Machine Learning for Florida Estuary Biodiversity Analysis: A Cal Poly Data Science Capstone

Edward Du
Computer Science

Ellyce Bilhorn
Statistics

Charles Ward
Computer Science

William Medwid
Statistics

Abstract

Estuary biodiversity analysis is a time intensive task for the expert ecologists at the Smithsonian Marine Station's Benthic Ecology Lab in Fort Pierce, Florida. To automate some of their work, we have developed an object detection model and desktop application that will classify and count the specimens in their benthic soil samples. Our best performing model achieved a mAP @ 50 of .583 and avg-F1 @ 50 of .643 (across 4 classes). Our desktop application allows sample analyzers to run our model on new images using a friendly Graphical User Interface (GUI), inspect and correct its predictions, and save the specimen counts to a standardized spreadsheet format.

I. Introduction

The biodiversity in Florida's lagoons, estuaries, and rivers is vital to local economies and the global climate. Unfortunately, decades of water control projects to make land suitable for urbanization and agriculture have decimated the Everglades' water inflow from Lake Okeechobee. To reverse this environmental harm, the $8.2 billion Comprehensive Everglades Restoration Plan was enacted in 2000 to restore water flows to the Everglades in southern Florida. While this transition is taking place, it is vital that the ecosystems' health and biodiversity is monitored to ensure that the restoration is as beneficial as possible.

The Smithsonian Marine Station's Benthic Ecology Lab is studying sites in the Indian River Lagoon, east of Lake Okeechobee, which will receive less, but cleaner, water due to the Comprehensive Everglades Restoration Plan. They collect soil samples from the bottom of several sites in the Indian River Lagoon, and count the number of distinct species they find under a microscope. These species counts are vital indicators of ecosystem health, however the manual process of labeling these specimens is very time intensive and requires skilled ecology experts.

To speed up and automate portions of this task, the Benthic Ecology Lab is partnering with our Data Science Capstone group to use machine learning to automatically label specimens. To make the model's results accessible and useful, we have developed a user interface for them to utilize our object detection model and perform manual quality control on its specimen predictions.

II. Data Description

The Benthic Ecology team provided us with two types of training images: 'Blended Photos' and 'Individual Specimen Photos'.

The Blended Photos such as Figure 2 (a) are developed from soil samples and contain a variety of organisms. This image type is ideal for training and testing our models as they are the ultimate target for our tool's operation. We were provided with 49 of these blended photos, which collectively contain a total of 627 individual specimens. The specimen are typically dyed pink in these photos, but due to inconsistencies in this process, we convert images to grayscale when training and utilizing our models.

The Individual Specimen Photos take a variety of visual styles: some are textbook-quality and zoomed into a clearly lit single specimen such as those in Figure 1, while others are similar to the Blended Photos, with many examples of a single specimen type. We were given 920 distinct Individual Specimen Photos, containing a total of 2069 specimens. However, the practicality of using many of these photos as training data is questionable due to differences in lighting, detail, and backgrounds as compared to the Blended Images we intend to classify.

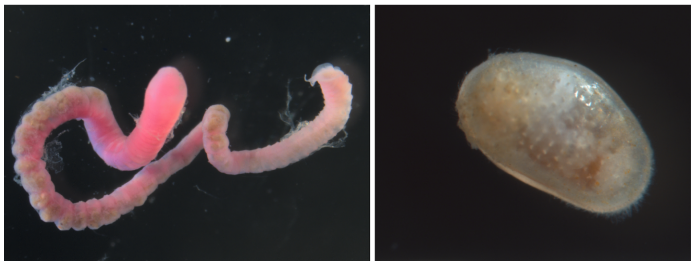
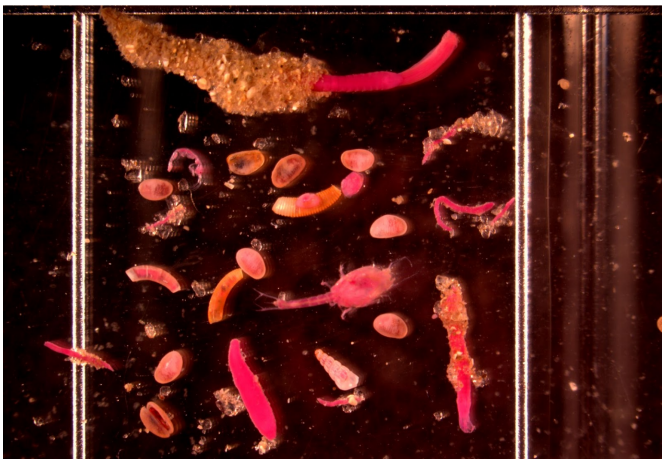
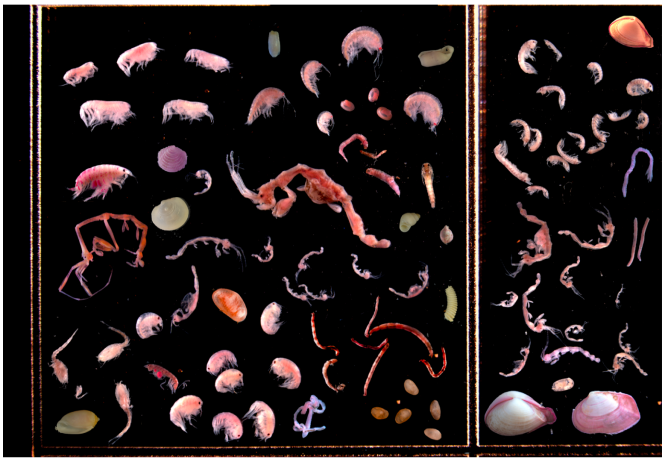


Fig. 1: Example Individual Specimen Photos: Specimen of major groups Polychaeta (left) and Ostracoda (right)

Labeling this data was complicated by the fact that organisms are sometimes broken into multiple pieces, but should only be labeled and counted once. Particularly, Polychaeta specimens in Blended Photos often have dismembered appendages scattered throughout their photos. These appendages resemble the organisms they originate from but ideally should not be double counted by our tool. Consequently, we labeled these Polychaeta appendages and other extraneous objects that could be confused for organisms as "Unknown". This



(a) Example Blended Photo



(b) Example Artificial Blended Photo

Fig. 2: Example Photos

approach ensures that the bounding box regression portion of our models does not suffer from the difficulty in distinguishing separated appendages from the specimen they came from.

A. Class Imbalance and Diversity

The provided image data are classified at three levels: 10 distinct major groups, 113 distinct family groups, and 441 distinct species. The 10 major groups are Amphipoda, Bivaliva, Cumacea, Gastropoda, Insecta, Ostracoda, Other, Other Annelida, Other Crustacea, and Polychaeta. However, the classes were significantly imbalanced, with very few examples of some major groups. For example, we only had 2 examples of "Other Annelida" in our blended photos, and 14 of them in our individual photos. Furthermore, some of these major groups encompass such a diverse set of species with little resemblance to one another that it may be difficult to recognize them as the same major group. This dataset, characterized by high-diversity and low-volume, poses substantial challenges for accurate object recognition. See Table I for a tabular overview of the Major Group class counts.

We only target the Amphipoda, Ostracoda, and Polychaete Major Groups due to these dataset size limitations. All other Major Groups and appendages/debris are treated as belonging to a separate 'Other' class for the purposes of model training and analysis.

B. Dataset Bolstering: Artificial Blended Images

The Individual Photo format presents issues for model training due to this data not being representative of the Blended Photos. However, there being only 49 Blended Photos also presents challenges in this small dataset size. Additionally, the Blended Photos have an uneven distribution of specimen classes that make it impossible to adequately split these images into strong train, validation, and test sets.

To remedy these challenges, we used Photoshop tools to create 24 'Artificial Blended Photos' from multiple Individual Photos to mimic the appearance of true Blended Photos. See Figure 2 (b) for an example. For each Artificial Blended Image, we created a realistic background by copying a real Blended Image and digitally removing the organisms. Then, we added individual specimen photos one by one, using the Magic Select tool to paste the specimen without their backgrounds. We shrunk the specimens to match their typical sizes and applied a blur to soften their hard edges and make them look like those pictured in real Blended Photos. Using this technique, we were able to select the best individual photos and turn them into information-dense training data. Note that some of these Artificial Blended Images contained only specimens from a single Major Group while others contained a variety of specimen classes.

The combination of Individual Images into a single Artificial Blended Photo also helps to reduce the computational costs of model training: many specimen examples can be evaluated in a small batch of Artificial Blended Photos that only require a small set of images to be loaded into memory.

TABLE II: Train/Validation/Test Split

Set Type	Image Types Contained in the Set
Train	1. Only true Blended Photos
	2. All true Blended Photos and only Artificial Blended Photos containing primarily specimens from the 3 target classes (Amphipoda, Ostracoda, and Polychaeta)
	3. All true Blended Photos and Artificial Blended Photos
Validation	True Blended Photos and Artificial Blended Photos
Testing	Only true Blended Photos

TABLE I: Major Group Counts by Image Type

Class Counts	*Image Type*			
Major Groups	Blended Photos	Artificial Blended Photos	Individual Photos	Grand Total
Amphipoda	145	123	222	490
Bivalvia	76	87	223	386
Cumacea	15	33	48	96
Gastropoda	29	134	502	665
Insecta	0	32	35	67
Ostracoda	61	81	205	347
Other	34	87	132	253
Other Annelida	10	26	14	50
Other Crustacea	20	80	183	283
Polychaeta	172	165	491	828
Grand Total	**562**	**848**	**2055**	**3465**

C. Train/Validation/Test Split Strategy

To test the efficacy of the Artificial Blended Images, we created 3 different training datasets as described in Table II. The validation and test sets were held constant across each training dataset.

III. Machine Learning Model

Our investigations yielded three main approaches to the specimen counting problem: (1) algorithmic computer vision techniques combined with standard Convolutional Neural Networks, (2) object detection models, and (3) image segmentation models. We decided to invest most of our research efforts into (2) object detection models after our evaluation of the other approaches indicated either a reduced ability for solution strength due to lack of available model complexity (1) or a necessitated reduction in user experience quality due to the model output not being conducive to user interaction/label correction (3).

Object detection, as a class of machine vision tasks, is different from commonly understood image classification tasks because object detection models must predict regions-of-interest (ROI) in addition to classifying those regions. It is important that the model be very good at both of these tasks as, using our problem as an example, specimen counts could not be accurate if the regions-of-interest proposed for classification are very poor, even if the actual classification stage is very accurate.

We use the metric intersection-over-union (IOU) to determine whether or not a predicted ROI matches a ground truth object region. When evaluating model predictions, one must set an IOU threshold above which you would consider two regions to be 'matched'. Predicted regions are often redundantly layered upon one another because the region proposal stage accurately identifies many regions that cover the same object, and is not significantly penalized for this repetition. As such, we perform non-max-suppression (NMS) with a given IOU threshold on all predicted regions to filter out these duplicate ROIs. We tune the NMS IOU threshold and box classification confidence threshold hyperparameters to maximize mean-average-precision (mAP) on the validation set via a grid search algorithm.

A. Object Detector Architectures

We trained and tested three object detector architectures. Each model was instantiated with open source COCO weights and fine tuned by replacing and training only the final classification layer and then training the entire model at a low learning rate until the best validation set loss was achieved. Training loops were stopped once a new minimum validation loss hadn't been achieved in 5 epochs.

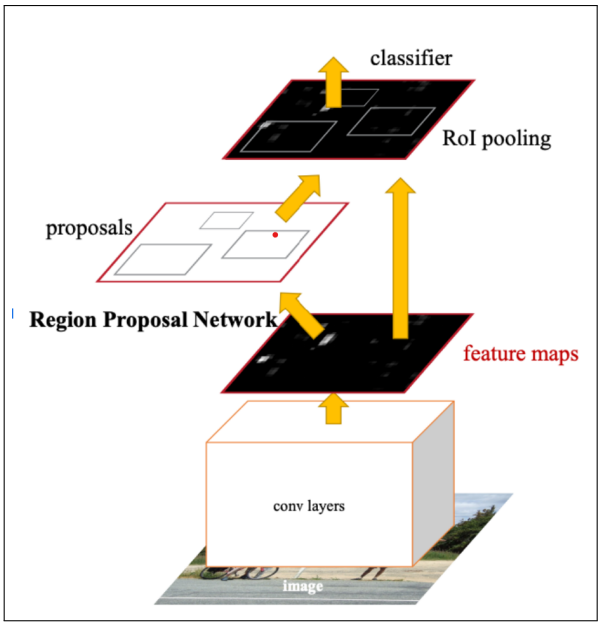


Fig. 3: Faster R-CNN Architecture

1) Faster R-CNN (Resnet 50 FPN Backbone): Faster R-CNN defines a family of object detection models that employ a 3 stage methodology [1]. As depicted in Figure 3, the first step is that a backbone Convolutional Neural Network, computes feature maps on the image. Then, these feature maps are input to a Region Proposal Network (RPN) that performs

bounding box regression. Lastly, the pooled feature maps for each proposed bounding box (rescaled to a preset dimension required by the classifiers dense network layers) are input to a classification stage usually consisting of a few convolutional and dense network layers.

2) YOLOv8: YOLOv8 is the most recent version of the popular YOLO object detection architecture. This model performs bounding box regression and classification all in a continuous series of network layers and thus is more computationally efficient than Faster R-CNN architectures. This computational effectiveness isn't very important for our research as our clients would rather have a model with better predictive power than one that performs faster analysis, however, other researchers have seen better results with YOLO than Faster R-CNN models.

3) Hierarchical: We defined our custom hierarchical architecture as using the Faster R-CNN (Resnet 50 FPN Backbone) for region proposals (RPN) and an EfficientNetV2 Large for classification. The EfficientNetV2 Large is one of the current top performing CNN architectures and is very complex. With lots of inter-class diversity in our data, we believed an extremely complex model such as this would perform well as it would be able to learn the similarities between species within each major group.

B. Model Results

Test set results for each model are displayed in Table III. We observe our best model to be the Faster R-CNN (ResNet 50 RPN Backbone) trained on the dataset with true Blended Images and the Artificial Blended Images that contain primarily the 3 target classes. This model's test set mAP @ 50 of .583 and avg-F1 @ 50 of .643 metric results are average. However, we are impressed by this model's bounding box regression performance and believe many of the test set misclassifications to be caused by debris and specimen appendages in the images. These extraneous predictions will be fairly easy for reviewers to correct and model performance can be bettered by sample photographers cleaning the photo area in greater detail prior to model analysis.

C. Error Analysis

As seen in Figure 4, we observe very strong bounding box regression performance. Even when specimens are nearly overlapping, the bounding boxes are very accurate.

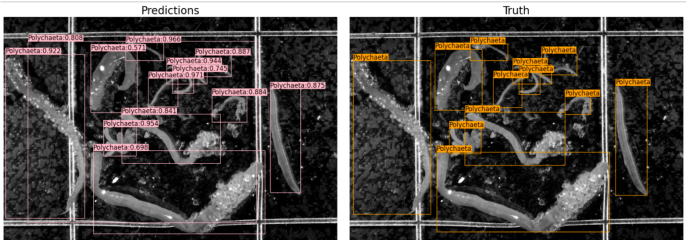


Fig. 4: Example of strong bounding box performance

Our model occasionally classifies debris or sediment incorrectly (Figure 5). Some of the major groups have characteristic crust or shell like structures that are sometimes predicted to be a separate specimen when separated from the main body.

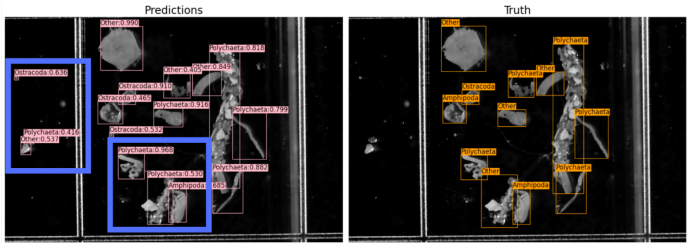


Fig. 5: Example of sediment misclassification errors

Occasionally, individual specimens are broken apart into multiple pieces when photographed (Figure 6). This leads the model to make correct but duplicate classifications of the specimen appendages. Our ground truth data used the 'Other' label as reference to any object in the image other than the 3 main target classes. We see that specimen appendages in our test set, with class label 'Other', are being predicted as belonging to the class the appendage would have otherwise resided in. The human sample photographers can help reduce model errors by removing specimen debris and appendages prior to taking the sample photograph. When misclassifications are made on debris and appendages, human reviewers should have an easy time deleting the corresponding bounding boxes using the Sample Analyzer Application.

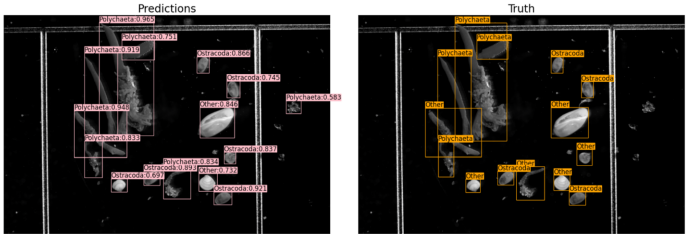


Fig. 6: Example of appendage misclassification errors

IV. Sample Analyzer Application

A. Motivation

1) Quality control: It is very important that sample analyzers have a convenient way to edit model results as the model is not accurate enough for its results to be used for ecology analysis without manual review. We created an intuitive, easy-to-use interface that will allow researchers to upload multiple photos for the model to predict on, review and alter those predictions, and save the results to Excel format. This way the specimen labeling process can be sped up by first running the model whilst the quality of the data recorded about the samples is assured to be accurate by human reviewers.

Before creating the application, a detailed mock-up was created using the design tool Figma. This allowed us to detail all the necessary features that the application should have prior to development. A feedback oriented relationship with the client was maintained throughout the application development

TABLE III: Test Set Results

Model	Training Dataset	mAP @ 50	Avg.-F1 @ 50
Faster R-CNN (ResNet50 FPN Backbone)	All True and Artificial Blended Images	.493	.589
Faster R-CNN (ResNet 50 FPN Backbone)	**Artificial Blended Images primarily for the 3 target classes + all true Blended Images**	**0.583**	**0.643**
Faster R-CNN (ResNet 50 FPN Backbone)	Only True Blended Images	0.529	0.621
Hierarchical: Faster R-CNN (ResNet 50 FPN Backbone) → EfficientNetV2 Large	All True and Artificial Blended Images	0.069	0.144
YOLOv8	All True and Artificial Blended Images	0.399	0.453

process so that all our client's wishes could be incorporated at each stage. A photo of the bounding box editor page from the Figma mock-up can be seen in Figure 7.

2) Non-technical usage (GUI): The Sample Analyzer Application allows ecology researchers to interact with our machine learning model without technical knowledge about Python or machine learning techniques. The application makes it simple to run model analysis and review the model's results. The only dependency for our software is that the user must install Python, with related packages such as Pandas and Pytorch, as described in our installation instructions.

3) Setting up future success: The way in which the Sample Analyzer Application allows for the correction of misclassifications and bounding box errors also sets the groundwork future model improvements. These corrected images can be used as ground truth data that can be used to train new models and achieve stronger results. The application architecture employs machine learning models in a plug-in manner such that newly trained models can be easily integrated.

B. Features

1) Project Saving: Upon launching our application, users will be presented with the option to create a new project or upload an existing project to the application, as seen in Figure 8 (a). If the user opts to upload an existing project, they can select a saved JSON file from their projects directory and the bounding box editor will be populated with the previously saved work and available to continue making revisions. If the user chooses to create a new project, they can choose any number of images to upload for analysis. Once the images are uploaded, our model processes the uploaded images in batches and generates a JSON output file. This file contains all the information about the predictions made by the model on the specimen within the images. Then, the user can launch the bounding box editor to review the model output.

2) Review Model Output: The objective of the bounding box editor is to provide the user with a straightforward way to correct the any inaccuracies in the model's predictions. The editor, shown in Figure 8 (b), is an interactive canvas where the user can create, resize, and delete bounding boxes. Upon clicking a bounding box, the user can also edit the label. To enhance the clarity of the editor, we created a color legend to differentiate the colors between major groups. Additionally, each bounding box is accompanied by text displayed in the top-left corner, which contains the major group label and the confidence score.

We also provide an easy way to navigate through batches of images, by providing a drop-down selection option to select an image to view based on its filename. Furthermore, we have incorporated a 'Next Image' button and a 'Previous Image' in the case that the user wants to incrementally step through images.

3) Output Organization: Upon completion of any necessary edits to the model output, our application offers various saving options. Users can choose to save the project for future continuation, export the results as Excel spreadsheets, or opt for both options. If the user chooses to save the project metadata, they will need to save it to the designated project directory folder within the application's file structure. They can then reopen the project metadata at a later time from the applications home page to continue making revisions. Alternatively, the user can choose to export the results into an Excel spreadsheet, which can be saved anywhere. The spreadsheet provides a comprehensive summary of the batch of images, including the filename, major group, the number of specimens within the major group, as well as total counts of specimen major groups across all images.

V. Conclusion

The results obtained from the model are satisfactory, and combined with the Sample Analyzer application, we are confident that our project will expedite the specimen counting process for the Smithsonian Benthic Ecology lab. The final version of the application has been provided to The Smithsonian, and they have had time to practice working with it on their own machines. Along with the application, we have also

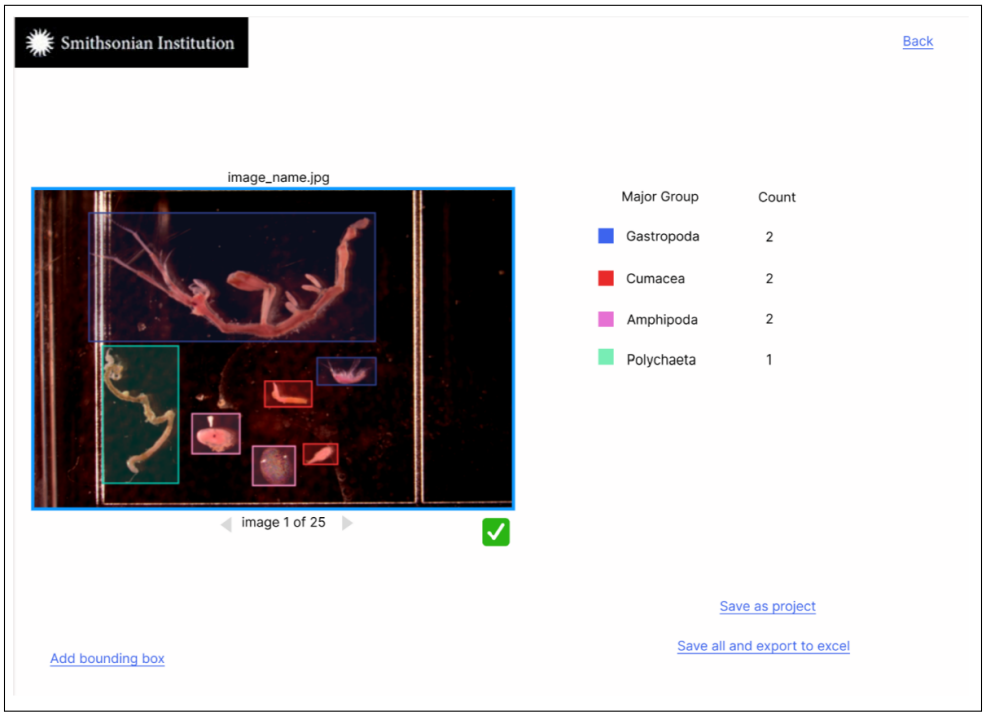
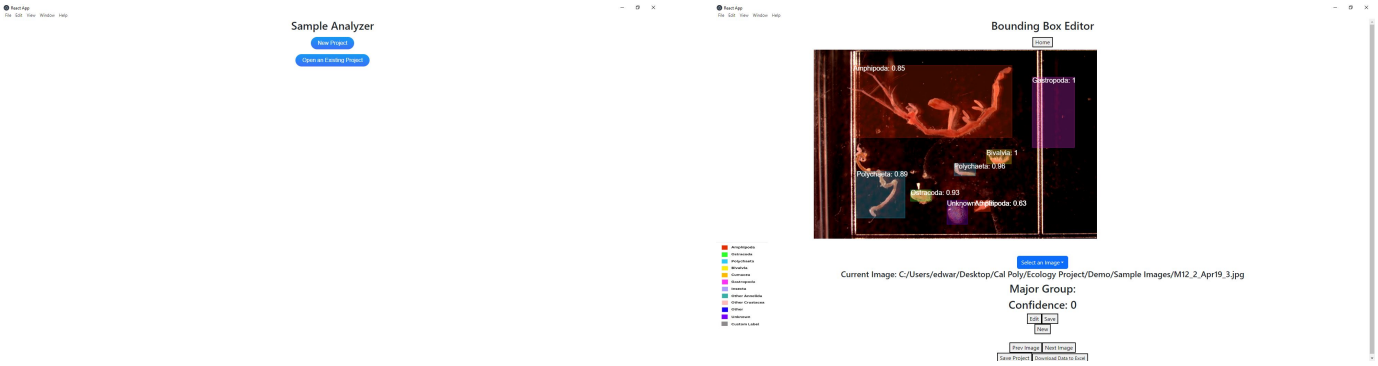


Fig. 7: Figma Prototype



(a) Sample Analyzer Home Page

(b) Bounding Box Editor

Fig. 8: Sample Analyzer Photos

provided all of our documentation and development files to our client so that future developers can improve this project. A Github Organization containing repositories with all the model training code as well as the code for building the Electron application has been created for the client. Although improvements can be made to this project, we are extremely satisfied with the results we were able to accomplish and the final product we delivered to The Smithsonian Benthic Ecology Lab.

VI. Future Work

The first improvement that could be made to the project is improving the model's predictions. With the addition of more ground truth Blended Photo data for model training, we believe our current model's results would improve and additional models could be trained on the new data to better the overall results. With more data, additional models could also be trained that have more target classes than the current model (i.e. all 10 major groups, specimen family groups, and individual species classification). Image segmentation models could also be investigated to determine if they could be more successful than object detectors for specimen classification.

If multiple models are created in the future, another useful improvement would be to add the ability within the application to select which model to run the analysis on via a drop-down menu. This could easily be implemented within our current application design.

The client could benefit from other general improvements to the Sample Analyzer application such as a progress bar for the model running process. As the Benthic Ecology team uses

the application, we anticipate that they will find workflow improvements to suggest to future developers. If other ecologist teams wish to use this application or modify it for their own needs, future developers may be able to tailor this application to alternative uses.

VII. Appendix

A. Application Logistics

There are many languages and frameworks that can be used for building a desktop application, each with unique drawbacks and benefits.

Frontend Programming Language: We examined three primary programming languages for front-end development purposes: Python, TypeScript, and JavaScript. Because TypeScript is a superset of JavaScript, meaning that TypeScript understands all of JavaScript's syntax while also including additional features, we do not compare Python and TypeScript.
Python vs JavaScript: The key aspects of a programming language we were looking for were:

1) Easy-to-use
2) Well-documented

Our team has extensive experience working with Python but limited experience with JavaScript. However, we chose to use JavaScript because the documentation and off-the-shelf capabilities for frontend development in Javascript is better than that in Python. We are confident that adopting JavaScript as the primary frontend programming language will best enable us to meet our client's user interface needs.
JavaScript vs TypeScript: There is a substantial difference between the dynamic typing in Javascript and the static typing in TypeScript. With dynamic typing, variables are not assigned a data type until runtime, making the code more flexible but potentially more error-prone. In contrast, static typing requires variables to be declared with a specific data type, providing greater clarity and reducing the chance of type-related errors. However, our project is small enough such that the likely reduction of type-related errors in TypeScript are not worth the added complexity and longer development time. Furthermore, there is not a significant performance difference between JavaScript and TypeScript in real-world applications, and JavaScript remains the most widely used language for frontend development. Therefore, we have decided to utilize JavaScript rather than Typescript as the primary language for our application, prioritizing ease of implementation and a more straightforward learning curve.

Desktop Application Frameworks: There are a plethora of popular application development frameworks available for us to choose from. However, our project requires the application to be run exclusively on Windows machines and must be compatible with our chosen programming language, JavaScript.

Electron is an application development framework that is solely focused on desktop applications and is suited well to our purposes. Another advantage of Electron is that it runs on a Chromium engine, which provides access to Chrome's developer settings. Finally, for this application, data security is also guaranteed, because all the information is kept in the system at a local level.

To create a user-friendly interface, we have utilized React.js, a flexible and efficient JavaScript library. React.js provides a wide array of off-the-shelf frontend components that we will use.

Electron-Python communication: We use PyShell to create a child process that executes the model script. Once the script is finished, a model output file is written to the project directory. The child process sends a IPC signal back to the main process and the completion of the script is registered by Electron and properly handled.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.