Journal of High School Science

Using Machine Learning to automate trash detection on a limited dataset with the YOLOv5 object detection network

Qureshi M[1], Fox S[2]

Submitted: July 8, 2023, Revised: version 1, August 20, 2023
Accepted: August 21, 2023

Abstract

Waste in the ocean is one of the most demanding environmental challenges in today's world. Marine debris poses an extensive list of threats to aquatic life and places financial burdens on local governments that may be responsible for funding its removal. In recent years, there have been increased efforts to develop technology that can automate the process of waste removal from rivers to prevent the flow of litter into oceans. However, because the process of automatically removing trash requires highly advanced technology, these methods can be very costly to implement and are therefore only used in select areas. In order to widen the impact of financial resources, it is more effective to automate the process of trash detection and notify the appropriate authorities to take action accordingly. In this paper, we propose a mode of trash detection that uses the object detection architecture *YOLOv5* to detect trash in rivers. To supplement the insufficient datasets for use in our model, we manually photographed and annotated images of surface-polluted rivers to enable us to train, validate, and test the model. Even with the lack of extensive data, the performance of our model is similar to that of other adequate waste detection networks.

Keywords

Machine learning, Computer vision, Object detection, Transfer learning, Environment, Pollution, Waste removal, Great pacific garbage patch, Ocean cleanup, Aquatic waste

---

[1]Corresponding author: Manaal Qureshi, Chatham High School, 255 Lafayette Ave, Chatham Township, NJ 07928, USA. manaal.uq@gmail.com.
[2]Susan Fox, Professor, Computer Science Department, Macalester College, 1600 Grand Ave, St Paul, MN 55105, USA. fox@macalester.edu.

Introduction

The increasing severity of water pollution is leading to a number of concerns that impact the environment, society, and climate change. Aquatic species' entanglement and ingestion of marine debris, which threaten the health of these animals and their ability to acquire food, pose one of the most prominent environmental impacts. Furthermore, plastic waste frequently fragments into microplastics instead of biodegrading, making it exceedingly difficult to clear it from the environment. Not only does this affect the diet of aquatic species, but researchers have also found that these plastics and the chemicals they contain might be present in certain species of fish that people consume. Additionally, aquatic waste leads to negative social impacts; for example, it can harm communities through decreasing fish populations that communities may depend on to meet a variety of needs (1). Considering the alarming effects of marine debris, special attention should be placed on the means with which waste finds its way into oceans and seas. For plastics, this main channel is by rivers. While scientists previously believed that a few select rivers brought 90 percent of river-borne plastic to oceans, new research suggests that over 1,000 rivers carry 80 percent of plastic marine debris (2). These recent findings emphasize the need for an approach that combats water pollution in a way that is both effective and financially-attainable to enable widespread implementation, *viz.* a computer vision approach. The computer vision model that we describe in our paper is very feasible economically, since it does not require expensive hardware to execute. This allows it to be broadly applied to monitor a wide range of rivers, rather than a select few, which addresses the previously-mentioned concern of concentrated impact. While there are technologies that offer a more advanced and automated approach to waste removal, such as the Interceptor (3) and The Bubble Barrier (4), their cost and accessibility is a significant impediment to their widespread usage. Through combining existing resources, i.e. previously-established cleanup groups, with an automated approach for monitoring, we provide a method of trash detection that bypasses the high costs of automated trash removal and only focuses on monitoring. While our approach would involve the cost of obtaining cameras that can run our model and connect to a cloud storage platform that can be easily accessed from the front end of trash cleanup groups, as well as the cost of installing them around rivers and floodlights, these costs would certainly be less than the approximate $380,000 that a single Bubble Barrier costs (4). In the same way that the Bubble Barrier is strategically placed in areas such that trash flowing from different river channels join together, we propose that the cameras should be placed near channel intersections in order to minimize the amount of cameras needed and associated costs. This way, cameras do not have to maintain surveillance over all sections of rivers, but rather focus on points that all channels lead to. Different from the Bubble Barrier, which would only be placed in one single location, these cameras could be placed at multiple intersections to widen the impact.

In our paper, we develop a model that uses a novel convolutional neural network, YOLOv5, to detect litter in rivers. To do this, we first annotate the data by localizing the object in the image by creating bounding boxes and then labeling the segmented areas. We then implement the object detection architecture,

YOLOv5, to enable our model to find important features of each trash type, so that it can detect these distinct components in new images at a relatively high speed. We also use transfer learning to save time during the data training process. By developing a model that performs real-time detection of trash in littered rivers, we aim to prevent trash from flowing into larger bodies of water, where the waste would become progressively harder to locate and remove. In application, our model would send the aforementioned information as well as location to any water pollution or community waste-management groups that are focused in that area. By sending this information to them, their otherwise random cleanup searches are anticipated to become more targeted in terms of location, which will enhance the efficiency of their efforts and limit the amount of people and resources that would otherwise be needed. As a whole, these saved resources can be used to spread cleanup efforts to a broader range of rivers. Additionally, along with the Trash Annotations in Context (TACO) dataset (5), we found and collected our own data from polluted rivers so that we may create our own dataset that is used to train, validate, and test our model and offer new data to the research community. As a whole, we experimentally determine the set of hyperparameters and conditions that yield the best performance in our model. Despite the limited size of our combined dataset, our model is able to perform similarly to existing trash detection networks.

Related Work

Recent work by Tharani et al. (6) introduced a new category of visual trash detection that focused on waste floating on top of canals. The primary purpose of the model was to detect smaller objects that are often overlooked by other models, but other trash is categorized as either medium or large. Tharani et al. compared commonly-used deep learning object detection models, such as SSD, YOLOv3-Tiny, YOLOv3, and RetinaNet, with models that have an added attention layer to evaluate which models have the best performance in a given set of circumstances. They found that their model, YOLOv3 combined with a log-based attention layer that focuses on smaller objects, had an Average Precision (AP) that surpassed those of other models on the easy test set. On the hard test set, their model performed relatively similar to others in terms of AP. Another indicator of performance used in the experiment was Intersection over Union (IoU), which represents the amount of overlap between two ground truth and predicted bounding boxes. The models all had a similar IoU.

Considering the results of this paper and the extent to which YOLOv3 coupled with an attention layer outperformed other popular object detection models, we focus particularly on a YOLO architecture in our paper. However, we use a more recent version, YOLOv5, in our model because of its enhanced accuracy and runtime compared to previous versions. Also, rather than dividing the detected trash into classes, our model performs binary detection.

In another paper, Conley et al. (7) compared three models—Mask R-CNN, SOLO, and YOLOv6—to determine which one would most effectively quantify the trash available for transport into storm drain systems. With this goal in mind, the models were trained to detect the presence of trash rather than to classify the trash by object type. The outputs of the trained models were used to calculate the ratio of trash

to non-trash pixels in the image. Variations in the distance of the objects from the camera were taken into account. Because the Mask R-CNN model performed best in terms of recall, precision, and accuracy, it was examined further through the use of linear regression analysis. This was used to determine how effectively the trash pixel ratios the model produced represented the real, measured volumes of trash.

While these models focused on quantifying the trash featured in the images, our model performs binary detection and does not quantify or classify the waste further. In addition, this paper did not place much weight on the remarkably higher amount of frames per second that YOLO can process images for training as opposed to the other models tested. We use the YOLO architecture to take advantage of its greater training speed.

In another paper, Panwar et al. (8) used AquaVision, a state-of-the-art model, to detect and classify underwater trash with the help of RetinaNet to train the model and Resnet50 and FPN as backbone models. The objects in the images were classified by material: glass, metal, paper, and plastic. They found that the single-stage object detection approach that AquaVision proposed proved to be more accurate than two-stage methods such as Faster R-CNN. Additionally, despite the fact that the model was only trained for non-aquatic images, it was nevertheless able to effectively detect and classify objects underwater. It was also trained on very few images but still produced satisfactory results when random images were passed through.

This paper emphasized the increased efficiency and accuracy of single-stage object detectors compared to the two-stage approach. In light of this, we employ the single-stage YOLO architecture to make use of its advantages. In addition, given their model's ability to perform in different environments, we use a combination of aquatic and land trash in our dataset.

Recent work by Srivastava et al. (9) aimed to detect trash on roadways and determine whether it would constitute a driving hazard. After going into a breakdown of the components that make up the YOLO architecture, they used transfer learning to retrain both a YOLO model and an M-RCNN model to improve upon their accuracy and precision. They then eliminated classes that had exceedingly low annotations associated with them to increase mAP. Next, they fine-tuned the object detectors for parent categories and mapped training labels of what the object was (e.g. cigarette, bottle) to either "drivable" or "non-drivable." In addition, four versions of the YOLOv5 architecture were compared: small, medium, large, and extra-large. They found that as the YOLO architecture increased in complexity, the loss and precision increased as well.

Similar to how Srivastava et al. classified the trash by object type, we experimented with classification by material as well as binary detection. We also implemented transfer learning to yield results in a shortened amount of time similar to Srivastava et al.

Background and Materials

General

To evaluate how well our model performed during training and validation, we used a number of performance metrics. Each of them re-

late to the accuracy of the model and how well it is able to find objects in images. Because the relative importance of each metric is subjective to the context of the problem, analyzing the performance of our model as a whole is more effective than considering each metric in complete isolation from each other.

Metrics

1. Intersection over Union (IoU) - IoU is frequently used in object detection to measure how well the model found the object. It is calculated by dividing the area of overlap of the two bounding boxes (the ground truth box and the predicted box) by the total area of both bounding boxes combined (10). In our paper, we did not analyze IoU independently but rather factored it into another metric which we did analyze, *viz.* the mean Average Precision (mAP).

2. Precision - Precision measures the amount of predictions the model made that are correct out of the total amount of predictions made (10).
Precision = TP / (TP + FP)
where, TP = true positives (accurately predicted the presence of an object) and FP = false positives (inaccurately predicted the presence of an object)

3. Recall - Recall, also known as sensitivity or true positive rate (TPR), measures how well the model finds objects present in the images. It evaluates the amount of objects the model accurately predicted out of all the objects present in the image (10).
Recall = TP / (TP + FN)
where, TP = true positives (accurately predicted the presence of an object) and FN = false negatives (object is present but no prediction is made)

4. Specificity - Specificity, also known as the true negative rate (TNR), measures how well the model is able to find true negatives.
Specificity = TN / (TN + FP)
where, TN = true negatives (no prediction is made and no object is present) and FP = false positives (inaccurately predicted the presence of an object)

5. mean Average Precision (mAP) - mAP is a commonly used metric to evaluate object detection models, since it incorporates elements of both precision and recall. It is found by first obtaining the Average Precision (AP) for each class, which is done by finding the precision at different recall values and averaging all of them together (11). These values are then averaged over all of the classes to obtain the mAP (12). Because our model performed binary detection and did not consist of multiple classes, the mAP and the AP had similar values. Additionally, IoU thresholds are often used to restrain the values used to calculate mAP. For example, mAP_0.5 indicates that all predicted bounding boxes with an IoU below 0.5 are suppressed, and only the values above this threshold are returned. This is done to ensure that invalid predictions are not counted towards the mAP.

6. Loss - There are two different kinds of loss we implemented in our paper: box and objectness. They were both calculated separately for our training and validation sets. Box loss measures how well the model is able to find the center of an object and how well the predicted bounding box covers the object. Objectness loss represents the probability of an object ex-

isting in the predicted bounding box (13). As a whole, low loss values represent desirable model performance. They indicate that the model makes few errors and that they are small, rather than large (14). Subtracting the loss from 1 returns the overall accuracy of the model.

7. F1 Score - The F1 Score measures the model's accuracy using a combination of precision and recall scores.
F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

Machine Learning Language
1. CNN - A convolutional neural network (CNN) is a type of network architecture in deep learning. It first takes in an input image, then determines the importance of objects and other image aspects based on weights, and finally differentiates all the image components from one another (15). It predicts labels, bounding boxes, and confidence probabilities for the objects in the image (16).

2. YOLOv5 - YOLO stands for You Only Look Once, referring to the model's ability to make predictions on an image from only one pass through. We used version 5s. It was developed by Ultralytics, a platform that makes machine learning models available to developers. YOLOv5 is a novel CNN, widely known for its remarkable speed. Furthermore, YOLO is a one-stage detector, meaning that in addition to its high inference speed, it predicts bounding boxes in only one step. It does not go through the process of proposing any candidate bounding boxes or regions. Using a grid box and anchors, the model is able to detect objects with just a single pass through (as mentioned above), as opposed to other models that may require many (17). Not only is YOLOv5 a relatively recent development, but it is also well-established and has a number of resources associated with it. In addition, it does not require extensive computational resources, and it has a relatively high frames per second (FPS) rate compared to other models, so it trains remarkably faster which is useful when comparing the effectiveness of different training approaches. These factors contributed to our determination of YOLOv5 being the best model for our purpose of trash detection.

Description of YOLOv5
YOLO consists of three main parts: the backbone, neck, and head (18).

The model backbone is primarily used for key feature extraction. This specific version of YOLO uses the Cross Stage Partial Dark Network (CSPDarknet) as a backbone, which is used in place of other approaches that require extensive computation resources in order to function. Moreover, CSPDarknet significantly improves processing time with deeper networks. Within the backbone, there are multiple blocks: the Focus layer, BottleNeckCSP and Spatial Pyramid Pooling (SPP). The primary purpose of the Focus layer is to reduce layers and parameters, while increasing forward and backward speed (19). BottleNeckCSP is used to reduce parameters and matrix multiplications. SPP is a pooling strategy that eliminates the need for an image of a fixed size to be inputted into the CNN, which may otherwise reduce recognition accuracy within images. SPP allows the computation of feature maps to occur only once for the image as a whole. Once this is done, features are pooled in a certain number of cells (20).

The model neck creates feature pyramids, which aid in the generalization and identification of trash objects of various sizes and scales. As a whole, they improve the performance of the model on unseen data. YOLOv5 uses the Path Aggregation Network (PANet) for feature aggregation (18). The image features extracted from the backbone are input into the neck, which fuses them to make the semantic, i.e. meaningful, information richer. The UpSample component rescales small data up, which is needed in order to combine data that has been pooled with the larger data. The highlighted features are then passed on to the head to predict the input features (21).

The model head performs the final task of detection. It predicts the features input from the neck and applies anchor boxes on them to create vectors (18). The vectors store the following information: the probability that the bounding box contains an object, the x and y coordinates of the box's center, the height of the box expressed as a percent of the cell's height, and the conditional probabilities that describe how likely the object in the cell is to belong to each class, given there is an object present in the box. If there is more than one bounding box in a cell, then the above information is added on to the same vector as it pertains to the new bounding box. The only pieces of information that are not repeated are the conditional probabilities of the cell containing objects for each class, since this pertains to the cell itself rather than to individual boxes (16). YOLOv5 uses a YOLO layer as the head, which outputs the results of detection, such as class, score, location, and size (22).

Methods

First, this section describes the equipment used to run the experiments, and it then describes the datasets we used and how we acquired them. We proceed to describe what the model does and the experimental approaches we took to determine the hyperparameters and conditions that yielded the best model performance. Finally, the section displays a side-by-side comparison of a sample set of ground truth data and the predictions our model made on those images.

Environmental Setup

We ran our experiments using Python 3.10 on a 2022 MacBook Pro (13-inch) with a 16 GB LPDDR5 Memory and macOS Monterey 12.4 installed. We ran the python code using Google Colaboratory, which is a Jupyter Notebook service, and we used TensorBoard in conjunction with Weights & Biases to visualize our results.

Datasets

There are a limited number of open-source datasets that primarily contain images of waste on surfaces of water. To overcome this, we used a combination of datasets: one that we gathered ourselves, and the TACO dataset found online. Since the TACO dataset is composed of images of trash in a number of natural environments (and not only rivers), it is beneficial for our model to have both datasets to train on so that it is exposed to images in the target environment.

Our Dataset

Figure 1. Sample images we collected for our dataset (see supplementary file for a full set of images)

To obtain a dataset with the desired conditions, we selected a densely littered river in the area and photographed the individual pieces of trash to create our own dataset of (originally) around 1,800 trash objects annotated among 400 images. There are numerous varieties in the dataset, such as the frequency of trash in a single image and its rotation in the water, but there are also some deliberate consistencies. For example, the overhead angle at which the images were captured generally remains constant; this is to simulate the placement of overhead cameras that the trash would be monitored with in actual implementation. Furthermore, there are a number of challenges present in the images that aid in our model's ability to learn based on realistic conditions. For example, vegetation most frequently creates reflections in the water, but other objects such as trash on the water surface also create reflections that can confuse the model. The trash objects used in the dataset do not follow one consistent shape, structure, or size. This may be because damaged trash items were used (e.g. crushed bottles and ripped pieces of cardboard) or the shapes of some objects changed over the duration of our data collection. For example, the shapes of plastic bags were constantly changing. Furthermore, some items were partially submerged in water or caught in rocks, causing irregular shapes to protrude from the surface.

Including these challenges in the dataset is crucial for our model to be able to perform well in a real world application. Controlled environments provide unrealistic conditions and therefore cannot be indicative of true model performance.

TACO

TACO (5) is an open-source dataset that includes images of trash in natural environments. It contains 1,500 images with polygonal segmentation annotations, among other data. These segmentations contour the objects of interest, making the annotations

more precise than those of other techniques. However, because complex annotation techniques require high memory storage, which in turn requires increased computing resources, they can be costly to implement (23). For these reasons, we only used the images provided by TACO, rather than their annotations as well, and manually annotated them using a less computationally- intensive technique.

Figure 2. Sample images from TACO dataset (5).

Annotation

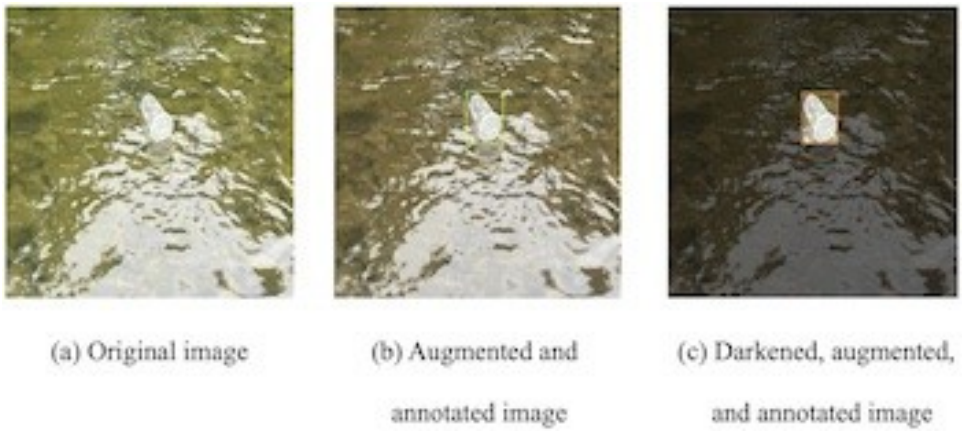

Figure 3. Transformation of a sample image from our dataset after augmentation and annotation using Roboflow (24). Images (b) and (c) are augmented with the following: 12° hue, 20% saturation, and 4% brightness. Image (c) is a darkened version of Image (b) for foreground and annotation visibility. Both bounding boxes in Images (b) and (c) are labeled with the "trash" class.

Both our own dataset and that of TACO, originally totaling around 1,900 images combined, were manually annotated using Roboflow (24): a computer vision developer platform for data collection, preprocessing, and model training techniques (11). In Roboflow, we used bounding boxes to segment the waste, and did not include any reflections of trash in the boxes, nor were they separately segmented. The objects were all categorized under "trash." After annotating, we performed a series of augmentations on the combined dataset of TACO and the images we gathered using Roboflow. The augmentations included changes in rotation, hue, saturation, and brightness. Our overall dataset grew from around 1,900 images to 4,100 through augmentation.

Model description

First, our model decomposed the input image into a grid of S by S cells (where S is typically 19). Each cell then predicted a certain number, B, of bounding boxes, where the center of the bounding box fell within the cell, even if the box extended into other cells. For each cell, vector y was then created, which stored all of the information regarding the cell's bounding boxes and probability of the cell containing trash. Any bounding boxes that yielded confidence probabilities above a given threshold were returned, and any boxes that did not meet this criteria were suppressed. The result was a multidimensional matrix of size SxSx(5B+C), where C represented the conditional probabilities of the cell containing trash, and 5 was the number of variables that directly related to each bounding box in the cell (16). Finally, our model output predicted bounding boxes labeled with the class (which is always "trash" in our case) and the confidence probability.

(a) Ground truth data (b) Model predictions

Figure 4. Sample images from validation set.

Trials

To determine the best conditions for our model to be trained in, we experimented with training it a number of ways and compared the results they produced. The control model was run under the following conditions and hyperparameters: trained on both raw and augmented data, had a batch size of 16, used preloaded weights from the Microsoft (MS) Common Objects in Context (COCO) dataset (25), and performed binary detection. In each trial of the experiment, one condition was changed in order to examine the significance of each variable relative to its impact on model performance.

Augmented Data

To determine whether generating augmented data improved the model mAP, we executed our model on a dataset of both raw and augmented data ("Control" model) and on entirely raw data ("No augmentations" model). We found that generating three images per training example significantly improved mAP. The best mAP produced by the "No augmentations" model was 0.403, and the best mAP produced by the "Control" model was 0.503. Additionally, the control model reached its maximum mAP in significantly fewer steps than the "No augmentations" model did. The graph characteristics showed that that running the "No augmentations" model for more steps/epochs would likely result in higher mAP, but it was still less efficient than running on augmented data.

Batch Size

Batch size determines the amount of images that are propagated through the model. We tried running our model on three different batch sizes: 8 ("Batch size 8" model), 16 ("Control model"), and 32 ("Batch size 32" model). We observed a non significant trend between batch size and mAP: the greater the batch size, the greater the mAP. This was unusual; because weights are updated after each propagation, networks usually run faster on smaller batch sizes. In our case, the opposite occurred. However, as Devansh points out (25), it is important to not read too far into the effect of batch size when it has such a small impact on mAP. In future work, we intend to further experiment with this hyperparameter by testing our model on more extreme differences in batch size.

Transfer Learning

We used transfer learning by running our model on pre-trained weights from the MS COCO dataset, and compared its performance to that of our model without any preloaded weights. To do this, we used the overall mAP with an IoU threshold of 0.5 and above. Using transfer learning, our model reached greater mAP values in significantly fewer steps, which was beneficial in reducing the training time. However, the best mAP values produced by the different methods of training were almost identical: 0.496 without weights and 0.503 with the COCO weights. Since the main purposes of transfer learning were to save resources and increase efficiency, rather than to produce better accuracy, it was evidently effective in this case and was preferred over training our model without preloaded weights.

Multi-Class Classification

To test our model's ability to both detect and classify trash by material, we ran our model on the same dataset as the control model but with the following annotations: plastic, paper, and other. Keeping in mind the limited size of the dataset and the model's need for plentiful instances of each class, we expected performance to degrade significantly. This was indeed reflected in our results: the highest mAP of the control model, which performed binary detection, was 0.503. The highest mAP achieved by the "Multi-class classification" model was 0.181.

Image Size

Across all trials, we consistently presented images of size 416 x 416 pixels in order to balance speed and accuracy. Using larger images would result in a slower processing speed during training, while smaller images gradually compromised performance, hence we selected

this size as a balanced medium. Also, because we did not foresee the use of high-resolution cameras in actual implementation, we deliberately chose this relatively small size.

Model Configuration

We implemented a custom yolov5s configuration developed by Roboflow (24) for all of our initial trials because of the relatively small size of our dataset. However, upon review, we found that using the yolov5x configuration significantly improved our results. In light of this, we conducted an additional trial maintaining the hyperparamter selection of our control model and only changed the configuration from the custom yolov5s to yolov5x.

Figure 5. Performance of model variations measured by mAP_0.5 with different hyperparameters/conditions. Note: model variations were run for 50 or 100 epochs depending on the graph trend and the likelihood to change based on their nature (common or distinctive).

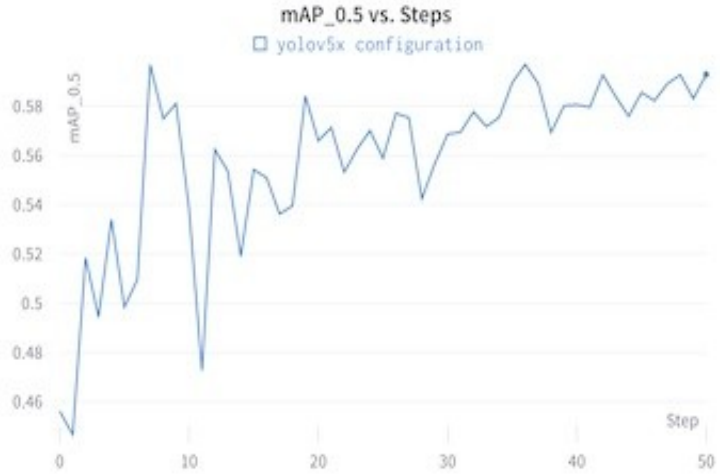

Figure 6. Performance of revised model maintaining hyperparameter selection from the control model and only changing the configuration to yolov5x.

Loss

From the loss curves in Figures 7a and 7b, it was evident that our model was making accurate generalizations of the data and was not overfitting or memorizing the datasets. The training box losses of our model variations plateaued between 0.027 and 0.037, and the validation box losses plateaued between 0.051 and 0.055, which were similar in range. This indicated that during both training and validation, our model was able to find the center of the pieces of trash and predict their bounding box well. Similarly, the objectness losses of the training and validation sets were very close: 0.015 to 0.018 (training) and 0.013 to 0.016 (validation). This demonstrated that there was a very high probability of waste existing in the predicted bounding boxes.

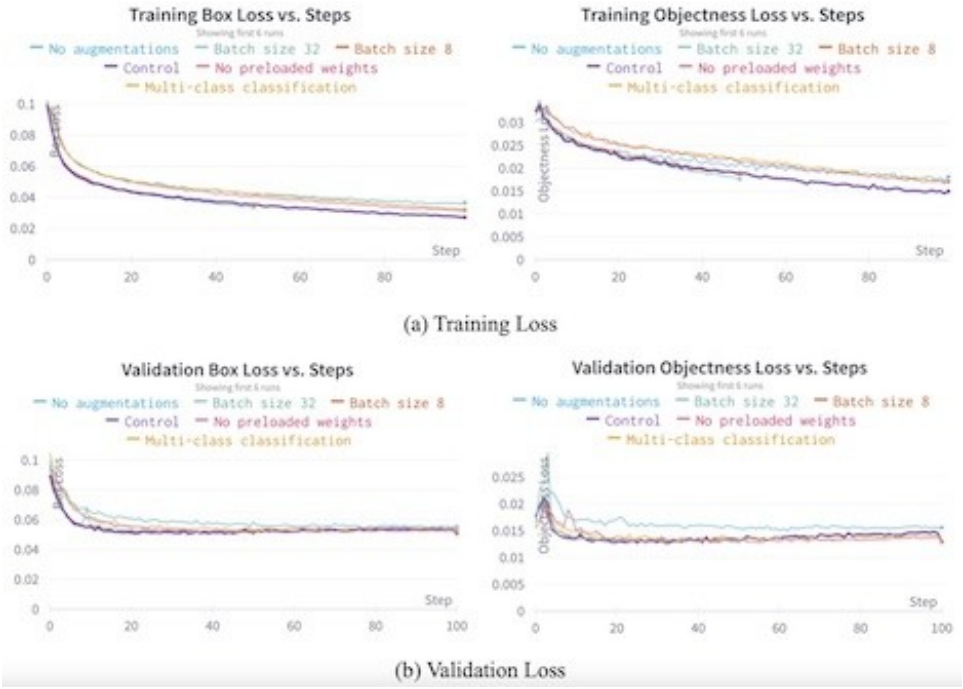

Figure 7. Learning curves during the training and validation stages. On the left are box loss curves, and on the right are objectness loss curves.

Table 1: Highest metric values achieved by model under different conditions.

Model	MAP_0.5	Precision	Recall
No augmentations	0.403	0.582	0.390
Batch size 32	0.499	0.712	0.461
Batch size 8	0.485	0.680	0.436
Control	0.503	0.672	0.470
No preloaded weights	0.496	0.624	0.491
Multi-class classification	0.496	0.327	0.335
Yolov5x	0.597	0.705	0.532

Results and Discussion

As seen from the learning curves in Figure 7, it is evident from the high similarity and overlap between the training and validation accuracies, that each of our model variations did not overfit or memorize the data. Overall, taking into consideration mAP_0.5, precision, recall, and accuracy/loss, we deduced that our control model performed the best out of all of initial trials. Since our control model was executed under the conditions and hyperparameters that we hypothesized to be the most effective, our prediction of the ideal batch size, inclusion of weights, and mode of detection (binary or classification) remained true. However, the model that used the yolov5x configuration performed better than all the others, including the control model, which was an unexpected outcome.

Table 1 shows that, out of our initial trials, our control model obtained the highest mAP_0.5 of 0.503, the third highest precision of 0.672, and the second highest recall of 0.470. In terms of precision, our control model under performed only those with different batch sizes. In terms of recall, our control model performed less accurately than the model without preloaded weights. However, we placed more importance on the mAP scores because they combined the precision at different recall values, thereby providing a more accurate summary of the performance as opposed to obtaining only the best precision or recall value in isolation. Furthermore, looking at Figure 7, our control model also demonstrated the lowest/best loss and highest accuracy out of the initial trials. It presented with the lowest training box loss (0.027), the lowest training objectness loss (0.015), the lowest validation box loss (0.051), and tied for the lowest validation objectness loss, when compared with the model run without preloaded weights (0.013). Therefore, when considering all of the metrics as a whole, our control model was the most accurate at detecting trash out of our first experiments. Our revised model with the yolov5x configuration, however, outperformed our control; with an map, 0.5 of 0.597, precision of 0.705, and recall of 0.532, it was more effective than the other models we experimented with. This figure could have been further improved had we used polygonal annotations rather than bounding box annotations, as described in a number of sources. Based on data provided by Roboflow, in which a similar YOLO model was executed with similar means of augmentation, we deduce that polygonal annotations significantly improve performance. The Roboflow model had an mAP_0.5 of 0.772 when presented with bounding box annotations, and had an mAP_0.5 of 0.851 when presented with polygonal annotations (27**)**. This is an increase of 10.23%. Based on this data, applying the 10.23% increase, our model would increase from an mAP_0.5 of 0.597 to 0.658 with polygonal annotations. Other papers such as *Efficient Interactive Annotation of Segmentation Datasets with Polygon-RNN++* also support this notion with data (28). In addition, had we presented images of a larger size for training, we anticipate that our model would have performed even better (29). While we did not implement polygonal annotations or larger images in these experiments in an effort to maintain a high processing speed and relatively low consumption of computing resources, if adopted into the model, it is anticipated that they would significantly improve model performance.

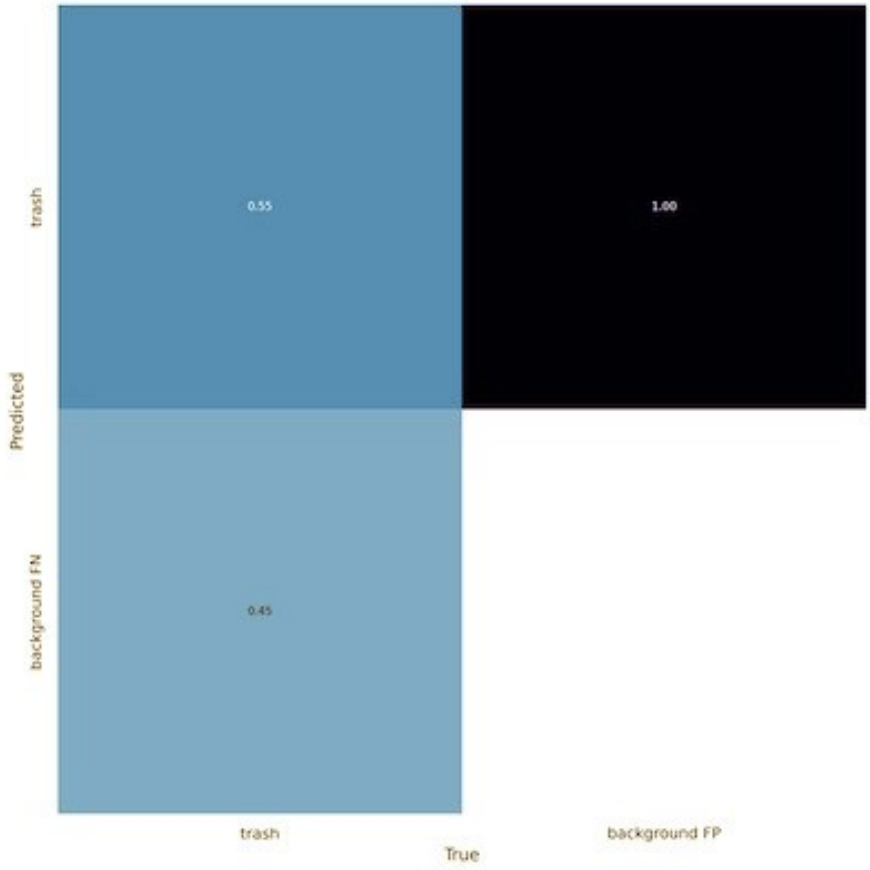

Figure 8. Confusion matrix of model with yolov5x configuration, which had the best performance. True Negative (TN), does not apply. It would represent a corrected misdetection. In the object detection task there are many possible bounding boxes that should not be detected within an image. Thus, TN would be all possible bounding boxes that were correctly not detected (many possible boxes within an image).

Figure 8 shows that the true positive rate, or sensitivity, was 0.55, and the false negative rate was 0.45. The true negative rate (or specificity) does not apply in the context of object detection tasks however, since we were concerned with the location of objects and not the background. Low specificities often result from duplicate bounding boxes that correctly segment a single ground truth object, but because there can only be one accurate predicted bounding box for every ground truth bounding box, only one prediction is considered a true positive and the rest are misleadingly considered false positives (30, 31).

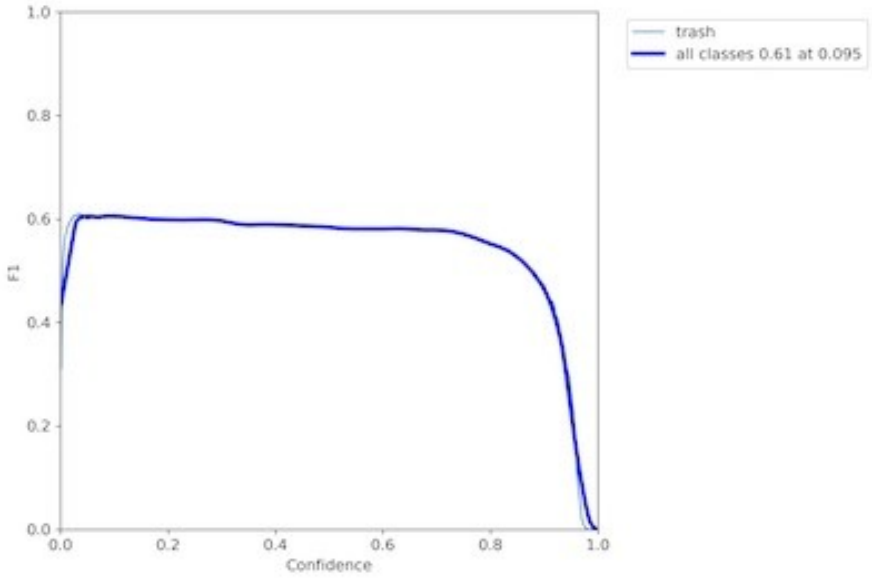

Figure 9. F1 score curve of model with yolov5x configuration, which had the best performance.

The F1 score curve displayed in Figure 9 allowed us to visualize how well our model balanced precision and recall. Based on the model's ability to maintain a relatively consistent F1 score across different confidence values represented by the shape of the curve, as well as the highest F1 score reached of 0.61, our model performed relatively well with balancing precision and recall.

Future Work

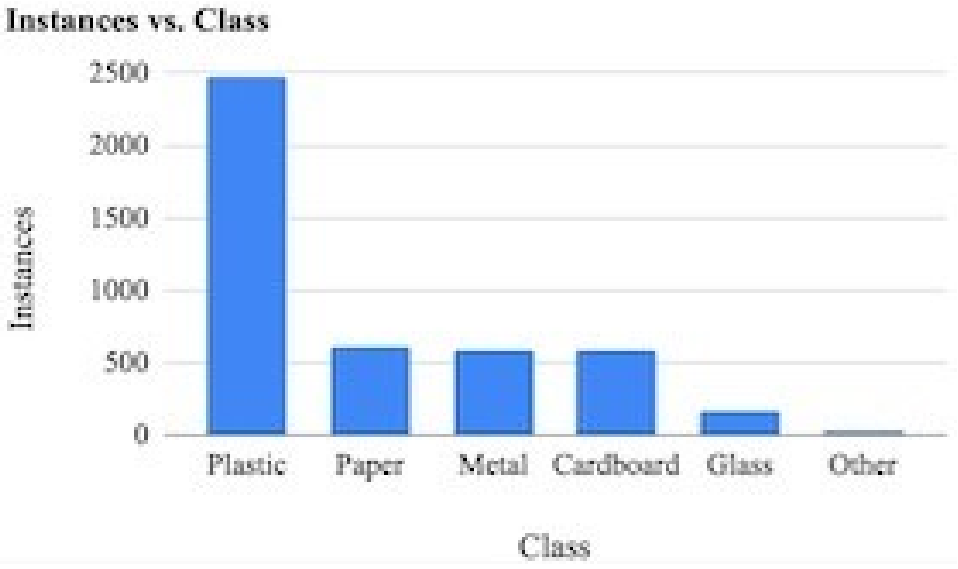

Figure 10. Distribution of object instances by class.

To better address the issue of trash detection by enabling our model to classify trash by material, we should collect a larger dataset in the future. Not only will our model have a high frequency of images to train on, but the dispersion of class instances (i.e. the total number of instances of objects of each class present in the images) will also be more evenly distributed. In our current dataset, there is an extreme imbalance between the instances of the "plastic" class and the other classes. The "plastic" class is over represented, and the "glass" and "other" classes are under represented. The lack of evenly represented classes largely contributes to our model's inability to detect trash by material type.

Furthermore, based on our model's enhanced performance on the augmented dataset compared to the raw dataset, we propose that generating more images per training example will further increase performance without overfitting. Including more types of augmentation, such as rotation and exposure, will likely yield favorable results as well.

Lastly, as previously mentioned, presenting polygonal annotations and larger images to the model could potentially further improve our results significantly.

Conclusion

In this paper, we applied a machine learning approach to automate the detection of waste in rivers. We presented a model of the one-stage object detection architecture YOLOv5 that combines effective hyperparameters for accurate and efficient trash detection. Our model performance was similar to that of other trash detection networks that use much more extensive datasets compared to ours, which is composed of 1,900 original images and 2,200 augmented images. We assembled an original dataset, which includes 400 manually-obtained images and almost 500 augmented images of littered rivers. Furthermore, we assembled a new version of the TACO dataset (5) that is annotated using bounding boxes rather than the original polygonal segmentations. The latter is a more computationally-intensive technique. We have two versions of this combined dataset: one with the waste categorized by material, and one with it all categorized under one class ("trash"). Overall, although it is a challenging task, automating the detection of waste in rivers will help combat water pollution and will allow resources to be saved and allocated more effectively.

References

1. US EPA, OW. *Learn About Aquatic Trash*. 19 Nov. 2021, https://www.epa.gov/trash-free-waters/learn-about-aquatic-trash.

2. Parker, Laura. "Plastic Gets to the Oceans through over 1,000 Rivers." *Environment*, National Geographic, 30 Apr. 2021, https://www.nationalgeographic.com/environment/article/plastic-gets-to-oceans-through-over-1000-rivers.

3. "Rivers • The Interceptor • The Ocean Cleanup." *The Ocean Cleanup*, https://theoceancleanup.com/rivers/.

4. “Amsterdam.” *The Great Bubble Barrier*®, https://thegreatbubblebarrier.com/amsterdam/.

5. Proença, Pedro F., and Pedro Simões. *TACO: Trash Annotations in Context for Litter Detection*. 2020. https://doi.org/10.48550/ARXIV.2003.06975.

6. Tharani, Mohbat, et al. *Attention Neural Network for Trash Detection on Water Channels*. 2020. https://doi.org/10.48550/ARXIV.2007.04639.

7. Conley, Gary, et al. “Using a Deep Learning Model to Quantify Trash Accumulation for Cleaner Urban Stormwater.” *Computers, Environment and Urban Systems*, vol. 93, Apr. 2022, p. 101752. https://doi.org/10.1016/j.compenvurbsys.2021.101752.

8. Panwar, Harsh, et al. “AquaVision: Automating the Detection of Waste in Water Bodies Using Deep Transfer Learning.” *Case Studies in Chemical and Environmental Engineering*, vol. 2, Sept. 2020, p. 100026. https://doi.org/10.1016/j.cscee.2020.100026.

9. Srivastava, Ishan. *Retraining of Object Detectors to Become Suitable for Trash Detection in the Context of Autonomous Driving*. Dresden University of Technology , Mar. 2022, https://www.researchgate.net/profile/Ishan-Srivastava-8/publication/360688760_Object_detection_in_self_driving_cars_using_YOLOv5/links/6285697c50c4566fc2744ac0/Object-detection-in-self-driving-cars-using-YOLOv5.pdf.

10. Yohanandan, Shivy. “MAP (Mean Average Precision) Might Confuse You!” *Medium*, 9 June 2020, https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2.

11. Bhattacharyya, Jayita. “Step by Step Guide To Object Detection Using Roboflow.” *Analytics India Magazine*, 11 Oct. 2020, https://analyticsindiamag.com/step-by-step-guide-to-object-detection-using-roboflow/.

12. *Mean Average Precision (MAP) Explained: Everything You Need to Know*. https://www.v7labs.com/blog/mean-average-precision.

13. Kasper-Eulaers, Margrit, et al. “Short Communication: Detecting Heavy Goods Vehicles in Rest Areas in Winter Conditions Using YOLOv5.” *Algorithms*, vol. 14, no. 4, Mar. 2021, p. 114. https://doi.org/10.3390/a14040114.

14. Seif, George. “Understanding the 3 Most Common Loss Functions for Machine Learning Regression.” *Medium*, 11 Feb. 2022, https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3.

15. Saha, Sumit. “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 Way.” *Medium*, 17 Dec. 2018, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

16. Wakamiya, Chad. *Object Detection with YOLO*. https://datax.berkeley.edu/wp-content/uploads/2020/07/Object-Detection-with-YOLO-1-1.pdf.

17. Lohia, Aditya, et al. “Bibliometric Analysis of One-Stage and Two-Stage Object Detection.” *Library Philosophy and Practice (e-Journal)*, Feb. 2021, https://digitalcommons.unl.edu/libphilprac/4910.

18. *YOLO V5 — Explained and Demystified – Towards AI*. https://towardsai.net/p/computer-vision/yolo-v5%e2%80%8a-%e2%80%8aexplained-and-demystified.

19. Jocher, Glenn. “YOLOv5 Focus() Layer · Discussion #3181 · Ultralytics/Yolov5.” *GitHub*, https://github.com/ultralytics/yolov5/discussions/3181.

20. He, Kaiming, et al. *Deep Residual Learning for Image Recognition*. 2015. https://doi.org/10.48550/ARXIV.1512.03385.

21. Guo, Zexuan, et al. “MSFT-YOLO: Improved YOLOv5 Based on Transformer for Detecting Defects of Steel Surface.” *Sensors*, vol. 22, no. 9, Jan. 2022, p. 3467. https://doi.org/10.3390/s22093467.

22. Xu, Renjie, et al. “A Forest Fire Detection System Based on Ensemble Learning.” *Forests*, vol. 12, no. 2, Feb. 2021, p. 217. https://doi.org/10.3390/f12020217.

23. Asadi, Khashayar, et al. *Real-Time Scene Segmentation Using a Light Deep Neural Network Architecture for Autonomous Robot Navigation on Construction Sites*. https://arxiv.org/pdf/1901.08630.pdf.

24. *Roboflow: Give Your Software the Power to See Objects in Images and Video*. https://roboflow.com/.

25. Lin, Tsung-Yi, et al. *Microsoft COCO: Common Objects in Context*. 2014. https://doi.org/10.48550/ARXIV.1405.0312.

26. Devansh. “How Does Batch Size Impact Your Model Learning.” *Geek Culture*, 5 Feb. 2022, https://medium.com/geekculture/how-does-batch-size-impact-your-model-learning-2dd34d9fb1fa.

27. Jul 19, Arty Ariuntuya, and 2023 5 Min Read. “Improve Accuracy: Polygon Annotations for Object Detection.” *Roboflow Blog*, 19 July 2023, https://blog.roboflow.com/polygon-vs-bounding-box-computer-vision-annotation/.

28. Acuna, David, et al. Efficient Interactive Annotation of Segmentation Datasets with Polygon-RNN++. 2018. https://doi.org/10.48550/ARXIV.1803.09693.

29. Hirahara, Daisuke, et al. “Effects of Data Count and Image Scaling on Deep Learning Training.” *PeerJ Computer Science*, vol. 6, Nov. 2020, p. e312. https://doi.org/10.7717/peerj-cs.312.

30. Geng, Jamie. “How to Evaluate an Object Detection Model: Explain IoU, Precision, Recall and mAP with Examples.” *Medium*, 13 Dec. 2022, https://zihaogeng.medium.com/how-to-evaluate-an-object-detection-model-iou-precision-recall-and-map-f7cc12e0dcf6.

31. *Evaluation Metrics*. 19 June 2023, https://neuralception.com/objectdetection-evaluationmetrics/.