

# Detection of Recyclable Objects using MaskRCNN

Brennan Proudfoot  
bwproud at cs.unc.edu      Jay Rao  
jayrao at cs.unc.edu

November 2018

## 1 Introduction

Studies show that the amount of recyclable materials in the U.S. waste system could generate over \$7 billion if they were properly recycled<sup>1</sup>. This poses the incredibly relevant challenge of accurately identifying recyclable material to help prevent the introduction of recyclable materials into the general waste system. We trained a Mask RCNN model with a ResNet-101 backbone to serve as an object detection system that detects, bounds, and segments glass bottles, plastic bottles, and metal cans. While this system could be used solely at a user-level, it is also conceivable that this could be deployed at an enterprise level in a recycling plant for fast sorting and detection of incoming objects.

## 2 Problem

For the aforementioned reasons, we have decided to develop a model that can detect and classify classes of recyclable items. The classes of recyclables that we currently support are plastic bottles, glass bottles, and metal soda cans. The reasoning behind this was that these are three different materials that should ideally be separated as each has a different recycling strategy. For example, when plastic is brought to a recycling center it is first sorted and then shredded<sup>2</sup> while metal is sorted and then melted<sup>3</sup>. In addition, the three classes we chose are similar in the service they provide, so it's likely that people will recycle all three of these items together. Ultimately, our computer vision model looks to cut down on human labor, and increase the amount of material that gets properly recycled.

## 3 Data

### 3.1 Dataset

As training a model from scratch for this task would have taken weeks, we decided to use transfer learning to speed up the development process. To that end, we got the weights of an existing Mask RCNN model trained on COCO, and then trained that model on our three classes. As there is no existing dataset of segmented plastic bottles, glass bottles, and metal soda cans with labels, we had to create our own dataset, using a mix of photos we took ourselves and photos we found online. Our criterion for finding photos included finding photos that had interesting visual qualities with respect to the object that we are trying to detect. These characteristics included occlusions (other objects partially obscuring the object we are trying to detect), out of focus images, crushed bottle and can images, among others.

---

<sup>1</sup><https://www.byui.edu/university-operations/facilities-management/recycling-and-sustainability/recycling-statistics>

<sup>2</sup><https://www.norcalcompactors.net/processes-stages-benefits-plastic-recycling/>

<sup>3</sup><https://www.thebalancesmb.com/an-introduction-to-metal-recycling-4057469>



Figure 1: Sample image that was segmented using VGG Image Annotator that is managed by the University of Oxford.

### 3.2 Preparation and Data Augmentation

Once we found all of the photos we wanted to use, we segmented them, creating a mask around the area of interest, so the Mask RCNN model could learn how to generate masks. As seen in figure 1, we used a open source tool for segmenting images, in this case four metal cans, to provide training and validation data for our model. In the end, we had 100 training images for each class as well as 50 validation images and 50 test images to evaluate the trained model.

To prevent overfitting, we also augmented our data slightly, randomly flipping images 50% of the time, as well as applying a slight Gaussian blur. In addition to making our model more robust, these augmentations helped artificially increase the size of our image set, which was especially important in our case, as we didn't have a very large training set compared to most computer vision training sets.

## 4 Training

To train the model, we used 30 epochs of 100 steps each, where a batch of images was evaluated and loss calculated for the batch at each step. We chose 30 epochs as empirically we did not see much improvement in any successive epochs after 30. At each step, the loss calculated is a composite of many different categories of loss, including rpn\_class\_loss, rpn\_bbox\_loss, mrcnn\_class\_loss, mrcnn\_bbox\_loss, and mrcnn\_mask\_loss. RPN class loss and bounding box loss is the loss associated with the predicted bounding boxes and class prediction within the bounding box by the Region Proposal Network (RPN). MRCNN class loss, bounding box loss, and mask loss is the loss associated with the final refined bounding boxes, masks, and class predictions associated with each bounding box. On Google Colab's 12 core GPU, the model's we created took about 3-4 hours to train, depending on whether we trained just the heads of the networks or on all layers of the model.

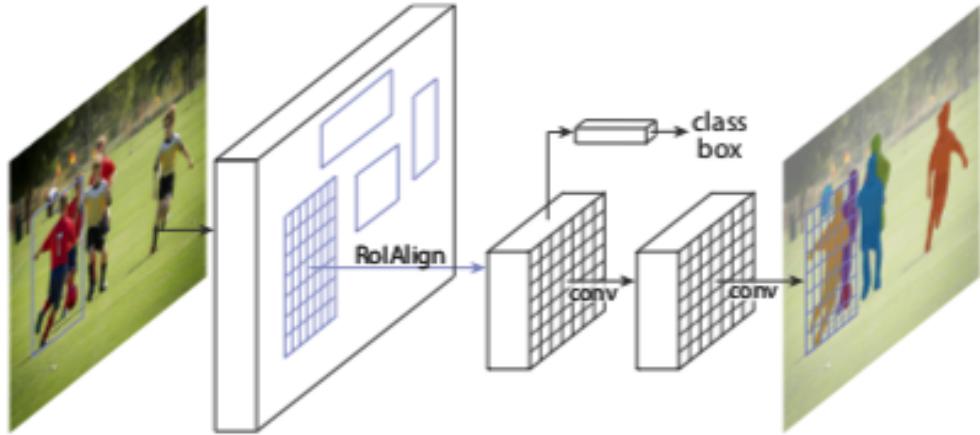


Figure 2: Image from the original MaskRCNN research paper released by Facebook Research. The image shows how regions of interest are both bounded in a box and masked[1].

## 5 Model

### 5.1 Mask RCNN

We chose to use a Mask RCNN for our model. A Mask RCNN is an extension of the Fast RCNN model which identifies Regions of Interest and convolutionally determines bounding boxes and classifications for these regions. The Mask RCNN extends this output to also include a mask for each region of interest [1]. We trained the Mask RCNN through the use of Transfer Learning.

### 5.2 Transfer Learning

The basic idea of Transfer Learning is to take some pre-trained model, which in our case is ResNet-101, and provide it data specific to further train it to something it was not originally trained to do. This allows us to take advantage of the extensive net that already exists and drastically reduces the amount of time we would have had to spend training a model from scratch [3].

We attempted two different types of transfer learning for this model - Fine tuning and Fixed Feature. In Fine Tuning we trained all of the nodes in the existing Neural Net with the new training data and changed weights throughout the model while with Fixed Feature we only changed the weights of a single layer of neurons. We chose two different styles of transfer learning in order to compare and contrast their relative value in the classification problem we were attempting to solve.

### 5.3 Models Trained

We trained four Mask RCNN models to see how altering the number of layers trained and the amount of training data affected accuracy. We had two main classes of models: those that were trained through fine tuning and those that used transfer learning as a fixed feature extractor. As explained previously, that means that half of the models trained every layer in each part of the model (region proposal network, feature pyramid network, ResNet) while the other half (fixed feature) only trained the heads of these networks, using the weights of the pre-trained model for every other layer. In addition, we trained both of these types with a small training set (15 images in each class) and a large training set(100 images in each class) to try to see what effect the training size had on accuracy.

## 5.4 Metrics Used in Evaluation

We use mean average precision (mAP) with an intersection over union (IoU) threshold of .5 to evaluate the performance of the different models. Mean average precision is defined through two different values, precision and recall. Precision is calculated by dividing the number of true positive results obtained by the total number of positive results obtained in general. This includes false positive results. The higher the precision of the model, the closer to 1 this value should be. Recall is calculated by dividing the number of true positive results by the number of correct results obtained overall which include true positives and false negatives. Recall can be thought of as how good your model is at detecting both good and bad results. Mean Average precision couples these two concepts together over a number of epochs to give us a single value from 0 to 1 that defines how accurate our detection was. We used a pre-existing implementation of mAP to calculate this value[2].

Intersection over Union is used to compare the bounding boxes that the model predicts with the ground truth bounding boxes that we define for the images.

## 6 Results

Table 1: Results

Mask RCNN Model	mAP on Val Set	mAP on Test Set
Fine Tuning(Small training set)	.873	.885
Fine Tuning(Large training set)	.845	.905
Fixed Feature(Small training set)	.766	.774
Fixed Feature(Large training set)	.842	.781

The results of calculating the mean average precision of each model on our validation and test image sets are contained in table 1. These results suggest that fine tuning the model definitely improved accuracy over just using the pre-trained model as a fixed feature extractor. The best fixed feature mAP results (those associated with model trained on the large image set) still don't match the worst mAP results of the fine tuned models. It does appear that increasing the size of the training image set increased the accuracy of the fixed feature model, however the same doesn't appear to be true for the fine tuned models. In fact, the fine tuned model on the small dataset actually had the highest overall average mAP between the validation and test image sets at .879.

The result that more training images didn't lead to an increase in mAP seems counter-intuitive, but there could be a few confounding variables that are causing this result. For example, in our large image set, we diversified and expanded the definition of what it meant to be part of each class. Where before we were just using water/beer bottles and wine bottles as examples as glass bottles, we expanded this to include glass liquor bottles which definitely have a different shape to our previous definition of what it meant to be a glass bottle. Adding to this, its possible we didn't update our validation/test set enough to reflect this expansion of the classes. Another possible reason could be that we effectively increased the size of the training set by a factor of 10 but kept the same number of epochs as we had with the small image set.

### 6.1 Results Overview

Overall, the results were very good across the board. The model performed very well at detection, placing tight bounding boxes, classification of the object within the bounding box, and then finally masking of the object. We've included many examples of the results of the model in the Appendix. We initially thought that the model was going to have a tough time telling plastic bottles from glass bottles due to the similarity in their shape, but that didn't seem to be an issue. In addition, the model seemed to deal with occlusion relatively well, which was one of our main goals before we started training.

## 6.2 Limitations



Figure 3: Mask RCNN False Positive

While the model was very good at detecting and classifying our intended targets when they existed, we did find that it had trouble with false positives, especially in the context of objects it had never seen before. Specifically, we found that our model tended to classify dogs and faces as examples of metal cans, likely because we didn't include many negative examples in our training set, and no images in our training set included dogs. This is likely a problem with over-fitting and could be resolved by adding in negative examples and further increasing the size of our training set. We also found that the model performed worse at detection of objects in context which is likely due to the images we used in training (normally a large centered objects with little to no background).

## 7 Conclusion

We believe that this is a good first step in prototyping a useful consumer and enterprise tool for detecting recyclable material through computer vision. Future work would include real time analysis of incoming video data to be able to do sorting in real time at recycling facilities. While our results were promising there is quite a lot of room for improvement, especially with regards to our gathered data in terms of variety as we did have some false positive identifications that could be eliminated with a stronger variety of training images.

## 8 Resources

We used a number of tools to complete this project. In order to gather the images, we used Google image search to find certain images we believed would increase our precision (like 'crushed cans' or 'person drinking glass bottle'). Once we had our images, we used VGG Image Annotator to segment each image and export the masks in a format we could use in training the model.

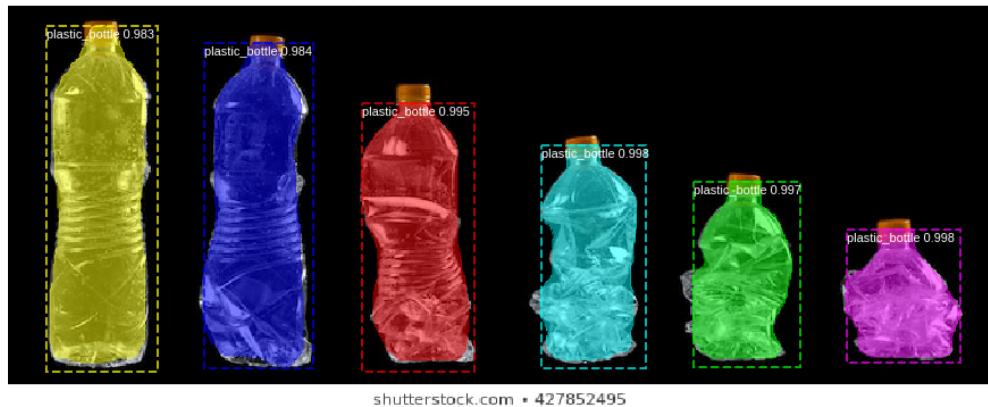
The base Mask RCNN implementation we chose was Matterport's Mask RCNN implementation as it was incredibly well documented, with samples/demos available which showed

how to train on your own data. The other option was Facebook’s new PyTorch Mask RCNN implementation, however this implementation had worse documentation and no examples about how to extend the model to our use case. While Facebook’s implementation is very new (it was released within the past month or so), there have been some preliminary results which say that Facebook’s implementation is slightly more accurate, but it takes longer to train and is slightly slower in the inference stage than Matterport’s implementation. As our starting point in transfer learning, we used a pretrained COCO model.

As we didn’t have a GPU at our disposal, which was very necessary to train our model, we used Google’s Colab software, which is essentially an online jupyter notebook that allows users to use a 12 core GPU for 12 hours each day. This allowed us to train our model in about 6 hours instead of the week to month long time frame it would have taken on a CPU.

## 9 Appendix

[bp!]



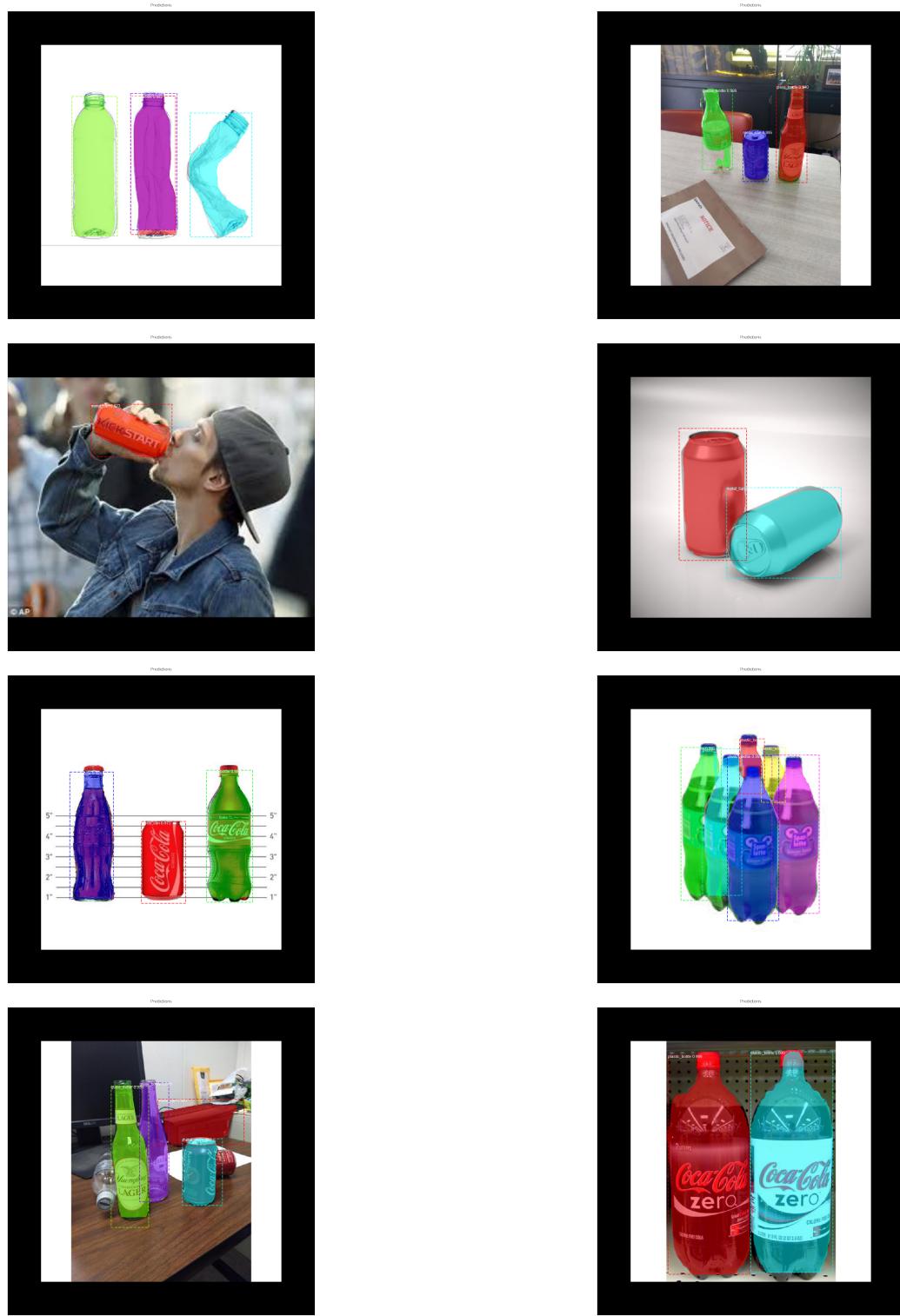


Figure 4: Examples of model output

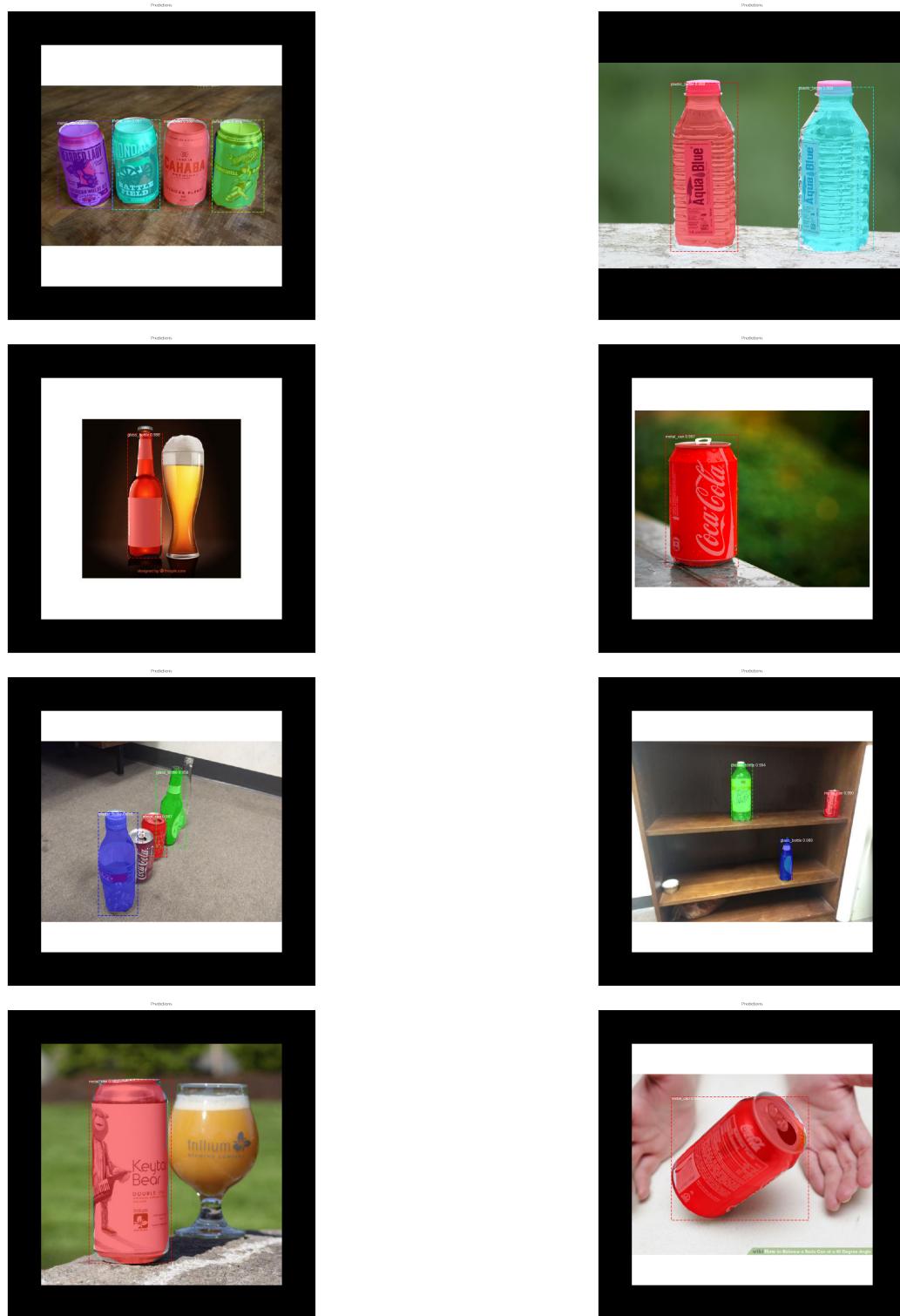


Figure 5: More examples of model output



## References

- [1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.
- [2] Jonathan Hui. mAP (mean average precision) for Object Detection. *Medium*.
- [3] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 242–264. IGI Global, 2010.