APPLYING OBJECT DETECTION TO MONITORING MARINE DEBRIS

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAI‘I AT HILO IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

TROPICAL CONSERVATION BIOLOGY AND ENVIRONMENTAL SCIENCE

MAY 2020

By

Leah Sherwood

Thesis Committee:

Michael Peterson, Chairperson
Steven Colbert
Megan Lamson

Keywords: marine debris, object detection, plastic pollution, beach surveys, deep learning, Darknet, YOLO
Acknowledgements

Thank you to my advisor Michael Peterson for his willingness to take me on as a graduate student, and for developing my confidence in myself to lead my own research and be successful. Thank you to my committee members Steven Colbert and Megan Lamson for providing me with support and guidance during my graduate career.

Thank you to my team of UH Hilo computer science undergraduate students Briana Noll, Charnalyn Crivello and Stacey Yanagihara for their hard work and determination in migrating Tiny YOLO onto a smartphone app. Thank you especially to Briana for all of her valuable computer science and engineering expertise.

Thank you to the Hawai‘i Wildlife Fund, who provided access to Kamilo Point, photographs of marine debris, and advice and guidance on marine debris issues.

Thank you to Francis Cristobal, Keith Edwards and Travis Mandal of the UH Hilo Department of Computer Science and Engineering for welcoming me into the department, and providing me with resources and advice when I needed them.

Thank you to the University of Hawai‘i Information Technology Services – Cyberinfrastructure for providing me with technical support and advanced computing resources.

Thank you to the Awesome Foundation, O‘ahu chapter for awarding me a scholarship to purchase the Samsung S10e.

Finally, I would like to thank my husband for his friendship, love and support.
Abstract

Most of what is known about the type and distribution of plastic marine debris has been learned from beach surveys conducted by hundreds of researchers and volunteers since the 1980s. However, beach surveys of plastic marine debris require significant manual labor and lack harmonization across survey sites. In this thesis, I demonstrate how object detection technology based on deep learning can be deployed to partially automate the manual labor required to conduct beach surveys and upload the survey results to a centralized marine debris database. To create a proof-of-concept implementation, I developed an object detection system for marine debris using Darknet, an open source framework for convolutional neural networks, and the detection algorithm YOLOv3. I trained the detector on nine object classes: bags, bottlecaps, bottles, buoys, containers, hagfish traps, nets, oyster spacers, and “other,” and achieved a mean average precision (the standard metric of accuracy in the object detection literature) of 52%. The best performing class was hagfish trap, with an average precision of 80%, and the worst performing class was “other,” with an average precision of 34%. Next, a team of UH Hilo undergraduates and I migrated the system to the Android smartphone platform using Tiny YOLO, a smaller version of YOLO that was developed for running on low-powered computing devices. I compared the performance of YOLOv3 and Tiny YOLO at different image sizes with and without transfer learning (pre-training). My results demonstrate that it is possible to deploy object detection technology at beach survey sites to identify and count marine debris objects in real time. The technology is also applicable to other scenarios such as monitoring for plastic marine debris underwater or on the ocean surface. Ultimately, I expect the technology to be deployed as part of a “human in the loop” system in which the object detection component
interacts with the person performing the beach survey so that the system can continuously improve in accuracy as it is used in the field while reducing the time and human labor costs associated with beach debris surveys.
# Table of Contents

Acknowledgements .................................................................................................................. ii

Abstract.................................................................................................................................. III

List of Tables .......................................................................................................................... VIII

List of Figures ........................................................................................................................ IX

Chapter 1: The problem: monitoring plastic marine debris ....................................................... 1

Chapter 2: Object detection technology .................................................................................... 8

  - About object detection ........................................................................................................ 8
  - Object detection with Darknet and YOLO ......................................................................... 8
  - Transfer learning for object detection .............................................................................. 9
  - Evaluation of object detection systems .............................................................................. 10

Chapter 3: Materials and methods .......................................................................................... 13

  - Study site: Kamilo Point on the island of Hawai‘i ............................................................... 13
  - Definition of nine marine debris object classes ................................................................. 13
  - Assembling the dataset of annotated photos .................................................................... 14
  - Management of EXIF metadata ....................................................................................... 15
  - Annotating and managing photos in RectLabel ............................................................... 16
  - Development environment ............................................................................................... 16
  - Installing Darknet ............................................................................................................ 17
  - Darknet and YOLO parameters ....................................................................................... 17
  - Models trained: YOLOv3 and Tiny YOLO, with and without transfer learning ............... 18
  - Precision-recall curves .................................................................................................... 19
Evaluating the mAP scores for small, medium, and large objects ........................................20

Smartphone app .................................................................................................................20

Android smartphone app .....................................................................................................21

Chapter 4: Results ...............................................................................................................22

Dataset of annotated photos .............................................................................................22

Overall (not class-specific) results ....................................................................................22

Small, medium, and large objects ....................................................................................23

Per-class precision and recall results ...............................................................................24

Precision-recall curves .......................................................................................................26

Android smartphone app .....................................................................................................26

Chapter 5: Discussion and future work ............................................................................28

Benefits of increasing image size ......................................................................................28

Precision of Tiny YOLO system ........................................................................................28

Evaluating by size ................................................................................................................29

Average Precision of specific object classes ....................................................................30

Improving the precision of the system .............................................................................30

iPhone smartphone app ......................................................................................................31

Conclusion ..........................................................................................................................31

Tables ..................................................................................................................................34

Figures .................................................................................................................................47

Literature Cited ....................................................................................................................54

Appendix A: Darknet commands .......................................................................................53
map command .................................................................................................................. 53

valid command .................................................................................................................. 53
List of Tables

Table 1. Comments in MDMAP database indicating that data is not recorded or incorrectly recorded ..........................................................................................................................34

Table 2. Comments in MDMAP database indicating data is not being counted ............35

Table 3. Comments in MDMAP database indicating that data entry was delayed long past date of survey ..........................................................................................................................36

Table 4. Definitions of the nine major object classes in alphabetical order ..................37

Table 5. Hashtags of photos scraped from the Instagram photo sharing service ..........38

Table 6. Copyright designations applied to the photos in the dataset .........................39

Table 7. Overall (not class-specific) results from Darknet’s built-in map command .........40

Table 8. Size-specific mAP scores with number of photos and objects when training and validating at Darknet’s default image size of 416*416 when the validation dataset is restricted to only small, medium or large objects. ..........................................................................................................................41

Table 9. Evaluation scores from the map command for YOLOv3 without transfer learning when trained and validated at YOLO’s default image size 416*416 .........................................................42

Table 10. Evaluation scores from the map command for YOLOv3 with transfer learning when trained and validated at YOLO’s default image size 416*416 .........................................................43

Table 11. Evaluation scores from the map command for YOLOv3 with transfer learning when trained at YOLO’s default image size 416*416 and validated at 608*608 .........................................................44

Table 12. Evaluation scores from the map command for Tiny YOLO with transfer learning when trained and validated at YOLO’s default image size 416*416 .........................................................45

Table 13. Built-in Darknet commands for training, testing, validation, and evaluation .......46
List of Figures

Figure 1. Bounding boxes generated by YOLO at different confidence thresholds. ...............47

Figure 2. Examples of photos of the nine major classes of marine debris objects used for training. .........................................................................................................................................................................................48

Figure 3. Hagfish trap in bounding box drawn in purple using RectLabel; blue lines are crosshairs .........................................................................................................................................................................................................................................................49

Figure 4. Total number of annotations of objects in each class and total number photos in which at least one object of that class is annotated (note the log scale). .................................................................50

Figure 5. Precision-recall curves for each of the nine object classes in the YOLOv3 + transfer learning platform. .........................................................................................................................................................................................................................................................51

Figure 6. Screenshot of the prototype Plastic Marine Debris Detector smartphone app ..........52
Chapter 1: The problem: monitoring plastic marine debris

Plastic litter is introduced into the marine environment by indirect or deliberate actions ranging from waste mismanagement to illegal dumping into the ocean. Plastics have been found in the most remote locations in every ocean from its surface waters to the sea floor (Goldberg, 1997; Derraik, 2002; Barnes & Milner, 2005). Because of its wide reach and longevity in the marine environment, plastic has become recognized by the scientific community as one of the largest threats to marine ecosystems (Andrady, 1989; Derraik, 2002; Endo et al., 2005; Barnes et al., 2009; Thompson et al., 2009; Erikson, 2014; Geyer et al., 2017). Plastic poses threats to wildlife including suffocation, entanglement, drowning, and physical rupture of internal organs (Jacobsen et al., 2010; Fossi et al., 2017) and can be a vector for invasive organisms into other environments (Barnes & Milner, 2005). Over time larger plastics can fragment into microplastics (< 5mm in size), which can be mistaken for food by marine organisms. Seabirds, turtles, fish and benthic feeders can ingest plastic fragments, affecting the organism’s viability, or resulting in its starvation, dehydration, or malnourishment from false satiation (Fry, 1987; Mascarenhas et al., 2004; Graham and Thompson, 2009; Van Franeker et al., 2011; Wilcox et al., 2015).

Monitoring of plastic marine debris typically consists of documenting the amount of debris (in numbers or mass) and the distribution and composition of plastic pollution in the environment surveyed. These assessments serve a variety of goals, including discovering source point pollution, identifying marine debris hotspots, assessing wildlife risk, estimating the amount and types of debris in the ocean, extrapolating global and regional trends in plastic pollution, informing public policy and education and outreach efforts, and evaluating the effectiveness of
laws and regulations against plastic pollution (Ryan et al., 2009; Worm et al., 2017; GESAMP, 2019).

There are existing protocols and methodologies for monitoring plastic litter on the shoreline, the sea floor, the sea surface and water column. Since shoreline surveys are the most straightforward to implement, most of what is known about the distribution and composition of marine debris has been gleaned from the hundreds of shoreline surveys conducted since the 1980s (Ryan et al., 2009). These surveys show that plastic is the most frequently cited material type on global debris surveys, accounting for 80% to 100% of the items counted on debris surveys (Ocean Conservancy, 2019; GESAMP, 2019).

The U.S. National Oceanic and Atmospheric Administration (NOAA) has published a field guide for shoreline surveys (Opfer et al., 2012). Under NOAA’s protocol, survey tasks include identifying, hand-counting, and categorizing (based on item type and material) marine debris objects found on the shore that are larger than 1 inch.

NOAA’s Marine Debris Program instituted the National Marine Debris Monitoring and Assessment Project (MDMAP) in 2012. The MDMAP database is a central location where marine debris survey results can be entered by a participating site. Survey results are organized in the MDMAP database by site and date. The database allows for the input of dozens of categories of plastic litter, including large (greater than 1 foot) and unique debris items, which are photographed, uploaded to the site, and recorded as custom data in MDMAP.

I pulled accumulation survey data from the MDMAP database to gain an understanding of the time and labor required to carry out surveys across marine debris sites. Between 2012 and 2020, a total of 3,639 accumulation surveys were entered from 72 partner MDMAP survey sites. Among the 3,639 surveys, the average number of plastic objects counted and removed was 175,
with a range between 0 and 8,313 plastic objects. The average number of persons assisting in each survey was 3.7, with a maximum of 80. The average amount of time required to hand count marine debris objects on the beach and secure the items for removal was slightly over 60 minutes. This excludes the time spent discarding litter at the transfer station and the time entering data and uploading photos into the MDMAP database. These figures suggest the average productivity for shoreline surveys is about 50 plastic objects per person per hour for the MDMAP database as a whole.

Focusing on the MDMAP data for Kamilo Point on Hawai‘i Island (my research site), which has been submitted to MDMAP since 2012 by Hawai‘i Wildlife Fund (HWF, a nonprofit organization in Hawai‘i), the average survey time is 1.9 hours over 34 marine debris surveys. The average number of plastic objects removed and secured from the shoreline is 2,525 with a range between 567 and 6,822. The average number of persons assisting in each survey was 8.7, with a range between 2 and 26. These figures suggest the average productivity for shoreline surveys is about 500 plastic objects per person per hour for HWF. In other words, according to the MDMAP data, HWF surveys are about ten times more productive than the average survey recorded in MDMAP. This may be due to the greater concentration of plastic marine debris in the Hawaiian archipelago, which is a recognized a hotspot for marine debris accumulation (Dameron et al., 2007; Morishige & McElwee, 2012; Blickley et al., 2016; Moy et al., 2018) due to its close proximity to the center of the North Pacific Gyre or the “Great Pacific Garbage Patch” (Lebreton et al. 2018). The complex behavior of the ocean currents and the winds in this location make Kamilo Point an accumulation point for plastics routinely moved out of the gyre (Moore, 2011; Maser, 2014).
The MDMAP database\(^1\) includes a “Notes” field for each survey where survey leaders can enter comments. Examining these comments, I noticed many that indicated problems encountered in the process of carrying out the survey manually that could have been avoided if the survey process were more automated (Tables 1-3). For example, by automating the process of recording data from a survey such as the debris counts and parameters like the time and date (instead of writing the data manually on paper forms), it would be possible to avoid missing data problems (Table 1). Making the debris counting process faster and easier would ameliorate counting failures (Table 2). And by automating the process of updating the MDMAP database with the survey data as soon as the survey is over, it would be possible to avoid data entry delay problems (Table 3).

Other comments in the MDMAP “Notes” field indicated problems with volunteer burnout: “Volunteer gave notice this will be their last survey as it is ‘getting to be too much for them.’” Or as one volunteer succinctly put it: “I have decided to end my debris monitoring at this time.”

The data and comments in the MDMAP database suggest that a significant amount of time and manual effort is currently being expended by scientists and volunteers on shoreline marine debris surveys. In the marine debris literature, shoreline surveys have been characterized as “labor-intensive” (Opfer 2012) and “demanding” (Ryan et al. 2003).

Having spent two years conducting monthly shoreline surveys on Hawai‘i Island, I can relate to many of the issues noted in the comments and can attest from personal experience that current marine debris surveys are inefficient and can be overwhelming when there is a lot of debris to count and remove. Furthermore, once the beach survey field work is complete, the task

\(^1\) https://mdmap.orr.noaa.gov/
of manually updating the NOAA MDMAP database, which is currently a separate process, is time-consuming and repetitive.

Today, manual tasks that are repetitive and labor-intensive are ripe targets for automation by Artificial Intelligence (AI). This fact has been demonstrated in a range of fields including accounting, law, life sciences, manufacturing, and medicine (LeCun et al., 2015; Furman et al., 2019). If marine debris survey work could be partially automated by AI, this would mean less time spent collecting plastic debris, tallying objects, and entering survey data into the NOAA MDMAP database, including the process of uploading photos of large debris items with descriptions and measurements. Full or partial automation of manual survey work would free up time for scientists, researchers, and volunteers to spend on more important work such as plastic pollution education and outreach efforts, including engaging with legislators.

A second benefit of greater automation is greater harmonization of monitoring approaches between survey sites and better data sharing and management. These issues are recognized as problems in the marine debris community. In 2019, the Joint Group of Experts on the Scientific Aspects of Marine Environmental Protection (GESAMP) concluded that marine debris surveys suffered from a “lack of harmonization of sampling methods,” and that “a greater harmonization of sampling protocols and reporting” would ease barriers to effective global data sharing and data management (GESAMP, 2019). To the extent individual and site-specific human judgments can be replaced by algorithms, the sampling protocols and reporting methods used by the marine debris community can be made more uniform, easing barriers to effective global data sharing and data management.

In this thesis, I explore one critical facet of the problem of automation of plastic marine debris surveys, namely the problem of automatically identifying and categorizing plastic marine
debris in beach environments in real time using object detection technology embedded in a smartphone app.

A previous example of the use of a smartphone app to address the problem of global data management and tracking plastic marine debris is Marine Debris Tracker (Jambeck & Johnson, 2015), which was developed jointly by the NOAA Marine Debris Program and the Southeast Atlantic Marine Debris Initiative. The Marine Debris Tracker app allows users to report marine debris or litter they find along coastlines or waterways, and works as an electronic notebook recording marine debris objects discovered by citizen scientists. It is a good example of an effective implementation of marine debris tracking that combines mobile technology with the power of crowdsourcing. However, it does not use object detection technology to facilitate the process of on-site identification of marine debris.

Fulton et al. (2019) describe an effort to apply object detection technology to identify plastic marine debris underwater from an autonomous underwater vehicle. They compared the effectiveness of four deep learning algorithms (YOLOv2, Tiny YOLO, Faster R-CNN, and SSD). They found that Tiny YOLO produced results that were comparable in accuracy to the larger systems but at much higher speeds, suggesting that real-time marine debris detection using visual deep learning models is feasible. Although this work was applied to underwater marine debris rather than shoreline marine debris, it established the feasibility of the idea of using Tiny YOLO object detection technology in my own project.

The goals of my thesis are as follows:

1. To use the YOLO algorithm to build an object detection model that is able to detect common plastic debris objects as a proof of concept system; and
2. To migrate the model onto a low powered computing device in order to demonstrate that the technology can work on-site at a beach survey.

In the remaining chapters, I explain what object detection technology is and how it works, how I applied the technology to the problem of identifying plastic marine debris on the beach, and what results I achieved.
Chapter 2: Object detection technology

About object detection

Object detection is an image processing technology that is used to identify and locate objects of particular classes (e.g., humans, automobiles, and traffic signs), ideally in real time (Liu et al., 2020). Object detection models are built by defining semantic classes (in this case, categories of plastic debris such as bottles and fishing nets) over the dataset of relevant images and then annotating the dataset with bounding boxes that identify objects of those classes for purposes of training a machine learning system.

Currently, a popular automated process used for object detection is “deep learning,” which is a machine learning approach based on convolutional neural networks (CNNs). CNNs process images by creating associative connections between nodes organized in a network of layers (Krizhevsky et al., 2012). Different layers effectively act as filters for recognizing different properties (shapes, colors, textures, edges, etc.) of the objects in the target image (Wu, 2017).

Object detection with Darknet and YOLO

In this project I used the combination of Darknet and YOLOv3 as my deep learning framework. Darknet features 75 convolutional layers followed by two fully connected layers (Redmon & Farhadi 2018). YOLO, which stands for You Only Learn Once, is the image detection algorithm used in Darknet. YOLOv3 works by applying a single neural network to the full photo to be analyzed in order to divide the image into regions and predict bounding boxes and probabilities (associated weights) for each region.
There is also a smaller version of YOLOv3 called Tiny YOLO designed for mobile machine learning and low-powered computing devices such as Internet of Things (IoT) devices. The CNN created by Tiny YOLO is only about 20% of the size of the CNN created by the full YOLOv3 and runs several times faster.

YOLOv3 is a popular choice for the development of object detection systems (e.g., Luo et al. 2019; Zhong & Meng, 2019) because it offers accuracy that is comparable to the best performing object detection algorithms (as measured in the Common Objects in Context (COCO) and Pascal VOC competitions), yet runs much faster than its competitors (Redmon & Farhadi, 2018). Furthermore, Tiny YOLO, the lightweight version of YOLOv3, is a popular choice for the deployment of object detection applications on relatively low-powered devices such as mobile phones, laptops, and the compact NVIDIA Jetson TX2 GPU module because it had a proven track record of effective performance on these platforms (Barry et al., 2019; Fulton et al., 2018; Redmon & Farhadi, 2018).

Transfer learning for object detection

In machine learning, “transfer learning” refers to the application of knowledge gained in one problem domain to a different but related one (Ng, 2016). For example, knowledge gained while learning to detect cars can be applied to recognizing trucks. In Darknet, transfer learning is performed by “freezing” all but the last several layers of a pre-trained CNN before training so that only the last few layers get their weights updated (see e.g. Fulton et al., 2018). In this way, one can build on the edge detection and other basic detection capabilities implemented in the low-level layers of a CNN that has already been successfully pre-trained on a huge dataset. Once these low-level layers are transferred over to the new domain, where there is less data available,
training can proceed by training only the higher-level layers that identify the specific objects in the new problem domain (in my case, plastic marine debris).

In the object detection field, two popular sources of data for transfer learning are the ImageNet dataset (Deng et al. 2009), which comprises 14 million total images, and the COCO dataset (Lin et al. 2014), which comprises 120,000 total images (organized into a training set of 80,000 images and a validation set of 40,000 images).

Evaluation of object detection systems

Machine learning systems are evaluated using the performance metrics of precision and recall (Russell & Norvig 2009). In the domain of object detection, precision measures how often identified objects are correctly identified to their object class, while recall indicates how often objects are actually correctly identified (and not overlooked) (Everingham et al., 2010). Both are computed as values between 0 and 1, with 1 being the best. In order to compute the precision and recall, it is necessary to first compute the number of true positives (TP) (i.e., objects successfully recognized), false positives (FP) (i.e., cases in which the machine identifies a nonexistent object), and false negatives (FN) (i.e., the number of true objects that the machine overlooks). The system's precision is defined as the ratio TP/(TP+FP), and recall is defined as the ratio TP/(TP+FN). Note that in general there is no notion of a true negative (TN) in the object detection task, except under special circumstances (for example, if counting only one object on each image).

Sometimes the precision and recall results are combined into a single measure called the F1 score, computed as \(2PR/(P+R)\), which represents the harmonic mean of the precision \(P\) and recall \(R\) (Russell & Norvig 2009). An F1 score of 1 indicates perfect precision and recall.

Precision and recall are obtained by comparing the system-generated bounding boxes to
the human annotations. To evaluate the quality of bounding boxes generated by the system, each machine-generated bounding box in the validation dataset is compared to the corresponding bounding box produced by the human annotator for the same object in the same image, and then the metric IoU (Intersection over Union) is computed. IoU is calculated as the ratio $I/U$, where $I$ is the area of the intersection of the two bounding boxes (machine- and human-generated), and $U$ is the area of the union of the two bounding boxes. The result is a value between 0 and 1, with 1 indicating that the machine-generated bounding box is identical to the corresponding bounding box produced by the human annotator.

A detection is considered a true positive when the IoU exceeds 50% (0.5). As Everingham et al. (2010) explain, “the threshold of 50% was set deliberately low to account for inaccuracies in bounding boxes in the ground truth data, for example defining the bounding box for a highly non-convex object, e.g. a person with arms and legs spread, is somewhat subjective.”

One commonly used metric for evaluating an overall object detection system is the average precision (AP), which is calculated as the average precision value over a set of different recall thresholds. In the case of the 2007 VOC competition, the AP is defined as the mean of precision values at a set of 11 equally spaced recall levels $[0,0.1,\ldots,1]$ (that is, 0 to 1 at step size of 0.1) (Everingham et al., 2010). In the object detection literature, the AP score is often computed at a particular IoU threshold, typically 0.5 (Everingham et al., 2010). For example, an evaluation score labeled AP$_{50}$ indicates an AP score calculated using the standard IoU threshold of 0.5.

AP scores are typically reported on a per-class basis. In order to evaluate the performance of a system as a whole, taking into account all object classes, researchers calculate the
mean average precision (mAP), which is the AP averaged over all object classes. The mAP score has been used as a convenient single metric for judging the Pascal VOC and COCO object detection competitions (Everingham et al., 2010; Lin et al., 2014; Everingham et al., 2015). In the object detection literature, the mAP score is traditionally reported as a percentage value to two decimal places (for example, 67.89%).

In Darknet, the recall of the system can be manipulated by setting the confidence threshold as a parameter. Each predicted object bounding box is assigned a confidence score, defined as Pr(object) * IoU, and the bounding boxes that fall below the confidence threshold are ignored (Redmon & Farhardi 2018). For example, if the confidence threshold is set to 0.4, then all predicted bounding boxes with a confidence score below 0.4 are pruned from the output. When the threshold is high, recall is high, and more bounding boxes are generated by the system (Figure 1).

Another variation of the AP score is the size-specific AP score, based on the size of the detected object in pixels. For example, in the COCO competition (Lin et al., 2014), objects less than $32^2$ pixels are classified as “small,” objects greater than $96^2$ pixels are classified as “large,” and objects in between these sizes are classified as “medium.” Research teams participating in the COCO competition compute and submit three size-specific AP scores, $AP_S$, $AP_M$, and $AP_L$. The reason for reporting size-specific AP scores in this way is to assess whether a given object detection system is better at detecting small objects than large objects or vice-versa.
Chapter 3: Materials and methods

Study site: Kamilo Point on the island of Hawai‘i

While completing this thesis, I spent two years as a volunteer with HWF participating and leading marine debris surveys on the beach at Kamilo Point on the island of Hawai‘i.

Kamilo is an isolated beach in the middle of an 80-mile stretch of rocky, undeveloped coastline. It is located within the Ka‘ū Forest Reserve in the Waiʻōhinu ahupua‘a (land division), owned by the State of Hawai‘i Department of Land and Natural Resources (DLNR), and managed by the Division of Forestry and Wildlife (DOFAW). HWF has had the necessary permits, permission, and 4WD transport equipment to conduct marine debris surveys and community-based removal efforts in this area since 2003.

According NOAA’s Marine Debris Shoreline Survey Guide, the survey area should be at least 100 m long, and the area of debris sampling should occur between the water’s edge and the first barrier (i.e. the vegetation line).

In the HWF surveys, marine debris is sampled from a 100 m by 10 m area along the shoreline of the beach from the sea edge to the vegetation line with no random transects created. All debris, regardless of material type, over 2.5 cm in diameter is hand counted, recorded by type and material, and then removed from the area for disposal. Any debris over 100 cm in length is photographed and input in a separate “large debris” category. The results of each HWF survey are entered into NOAA’s MDMAP database.

Definition of nine marine debris object classes

I defined eight major classes of marine debris objects comprising a representative sampling of both consumer plastics (bags, bottles, bottlecaps, and containers) and derelict fishing
and aquaculture gear (fishing nets, buoys, hagfish traps, and oyster spacers), and an “other” class for all other plastic objects (Figure 2 and Table 4). In selecting the eight major classes, I took into account the scientific literature on commonly found objects (Barnes et al., 2009; Blickley et al., 2016) and historical data from the Ocean Conservancy and HWF beach surveys.

I chose to focus on nine classes in this project as a compromise between tractability and representativeness. Although NOAA’s MDMAP database recognizes 47 different classes of marine debris, I decided to limit my project to fewer than nine in the interests of tractability (achieving results in a limited time). On the other hand, I wanted to be able to detect a range of object classes comprising a representative sampling of both major categories of plastic marine debris, namely consumer plastics and fishing and aquaculture gear (Barnes et al., 2009; Morishige & McElwee, 2012; Ocean Conservancy, 2019).

Assembling the dataset of annotated photos

My largest source of photos of plastic marine debris was Instagram, a popular photo sharing social networking service owned by Facebook. To efficiently download thousands of photos from Instagram, I used instagram-scrap (Rarcega, 2020) a command-line application written in Python for this purpose. I scraped approximately 200,000 photos from Instagram that were tagged with various tags associated with plastic marine debris (Table 5).

Thus far, I have only sorted, trained and tested on the photos tagged with #marinedebris and #oceanplastic, comprising a total of 3,606 Instagram photos. (In other words, scraping Instagram for photos yielded many more photos than I have actually been able to use.)

I also searched Flickr, an online photo sharing and hosting platform, to supplement the “bag” object class. Flickr allows users to share their content, and I only used photos that were
made downloadable by the original owner of the photo. Finally, I supplemented the dataset with HWF’s photos and my own.

When acquiring photos from Instagram and other sources online, I omitted the following types of photos from the training data:

- Photos with logos or text (for example, “Prevent plastic pollution!”) superimposed over the image
- Photos of jewelry, toys, or other artifacts made out of marine debris as arts and craft projects (e.g., bracelets made of derelict fishing nets)
- Debris items in piles where the object edges are not easily distinguishable, making the object difficult to draw a bounding box around
- Artwork such as oil paintings or digital renderings

The dataset includes occluded and small objects if they could be identified by the annotator and had clear edges. Most of the photos in the dataset are of marine debris objects found on the beach, but some photos show plastic litter in other settings such as on the street or in a park.

The Instagram and Flickr photos can legally be used for academic research under the “fair use” exemption to US copyright law (U.S. Copyright Act, 1976). However, I cannot distribute these photos to others without the express permission of the copyright holders. The usage permissions associated with each photo in my dataset are documented in its EXIF metadata as described below.

Management of EXIF metadata

Photos in JPEG format typically include metadata with information such as the source of the photo, the time and geospatial coordinates at which the photo was taken, technical attributes such as resolution, shutter speed, and exposure time, and information about the copyright on the photo and its permissible uses. To keep track of this information, I used the EXIF metadata
standard (CIPA, 2012), which was designed for this purpose. The JPEG format allows EXIF metadata to be stored directly in the JPEG file itself, which facilitates management of the information, since there is no need to synchronize the information across multiple systems or platforms (for example, in a separate database).

To search and manipulate the EXIF metadata, I used ExifTool (Version 11.38), a Unix program developed for this task. ExifTool allows the EXIF metadata to be manipulated at the individual photo level or in bulk (applied to an entire directory), which makes it particularly useful for managing the metadata of a large dataset of photos.

I also used ExifTool to strip the orientation metadata of each photo. This turned out to be necessary because if orientation metadata is present, RectLabel (the annotation software) and Darknet (the CNN framework) do not always interpret it in a consistent manner, which can cause the orientation of the photo and its bounding boxes to be different in each application.

Annotating and managing photos in RectLabel

I used RectLabel, a MacOS application for annotating photos with bounding boxes for machine learning applications, to annotate the classes of plastic marine debris objects in my photos (Figure 3) as well as to organize the folder structure of the photos.

Development environment

I used the High-Performance Computing (HPC) cluster operated by the University of Hawai‘i at Mānoa for development. The HPC node I used to compile Darknet, and on which I trained, tested, and evaluated my object detection models, is an Intel Xeon E5-2660 v3 (10 core 2.60 GHz) with 128 GB RAM and eight GPUs on a GeForce RTX 2080 Ti graphics card. The GPUs are essential for training CNNs in a reasonable time. For instance, using all eight GPUs, it
took roughly three days to train a YOLOv3 model, and roughly one week to train a Tiny YOLO model, depending on the size of the training dataset.

Installing Darknet

Two versions of the Darknet codebase are available on the GitHub development platform. The original code by Joseph Redmon is published online at Redmon’s site.² There is also a fork of the Darknet codebase on GitHub.³ The majority of Darknet users prefers the AlexeyAB fork (AlexeyAB, 2020) of Darknet because it fixes several bugs in Redmon’s original code and offers faster performance. For this project I used the AlexeyAB fork.

Darknet and YOLO parameters

Darknet’s training and testing parameters are manipulated using configuration files. Mostly, I re-used the default settings and parameters for the VOC 2007 data, which are supplied with Darknet in the configuration files voc.data and yolov3-voc.cfg.

The parameters I adjusted were as follows:

A. YOLO’s default image size is 416*416 pixels (training photos are automatically resized to these dimensions, with the empty areas of non-square photos blacked out). In some experiments, I changed this parameter to resize the image dimensions to 256*256 or to 608*608 to see how the lower or higher image size setting would affect performance.

B. For my Darknet training and testing configuration settings and parameters, I changed the parameter classes (number of classes) to 9 and the parameter filters (number of convolutional filters) to 42. The parameter classes refers to the number of object

² https://github.com/pjreddie/darknet
³ https://github.com/AlexeyAB
categories (bottles, nets, etc.) recognized by the plastic marine debris model. The parameter filters refers to the number of detection filters (vectors of weights for detecting the presence of specific features) in the final convolutional layer of the CNN. In YOLOv3 this parameter is always set to $3 \times (\text{classes} + 5)$.

Models trained: YOLOv3 and Tiny YOLO, with and without transfer learning

In addition to YOLOv3, which was the latest version of YOLO at the time, I also trained models on Tiny YOLO, which uses 16 convolutional layers. Following the default Darknet settings in the configuration files, I trained the YOLOv3 models for 52,000 iterations and the Tiny YOLO models for 100,000 iterations.

I trained and tested multiple Darknet CNNs, including some trained “from scratch” on only marine debris objects and others trained with the assistance of transfer learning. For the transfer learning on YOLOv3, I used an initial set of convolutional layers trained on ImageNet that I downloaded from the YOLO home page. For the transfer learning on Tiny YOLO, I used an initial set of convolutional layers trained on the COCO dataset that I downloaded from the YOLO home page. The ImageNet dataset is more than 100 times larger than the COCO dataset and therefore preferable for transfer learning, but unfortunately no pre-trained weights for ImageNet are available for the Tiny YOLO platform.

In the remainder of this thesis I will abbreviate the names of the different validation platform as follows. The “YOLOv3” platform refers to a YOLOv3 CNN trained from scratch. The “YOLOv3 + transfer” platform refers to a CNN trained not from scratch, but from an initial set of convolutional layers pre-trained on the ImageNet dataset. The “Tiny YOLO” platform refers to a Tiny YOLO CNN trained from scratch. The “Tiny YOLO + transfer” platform refers
to a Tiny YOLO CNN trained not from scratch, but from an initial set of convolutional layers pre-trained on COCO dataset.

I also took my two best performing models, “YOLOv3 + transfer,” and “Tiny YOLO + transfer” and adjusted the default image size of 416*416 to 608*608 during validation to test whether the higher resolution would boost object detection performance.

Precision-recall curves

I generated precision-recall curves for each of the nine object classes based on the output of the Darknet valid command (Appendix A). I created the graphs using pyplot, a Python library for interactive plotting.

Precision-recall curves illustrate the tradeoff between the precision and recall in the system and offer a visual snapshot of the AP score, since the AP is by definition a sampling of different points along the curve (Everingham et al., 2010). A large area under the curve represents both high recall and high precision and therefore indicates a high AP score. Precision-recall curves are also useful for comparing the performance of different algorithms and assessing how different parameters affect performance.

Conceptually, a precision-recall curve is generated by iteratively removing the least confident detection and recalculating the precision and recall for the remaining detections. To plot the point at the bottom right of the precision-recall curve (that is, the point with the lowest precision and the highest recall), all the detected objects with a confidence score above the minimal threshold of 0.005 are counted. To calculate the other points (that is, the points with increasing precision and decreasing recall), the confidence threshold is incrementally raised until only the most confident detection is left.
Evaluating the mAP scores for small, medium, and large objects

A Python script included in the Darknet installation package converts from Rectlabel’s XML annotation format to YOLO format. The YOLO format normalizes the width \((w)\) and height \((h)\) of the object’s bounding box to a value between 0 and 1 relative to the photo’s size. By modifying this Python script, I was able to calculate the area of each object’s bounding box by multiplying \(w \times h\). After computing the area of the bounding boxes in each photo in the validation dataset, I sorted the photos into four bins: photos containing only small objects, photos containing only medium objects, photos containing only large objects, and all other photos. In this way I was able to compare the mAP scores for photos containing only small, medium, and large objects. I defined a small object is defined as an object whose bounding box area is less than 10% of the photo’s full size, a medium object as an object whose bounding box area is between 10% and 50% of the photo’s full size, and a large object as an object whose bounding box area is greater than 50% of the photo’s full size.

Smartphone app

The marine debris object detection system is intended to be used by citizen scientists and volunteers who conduct shoreline surveys, most of whom are not data scientists with access to eight-GPU servers in a university supercomputer cluster. Therefore, to make the system useful for use on-site at a beach debris survey, I recruited three undergraduate students from the fall 2019 and spring 2020 Computer Science 460 and 461 (Software Engineering) class at UH Hilo: Briana Noll, Stacey Yanagihara, and Charnalyn Crivello, to implement the Tiny YOLO model on a smartphone platform.
Their assignment was to (1) migrate the Tiny YOLO model to a smartphone platform; (2) implement the system as a mobile app; and (3) design a user interface for interacting with the mobile app. These deliverables were used to satisfy the software engineering requirements of the class.

Android smartphone app

The team succeeded at migrating the Tiny YOLO model to the Android platform and developed a working prototype app in Android Studio using OpenCV, an open source computer vision and machine learning software library compatible with Darknet. The app was tested on a Samsung Galaxy S10e smartphone.

Although Android Studio is a Java development environment and OpenCV’s native language is C++, Android Studio supplies Java wrappers, allowing implementation of OpenCV in Android Studio. Once OpenCV was configured as a dependency in Android Studio, the Tiny YOLO weights file could be imported as an asset and called using built-in OpenCV methods.
Chapter 4: Results

Dataset of annotated photos

I assembled a total of 5,464 photos of marine debris, containing a total of 28,295 annotations of marine debris objects (bounding boxes and class labels) (Figure 4).

I used RectLabel’s Export command to randomly distribute the annotated data into a training dataset (80% of the data) and a validation dataset (the remaining 20% of the data), following standard practice in the machine learning field (Russell and Norvig, 2009). The evaluation results given in this chapter were computed on the validation dataset.

Overall (not class-specific) results

The overall results (Table 7) include the mAP score, which is the average AP score over all object classes calculated using the standard IoU threshold of 0.5, as well as the precision, recall, F1 score, total true positives (TP), false positives (FP), false negatives (FN) and average IoU when YOLO’s recognition confidence threshold is set to the default value of 0.25. The platforms marked “608*608” were validated after changing the image size parameter in the Darknet configuration file from the default 416*416 pixels to 608*608. The platform marked as “256*256” was validated after changing the same parameter to 256*256. Note that because the mAP score for Tiny YOLO at 256*256 was so low, I did not calculate the other results (precision etc.) for this platform.

The overall results show that the mAP score of the YOLOv3 model trained from scratch performed far worse than all the other models. In particular, Tiny YOLO, the smaller version of YOLO, obtained a mAP score three times higher than YOLOv3 when trained from scratch without transfer learning. Conversely, when transfer learning was used, YOLOv3 outperformed
Tiny YOLO. This behavior has been explored in a discussion prompted by user ssanzer (2018) on the Darknet GitHub site, where the consensus is that Tiny YOLO outperforms YOLOv3 on small training sets, whereas YOLOv3 outperforms Tiny YOLO on large training sets. The other takeaway from the overall results is that both the YOLO and Tiny YOLO models perform better when validated at the higher image size parameter of 608*608.

In summary, two techniques were effective in achieving higher mAP scores and F1 scores on the YOLOv3 system. The most effective technique by far was to train using the transfer learning technique. YOLOv3 without transfer learning achieved a mAP score of 12.16%, whereas YOLOv3 with transfer learning achieved a mAP score of 48.35%. In terms of F1 scores, YOLOv3 without transfer learning achieved an F1 score of 0.08, whereas YOLOv3 with transfer learning achieved an F1 score of 0.45.

The second technique was to increase the image size of the photos from YOLO’s default image size of 416*416 to 608*608 during validation. When this was done, the mAP score on YOLOv3 increased from 48.35% to 52.38% and the F1 score increased from 0.45 to 0.47.

On the Tiny YOLO platform, transfer learning and boosting the default image size were also effective at boosting the mAP and F1 scores, although the benefits were smaller than on the YOLOv3 platform. Specifically, on the Tiny YOLO platform, transfer learning boosted the mAP score from 37.14% to 39.92%, and then increasing the validation from 416*416 to 608*608 gave the mAP score a small boost to 40.91%.

Small, medium, and large objects
I sorted the photos in the validation deck into small, medium, and large bins (Table 8) based on the sizes of the objects in each photo in order to gain insight into how the sizes of the objects affected the overall mAP scores of the models.

Medium objects had the best mAP score at 63.66%, slightly better than the mAP score for large objects (60.09%), and far better than either of the two mAP scores for small objects.

The strategy of increasing the image size of the photos during validation seems to be effective at boosting the precision of the detection of small objects. When I increased the image size to 608*608, the mAP score of the small objects rose from 44.79% to 48.70%.

Per-class precision and recall results

This section gives the per-class precision and recall results generated by the Darknet map command (Appendix A) for each of the platforms. The tables give the total number of annotations in the training and validation set, the AP50 score, and the precision, recall and intersection over Union (IoU) values at Darknet’s default confidence threshold 0.25, along with the number of true positives (TP) and false positives (FP) at the 0.25 threshold parameter. The tables of the results for the various platforms are presented in the following order:

YOLOv3 (Table 9)

YOLOv3 + transfer (Table 10)

YOLOv3 + transfer 608*608 (Table 11)

Tiny YOLO and Tiny YOLO + transfer (Table 12)

This section shows per-class results for Tiny YOLO at the default 416*416 image size, but not at the other sizes (256*256 and 608*608). At 256*256 pixels, Tiny YOLO produced a very poor mAP score (Table 7), and so the low-resolution model was not analyzed further. At the higher 608*608 resolution, Tiny YOLO’s mAP score was only slightly better than at the default
416*416 image size (Table 7), but in practice, Tiny YOLO performed far more slowly at the higher resolution setting. Therefore, the results for Tiny YOLO below include only the precision and recall at the default image size of 416*416.

The results of YOLOv3 without using transfer learning (Table 9) are poor. The worst-performing class is bottlecap, with an AP50 score of 7.02%, and the best is net, with an AP$_{50}$ score of 18.84%. YOLO is known to perform poorly when trained on small datasets (Li et al., 2018), and these results confirm that our training dataset is still too small to produce a usable system without the use of transfer learning.

The results for YOLOv3+transfer learning at 416*416 (Table 10) and YOLOv3+transfer learning at 608*608 (Table 11) are much better than the results without transfer learning. The four best-performing object classes are bottle, bottlecap, hagfish trap and oyster spacer, all of which achieved AP$_{50}$ scores above 50% and precision scores above 0.6 at the default confidence threshold of 0.25. The four worst-performing object classes, excluding the “other” class were (ordered from worst to best in AP scores): bag, container, buoy, and net.

When validated at 608*608 (Table 11), almost every object class achieved a higher AP score with the exception of bag and net. The bag and net classes also showed steep drops in IoU scores at the higher image size, with the bag class dropping from an IoU of 0.41 down to 0.30 and the net class dropping from an IoU of 0.52 down to 0.35.

The per-class results for Tiny YOLO+transfer learning at 416*416 (Table 12) were worse than the corresponding results for YOLOv3+transfer learning (Table 10) in every category. However, the Tiny YOLO+transfer learning results are much better than the YOLOv3 without transfer learning (Table 9). In other words, the benefit obtained from transfer learning is far greater than the penalty imposed by switching from YOLOv3 to the smaller Tiny YOLO model.
In the Tiny YOLO results, just as for YOLOv3, bottles, bottlecaps, hagfish trap, and oyster spacers once again outperformed the other classes.

Precision-recall curves

The precision-recall curves for each of the nine object classes in the YOLOv3 + transfer learning platform (Figure 5) illustrate the tradeoff between the precision and recall in the system. The large areas under the curves for hagfish traps and oyster spacers reflect the fact that these are the two classes with the highest AP scores. Conversely, the “other” class, which has the lowest AP score, exhibits a convex curve with a low area under the curve.

Precision-recall curves are also useful for comparing the performance of different algorithms and assessing how different parameters affect performance. For example, examining the curves in the figure, for bottles, buoys, nets, containers, and oyster spacers, the graphs are particularly steep in the vicinity of the diamond, meaning a small decrease in recall produces a large increase in precision. These results can be used to set the confidence threshold parameter in Darknet to maximize the performance of the system across all the objects.

Android smartphone app

My team of undergraduates and I were able to deploy the Tiny YOLO model on a Samsung Galaxy S10e smartphone. A screenshot of the app running on the Samsung (Figure 7) shows that when the app is executed, the camera captures each frame and processes it with Tiny YOLO, which generates a bounding box and a predicted class for each detected object. The bounding boxes and class labels are printed directly onto the frame. The app allows users to confirm and capture the data by selecting the **Capture Data** button located at the bottom of the
screen. If the prediction does not match the object’s true class, the user can specify it manually using a drop-down menu in the top left corner of the screen. The user can also manually specify any “other” debris item counted that is not included the eight major classes. The captured data is recorded onto the device and can be exported and shared as a CSV file. Metadata and other data generated while the app is used is stored in resource files on the Android device. At the default 416*416 image size, the system is able to run without lag on the Samsung Galaxy S10e at a rate of at approximately 30-40 frames per second (FPS).

I tested the speed, detection capabilities, and durability (battery life and overheating) of the prototype smartphone app in different environments, including at a regularly scheduled HWF beach survey at Kamilo Point, where the prototype was able to successfully identify plastic marine debris objects and generate bounding boxes for them. After repeated tests in various environments, my team and I found no issues with speed or overheating, and even after heavy usage, the system did not drain the smartphone’s battery capacity.
Chapter 5: Discussion and future work

Benefits of increasing image size

Increasing the image size from 416*416 to 608*608 during validation of YOLOv3 with transfer learning improved the network’s recall. Specifically, the recall at confidence threshold 0.25 rose from 0.37 at 416*416 to 0.44 at 608*608. This increase in recall translated into an improvement in the overall mAP score (from 48.35% to 52.38%), demonstrating that boosting the image size was a good idea in general. On the other hand, boosting the image size caused the average IoU to fall. Specifically, the average IoU at confidence threshold 0.25 fell from 44.31% at 416*416 to 38.89% at 608*608. This result may be due to the fact that boosting recall means generating more bounding boxes at the margin, and the new marginal bounding boxes can be expected to align less well with the human-annotated bounding boxes.

Precision of Tiny YOLO system

The reason transfer learning was more effective on YOLOv3 than Tiny YOLO is because the YOLOv3 transfer learning used weights that were pre-trained on ImageNet, which consists of more than 14 million images. In contrast, the Tiny YOLO transfer learning used weights that were pre-trained on the COCO dataset, which consists of only about 120,000 images.

Increasing the validation image size from 416*416 to 608*608 yielded a small improvement in the Tiny YOLO + transfer results, boosting the mAP score from 39.92% to 40.91%. However, on an actual low-powered computing device deployed in the field, increasing the image size on Tiny YOLO is not always feasible. At higher image sizes, the speed at which the system generates bounding box predictions slows dramatically. Conversely, reducing the image size parameter to 256*256 would achieve a very responsive network (low inference time).
on a low-powered device (Redmon & Farhardi, 2018). To test this, I tried configuring the image size parameter to 256*256. However, this caused the mAP score to drop down to 30.53%. My results suggest that for Tiny YOLO, the default parameter of 416*416 offers a good balance of speed and precision on a low-powered device.

Evaluation by size

Medium and large objects had higher mAP scores than small objects. There were far more small objects than large objects in this validation set that I tested, and the system was worse at detecting them than the larger objects. This result is expected given that smaller objects are harder to recognize by detection algorithms (Everingham et al. 2010; Lin et al. 2014; Redmon et al. 2016; Zhao et al. 2019), and improving performance on small objects has been an ongoing goal in object detection research (Lin et al. 2014; Kisantal et al. 2019).

The strategy of increasing the image size of the photos during validation seems to be effective at boosting the precision of the detection of small objects. When I increased the size to 608*608, the mAP score of the small objects rose from 44.79% to 48.70%.

For an object detection system deployed in the field for beach surveys, it seems unlikely that the user would focus on small objects (defined here as objects whose bounding box area is 10% of the total area of the photo). Small objects would be more of a concern when performing the object detection task from a large distance (e.g., aerial drone surveys). The fact that the system performs worse on small objects is therefore not a major concern. In general, truly small objects such as microplastics are excluded from counting in beach survey protocols (Opfer et al., 2012).
Average precision of specific object classes

The four best-performing object classes were bottle, bottlecap, hagfish trap and oyster spacer, while the four worst-performing object classes, excluding the “other” class, are (ordered from worst to best in AP scores): bag, container, buoy, and net.

One reason for low the precision scores for these object classes is that object detection systems still have difficulty with occluded objects, since occlusion removes information about the object (Everingham et al., 2010; Op het Veld et al. 2015). Bags and nets are flexible objects that do not always present clear edges in photos, and the edges are often occluded by water, sand, leaves, and other objects. In general, neural networks have an easier time learning hard edges at right angles than complex or fuzzy edges (Wölker, 2019). It is also possible that other physical material properties of bags, for example their texture, make them more difficult to detect by deep learning. In general, deep learning systems are “black boxes,” meaning it is difficult to identify the specific features associated with the individual convolutional layers in the network.

Improving the precision of the system

There is no reason to view the current mAP scores as an upper limit. The amount of training data I used is still small compared to the larger datasets available to researchers with more resources than I have access to (e.g., Deng et al., 2009).

In addition to collecting and annotating more data, another technique for improving training is data augmentation (Zoph et al. 2019). This a strategy that increases the diversity of data available without actually collecting new data, for example by cropping, padding, or flipping the images horizontally. This approach could be particularly useful for some of my object classes such as buoys or containers that were under-represented in the training data.
A Smartphone App

Although my team of undergraduates succeeded in deploying the Tiny YOLO model on the Android platform, they were not able to migrate Tiny YOLO to the iOS platform to make an iPhone mobile app. The problem was that the Tiny YOLO weights file could not be converted from the Darknet YOLO format to a format compatible with Core ML, which is the software that integrates machine learning models into iOS apps. I learned that there is a forked version of Darknet for MacOS, implemented in Python, that is apparently able to generate weights compatible with Core ML. Unfortunately, I was unable to test this option because UH Hilo does not have a suitable high-end MacOS development environment (a multi-GPU system).

Conclusion

The system I developed is a successful “proof of concept” that demonstrates that modern day object detection technology can be developed and applied to the task of assisting surveyors in identifying and counting plastic marine debris in beach environments in real time. Additionally, because the CNN was trained on photos depicting marine debris in various environments (beaches, underwater, along riverways, and on the sea surface), this technology can be applied to other monitoring approaches, for example monitoring of plastic on or in the water.

The successful prototype demonstrates that this technology has the potential to allow researchers and citizen scientists to update existing databases of plastic marine debris (such as the MDMAP database) in a way that is possibly faster, cheaper, more consistent, and more precise than existing methods, thereby improving the accuracy of current estimates of the amount and type of shoreline marine debris.
In the future, I anticipate that this technology will be implemented in a “human in the loop” system in which the object detection component interacts with the person performing the beach survey. Such a system would be able to continuously improve in accuracy as it is used in the field, because the model could be updated by the judgments of the user. For example, the user would aim the device at an object, the model would detect “bottle,” and the human would verify “yes” or “no.” These human judgments would then be fed back into the system. In this respect, a marine debris recognition system would be very different from an object detection system for an autonomous vehicle. The latter is required to demonstrate extremely high precision and recall, because failing to detect a pedestrian or a stop sign has severe consequences, and furthermore the human driver is not helping to improve the system by telling it “this is a stop sign” or “that was a pedestrian.”

Future work on this project should include further data annotation and training to improve the precision and recall of the system, the addition of more classes of marine debris beyond the current eight, and continued development of the smartphone app for Android and iPhone, including its “human in the loop” components. The smartphone app should also undergo field testing under real-life beach survey conditions such as the ones at Kamilo Point.

With respect to adding new classes of marine debris, two obvious additions are plastic utensils and straws. Both of these classes are found in the top ten list of surveyed plastic debris items on global surveys (Ocean Conservancy, 2019) and locally in Hawai‘i (Blickley et al., 2016). Furthermore, photos of plastic utensils and straws on beaches are readily available on Instagram, and it is likely that both object classes would perform well, since they have unique, defined shapes (like bottles, bottlecaps, hagfish traps, and oyster spacers, all of which were detected well).
Finally, it is likely that there are several ways to improve the precision and recall of the system beyond acquiring more data. For example, it is likely that better performance could be achieved by experimentally fine-tuning the Darknet configuration parameters (as was done by Fulton et al. 2018) instead of relying on the default parameters.
Table 1. Comments in MDMAP database indicating that data is not recorded or incorrectly recorded

<table>
<thead>
<tr>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time not recorded on datasheet. entered 5AM</td>
</tr>
<tr>
<td>Time not noted on data sheet</td>
</tr>
<tr>
<td>No time recorded. 5AM not accurate</td>
</tr>
<tr>
<td>Time was not recorded on data sheet. Time put in only to allow survey to be created.</td>
</tr>
<tr>
<td>Time And Measurements for the large debris items were made up when entering this data to make it able to be entered</td>
</tr>
<tr>
<td>Front page missing from datasheet, guessed on timing and beach width according to past data sheets. low tide is 3.1'</td>
</tr>
<tr>
<td>Volunteer did not record.</td>
</tr>
<tr>
<td>Incorrectly categorized plastic fragments in the &quot;Plastic-Other&quot; category. Will correct for next survey.</td>
</tr>
<tr>
<td>Lots of lumber at this site. students do not count and measure due to the amount and lack of time</td>
</tr>
<tr>
<td>Forgot large items sheet - see description in notes section of field sheet.</td>
</tr>
<tr>
<td>Last months debris (6/14/2014) was accidentally discarded before I could categorize it.</td>
</tr>
<tr>
<td>Due to miscommunication, marine debris was picked up and surveyed buy another volunteer on 8/04/2015. The volunteers share this site.</td>
</tr>
<tr>
<td>Data sheets water-damaged; info transcribed from field notebook</td>
</tr>
</tbody>
</table>
Table 2. Comments in MDMAP database indicating data is not being counted

<table>
<thead>
<tr>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>So much plastic that it was not counted... estimated by garbage bag.</td>
</tr>
<tr>
<td>As for first section, there is too much plastic to count.</td>
</tr>
<tr>
<td>Continuation of Heavy D - very heavy with debris. Too much plastic to count.</td>
</tr>
<tr>
<td>This is the last part of the &quot;Heavy D&quot; accumulation survey. As for the rest, it was inundated by foam pieces, and uncountable. Bag count only.</td>
</tr>
<tr>
<td>Plastic is uncountable, so estimate based on number of bags collected</td>
</tr>
<tr>
<td>There is so much lumber at this site, it is too much to measure and count for the amount of time the students have.</td>
</tr>
<tr>
<td>We only counted floats that were whole (no partial pieces, that would have been too many to count).</td>
</tr>
<tr>
<td>Note that no lumber data is collected at this site due to the large numbers of lumber items and limited willingness from this group.</td>
</tr>
</tbody>
</table>
Table 3. Comments in MDMAP database indicating that data entry was delayed long past date of survey

<table>
<thead>
<tr>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data entry delayed due to amount of debris collected, volunteer dropped 3 bags of debris at OCNMS. 3 volunteers then took 1 hour to sort the debris (this effort should be added to the survey time in the field)</td>
</tr>
<tr>
<td>Data entered in 6/2019 so not all details recalled</td>
</tr>
<tr>
<td>Data being entered in june 2019 so not every detail recalled, .... and large items / photos will be uploaded at a later date.</td>
</tr>
<tr>
<td>Data being entered in june 2019 so not everything in memory, .... will upload large debris / survey photos later.</td>
</tr>
<tr>
<td>Data entered in june 2019 so some of the details are missing, ... will add in any large debris / photos at a later time just wanna get this data entered ASAP!</td>
</tr>
<tr>
<td>Data entered in 4/2019 ... photos to be uploaded later if possible. Yes, there were likely large debris items (besides the larger fragments listed in the custom section) but those will have to be uploaded at a later date. Memory of cleanup / debris survey is limited.</td>
</tr>
<tr>
<td>Found a datasheet with extra #s from another &quot;bag&quot; that was not included in the initial debris survey.... added them in here in june 2019.</td>
</tr>
</tbody>
</table>
Table 4. Definitions of the nine major object classes in alphabetical order

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bag</strong></td>
<td>Single-use plastic bags including grocery bags, trash bags, and plastic wrapping materials</td>
</tr>
<tr>
<td><strong>bottle</strong></td>
<td>Clear plastic single-use beverage bottles</td>
</tr>
<tr>
<td><strong>bottlecap</strong></td>
<td>Caps for single-use beverage bottles</td>
</tr>
<tr>
<td><strong>buoy</strong></td>
<td>Dark or brightly-colored ball-shaped floatation devices, typically for fishing gear</td>
</tr>
<tr>
<td><strong>container</strong></td>
<td>Jugs and other opaque sealed containers for non-beverage liquids such as bleach, shampoo, and petroleum products</td>
</tr>
<tr>
<td><strong>hagfish trap</strong></td>
<td>Black cone-shaped funnel traps</td>
</tr>
<tr>
<td><strong>net</strong></td>
<td>Fishing nets, fishing line, and ropes</td>
</tr>
<tr>
<td><strong>oyster spacer</strong></td>
<td>Black or green tubes used for oyster harvesting</td>
</tr>
<tr>
<td><strong>other</strong></td>
<td>All other plastic objects, including balloons, fragments, lighters, syringes, toothbrushes, and toys, among many others</td>
</tr>
</tbody>
</table>
Table 5. Hashtags of photos scraped from the Instagram photo sharing service

<table>
<thead>
<tr>
<th>#2minutbeachclean</th>
<th>#beachcleanup</th>
<th>#beachcleanups</th>
<th>#beachlitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>#beachplastic</td>
<td>#beachtrash</td>
<td>#beatplasticpollution</td>
<td>#breakfreefromplastic</td>
</tr>
<tr>
<td>#marinedebris</td>
<td>#oceanlitter</td>
<td>#oceanplastic</td>
<td>#oceantrash</td>
</tr>
<tr>
<td>#plasticfreeocean</td>
<td>#plasticfreeoceans</td>
<td>#plasticlitter</td>
<td>#plasticocean</td>
</tr>
<tr>
<td>#plasticpollution</td>
<td>#plasticpollutioncoalition</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6. Copyright designations applied to the photos in the dataset

<table>
<thead>
<tr>
<th>Copyright</th>
<th>Total</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Images</td>
<td>607</td>
<td>Photos taken from Google searches</td>
</tr>
<tr>
<td>Instagram</td>
<td>3,606</td>
<td>Photo scraped from Instagram</td>
</tr>
<tr>
<td>Flickr</td>
<td>106</td>
<td>Photo downloaded from Flickr</td>
</tr>
<tr>
<td>HWF</td>
<td>50</td>
<td>Photo owned by Hawaii Wildlife Fund</td>
</tr>
<tr>
<td>LS</td>
<td>1095</td>
<td>Photo owned by Leah Sherwood</td>
</tr>
</tbody>
</table>
Table 7. Overall (not class-specific) results from Darknet’s built-in map command

<table>
<thead>
<tr>
<th>Platform</th>
<th>mAP score</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 Score</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>Avg IoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLOv3</td>
<td>12.16%</td>
<td>0.39</td>
<td>0.03</td>
<td>0.08</td>
<td>236</td>
<td>364</td>
<td>5068</td>
<td>29.16%</td>
</tr>
<tr>
<td>YOLOv3 + transfer learning</td>
<td>48.35%</td>
<td>0.59</td>
<td>0.37</td>
<td>0.45</td>
<td>1942</td>
<td>1377</td>
<td>3362</td>
<td>44.31%</td>
</tr>
<tr>
<td>YOLOv3 + transfer learning 608*608</td>
<td>52.38%</td>
<td>0.52</td>
<td>0.44</td>
<td>0.47</td>
<td>2325</td>
<td>2181</td>
<td>2979</td>
<td>38.89%</td>
</tr>
<tr>
<td>Tiny YOLO</td>
<td>37.14%</td>
<td>0.55</td>
<td>0.20</td>
<td>0.30</td>
<td>1073</td>
<td>862</td>
<td>4231</td>
<td>41.22%</td>
</tr>
<tr>
<td>Tiny YOLO + transfer learning</td>
<td>39.92%</td>
<td>0.51</td>
<td>0.26</td>
<td>0.35</td>
<td>1385</td>
<td>1310</td>
<td>3919</td>
<td>38.35%</td>
</tr>
<tr>
<td>Tiny YOLO + transfer learning 256*256</td>
<td>30.56%</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Tiny YOLO + transfer learning 608*608</td>
<td>40.91%</td>
<td>0.48</td>
<td>0.30</td>
<td>0.37</td>
<td>1577</td>
<td>1722</td>
<td>3727</td>
<td>35.56%</td>
</tr>
</tbody>
</table>
Table 8. Size-specific mAP scores with number of photos and objects when training and validating at Darknet’s default image size of 416*416 when the validation dataset is restricted to only small, medium or large objects. The mAP score for small objects at 608*608 is also shown.

<table>
<thead>
<tr>
<th></th>
<th>Small objects</th>
<th>Small objects</th>
<th>Medium objects</th>
<th>Large objects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>416*416</td>
<td>608*608</td>
<td>416*416</td>
<td>416*416</td>
</tr>
<tr>
<td>mAP score</td>
<td>44.79%</td>
<td>48.70%</td>
<td>63.66%</td>
<td>60.09%</td>
</tr>
<tr>
<td>Number of photos</td>
<td>386</td>
<td>386</td>
<td>294</td>
<td>117</td>
</tr>
<tr>
<td>Number of objects</td>
<td>2,242</td>
<td>2,242</td>
<td>326</td>
<td>118</td>
</tr>
</tbody>
</table>
Table 9. Evaluation scores from the map command for YOLOv3 without transfer learning when trained and validated at YOLO’s default image size 416*416.

<table>
<thead>
<tr>
<th>Class</th>
<th>Training annotations</th>
<th>Validate annotations</th>
<th>AP$_{50}$</th>
<th>Precision</th>
<th>Recall</th>
<th>IoU</th>
<th>TP</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>bag</td>
<td>1473</td>
<td>342</td>
<td>9.78%</td>
<td>0.40</td>
<td>0.05</td>
<td>0.31</td>
<td>16</td>
<td>24</td>
</tr>
<tr>
<td>bottle</td>
<td>3464</td>
<td>799</td>
<td>11.20%</td>
<td>0.36</td>
<td>0.03</td>
<td>0.26</td>
<td>25</td>
<td>44</td>
</tr>
<tr>
<td>bottlecap</td>
<td>2932</td>
<td>698</td>
<td>7.02%</td>
<td>0.46</td>
<td>0.03</td>
<td>0.36</td>
<td>19</td>
<td>22</td>
</tr>
<tr>
<td>buoy</td>
<td>981</td>
<td>156</td>
<td>9.80%</td>
<td>0.25</td>
<td>0.03</td>
<td>0.20</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>container</td>
<td>1267</td>
<td>279</td>
<td>15.58%</td>
<td>0.33</td>
<td>0.10</td>
<td>0.25</td>
<td>28</td>
<td>56</td>
</tr>
<tr>
<td>hagfish trap</td>
<td>851</td>
<td>139</td>
<td>13.10%</td>
<td>0.44</td>
<td>0.05</td>
<td>0.33</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>net</td>
<td>2949</td>
<td>640</td>
<td>18.84%</td>
<td>0.52</td>
<td>0.12</td>
<td>0.39</td>
<td>74</td>
<td>67</td>
</tr>
<tr>
<td>other</td>
<td>8508</td>
<td>2124</td>
<td>11.56%</td>
<td>0.31</td>
<td>0.03</td>
<td>0.22</td>
<td>56</td>
<td>124</td>
</tr>
<tr>
<td>oyster spacer</td>
<td>379</td>
<td>90</td>
<td>12.50%</td>
<td>0.54</td>
<td>0.08</td>
<td>0.43</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>
Table 10. Evaluation scores from the map command for YOLOv3 with transfer learning when trained and validated at YOLO’s default image size 416*416. This table corresponds to the precision-recall curves in Figure 5.

<table>
<thead>
<tr>
<th>Class</th>
<th>Training annotations</th>
<th>Validate annotations</th>
<th>AP$_{50}$</th>
<th>Precision</th>
<th>Recall</th>
<th>IoU</th>
<th>TP</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>bag</td>
<td>1473</td>
<td>342</td>
<td>38.60%</td>
<td>0.54</td>
<td>0.37</td>
<td>0.41</td>
<td>127</td>
<td>109</td>
</tr>
<tr>
<td>bottle</td>
<td>3464</td>
<td>799</td>
<td>50.31%</td>
<td>0.64</td>
<td>0.43</td>
<td>0.48</td>
<td>347</td>
<td>195</td>
</tr>
<tr>
<td>bottlecap</td>
<td>2932</td>
<td>698</td>
<td>51.31%</td>
<td>0.63</td>
<td>0.47</td>
<td>0.48</td>
<td>325</td>
<td>190</td>
</tr>
<tr>
<td>buoy</td>
<td>981</td>
<td>156</td>
<td>41.34%</td>
<td>0.60</td>
<td>0.42</td>
<td>0.48</td>
<td>65</td>
<td>43</td>
</tr>
<tr>
<td>container</td>
<td>1267</td>
<td>279</td>
<td>39.20%</td>
<td>0.55</td>
<td>0.36</td>
<td>0.44</td>
<td>101</td>
<td>83</td>
</tr>
<tr>
<td>hagfish trap</td>
<td>851</td>
<td>139</td>
<td>77.95%</td>
<td>0.89</td>
<td>0.75</td>
<td>0.68</td>
<td>104</td>
<td>13</td>
</tr>
<tr>
<td>net</td>
<td>2949</td>
<td>640</td>
<td>45.60%</td>
<td>0.69</td>
<td>0.43</td>
<td>0.52</td>
<td>272</td>
<td>123</td>
</tr>
<tr>
<td>other</td>
<td>8508</td>
<td>2124</td>
<td>30.78%</td>
<td>0.48</td>
<td>0.25</td>
<td>0.36</td>
<td>548</td>
<td>597</td>
</tr>
<tr>
<td>oyster spacer</td>
<td>379</td>
<td>90</td>
<td>60.11%</td>
<td>0.69</td>
<td>0.59</td>
<td>0.52</td>
<td>53</td>
<td>24</td>
</tr>
</tbody>
</table>
Table 11. Evaluation scores from the map command for YOLOv3 with transfer learning when trained at YOLO’s default image size 416*416 and validated at 608*608.

<table>
<thead>
<tr>
<th>Class</th>
<th>Training annotations</th>
<th>Validate annotations</th>
<th>AP$_{50}$</th>
<th>Precision</th>
<th>Recall</th>
<th>IoU</th>
<th>TP</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>bag</td>
<td>1473</td>
<td>342</td>
<td>38.11%</td>
<td>0.39</td>
<td>0.39</td>
<td>0.30</td>
<td>135</td>
<td>208</td>
</tr>
<tr>
<td>bottle</td>
<td>3464</td>
<td>799</td>
<td>56.09%</td>
<td>0.63</td>
<td>0.53</td>
<td>0.48</td>
<td>424</td>
<td>244</td>
</tr>
<tr>
<td>bottlecap</td>
<td>2932</td>
<td>698</td>
<td>59.27%</td>
<td>0.64</td>
<td>0.55</td>
<td>0.50</td>
<td>387</td>
<td>221</td>
</tr>
<tr>
<td>buoy</td>
<td>981</td>
<td>156</td>
<td>42.71%</td>
<td>0.49</td>
<td>0.42</td>
<td>0.40</td>
<td>66</td>
<td>68</td>
</tr>
<tr>
<td>container</td>
<td>1267</td>
<td>279</td>
<td>42.43%</td>
<td>0.45</td>
<td>0.44</td>
<td>0.36</td>
<td>122</td>
<td>148</td>
</tr>
<tr>
<td>hagfish trap</td>
<td>851</td>
<td>139</td>
<td>80.46%</td>
<td>0.82</td>
<td>0.86</td>
<td>0.63</td>
<td>119</td>
<td>27</td>
</tr>
<tr>
<td>net</td>
<td>2949</td>
<td>640</td>
<td>44.37%</td>
<td>0.49</td>
<td>0.45</td>
<td>0.35</td>
<td>285</td>
<td>291</td>
</tr>
<tr>
<td>other</td>
<td>8508</td>
<td>2124</td>
<td>33.79%</td>
<td>0.43</td>
<td>0.33</td>
<td>0.32</td>
<td>719</td>
<td>947</td>
</tr>
<tr>
<td>oyster spacer</td>
<td>379</td>
<td>90</td>
<td>74.23%</td>
<td>0.72</td>
<td>0.76</td>
<td>0.53</td>
<td>68</td>
<td>27</td>
</tr>
</tbody>
</table>
Table 12. Evaluation scores from the map command for Tiny YOLO with transfer learning when trained and validated at YOLO’s default image size 416*416.

<table>
<thead>
<tr>
<th>Class</th>
<th>Training annotations</th>
<th>Validate annotations</th>
<th>AP$_{50}$</th>
<th>Precision</th>
<th>Recall</th>
<th>IOU</th>
<th>TP</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>bag</td>
<td>1473</td>
<td>342</td>
<td>24.14%</td>
<td>0.47</td>
<td>0.20</td>
<td>0.35</td>
<td>67</td>
<td>75</td>
</tr>
<tr>
<td>bottle</td>
<td>3464</td>
<td>799</td>
<td>40.32%</td>
<td>0.61</td>
<td>0.33</td>
<td>0.45</td>
<td>262</td>
<td>171</td>
</tr>
<tr>
<td>bottlecap</td>
<td>2932</td>
<td>698</td>
<td>41.46%</td>
<td>0.61</td>
<td>0.36</td>
<td>0.46</td>
<td>254</td>
<td>164</td>
</tr>
<tr>
<td>buoy</td>
<td>981</td>
<td>156</td>
<td>28.11%</td>
<td>0.62</td>
<td>0.23</td>
<td>0.46</td>
<td>36</td>
<td>22</td>
</tr>
<tr>
<td>container</td>
<td>1267</td>
<td>279</td>
<td>27.81%</td>
<td>0.51</td>
<td>0.18</td>
<td>0.41</td>
<td>49</td>
<td>48</td>
</tr>
<tr>
<td>hagfish trap</td>
<td>851</td>
<td>139</td>
<td>74.05%</td>
<td>0.78</td>
<td>0.75</td>
<td>0.57</td>
<td>104</td>
<td>30</td>
</tr>
<tr>
<td>net</td>
<td>2949</td>
<td>640</td>
<td>33.17%</td>
<td>0.62</td>
<td>0.25</td>
<td>0.44</td>
<td>158</td>
<td>95</td>
</tr>
<tr>
<td>other</td>
<td>8508</td>
<td>2124</td>
<td>21.78%</td>
<td>0.37</td>
<td>0.18</td>
<td>0.27</td>
<td>399</td>
<td>688</td>
</tr>
<tr>
<td>oyster spacer</td>
<td>379</td>
<td>90</td>
<td>67.82%</td>
<td>0.77</td>
<td>0.62</td>
<td>0.59</td>
<td>56</td>
<td>17</td>
</tr>
</tbody>
</table>
Table 13. Built-in Darknet commands for training, testing, validation, and evaluation

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>train</td>
<td>Train a new CNN.</td>
</tr>
<tr>
<td>test</td>
<td>Test a CNN on a single image, producing a new image with bounding boxes drawn around the identified objects and a confidence score for each object. This command accepts a minimum confidence threshold as an argument. The default confidence threshold is 0.25.</td>
</tr>
<tr>
<td>valid</td>
<td>Validate a CNN on a set of unseen images (the validation dataset), producing bounding box coordinates and confidence scores for each object in the validation dataset. This command is used to generate the data used to plot the precision-recall curves.</td>
</tr>
<tr>
<td>map</td>
<td>Compute the individual AP_{50} scores (average precision using the standard IoU threshold of 0.5) for each object class, as well as the overall mAP (mean average precision) for the validation dataset. This command also calculates the per-class and average IoU, Precision, Recall, and F1 scores at the default confidence threshold of 0.25.</td>
</tr>
</tbody>
</table>
Figure 1. Bounding boxes generated by YOLO at different confidence thresholds. (a) original photo; (b) YOLO output at confidence threshold .005; (c) YOLO output at confidence threshold .05; and (d) YOLO output at confidence threshold .5.
Figure 2. Examples of photos of the nine major classes of marine debris objects used for training.
Figure 3. Hagfish trap in bounding box drawn in purple using RectLabel; blue lines are crosshairs
Figure 4. Total number of annotations of objects in each class and total number photos in which at least one object of that class is annotated (note the log scale)
Figure 5. Precision-recall curves for each of the nine object classes in the YOLOv3 with transfer learning platform. The y-axis shows precision and the x-axis shows recall. In each plot, the diamond indicates the precision and recall when confidence threshold is set to 0.25. Note that in some cases the diamond is not exactly on the curve due to slight differences between the way the diamond is calculated (in C by the Darknet map command) and the way the curves are calculated (in Python by the program reval_voc_py3.py).
Figure 6. Screenshot of the prototype Plastic Marine Debris Detector smartphone app
Appendix A: Darknet commands

Darknet comes with several built-in commands (Table 13).

map command

To compute the precision, recall, and related metrics, I ran Darknet’s built-in map command on the validation dataset under the four different platforms (YOLOv3 and Tiny YOLO with and without transfer learning). I ran the map command with the argument `--points 11`, which specifies that the AP is to be computed as the mean of precision values at a set of 11 equally spaced recall levels \([0,0.1,…,1]\) (that is, 0 to 1 at step size of 0.1), as done in the 2007 VOC competition (Everingham et al., 2010). In the class-specific results given in Chapter 4, AP indicates the AP score calculated using the standard IoU threshold of 0.5, TP is the number of true positives, and FP is the number of false positives.

valid command

The Darknet valid command outputs bounding box coordinates and confidence scores for each object in the validation dataset that was detected. I fed the output from the valid command to the python program reval_voc_py3.py written by Yaping Sun in order to draw the precision-recall curves (Figure 5).
Literature Cited


