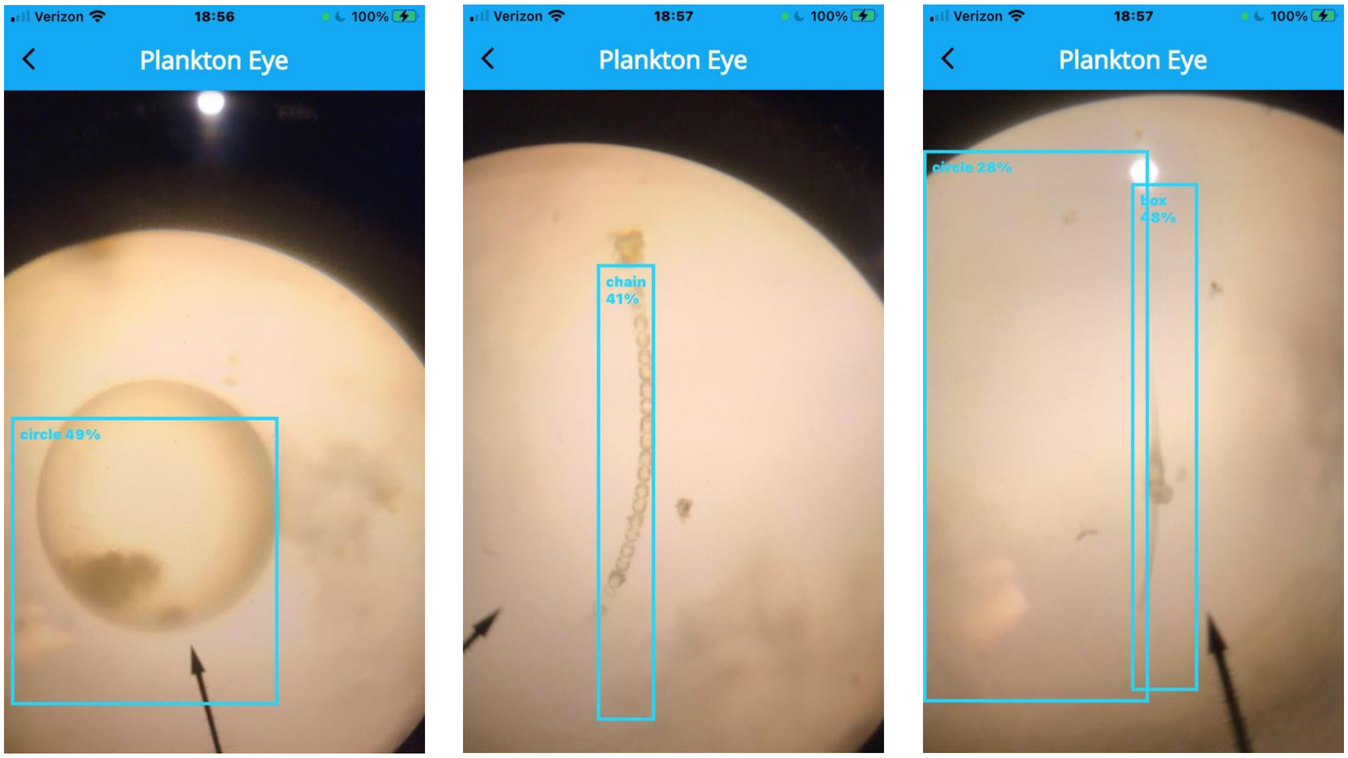


Figure 5 shows the localization loss of the object detection model. Beginning with a value of 3.10 and ending with oscillation around .1 after 90,000 steps.



Figures 6-8 show the mobile app in use. It draws a box around the object it is attempting to identify, labels the shape, and shows its confidence.

Discussion

The results support the previous research performed by Embleton et al., Ellen et al., and Faillettaz et al. that show the automated identification of plankton. The ability to automatically identify the shape of phytoplankton is greatly beneficial as standard classification with large samples can take more time. While this is merely identifying the 3 most common shapes in phytoplankton (circle, chain, box), it offers the ability to perform real-time identification with technology separate from a standard computer. When developing the ML model, the choice to use object detection instead of image classification came down to the ability to identify multiple plankton in each frame, as image classification is limited to only one output. This allowed a more practical use of the model. Furthermore, it was recognized that a mobile app could be beneficial to both simplify data collection and create opportunities for on-site field research. The mobile app was named Plankton Eye, and it demonstrated the early success of on-device object recognition.

In Figures 2-5, the data is generated from a dataset collected from one sampling. This created many limitations. One factor affected by the small dataset are false positives. The loss may be near 1, but the ability to properly differentiate between a chain and a box is difficult with the small training data. As the dataset only consists of 504 images, it does not have the substantial amount needed for consistent detection. This is most prevalent as the images are compressed to a 300x300 pixel ratio due to MobileNet standards. This creates challenges with detection and classification as the fine details in the phytoplankton are unable to be determined due to compression. Another limiting factor for training is accessibility for computer hardware.

As graphics processing units (GPUs) are the hardware component used for this high data processing, powerful components make the process more efficient.

Continued research could include expanding the dataset to train a higher caliber model, species-specific classification, and developing an automated quantification feature for the application. Expanding the dataset would help improve the accuracy of the model, and species-specific classification would make the application more accessible overall. Automated quantification would allow researchers to cut down on data collection times. Improving the app allows for the opportunities to aid in initial analysis for immediate results. By building upon this research, benefits would impact both the Marine Chemistry Program as well as plankton researchers throughout the world.

Works Cited

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems.

Culverhouse, P. F., Williams, R., Reguera, B., Herry, V., & González-Gil, S. (2003). Do experts make mistakes? A comparison of human and machine identification of dinoflagellates. *Marine Ecology Progress Series, 247*, 17-25.

Ellen, J., Li, H., & Ohman, M. D. (2015). Quantifying California Current Plankton Samples with Efficient Machine Learning Techniques. *Institute of Electrical and Electronics Engineers.*

Embleton, K. V., Gibson, C. E., & Heaney, S. I. (2003). Automated counting of phytoplankton by pattern recognition: a comparison with a manual counting method. *Journal of Plankton Research, 25*, 669-681.

Faillettaz, R., Picheral, M., Luo, J. Y.,Guigand, C., & Cowen, R. K. (2016) Imperfect automatic image classification successfully describes plankton distribution patterns. *Methods in Oceanography, Elsevier, 2016*, doi:10.1016/j.mio.2016.04.003ff.

Karki, R. (2020, October 10). object_detection. Github repository. Retrieved from https://github.com/rupakkarki27/object_detection.

Luo, T., Kramer, K., Goldgof, D., & Hall, L. O. (2003). Learning to Recognize Plankton. *IEEE International Conf. on Systems, Man and Cybernetics, 2003*.

Sosik, H. M., & Olson, R. J. (2007). Automated taxomonic classification of phytoplankton sampled with imaging-in-flow cytometry. *The American Society of Limnology and Oceanography, Inc, Methods 5,* 204-216.

Vilamala, A., & Autonomous University of Barcelona (Ed). (2010). Detection and Identification of Phytoplankton Assemblages using Case-Based Reasoning. *Universitat Autònoma de Barcelona*.

Wilkins, M. F., Boddy, L., Morris, C. W., & Jonker, R. R. (1999). Identification of Phytoplankton from Flow Cytometry Data by Using Radial Basis Function Neural Networks. *Applied and Environmental Microbiology, Oct 1999*, 4404-4410.

Xiaoou, T., Stewart, W. K., Vincent, L., Huang, H., Marra, M., Gallager, S. M., & Davis, C. S. (1998). Automatic Plankton Image Recognition. *Artificial Intelligence Review, 12*, 177-199.

Zheng, H., Wang, R., Yu, Z., Wang, N., Gu, Z., & Zheng, B. (2017). Automatic plankton image classification combining multiple view features via multiple kernel learning. *BMC Bioinformatics*. doi:10.1186/s12859-017-1954-8