Object Detection in an Aquatic Environment

Sayan Ghosh sayghosh@umich.edu
Michael Rakowiecki mrako@umich.edu
Atishay Singh atishays@umich.edu
Hao Wang haowwang@umich.edu
Michael Wolf mjwolf@umich.edu

Abstract

The Michigan Mars Rover Team, a student-run design team at the University of Michigan, is interested in overhauling their computer vision to prevent the rover from colliding with obstacles. We apply a Faster-RCNN neural network using Facebook's Detectron2 to detect obstacles in an aquatic environment, which has many similarities to the desert environment that their rovers operate in. We also experiment with different methods of data augmentation to examine how they affect the performance of the obstacle detection model.

A video summary can be found here.

1. Introduction

Obstacle avoidance is of particular importance to the Mars Rover Team. Their competition, the University Rover Challenge, has a task focusing on autonomous navigation, in which the rover must autonomously navigate to a set of GPS waypoints and find nearby markers. It is easy for the rover to become stuck or or damaged if it collides with an obstacle. In fact, in the 2018 University Rover Challenge competition, the rover was unable to complete the given task because it became lodged in a small bush that it failed to detect.

Some of the challenges posed by this problem included the lack of suitable training data and in particular, training data that would include some of the same challenges that an obstacle detection system on a rover would have. While object detection datasets such as COCO [1] exist and are well-documented in terms of benchmarks, they do not provide the challenges that a rover's obstacle detection system would have to face, such as motion blur and a variety of small objects such as rocks that can clutter the view but may not be the relevant objects of interest. In addition, it takes much effort to create a sizable dataset for object detection, especially with regards to annotating bounding boxes. Thus, we experimented with a variety of data augmentation techniques that would in theory allow our model to be more robust to variations and distortions in the data it is fed.

2. Background

Deep learning models for visual tasks have achieved tremendous success and continue to improve in the past 8 years. The rise of computation platforms built on GPUs has facilitated the growth of deep learning model complexity [2] [3], which plays a critical role in the improving performance of the models. Another important factor is the increasingly large volume of diverse data, and more high quality data can improve the performance of the model. Sun et al. [4] has shown that the performance of model in vision tasks improve logarithmically with the volume of data. While collecting and organizing high quality data are not always feasible, the alternative is to employ data augmentation to generate more data. Data augmentation is one of the most popular techniques used to reduce overfitting of deep learning models [5], and many data augmentation techniques have been proposed. They can be roughly divided into two categories: traditional image manipulation and policy learning [6]. Even though traditional image manipulation techniques are not as effective in practice, but they do improve the performance of the model in image classification as shown in [7].

3. Dataset Selection

Since datasets that closely resemble desert environment are not readily available, we decided to train and evaluate our model on the Multi-modal Marine Obstacle Detection Dataset 2 (MODD2) [8], which consists of 11675 images taken in aquatic settings and annotated for object detection and semantic segmentation application. MODD2 largely resembles our expected desert landscape in the sense that there is sky in the background, homogeneous landscape in front (water in this case), and obstacles that appear on/in this homogeneous landscape. We expect that a model that performs well on this dataset should be similarly successful on the desert once retrained with a desert dataset.

The MODD2 dataset consists of images taken from unmanned service vehicles (USVs) in a variety of conditions and with a variety of disrupting factors including motion blur and a variety of small obstacles. All of these pose unique challenges to this dataset, and in particular, are relevant to the types of challenges that would be present in the situation where obstacle detection would have to be conducted on a portable rover. The obstacles in the dataset are annotated by rectangular bounding boxes (with no rotation) and can be classified into 2 classes: small and large. Figure 1 depicts an example from the training split of the dataset, including bounding box annotations.

In our use of the MODD2 dataset, we were faced with a few challenges. When we reformatted the data to the Detectron2 required format for their data loaders, we had an issue where the water edge segmentation would cause errors in the COCO evaluation code. For this reason we removed that water edge information from our data when loading it. It was also realized during the reformatting of the dataset that there was no class information associated with the annotations. Therefore, we only used one class for the instances. Additionally encountered was an issue where certain annotations would exist without a corresponding image for said annotation. There seemed to be on the order of 10 annotations with this issue, so we removed them. Later, when creating augmentations of the data for our training set, we noticed that the bounding boxes would not line up correctly on a small subset of images. Because this occurred on a very small subset, we realized that there was another issue in the dataset. Some images had duplicate filenames, causing the augmentation functions to associate the wrong annotation with the wrong image. To fix this issue, we removed the images with duplicate file names from the augmented dataset.

Figure 1. Example labeled image from MODD2 Dataset

4. Experimental Setup and Procedures

In our experiment we used a pre-trained model from the Detectron2 model zoo, and trained it on the MODD2 dataset with different data augmentation techniques.

4.1. Detectron2 Model

We built off of the Detectron2 model [9] to develop our object detection algorithm. In particular, we used the Faster RCNN [10] model provided, due to it having a relatively fast runtime compared to many earlier object detection systems. Faster RCNN is similar to the Fast RCNN [11], with its key feature being a region proposal network, which is itself a fully convolutional network whose purpose is to detect regions of interest in the input image. This makes the region proposal step much faster compared to that of a Fast RCNN.

In order to use a custom dataset with Detectron2, we developed a script to manipulate the dataset into the required format. Subsequently, data augmentation datasets, generated from MODD2, were developed and incorporated into this project.

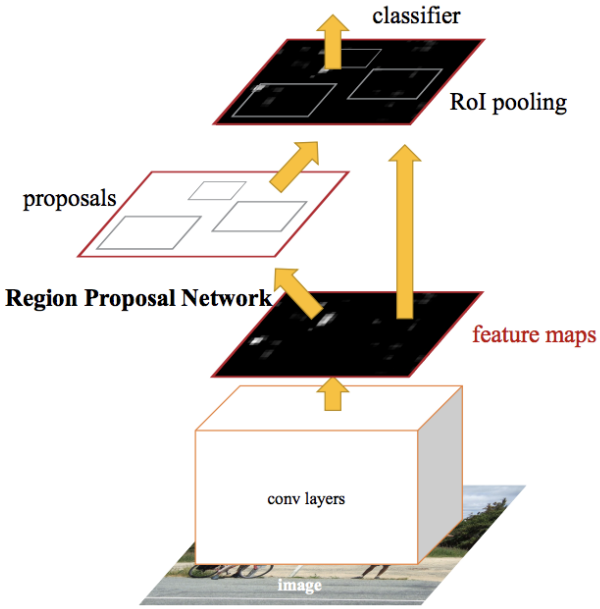

Figure 2. FasterRCNN architecture (from [10])

4.2. Training details

To create our baseline model, we trained the "rectified and undistorted images" from the MODD2 dataset on the pre-trained Faster RCNN Resnet-101. We used 800 iterations, a base learning rate of 5×10^{-4}, and a momentum of 0.9. The model took about 4 minutes and 40 seconds to train on Colab's K80 GPU. The Loss vs. Iteration graph for this model can be seen in Figure 3. The dataset annotations

did not have any class information as the website for the dataset suggested, so all the models were trained with only one object class, 'obstacle', included.

Our initial training set without augmented data included 17807 instances. The Detectron2 trainer was set to remove images from training that included no obstacles within them. This reduced the initial training set size from 7516 images to 6252 images. In our first augmented training set, we included images that were mirrored about the vertical axis. This increased our dataset size from 7516 to 15016 images. To produce this increase, we mirrored every image in our dataset that did not have a duplicate file name. With this augmented dataset, we had essentially doubled the number of instances to train on, from 17807 to 35614.

Our next augmented training set still included the original data and mirrored data, as well as a few new augmentations. The new augmented data was created by randomly changing brightness and contrast of the dataset. The changes were kept within a reasonable limit to ensure that we didn't create unrealistic images. We also added translations to our original dataset to create even more augmented images. This augmented training set totalled to 36738 images, and had 79303 instances. The size of the dataset was quadrupled when using all of these augmentations.

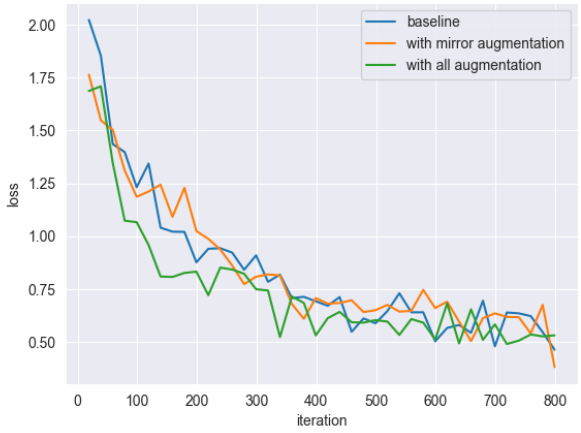

Figure 3. Iterations vs. loss for all models

4.3. Data Augmentation Methods

Data augmentation has been used by researchers [12] in order to increase model performance by increasing the amount of training data available during training. Data augmentation improves the performance of the model also by reducing overfitting [6]. As noted in [13], data augmentation is particularly relevant in the domain of object detection, as it is costly to create large datasets. At the same time, the complex deep models that are used for obstacle detection perform best when trained on large amounts of data.

Figure 4. Example of image classification from baseline model

The domain of object detection poses several additional challenges when augmenting a training set, as one must also be cognizant of the effect of transforming an image on the corresponding bounding box. We therefore experimented with a few different types of transforms to generate more training data, including some that affect the dimensions and positions of the bounding boxes. The purpose of employing different augmentation techniques is two-fold. First, we wanted to improve the performance of the model by increasing the volume of the training set. Next, we wanted to improve the robustness of the model in reaction to changes of images' lighting condition, directionality, etc.

4.3.1 Translation

The images are translated in both the horizontal and vertical directions. In order to incorporate more varieties into the augmentation dataset, the spatial extent of the translations are randomly and independently generated in the interval $[-100, 100]$. In the case where the obstacle(s) are translated out of the frame, the image is not included in the augmentation dataset. It is worthwhile to mention that noticeable artifacts are introduced given the fact that the images are generated by rolling, where the pixels pushed out of the frame are stitched to the other end of the image.

Figure 5. Image translation, original (left), shifted (right).

4.3.2 Mirroring

To ensure that the network does not rely on any sort of directionality present among obstacles in the dataset, we create a set of augmented images for which each is a mirror of an image in the original dataset, mirrored about the central vertical axis. The figure below shows the mirrored image and bounding box.

Figure 6. Image mirroring, original (left), mirrored (right).

4.3.3 Contrast

We randomly and independently transformed the training images to 2 levels of 0.5 or 1.5 times the original contrast value. Transforming the images into two different levels can help the model to learn to recognize two extreme contrast level and hence become more robust to change in contrast. This would help in cases where there is heavy fog or otherwise low visibility. Examples can be seen below. The transformation were implemented using the Pillow library.

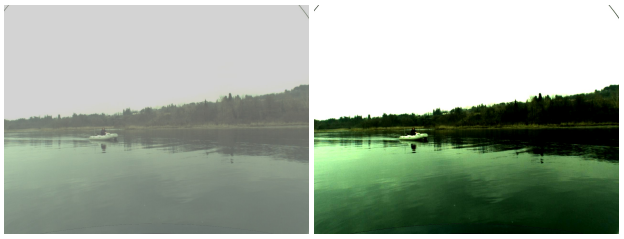

Figure 7. Contrast adjustment, low (left), high (right)

4.4. Brightness

Similarly, we transformed the training images into 2 extreme brightness levels randomly and independently. The purpose of this transformation is to help the model to become more robust in different lighting conditions. The transformations were implemented using the Pillow library.

Figure 8. Brightness adjustment, low (left), high (right)

5. Result and Discussion

The different trained models were evaluated using the MODD2 evaluation metrics. Based on our interpretation of the methods discussed, our model was highly competitive with the other models on their leaderboard.

5.1. Evaluation Method

To evaluate the performance of our dataset, we adopted the scoring metric used in the MODD2 paper and on the MODD2 leaderboard, where they compare various models against each other. As per this metric, to count a detection as a True Positive (TP), an instance detection only has to have an overlap of 0.15 with the ground truth annotation. This minimum overlap corresponds to a minimum Intersection over Union (IoU) score of roughly 0.10. In addition, any detections above the water edge are disregarded and not counted as either true or false positives, since such an obstacle would not . In order to rank the performances of the different models trained on the MODD2 dataset, the F-measure of each model was calculated and listed. F-measure is described as a way to measure a test's accuracy by considering both the precision and the recall (i.e. $F = \frac{2*\text{precision}*\text{recall}}{\text{precision}+\text{recall}}$). It can be calculated using the count of TP, FP, and FN instances, or by the harmonic mean of precision and recall. A demonstration of the equivalence of both methods can be seen in the calculations below.

$$AP = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$
\begin{aligned}
F &= \frac{2 * AP * Recall}{AP + Recall} \\
&= \frac{2TP^2}{TP^2 + TP(FP) + TP(FN) + FP(FN)} \\
&\quad * \frac{TP^2 + TP(FP) + TP(FN) + FP(FN)}{2(TP)^2 + TP(FN) + TP(FP)} \\
&= \frac{2TP^2}{TP(2TP + FN + FP)} \\
&= \frac{2TP}{2TP + FN + FP} \qquad (3)
\end{aligned}
$$

Although the metric used in the paper allows for poor detections, we have decided to try to match it in order to verify our model against the statistics they provide. In our evaluations we use Average Precision and Average Recall, both with an IoU threshold of .10 to calculate the F-measure. To get a better idea of our model's robustness, we also decided to count detections above the water edge as false positives unlike the original MODD2 evaluation. Because we know that an IoU threshold of .10 is low and does not relate well to many other evaluation methods, we kept track of the Average Precision (AP) and Average Recall (AR) scores for IoU thresholds ranging from .10 to .95 with a step of .05. To get these statistics, we adapted both the source code of the Detectron2 COCOEvaluator class and the pycocotools COCOEval class to be able to return any combination of IoU threshold, area range, and max determinations per image values.

5.2. Resulting Statistics

When only using the original training set we had surprisingly good results in comparison to the MODD2 leaderboard. Our F-measure score was .820, which places third compared to other models that have trained on the dataset. The highest F-measure on the leaderboard was .827, and it was achieved by BiSeNet[14]. Our model that was trained with only the mirrored additional augmentation performed even better than this. We achieved an F-measure of .858 as can be seen in Table 2.

Tables 1-3 show our AP and AR values for an IoU threshold of .10, as well as the AP and AR values averaged over the IoU thresholds from .10 to .95. We also have the F-measure for each, which was calculated using the AP and AR values with a .10 IoU threshold. More statistics were recorded for each model, and they can be found on our github.

Metric	Value
AP @[IoU = 0.10:0.95]	0.507
AP @0.1 IoU	0.733
AR @[IoU = 0.10:0.95] (maxDets=100)	0.745
AR @0.1 IoU	0.930
F-Measure @0.1 IoU	0.820

Table 1. Results for baseline model on validation set.

Metric	Value
AP @[IoU = 0.10:0.95]	0.551
AP @0.1 IoU	0.792
AR @[IoU = 0.10:0.95] (maxDets=100)	0.768
AR @0.1 IoU	0.935
F-Measure @0.1 IoU	0.858

Table 2. Results for mirrored data augmentation model on validation set.

Metric	Value
AP @[IoU = 0.10:0.95]	0.497
AP @0.1 IoU	0.711
AR @[IoU = 0.10:0.95] (maxDets=100)	0.771
AR @0.1 IoU	0.931
F-Measure @0.1 IoU	0.806

Table 3. Results for all data augmentation model on validation set.

As shown in the tables, the model trained with the base data augmented with mirrored data performed better than the model trained only with the base data. This is expected because the augmented data has a much larger volume and slightly improved diversity, due to added directionality invariance. Unlike what we expected, the model trained on the base data augmented with all transforms was the worst performing in terms of F-measure. While this result is unexpected, there are a few reasons this drop in performance could have occurred. While loading and training on the dataset, the model we were using detected false duplicates, where similarly named image files were treated as duplicate images despite either being different images entirely or being separate transformations of the same image. We did attempt to resolve this issue, but it is possible that some of the data for the augmented images was either thrown out or incorrectly combined with that of other images. Alternatively, it could be that some of the augmentations added fail to represent plausible or consistent obstacle locations or image conditions. If this is the case, then training on these augmentations would decrease the performance on the validation set, which was free of augmentations.

6. Future Work

While the work done for this project provides a solid basis for a robust object detection model, we have really only tested it on an aquatic dataset so far. However, the purpose of this research was to develop and train a model for use

in a desert-like setting. While the basic architecture of the model and the techniques used in its creation are can be applied to a dataset more in line with our original goals, there is still a lot of work to be done in improving the performance and robustness of the model and adapting it to the desired use case, as described above.

Figure 9. The rover in the desert environment

Our ultimate goal for this project is to prepare ourselves for the future project of training a network to detect obstacles that our rover will encounter during competition in the desert. Now that we have demonstrated the effectiveness of this architecture training on a similar dataset, our next goal is to build our own dataset of images from the rover's operation environment (Figure 9) in and retrain the model on this data.

We also would like to expand upon our data augmentation, exploring the effectiveness of other methods. This will be essential if we decide to generate our own dataset, as such acquiring of enough data to train a model that performs well in real environments is challenging and time-consuming. We are interested in methods such as those introduced by [13], which involve learning the data augmentation strategies based on the data itself. In particular, the problem of selecting an augmentation strategy would be framed as a search problem, starting with a base set of operations such as the ones we implemented, and then combining them probabilistically to learn the best sequence of transformations.

In addition, we plan to eventually modify the network produced here to handle semantic segmentation tasks as well as object detection. The purpose of this would be to allow the rover to identify and distinguish the ground plane from the sky, including cases where the ground is rocky or otherwise not level. Since the rover is expected to autonomously navigate a wide variety of terrain, the ability to identify the ground plane would help the rover identify hilly regions or regions with sharp drop-offs, reducing the likelihood of damage.

7. Acknowledgement

We would like to thank Prof. Justin Johnson and Mr. Mohamed El Banani for their guidance and support in the project. We would also like to thank Google for their donation of Compute Engine hours for our project.

References

[1] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[4] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era, 2017.

[5] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy, 2017.

[6] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 07 2019. Copyright - Journal of Big Data is a copyright of Springer, (2019). All Rights Reserved.; © 2019. This work is published under http://creativecommons.org/licenses/by/4.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2019-07-07.

[7] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017.

[8] Borja Bovcon, Jon Muhovič, Janez Perš, and Matej Kristan. Stereo obstacle detection for unmanned surface vehicles by IMU-assisted semantic segmentation. *Robotics and Autonomous Systems*, 104:1–13, 2018.

[9] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[12] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis.

[13] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. *arXiv preprint arXiv:1906.11172*, 2019.

[14] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *The European Conference on Computer Vision (ECCV)*, September 2018.