#### DATA 608: Developing Big Data Applications

**Plastic Object Detection Models Comparison**

Final Report

**Prepared by**

Group 6

**Members**:

Yi-Chi Chang (30156294)

Yat Hei Fung (30183611)

Aung Khant Min (30160164)

Peerada Tarasub (30178664)

Hao Su (30191442)

Table of Contents

[1. INTRODUCTION 3](#_Toc132068117)

[2. DATASET 4](#_Toc132068118)

[2.1 Data Collection 4](#_Toc132068119)

[2.2 Data Pre-Processing 5](#_Toc132068120)

[2.2.1 Data Cleaning, Reading, and Processing 5](#_Toc132068121)

[2.2.2 Tools 6](#_Toc132068122)

[3. METHODOLOGY 7](#_Toc132068123)

[3.1 Image Processing Pipeline – OpenCV 7](#_Toc132068124)

[3.2 Convolutional Neural Network (CNN) 9](#_Toc132068125)

[3.3 Faster Region-based Convolutional Neural Network (Faster R-CNN) 9](#_Toc132068126)

[3.3.1 Faster R-CNN Resnet50 FPN V2 Model using PyTorch 10](#_Toc132068127)

[3.3.2 Faster R-CNN using ArcPy 10](#_Toc132068128)

[3.4 YOLOv8 (You Only Look Once) 11](#_Toc132068129)

[3.5 Implement Cloud Services 11](#_Toc132068130)

[4. RESULTS 12](#_Toc132068131)

[4.1 Image Processing Pipeline – OpenCV 12](#_Toc132068132)

[4.2 Convolutional Neural Network (CNN) 12](#_Toc132068133)

[4.3 Faster Region-based Convolutional Neural Network (Faster R-CNN) 13](#_Toc132068134)

[4.3.1 Faster R-CNN Resnet50 FPN V2 Model using PyTorch 13](#_Toc132068135)

[4.3.2 Faster R-CNN using ArcPy 15](#_Toc132068136)

[4.4 YOLOv8 (You Only Look Once) 16](#_Toc132068137)

[4.5 Implement Cloud Services 16](#_Toc132068138)

[5. CONCLUSION 17](#_Toc132068139)

[5.1 Model Comparison 17](#_Toc132068140)

[5.1 Cloud Deployment 18](#_Toc132068141)

[6. CHALLENGES & NEXT STEPS 18](#_Toc132068142)

[6.1 Challenges 18](#_Toc132068143)

[6.2 Next Steps 18](#_Toc132068144)

[7. LEARNING OBJECTIVES 19](#_Toc132068145)

[7.1 Yi-Chi Chang (Lawrence) 19](#_Toc132068146)

[7.2 Aung Khant Min 19](#_Toc132068147)

[7.3 Hao Su (Leo) 20](#_Toc132068148)

[7.4 Peerada Tarasub 21](#_Toc132068149)

[7.5 Yat Hei Fung (Boris) 22](#_Toc132068150)

[8. REFERENCES 23](#_Toc132068151)

# **INTRODUCTION**

Nowadays, plastics, or synthetic polymers, have been extensively used for commercial, industrial, medicinal, and municipal purposes. Their disposal has become a significant environmental concern due to their durability and high resistance to degradation (Lo et al., 2018)[1]. Internationally, National Oceanic and Atmospheric Administration (NOAA), the Intergovernmental Oceanographic Commission (IOC), Outsourced Service Provider's Audit Report (OSPAR), and European Commission have provided several protocols for monitoring marine debris on beaches. In this project, we will focus on plastic disposal that can be seen regularly on the beach.

Normally, the most common method to quantify and characterize beach litter is the visual survey. Even though it is widely used, the downside is that it requires a lot of labor, time-consuming, and difficult to carry out on beaches that are remote or difficult to access. Thus, unmanned aerial vehicles (UAVs), such as aerial drones, are introduced for automated image acquisition and image processing. This makes it a lot faster to collect the data compared to a traditional visual survey. The development of this type of method has been improved continuously by implementing deep learning techniques to enhance the performance and speed of object detection and identification.

Hence, the following objectives are constructed:

1. Develop models using different libraries to detect plastic objects on the beach from photos.
2. Compare the performance of the obtained models in terms of accuracy.
3. Improve the performance of the obtained models (if applicable)
4. Implement cloud service (e.g., Google Cloud Platform) for data storage and running models (if applicable)

# **DATASET**

## 2.1 Data Collection

Twelve ecologically important and vulnerable sites scattered from northwestern New Territories, Lantau Island to Hong Kong Island are selected. A mid-range commercial drone DJI Mavic Pro equipped with a 12 megapixels camera is used to collect data from the beach. It was controlled by a DJI Ground Station Pro application to automate flight for scanning the subsample areas. A series of photos were taken by the drone flying at lower operating heights (5–10 m) above ground level on a sunny day, and the camera was pointed at the nadir (90° to the ground) to generate a high resolution orthomosaic photo by Agisoft Photoscan Professional (Lo et al., 2020)[2].

Map

Description automatically generated  
*Figure 2.1: Hong Kong Shore Map*

After collecting the drone pictures of beaches in Hong Kong, trashes are manually labeled using ArcGIS Pro, then and sliced into 6000+ images in the form of XML files and tiff files. The figure below shows one of the sites, Discovery Bay, that the drone took pictures from, in which the blue dots represent the trash location and distribution as annotated manually. The middle one is an example of how the bounding box is located and labelled the trash. The list on the right is the structure in the XML files that shows the class of the object and coordinate information of the bounding box by indicating the maximum and minimum of x-axis and y-axis coordinates.

*Map

Description automatically generated*

*Figure 2.2: An example of the data image, labels, and the corresponding bounding boxes in XML file*

## 2.2 Data Pre-Processing

### 2.2.1 Data Cleaning, Reading, and Processing

**Data Reading and Processing:**

We used glob to get all the tif formatted images from the folder and read them using opencv-python. The read image is accepted as a numpy array in python. We resize the image to 256\*256\*3 to make sure that they are all the same size. After resizing, we extract the file name and saved the images in a dictionary with key as file names and values as resultant images. Afterward, we glob to another directory for getting all the corresponding XML files for the images.

**Data Cleaning:**

The structure of the dataset is Pascal VOC (Visual Object Classes), meaning each image will pair with one XML file indicating the location of the trash object in the image with the coordinates that describe the bounding boxes in the image. The XML file contains tags for image file name, image size, object name, and the coordinates of the bounding boxes. To be able to match the label and the image together for the input of machine learning model, we first parsed the xml file and read the image data. Next, we stored the data into two dictionaries for with filename as key, and image/ labels as value. As the input of CNN are required to be lists or arrays (one for labels and another for images), we also set up a function that can return two list with matching order.

During the XML parsing and image reading process, two issues are addressed:

1. Filter out the target object from XML file: The data producer not only labeled the trash object but also labeled some background and plastic-liked object for training purpose. The type of the objects is annotated through name tag under the object, with 1 representing real plastic object, 2 representing plastic-liked object such as white rocks or woods, and 5 representing background. To simplify the project, for some of the model, we filtered out the real plastic object only for machine learning training and testing.
2. Uncertainty of the quantity of plastic objects in one image: The images data was sliced from another photo into equal sizes, but the plastic objects are scattered randomly, meaning that some of the image might have more than 1 plastic object and some might even have no plastic object. (As shown in figure 2.2 above) To address this issue, for some of the model that doesn’t accept multiple labels (like CNN), we simply choose the first plastic object in the XML file and for those model that accept multiple labels, we keep all the objects for training and testing.

Last, to check whether the cleaning process affects the precision of the label, we plot the image data with bounding box in Python as shown below.

一張含有 圖表 的圖片

自動產生的描述

*Figure 2.3: Data cleaning process.*

### 2.2.2 Tools

1. **ElementTree XML API (xml.etree.ElementTree):​**

We used ElementTree XML to parse the xml file to extract the value of tag. We start with parse and root function to read the XML file. Next, to extract the value, we use find all and find to visit the tags we are interested in. After that, we can simply use .text to extract the coordinates of the bounding box and stored it into a list with length of four.

1. **Glob:**  
   Glob is a library used for getting all the files within a folder. It is often used together with a library called “os” and it is mainly used for scripting purposes. We use glob.glob for extracting all the files with specific file formats we need for this project.
2. **OpenCV:**

OpenCV is one of most powerful image processing libraries in python. For our project, it is mainly used for image reading, displaying, and processing. For reading and displaying, we used imread() function to read the image and use imshow() to show the image in another window. cv2.rectangle function is also used to draw the bounding box that we parsed from the xml file in the previous step. Regarding the image processing, we used image resizing techniques along with image enhancement transformations.

# **METHODOLOGY**

Three deep learning methods will be used to develop object detection models in order to detect plastics on the beach as long as accuracy computation. All models will be compared based on accuracy to determine the best model. After that, such models will be deployed to Google Cloud Platform.

## 3.1 Image Processing Pipeline – OpenCV

To improve accuracy, we decided to build an image processing pipeline using opencv-python. Instead of passing in the raw image data into the convolutional neural network, we went for a processed image approach. The pipeline goes as follows: the image is extracted from the dataset. Afterwards, we save the image in a numpy array. The image is made bigger using an interpolation-based image enhancement. We tried using FSRCNN (Fast Super Resolution Convolutional Neural Networks) but that OpenCV function does not seem to be working. We even dug into the opencv-contrib-library to see whether we can make some quick fixes. We investigated the C programming-based files and their respective header files in the library. We installed gcc library to cmake the opencv library from source[3]. We used commands for getting build essentials and cmake builds. Unfortunately, we had to stick with the interpolation methods. The pixel counts increase and the file size increases. After applying interpolation method, we get a bigger image at 512\*512\*3 size. After getting that aggrandized image, we pass the image to transformations as Figure 3.1 below depicts. Upon testing the model, we disregarded the interpolation method as it decreases accuracy.

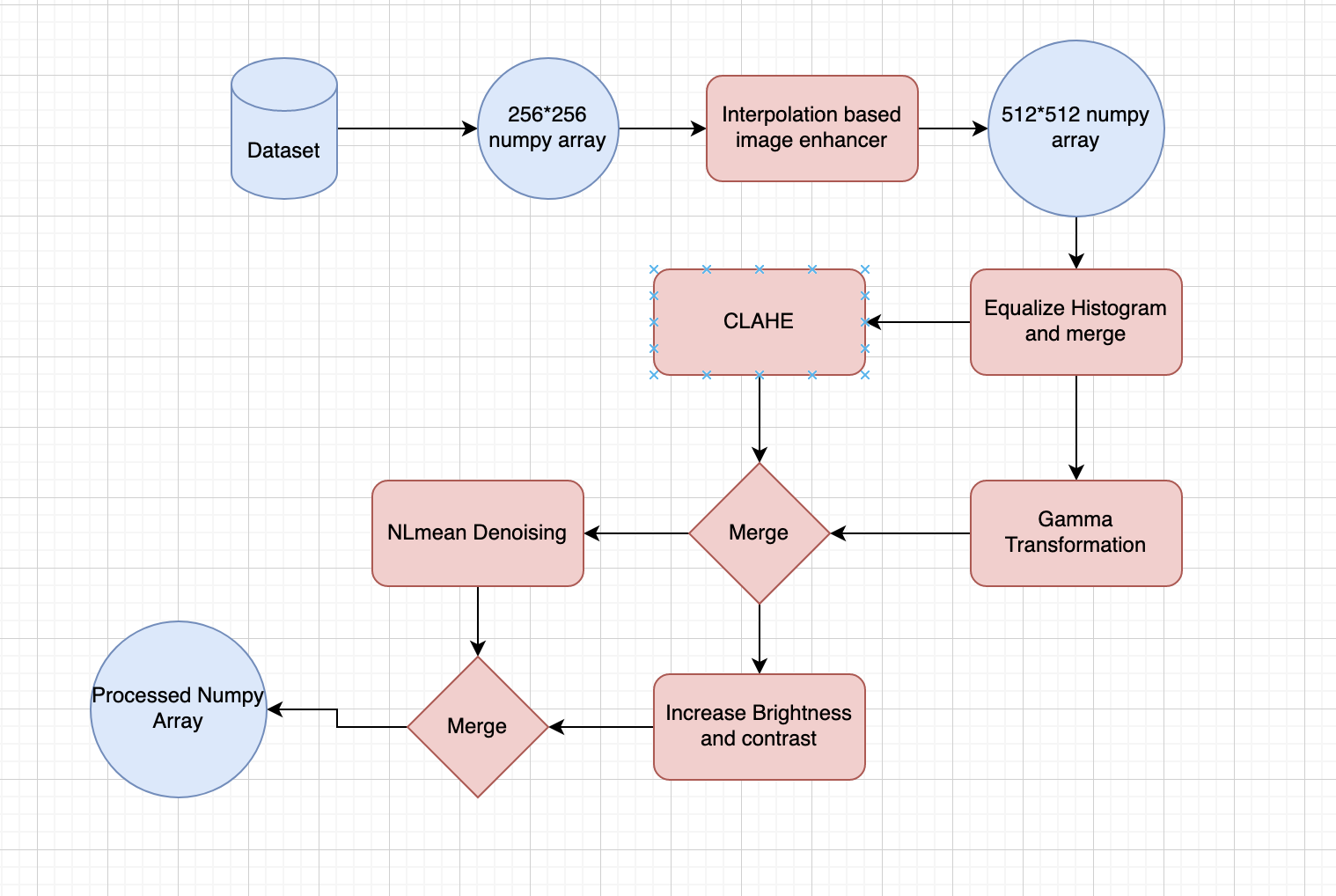


Figure 3.1: OpenCV architecture

The first layer is histogram equalization. The passed in image is turned into a grayscaled image and passed into the histogram equalization from opencv. The histogram equalization is applied for making images have more contrast and more distinct. In Figure 3.2, we can see the image after histogram equalization is much more distinct and easier to recognize objects from. The downside to histogram equalization is that it can cause bright images to be brighter making the image less distinct. As a result, what we did was we blended the original image with the histogram equalized image. Then, we can create some distinct features while reserving some important contents of the original image.

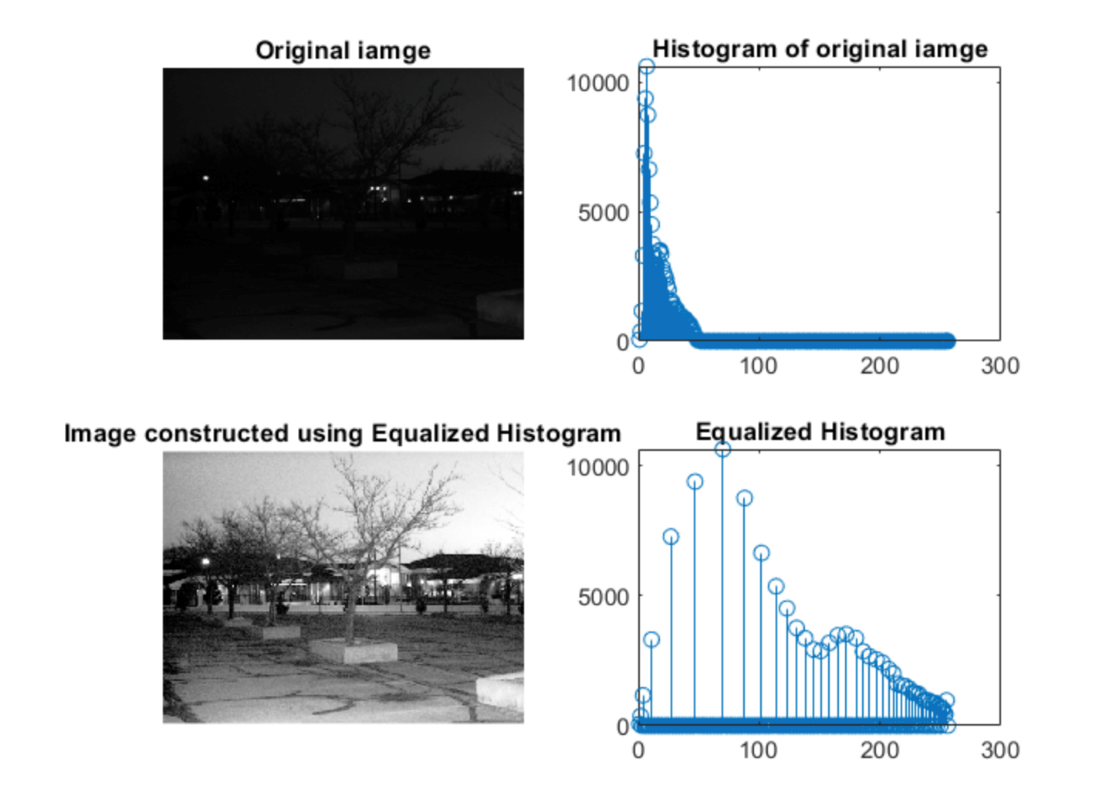


Figure 3.2: Histogram Equalization example[4]

Upon merging, the same image is sent to two different types of transformations and merged again. The first transformation is we did gamma correction on the image to make sure the images with low light or low contrast to be corrected. Gamma correction allows low light level images to be brighter. On a separate node, we passed the image into Clahe transformation for another type of histogram equalization. This time, we passed in the image with colorway to conduct histogram equalization. As shown in figure 3.3, we combined these two images to get a higher light level image with proper contrast.

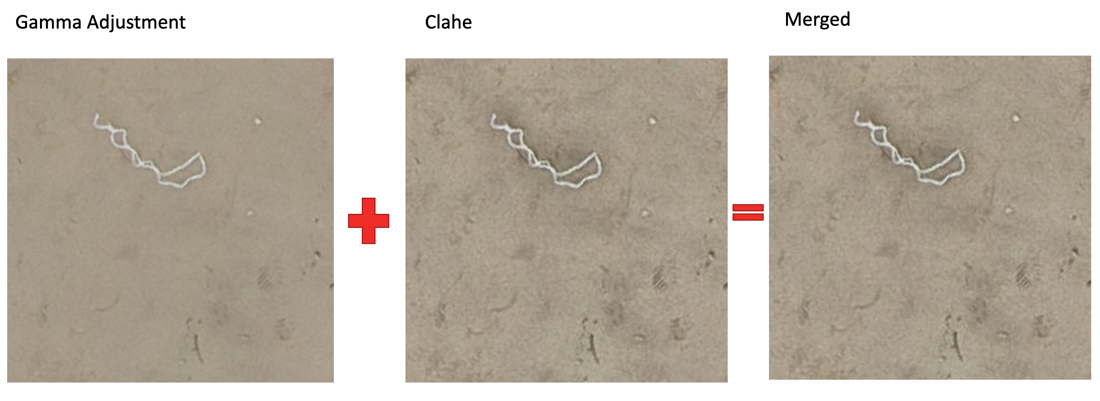


Figure 3.3: Merging of Gamma and Clahe

The resultant image from the previous part is then passed in to alpha and beta changes in brightness adjustment. Based on the image’s need, the alpha and beta will adjust the brightness of the image and the contrast. As for our case, the image gets increased brightness and contrast after going through the transformation. We also denoised the image so that we can combine similar pixels alike and assign a color. Denoising is very important as it leaves out parts with distinct shapes and tries to blur the background. The effect on the image is almost like a gaussian blur with feature extraction. As figure 3.4 depicts the images are merged to get a middle ground of denoised and original noise. The merging step is applied so that if the denoising affect wipes out the important features of the image, the image will have some pixels that mark the features.

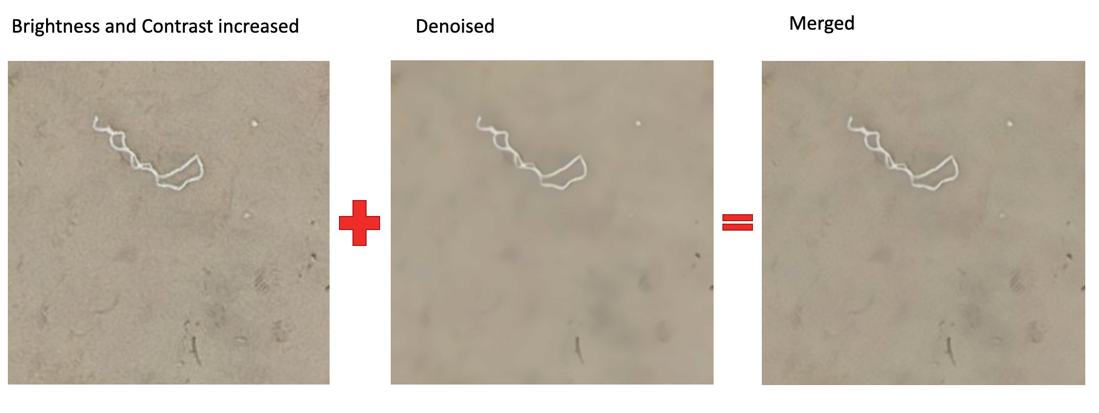


Figure 3.4: Merging of Denoised and Brightness Adjusted

The process is repeated for all the images in the dataset. We save the images that are outputted from the pipeline to be used in the Convolutional Neural Network.

## 3.2 Convolutional Neural Network (CNN)

We will develop our own 2D Convolutional Neural Network model for object detection. How we conducted convolutional neural network was that we did a train\_test split on the function to split the images to training and testing. We used 0.8 of the image data as our training data and 0.2 as our testing data. The images are then divided by 255 to normalize and scale the values between 0 and 1. We used convolutional neural network model architecture from Sequential class of TensorFlow Keras. The model has 3 convolutional filtering layers and a max pooling layer. We use rectified linear unit (ReLU) as our activation function as it is a regression problem. We used MSE as our loss function with a learning rate of 10-2. Afterward, we implemented early stopping to make sure that the model stops training when the validation loss does not improve after 2 epochs. After specifying all the parameters, we fit the data for 10 epochs and used it for predicting results. We computed mean squared error of our testing set.

## 3.3 Faster Region-based Convolutional Neural Network (Faster R-CNN)

Faster R-CNN is one of the most advanced multi-stage object detectors. It applies Region Proposal Network (RPN) as its algorithm with the CNN model. It can predict the object bounding boxes and corresponding scores at the same time. The brief architecture is that the framework will take an input image and pass it to the convolutional network which returns convolutional feature maps. Then, it will apply RPN on these features maps and get object proposals. Next, the predicted region proposals are reshaped using a ROI pooling layer. These proposals will be passed to another layer to classify the bounding boxes for the image[4] [5].

Diagram, engineering drawing

Description automatically generated

Figure 3.5: The architecture of Faster R-CNN

### 3.3.1 Faster R-CNN Resnet50 FPN V2 Model using PyTorch

The Faster R-CNN ResNet-50-FPN V2 combines the ResNet-50 architecture, which is a widely used Convolutional Neural Network (CNN), with the Feature Pyramid Network (FPN), which is used to generate feature maps. This new version of Faster R-CNN is developed by PyTorch, which is an optimized tensor library for deep learning using GPUs and CPUs.

We plan to implement this model with our dataset by starting with data image transformation, then loading the dataset and dividing it into 80% training set and 20% testing set. The model will be instantiated and fit with the training data. Finally, we plan to visualize the results – which is the image along with the predicted bounding boxes – and compute the accuracy.

### 3.3.2 Faster R-CNN using ArcPy

Since the dataset is generated from ArcGIS Pro, we will try to use the build-in object detection model provided in the library of ArcPy, which is a package that provides a useful and productive way to perform geographic data analysis, data conversion, data management, and map automation. We will use Arcgis’s prepare\_data() to pre-process the images e.g., rescale, sheer, rotate, flip the images. [6]Then, the dataset will be split into 90% training data, 10% validation data​. After instantiating the model with Faster R-CNN, the data will pass and fit the model.

## 3.4 YOLOv8 (You Only Look Once)

This object detection Developed by Ultralytics. Since the data is exported from Acrgis Pro, it includes tiff, tfw image files and xml label files while Yolo only uptake image and txt format label, xml files are converted to txt. Then, the data is split into 3 parts, training, testing, and validation sets. In particular, the data is split into 70% training data, 20% validation data, and 10% testing data. A YAML file is set up a path to the model. After instantiating the pretrained model YOLOv8l.pt​ provided by Ultralytics (size (pixels): 640), mAPval: 52.9, Speed CPU ONNX (ms): 375.2, Speed A100 TensorRT (ms): 2.39, params (M): 43.7, FLOPs (B): 165.2)

## 3.5 Implement Cloud Services

In this study, we deployed our trained model on the Google Cloud Vertex AI platform to provide a scalable, reliable, and secure prediction service. We converted the model to TensorFlow SavedModel format and uploaded it to a Cloud Storage bucket. We then created a Vertex AI endpoint, deployed the model, and managed multiple versions of it throughout development. The implementation of access control and authentication ensured secure access, and API integration allowed client applications to make prediction requests. By leveraging Google Cloud Vertex AI, we effectively managed and monitored our model while offering a seamless experience for accessing its predictions.

# **RESULTS**

## 4.1 Image Processing Pipeline – OpenCV

Upon running through all the image transformation, we tested the image using a feature detection method. In figure 4.1, we can see three distinct pictures. The first one is the original picture. The middle picture is the difference of gaussian of the original picture. The third one is the difference of gaussian result of the processed picture. From the picture we can see that the noise in the image is reduced after performing the processing pipeline. The dark pixels are more focused on the distinct object instead of having scattered across, thus making the object more prominent.



Figure 4.1: Result of the pipeline

## 4.2 Convolutional Neural Network (CNN)

In our approach, we utilized a Convolutional Neural Network (CNN) to address the detection problem. The initial step involved splitting the data into training and validation sets, with an 80% allocation for training and 20% for validation. Subsequently, we normalized the data by scaling the values between 0 and 1, a common practice to expedite model convergence during training.

Our CNN model comprised three convolutional layers, followed by max pooling layers. After flattening the outputs, we added two dense layers, with the last dense layer consisting of four output neurons and a linear activation function. The model was compiled using the Adam optimizer with a learning rate of 0.001 and a Mean Squared Error (MSE) loss function, while monitoring accuracy metrics. We then trained the model for 10 epochs using the training data and evaluated its performance on the validation data.

For simplicity, this model was designed to identify and output a single bounding box for plastic detection.

We trained the model using both the original data and the processed data to compare the effects of preprocessing. We found that image processing indeed had a positive impact, as the validation set accuracy of the model improved from 52.36% to 58.15%.

In the example image below, the green box represents the original label, and the red box signifies the model's output. While the model successfully detected plastic on the beach in some cases, it made incorrect detections in others. We believe this may be due to inaccuracies in the data labels and that additional time is needed for model adjustments.

在山上

中度可信度描述已自动生成

Figure 4.2: An example of the output image

## 4.3 Faster Region-based Convolutional Neural Network (Faster R-CNN)

### 4.3.1 Faster R-CNN Resnet50 FPN V2 Model using PyTorch

Firstly, a class to load and process the dataset was created. The class contained methods to take the directory path where the dataset was stored, load their object bounding boxes and labels from XML files, calculate the number of images in the dataset, return the managed data which contained image ID, bounding boxes, area of the bounding boxes, and labels.

Next, The Faster R-CNN model was instantiated with several setups. The model was constructed with the ResNet50 architecture with the option to be moved to the GPU if available.

Then, some transformations were made to the dataset in order to be compatible with model requirements. All images were transformed into tensor form and were normalized. All transformed data were loaded using the aforementioned class with the help of DataLoader and divided into 80% training set and 20% testing set.

A picture containing map

Description automatically generated

Figure 4.3: An example of the training data image

The model was then trained with the training set and tested with the testing set. Unexpectedly, the model took around 4 hours to complete. Hence, we decided to generate a subset of the dataset by randomly choosing 20% out and divided the training and testing set from this subset instead. The visualization of the testing set’s results is shown below:

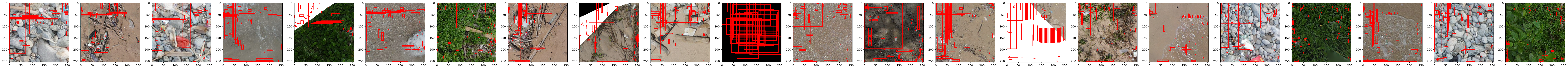


Figure 4.4: An example of the training data image

Obviously, the results obtained were not as expected. The bounding boxes appeared to be unacceptably mislabeled and were not meaningful. This might happen because of the sensitivity to mislabeled original data or too little amount of training data. Modifying the dataset and results tends to consume too much time and does not guarantee the results. Thus, we were moving to the next model, which was the Faster R-CNN using ArcPy.

### 4.3.2 Faster R-CNN using ArcPy

A picture containing stone, rock, wall, nature

Description automatically generated

Figure 4.5: An example of the results obtained by Faster R-CNN using ArcPy

Since the Faster R-CNN from PyTorch did not generate satisfactory results, we tried to use the Faster R-CNN Resnet50 RPN from ArcPy library provided by Arcgis Pro. ​Firstly, the whole raw dataset are preprocessed by the prepare\_data() function, which includes rescale, sheer, rotate, flip the images, and then split the data by default. Then, I used faster R-CNN to train the model for 20 epochs, and some trash was detected, but the precision is not high. It achieved 0.3081 precision, 0.3311 training loss, and 0.4330 validation loss. The image above displays some plastic waste from the beach that was utilized for performance testing. The Faster R-CNN successfully detected multiple waste objects, localized by two bounding boxes, and were classified as trash.

## 4.4 YOLOv8 (You Only Look Once)

A picture containing text, shop

Description automatically generated

Figure 4.6: An example of the results obtained by YOLO

Unlike classical object models, Yolo and its derivatives are one-stage frameworks converting object detection to a bounding box regression problem, which means Yolo is advantageous on calculation speed with less parameters required to be set compared to two-stage frameworks (Kim et al., 2020). Considering the future application of real-time trash detection on the beach, Yolo seems to be a more promising approach. Instead of resnet50 R-CNN, the backbone network is CSPDarknet53 that is an enhanced version of the Darknet53 network with Cross Stage Hierarchical (CSP) connections for better gradient flow and feature fusion, providing strong feature extraction capabilities. After training the Yolov8 model for 20 epochs, some trash and background was detected, and the precision is much better than Faster R-CNN. It achieved 0.54612 precision, 3.8806 training loss, and 4.454 validation loss. As shown in the above figure, some objects are labeled by bounding boxes, in which the number is representing the confidence that classification, and the trash is mostly classified correctly. The average precision is 0.5461.

## 4.5 Implement Cloud Services

By leveraging Google Cloud Vertex AI, we successfully provided a scalable, reliable, and secure future prediction service. The platform's robust features enabled effective management and monitoring of our model. We developed a function to convert test images to JSONL format files. And we planned to develop a user-friendly interface for accessing its predictions. Unfortunately, we encountered issues during testing, as there were errors in the output and all our credits were depleted. Consequently, we could not access the model any further and were unable to proceed with subsequent attempts.

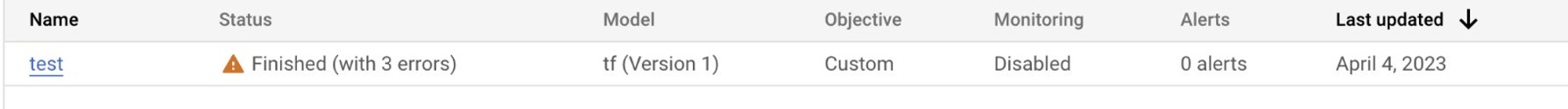


Figure 4.7: The failure prediction in Google Cloud Vertex AI

# **CONCLUSION**

## 5.1 Model Comparison

Since each model use different loss function, which will lead to a different scale of loss value, making the loss value not comparable. In a CNN, the uses loss function is the categorical cross-entropy loss, which is used to classify objects into different categories. This loss function measures the difference between the predicted probability distribution and the true probability distribution of the target classes. On the other hand, Faster R-CNN uses a multi-task loss function, which based on categorical cross-entropy and uses an additional regression loss to predict the bounding boxes of objects in the image. The regression loss used in Faster R-CNN is the smooth L1 loss, measuring the difference between the predicted bounding box coordinates and the true bounding box coordinates of the objects in the image. Yolo uses a new loss functions for improved training, Complete IOU (CIoU) loss, that composes of three loss functions in consideration of the corresponding weight and number of grids in the input image: the confidence loss or IoU (the objectness of the box, or the overlapping area), the localization loss (the sum-squared error between the predictions and the ground truth, or distance between center points) and the classification loss (the squared error of the class conditional probabilities for each class, or the aspect ratio) (Benjdira et al., 2019[7]; Mao et al., 2022[8]).

Since the loss values were not comparable, we compared the average precision of each model instead.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Epoch** | **Train Loss** | **Valid Loss** | **Average Precision** |
| CNN | 10 | 6880.3696 | 6823.4814 | 52.36% |
| CNN  (With Processed Image) | 7 | 6526.1787 | 6442.1226 | 58.15% |
| Faster R-CNN | 20 | 0.331076 | 0. 433003 | 30.81% |
| YOLOv8 | 20 | 3.8806 | 4.454 | 54.61% |

Table 5.1: The Performance of the Model using CNN, CNN with Processed Image, Faster R-CNN, and YOLO

To avoid overfitting, the training process implemented early stopping method for CNN training process, and it stopped at 7 epochs. From the table above, it can be seen that CNN model that was trained with processed image from OpenCV provided the best result of 58.15% of average precision, followed by YOLO, CNN, and Faster R-CNN. The CNN was then selected to deploy to the cloud environment.

## 5.1 Cloud Deployment

Our model has been deployed on Google Vertex AI to allow prediction from external data. However, although the deployment process is successful, there were error occurs in the prediction output. We are unable to resolve the issue as we used up all the credit we got. More future next step for cloud deployment will be addressed in the following section.

# **CHALLENGES & NEXT STEPS**

## Challenges

While we used the training data to implement Faster R-CNN model, the result could not locate the and classify the trash correctly. We then checked the image data that integrated tiff files and coordinates in the xml files, but some bounding boxes are mislocated inside the original photos, which is very likely to be the reason leading to the wrong object detection results. Also, there are more than 6000 photos in the dataset, which is around 1.22 GB in total, leading to drastically long runtime when developing and testing the model, as well as high CPU & GPU consumption. In this project, we used the resolutions of 256 pixels, which is relatively low to reflect all the details of the target and background.

## 6.2 Next Steps

As mentioned previously, we suspected that the original dataset was mislabeled or the preprocessing step could not mount the bounding boxes onto the image correctly, which led to the wrong predicted object bounding boxes, and the accuracy of the label should be improved. To solve and improve this issue, .tfw files could be considered as it is used in saving the world location of an aerial satellite photograph. Also, the OpenCV-processed image can also be applied to the Faster R-CNN models and might lead to better accuracy of the results. Since this project is in a preliminary stage, all obtained models’ parameters can be further fine-tuned, for example, learning rate, batch size, weight initialization, anchor scales and aspect ratios.

For our next steps in the cloud service, we plan to address the ‘errors’ issue encountered during the prediction phase on Vertex AI, identifying the root causes and implementing solutions to fix the error. Once this issue has been resolved, our objective is to develop a web application that will integrate the model's API, allowing for practical use of the model in detecting plastic on beaches. This web application will serve as a user-friendly platform for clients to upload images and receive predictions on the presence and location of plastic debris, thereby demonstrating the real-world applicability of our model and contributing to efforts in reducing plastic pollution.

# **LEARNING OBJECTIVES**

## 7.1 Yi-Chi Chang (Lawrence)

My learning objective is to:

* Learn how to apply OpenCV to read, show, resize and process the images picture by wrangling 6,000+ images for this project by 31 March 2023.
* Learn how to parse XML files with xml tree package in Python by parsing the 6,000+ labels dataset for this project for the preparation of the CNN model by 31 March 2023.

What I had planned on learning

Without any image process or Pascal VOC data wrangling experience, I had planned to learn Open CV from online resources such as the documentation of Open CV package and YouTube videos. I will also apply some basic image process method such as slicing, blur, canny …etc. To familiar with the function available on OpenCV. For XML parser, I planned to read the documentation of XML file to learn how to parse and extract the value from the file.

Demonstrate evidence of your learning

The corresponding task associated with my learning objective is to read all images data and parse all XML files to filter and extract the bounding box information. Afterwards, write a code to draw bounding boxes on the image. All the tasks are successfully completed.

Overall learning result

From my perspective, I would conclude that I complete my learning objective as I am able to generate lines of code for my teammate to read in the image data and match it with the annotations so that they are able to construct the machine learning model. I also generate a function to draw the original and predicted bounding boxes on the image to evaluate the performance of the model. Besides, as my task is at the early stage of the project’s life cycle. I also spent some time on learning CNN model to support Leo on improving the performance of the model.

## 7.2 Aung Khant Min

My learning objective is to:

* To enhance my understanding of image processing, I used glob, OpenCV, multiprocessing and scripting techniques to process all the images for passing onto Leo's CNN by 26 March 2023.
* To learn how to use tensorflow for deep learning, I assisted Leo and Lawrence by debugging the CNN model's problems and testing the model they built with different data.

What I had planned on learning

I had planned on understanding the image processing library “OpenCV” more in depth. I have prior experience with OpenCV as I have parsed video and image data and ran bar code scanning algorithm on the image data. I used OpenCV briefly on raspberry pi during that project. I noticed that the library has a ton of functionalities. For this project, I intended to research what kind of transformations can be done to enhance the results of the convolutional neural network. I planned to read the opencv documentation and numerous websites to debug the opencv codes and how to make the models work. I also intended on learning how to build a convolutional neural network and hyperparameter tuning by helping Leo build it.

Demonstrate evidence of your learning

In order to enhance my understanding of image processing, I built a multi-layer image processing pipeline from scratch that utilizes various types of image transformation techniques. In order to develop the pipeline, I had to dive into package files and documentation. I wrote multiple scripts that take in data from the folder, process them, and rename them to save them. I was also able to change the code in a multiprocessing format to make the processing time faster. I helped with Leo’s CNN and implemented early stopping in Leo’s CNN code as well as tested out a couple different versions of CNN with different epochs and parameters.

Overall learning result

For my overall learning result, I believe that my image processing knowledge is strengthened along with my TensorFlow deep learning knowledge. Firstly, I learnt how to write python scripts to extract different types of data and process the images efficiently using a pipeline. As I resized images, I had to script through the labels and adjust to the changes based on the size of my new images. I also learned to read CNN code, play around with it, change parameters, and implement early stopping. The unexpected thing I learned during this project is how to build OpenCV from source and how to dig into the code of packages that are pip installed to change the code inside. The digging into the code base part was very useful for me as I was able to change the code in the LazyPredict library to make it run for 607. My contributions are image processing pipeline, scripting to get all the images from a directory, running multiple scripts to get the processing to work, running the CNN with different data multiple times, and debugging for people whenever someone in the group needed help. Overall, I would say the project was a success as I was able to complete all my learning objectives as well as improve my Python coding skills.

## 7.3 Hao Su (Leo)

My learning objective is to:

* Train the CNN model and choose the suitable number of layers and neurons and suitable activation functions by 31 March 2023.
* Learn how to use cloud services by deploying the project on GCP by 7 April 2023.

What I had planned on learning

Considering my limited theoretical knowledge of CNNs from a few years ago, I planned to dive deeper into the topic, treating it as a nearly new concept for me. My learning objectives included revisiting the fundamental principles of CNNs, understanding the roles of convolutional, pooling, and fully connected layers, and exploring various activation functions. I aimed to apply this theoretical knowledge to design, train, and optimize a CNN model for the detection problem by experimenting with different configurations of layers, neurons, and activation functions. Simultaneously, I planned to familiarize myself with the deployment and management of machine learning models on cloud platforms, specifically GCP's Vertex AI, to create a robust, scalable, and secure solution for the future prediction service.

Demonstrate evidence of your learning.

Throughout the course of this project, I successfully trained a CNN model with multiple layers and neurons and experimented with various activation functions to optimize its performance. The validation set accuracy improved significantly after preprocessing the images, indicating a well-designed model. Furthermore, I effectively deployed the trained model on GCP's Vertex AI platform, which involved setting up a project, uploading the model to a Cloud Storage bucket, and creating an endpoint for API access. In the process, I gained valuable experience in cloud service deployment and management.

Overall learning result

The learning objectives set forth at the beginning of this project were effectively achieved. I gained valuable insights into training and optimizing CNN models for detection tasks, particularly in selecting suitable layers, neurons, and activation functions. This understanding was evident in the performance improvement observed when comparing the model's validation set accuracy before and after preprocessing the images. The project allowed me to develop a deeper comprehension of the nuances of neural network design and optimization, which is essential for tackling similar problems in the future.

In addition to honing my skills in neural networks, I also gained hands-on experience in deploying and managing machine learning models on cloud platforms, specifically GCP's Vertex AI. This aspect of the project expanded my knowledge of cloud-based solutions, resource management, and API integration, making me proficient in handling real-world applications that require scalable, reliable, and secure machine learning services. As a result, I am now better equipped to undertake future projects involving artificial intelligence and cloud-based solutions, contributing to my growth as a professional in this domain.

## 7.4 Peerada Tarasub

My learning objective is to:

* Learn how to perform object detection by implementing a Faster R-CNN model using PyTorch library and able to obtain the output by 31 March 2023.

What I had planned on learning

I had planned to learn how to perform such a task by starting on researching about the architecture of Faster R-CNN and how PyTorch works to build fundamental understanding of the method and algorithm. After that, I planned to follow closely on Debugger Café[9] to construct a model and parse the dataset into the model. Any transformation might be applied to the dataset in order to be compatible with the model requirement. After training and testing the model, I planned to display the original images with the predicted bounding boxes that I obtained from the model.

Demonstrate evidence of your learning

Even though my model did not work well as expected as mentioned in the previous section, I did learn a lot during the process. I was able to instantiate the model, transform the data and fit the data with the model, train and test the model, and eventually able to obtain some results and display some images. The results of my learning can be seen in section 4.3.1.

Overall learning result

From my perspective, I marked my learning objective as completed as I learned a lot about how to implement the model, able to get the model successfully run, and obtain the results – even though it was not as good as expected – by the indicated deadline.

## 7.5 Yat Hei Fung (Boris)

My learning objective is to:

* Learn and implement the related libraries (ArcPy and Yolov8) for object detection model development by 21 March 2023
* Familiarize and compare different Neural Network models (Faster R-CNN and Yolov8) for object detection modelling, such as the precision and loss functions, by 10 April 2023.

What I had planned on learning

With some experience in environmental studies, including field work and sample collection, I am interested in proceeding the project to the next stages. To achieve this objective, I planned to learn some background knowledge such as the history of object detection, difference between object detection models, and their architectures. Thus, I studied through the documentations provided by the related libraries, reviewed some literature through google scholar to gain a boarder understanding of application and rationale of implementation of different object detection models.

Demonstrate evidence of your learning

Trained object detection models using Faster R-CNN and Yolov8, and successfully obtained preliminary result showing the performance of the models and potential improvement by fine tuning parameters. Also, I am able to tell the difference of loss functions used in different object detection models as shown in section 5.1, model comparison, and the performance comparison in section 4.3 and 4.4.

Overall learning result

I completed my learning objective as I generated lines of code for the project to read in the image data, matching with the annotations, and constructed the machine learning object detection models that can detect some objects in the images and calculated the loss using the built-in functions. Besides, I am able to tell the difference of loss functions used in different object detection models.

# **REFERENCES**

[1] Lo, H. S., Xu, X., Wong, C. Y., & Cheung, S. G., “Comparisons of microplastic pollution between mudflats and sandy beaches in Hong Kong,” *Environ. Pollut.*, no. 236, pp. 208–217.

[2] Lo, H. S., Wong, L. C., Kwok, S. H., Lee, Y. K., Po, B. H. K., Wong, C. Y., Tam, N. F. Y., & Cheung, S. G, “Field test of beach litter assessment by commercial aerial drone,” *Mar. Pollut. Bull.*, no. 151, 2020.

[3] “Introduction to OpenCV,” Open Source Computer Vision. [Online]. Available: https://docs.opencv.org/3.4/df/d65/tutorial\_table\_of\_content\_introduction.html

[4] Neel Joshi, “Histogram equalization,” Jan. 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/01/histogram-equalization/

[5] Sharma, Pulkit, “A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 - with Python Codes),” Analytics Vidhya, Nov. 2018. Accessed: Mar. 30, 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/.

[6] Gandhi, Rohith., “‘R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms.,’” Medium, Jul. 2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

[7] “‘Faster R-CNN Object Detector.’ ArcGIS API for Python.” Accessed: Mar. 12, 2023. [Online]. Available: https://developers.arcgis.com/python/guide/faster-R-CNN-object-detector/

[8] Benjdira, B., Khursheed, T., Koubaa, A., Ammar, A., & Ouni, K. , “Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3.,” pp. 1–6, 2019.

[9] “Deep learning networks for real-time regional domestic waste detection,” *J. Clean. Prod.*, no. 344.

[10] “‘How to Train Faster R-CNN ResNet50 FPN V2 on Custom Dataset?’ DebuggerCafe,” Nov. 2022. [Online]. Available: https://debuggercafe.com/how-to-train-faster-R-CNN-resnet50-fpn-v2-on-custom-dataset/.