Food Waste Segmentation

Links to code and outputs of our model and Mask-RCNN

Ivan Zou and Mohit Garg

1. INTRODUCTION

Every year approximately 22 million pounds of food are wasted in college campus dining halls which equates to roughly 142 pounds of waste per student. These staggering numbers only hint at the massive food waste issue surrounding the United States with about 30-40% of the entire US food supply being thrown down the chute [2].

We believe much of this problem stems from the inability of restaurant kitchens and cafeterias to adapt to the ever-changing demands and palette of the consumers. Like how critical systems in robotics, industrial processes, etc. require some implementation of a controller feedback mechanism for the system's state to adapt and respond to its changing environment, restaurants kitchens and cafeterias need a more consistent, effective, and objective feedback mechanism beyond the irregular and bias consumer review.

To address this need, we've developed a computer vision application to tackle the classic instance segmentation problem but in the context of food waste: detect the frequency of different food waste given a video frame. Consequently, these frequencies will be the indicators that a kitchen needs to adapt and reduce waste. For evaluation of our model to existing instance segmentation solutions, we've fined tuned a state-of-the-art Mask-RCNN algorithm on the same dataset. The results are compared and discussed in section 5 below.

2. DATA COLLECTION

The data collection occurred in the dining halls of Georgia Institute of Technology with permission from the school's police department and dining directors.

During the span of a week, we attached cameras to record the waste being thrown out by the students. From the footage, we've sampled 116 meaningful video frames and labelled them using the MakeSense image annotation software [5]. These will be the images that we'll use to train our own model as well as fine tune Mask-RCNN.

From these images, we've partitioned all the different types of food into 9 classes.

FOOD CLASS	INDEX
Beef	0
Chicken	1
Pizza	2
Pasta	3
Fruit	4
Vegetable	5
Rice (serving)	6
Beans (serving)	7
Bread	8

Table 1. Food classes and their indices used in our study.

Figure 1. Camera setup to collect waste footage.

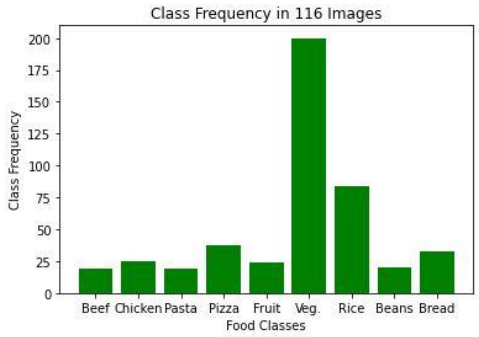

Figure 2. Data frequency distribution of the 116-image dataset.

3. PROPOSED TECHNIQUE

Although a popular choice for image classification problems, convolution neutral networks have since been complicated to extend its capabilities to tasks such as object detection and segmentation, giving rise to Fast-RCNN, YOLO, MASK-RCNN, etc [7, 8].

We've determined that the outputs of many of these highly effective algorithms, the exact pixel locations and bounding boxes of specific objects, have no purpose in our use case of waste detection, let alone the additional computation and time complexities they bring along. We're simply only interested in the frequency of different types of food in each video frame. Additional information like pixel locations and bounding boxes of the food are rather a luxury.

As a result, we've elected to use just use a CNN to detect the frequency of the different classes of food given an input image. More specifically, given an image, our model would output a class frequency vector in a 9-dimension real inner product space.

$$\begin{pmatrix}2\\0\\3\\0\\0\\1\\1\\0\\0\end{pmatrix} = \begin{pmatrix}2\ beef\\0\ chicken\\3\ pizza\\0\ pasta\\0\ vegetable\\1\ fruit\\1\ rice\\0\ bean\\0\ bread\end{pmatrix}$$

The argument is that the CNN would be able generate and classify frequencies of the types of food based on the optimal feature mapping that it learns from the different colors and textures of the types of food in the dataset.

3.1 Preprocessing

Figure 3. Processed images to feed into CNN.

To maintain consistency and eliminate noise such as human shoes, lightening, etc. that might affect CNN feature map, we've blackened every part of the frame that isn't the waste bin using a Hough circle transform so that the model is able to focus on the differences only within the waste bin [4]. Additional preprocessing requirements such as image resizing of our 1080 x 1920 x 3 images were also applied.

3.2 CNN Architecture

In terms of constructing our CNN architecture, we modelled our network after the VGG-16 [8, 11]. The summary of our network is given below.

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
Waste_CNN                                --                        --
├─Conv2d: 1-1                            [1, 8, 538, 918]          224
├─MaxPool2d: 1-2                         [1, 8, 269, 459]          --
├─Conv2d: 1-3                            [1, 16, 267, 457]         1,168
├─MaxPool2d: 1-4                         [1, 16, 133, 228]         --
├─Conv2d: 1-5                            [1, 32, 131, 226]         4,640
├─MaxPool2d: 1-6                         [1, 32, 65, 113]          --
├─Linear: 1-7                            [1, 1024]                 240,681,984
├─Linear: 1-8                            [1, 512]                  524,800
├─Linear: 1-9                            [1, 9]                    4,617
==========================================================================
Total params: 241,217,433
Trainable params: 241,217,433
Non-trainable params: 0
Total mult-adds (M): 631.73
==========================================================================
Input size (MB): 5.96
Forward/backward pass size (MB): 54.82
Params size (MB): 964.87
Estimated Total Size (MB): 1025.65
```

Figure 4. Summary of CNN Architecture.

A highlight is the small kernel and stride length. We cared less about spatial location of the waste within the image and wanted the network to focus more on the subtle details and differences within a specific highly dense region that contained the food waste. We also resized our image as little as possible to preserve as much information as we can, hence the large number of parameters, since we were already working with a small dataset.

A ReLU nonlinearity was applied after every convolution and linear layer both as an activation and to keep the final output layer as nonnegative integers.

$$f(x) = \begin{cases}0, & x < 0\\ x, & x \geq 0\end{cases}$$

For the back propagation of our network, we elected to minimize a mean squared error with L2 regularization.

$$L(x) = \frac{1}{n}\sum_{i=0}^{n-1}(g_i - x_i)^2 + C\sum_j {w_j}^2$$

$$g_i = gold\ label$$

$$x_i = output\ of\ forward\ propogation$$

And lastly since our final output requires non-negative integers, we apply a ReLU and nearest integer rounding.

$$output = ReLU(round(x))$$

3.3 Accuracy Measurement

The final piece of our solution was determining an appropriate way to measure the accuracy of our predicted class frequency vector against the actual class frequency vector. After different attempts, we settled on using the orthogonal projection length. More specifically, given

$$proj_{\vec{v}}\vec{w} = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\|^2}$$

We take whichever class frequency vector with the greater magnitude as the vector to be projected onto. The resulting percentage difference between the projected class frequency vector and the vector that was projected onto is the accuracy.

$$accuracy(x_i, g_i) = \begin{cases} \frac{\vec{x_i} \cdot \vec{g_i}}{\|\vec{x}\|^2} \times 100, & \vec{x_i} \cdot \vec{g_i} > 0 \\ 0, & \vec{x_i} \cdot \vec{g_i} \leq 0 \end{cases}$$

Take for example the following vectors

$$x_i = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, g_i = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

Using our formula, the accuracy of the output vector against the gold label would be 70%.

3.4 Accuracy Measurement Justification

Why is our formula a good accuracy indicator between two vectors? Our reasoning is that we wanted to consider both the angle and magnitude differences between the vectors, both of which play a role in the orthogonal projection of the vectors.

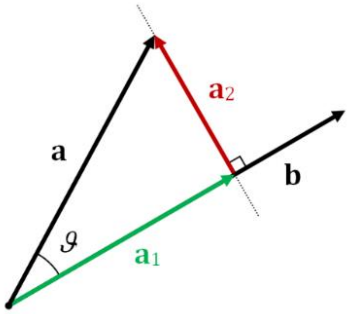

Figure 5. Orthogonal projection of 2 vectors. The percentage difference in magnitude between vector b and a is a good indicator of the difference between the two vectors.

To enforce this idea even more, we went through a series of cases of 2-dimensions vectors to see how the formula would performance against our intuition of the differences between the vectors. With no loss of generality, the same logic applies to higher dimensions.

$$x_i = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, g_i = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, accuracy = 100\%$$

As expected, two vectors that are the same give 100%.

$$x_i = \begin{pmatrix} 0 \\ 4 \end{pmatrix}, g_i = \begin{pmatrix} 0 \\ 3 \end{pmatrix}, accuracy = 75\%$$

Vectors in the same span have the same angle but differ in magnitude so accuracy should still be high if similar magnitude.

$$x_i = \begin{pmatrix} 1000 \\ 1 \end{pmatrix}, g_i = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, accuracy = 0.001\%$$

Vectors in same span but very different magnitudes should have very low percentage.

$$x_i = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, g_i = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, accuracy = 0\%$$

The orthogonal case is the only real flaw as they these two vectors have similar magnitude but different angles. To negate this, we decided that any angle greater than 90 degrees constitutes the case of a high difference in magnitude, the previous case. As a result, any output vectors in the subspace less than the orthogonal hyperplane of the gold label vector will have an accuracy of 0%.

4. EXPERIMENTATION

We used a 4-fold cross validation to test on our small dataset of 116 labelled images in batch sizes of 1 image with random shuffle and weight initialization of the dataset and CNN. Each validation was trained for 20 epochs on 87 images and evaluated on the remaining 29. The results can be summarized in the figures below.

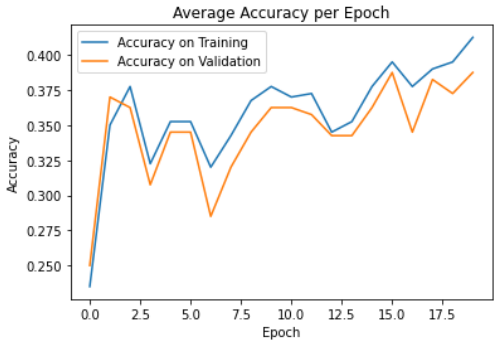

Figure 6. Averaged accuracy of each cross validation evaluated on the training and validation sets per epoch.

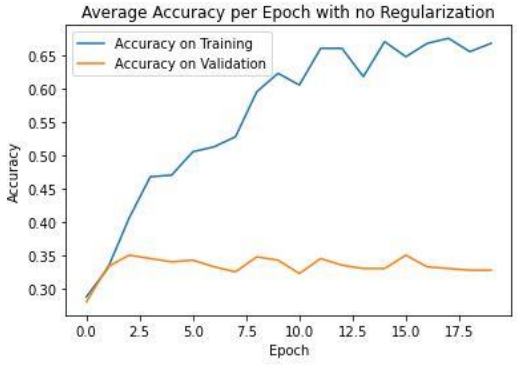

Figure 7. Averaged accuracy of each cross validation evaluated on the training and validation sets per epoch with no regularization.

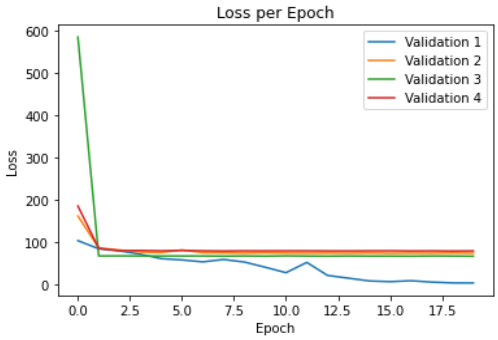

Figure 8. Loss per epoch for each of the four validations.

As we can see in figure 8, the loss for each of the validation converges within a few epochs. It is at the same time the accuracy rate reaches approximately its maximum, with additional increases in accuracy rate due to the contribution of the increased loss of the first validation.

Based on figure 6, we can see that there is a slight overfitting of the model on the training set.

An equally interesting result was the high accuracy variance of the model as slightly seen in the figure 6. Although we never measured the variance of the model, we got final average accuracies anywhere between 30-54% when evaluated on the validation set. Reasons for this vary from the random initialization of the CNN weights, small dataset, and our failure to formally fine tune the hyperparameters of the model.

Ultimately, our final average accuracy evaluated on the validation set was approximately 39% according to the metric we defined above. Or in other words, class frequency predictions that our model makes are approximately 39% similar to the gold vectors.

5. EVALUATION & DISCUSSION

To access our model on existing solutions for instance segmentation, we fine-tuned a state-of-the-art Mask-RCNN on our custom dataset with pre-trained weights from the MS COCO image dataset.

5.1 Fine-Tuning Mask-RCNN

To help fine tune a Mask-RCNN model on our custom dataset, we used an open-source implementation of Mask-RCNN from Matterport and adjusted some of the configuration parameters to better suit our dataset [1,3,6]. Specifically, we lowered the images per GPU count with a smaller batch, validation, and confidence threshold just due to our small dataset and low computation resources. We also lowered the anchor pixel sizes the network uses in the region proposal phase so that it would be able to pick up on smaller objects.

After setting up a proper configuration for our data, we fined tuned the Mask-RCNN model to populate the backbone layers with the pre-trained weights from the MS COCO dataset. The back-bone layers are the initial CNN layers (ResNet101 Model in our usage) for generating the appropriate feature map to pass to the other layers. The successive parallel layers after the backbone (Region Proposal Network, ROI Classifier, and Segmentation Mask) are initialized with random weights and will be learned by back propagating on our dataset with our classes while the backbone layer remains frozen throughout the fine-tuning process [3, 6].

5.2 Results

We used a hold-out validation to train and evaluate the Mask-RCNN model due to training times. The model was trained in 20 epochs on the same 116 images as experimented in our implementation and evaluated on 23 test images. The results can be summarized below.

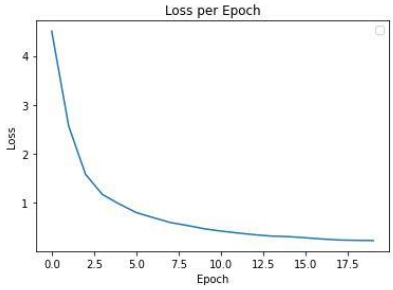

Figure 9. Loss per epoch during training phase. Converges to approximately 0 after 18 epochs.

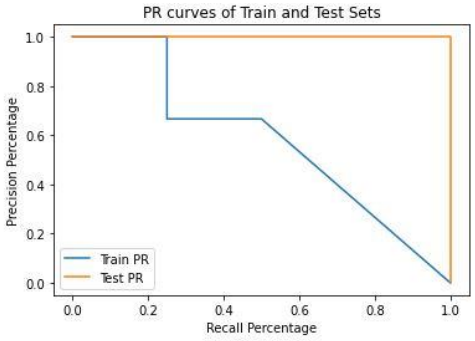

Figure 10. Precision-Recall curve of the Mask-RCNN model when evaluated on the training and test set with mAP of 97% and 40% for an IOU of 0% and a confidence threshold of 80%.

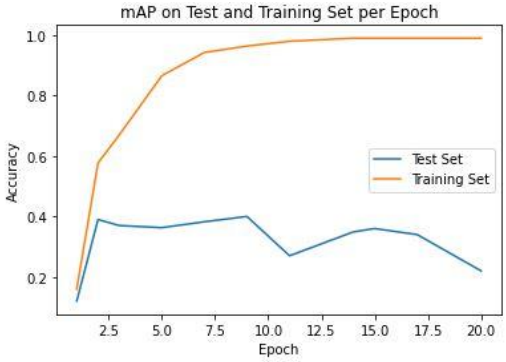

Figure 11. mAP of the model evaluated on the test and train models per epoch.

We can see that the loss converges but the model clearly starts overfitting on the training data after 2 epochs. The mAP of the model evaluated on the test set reaches its highest accuracy of 40% after 9 epochs while the evaluation on the training approaches 100% after 10 epochs [9].

Some explanations to the overfitting can be attributed to the small dataset used to tune the model, human labeling inconsistencies between the train and test sets, and the inappropriate weights pre-trained on the MS COCO dataset that generated the features map for our images.

5.3 Comparison with our Proposed Technique

Although it might not seem like our model and its accuracy metric can be compared against the mean average precision, which is the typical metric used in instance segmentation, upon closer inspection there isn't that much difference in the metrics when it comes down to our use-case: detecting the frequency of each food class in an image.

Taking just the precision and recall metrics and eliminating the IOU contribution (ignore masking and bounding box accuracy) to the mAP by setting it to 0% threshold, we're simply just computing the recall accuracy, the frequency of the model recalling objects for a given class, and the precision accuracy, the accuracy of the recall. The resulting average of the two is very similar to our metric, which basically just computes the similarity of the two vectors. Output vectors with high recall and precision rates will have very similar angle and magnitudes with the gold label vector therefore we believe the two are fair metric to evaluate against each other.

Taking that into consideration, our model's accuracy metric of 39% performs equivalent to the 40% mAP produced by the Mask-RCNN when it comes to just detecting the frequency of the different food classes in the evaluation set. The Mask-RCNN model does perform significantly better when evaluated on the training dataset, with mAP of 99% against 65% and seems like the better solution and will have higher validation set metrics when the dataset is larger.

Another metric taken into consideration was the training time. Although the Mask-RCNN model trained on 116 images while our CNN model only trained on 87 images, there is a huge difference in the training times.

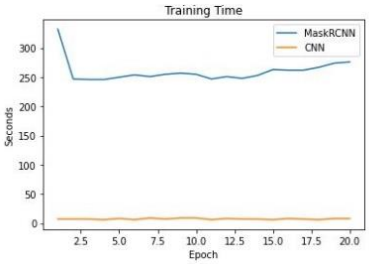

Figure 12. Training time per epoch.

But note that the Mask-RCNN model converged to its maximum mAP in less than 10 epochs while our CNN model took 20 epochs.

So, while the Mask-RCNN is the general solution for most instance segmentation problem, for our use case and the small dataset we worked with, the performance and functionality aspect of our simpler model is much preferred to the larger overhead that gets introduced with Mask-RCNN.

6. CONCLUSION & FUTURE WORK

Figure 13. Original image and the prediction output with confidences generated by the Mask-RCNN algorithm. As you can see, it correctly predicted the rice and vegetables in the original image.

Overall, the results of this study best emphasize how smaller and simpler models greatly outshine more complex models under certain limitations such as small datasets when evaluation comes down to functionality and performance. We saw in our study how a simpler CNN model produced a similar functional accuracy to the state-of-the-art Mask-RCNN model while having a drastically better training time performance.

In terms of the future of this study, it does seem like the Mask-RCNN model is the preferred solution as we obtain more images and therefore can train on all layers of the network. Possible solutions to enhance the Mask-RCNN model include incorporating our CNN into the Mask-RCNN algorithm so that it can calculate object frequencies to aid in the Mask-RCNN process or using depth cameras to add additional details to the model.

Beyond the software challenges, additional issues like battery life, networks, and computing resources adds to the complexity of implementing this solution.

References

[1] Abdulla, W. (2018, December 10). *Splash of color: Instance segmentation with mask R-CNN and TensorFlow*. Medium. Retrieved December 8, 2021, from https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46.

[2] *Food waste in America in 2022: Statistics & Facts: RTS*. Recycle Track Systems. (n.d.). Retrieved December 8, 2021, from https://www.rts.com/resources/guides/food-waste-america/.

[3] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

[4] *Hough Circle transform*. OpenCV. (n.d.). Retrieved December 8, 2021, from https://docs.opencv.org/4.x/da/d53/tutorial_py_houghcircles.html.

[5] *Make sense*. Make Sense. (n.d.). Retrieved December 8, 2021, from https://www.makesense.ai/.

[6] Matterport. (n.d.). *Matterport/MASK_RCNN: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. GitHub. Retrieved December 8, 2021, from https://github.com/matterport/Mask_RCNN.

[7] Seif, G. (2021, July 18). *How to do everything in computer vision*. Medium. Retrieved December 8, 2021, from https://towardsdatascience.com/how-to-do-everything-in-computer-vision-2b442c469928?gi=564b1e53ddf5.

[8] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[9] Solawetz, J. (2020, October 5). *What is mean average precision (MAP) in object detection?* Roboflow Blog. Retrieved December 8, 2021, from https://blog.roboflow.com/mean-average-precision/.

[10] *Training a classifier*. Training a Classifier - PyTorch Tutorials 1.10.0+cu102 documentation. (n.d.). Retrieved December 8, 2021, from https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html.

[11] *VGG16 - convolutional network for classification and detection*. VGG16 - Convolutional Network for Classification and Detection. (2021, February 24). Retrieved December 8, 2021, from https://neurohive.io/en/popular-networks/vgg16/.