

CS231n Final Report

EatRight: Calorie Counter

Jagriti Dixit
Stanford University
Stanford

jagritid@stanford.edu

Paavni Shukla
Stanford University
Stanford

paavni@stanford.edu

Abstract

In this project we attempted to simplify the task of finding the calories present in the food being consumed using Deep Learning. We experimented with different architectures such as Residual Network design and Mask-RCNN on a smaller subset of very large dataset Food-101. Mask-RCNN can successfully perform instance segmentation and has capability to record several detections of more than one type of food item present in an image. The detected labels are used to calculate total calories in the food image using standard serving size and calories associated with them.

1. Introduction

The statistics on world obesity rates indicate that the worldwide prevalence of obesity nearly tripled between 1975 and 2016. Overall, about 13% of the world's adult population (11% of men and 15% of women) were obese in 2016. Of course, obesity leads to a myriad of health problems for the people suffering from it.

Today, a lot of people are advised by nutritionists to control their daily calorie intake by eating healthier foods. Most people are also cautious about what they eat as a measure of prevention is better than cure. However, although food packaging comes with nutrition (and calorie) labels, it's still not very convenient for people to reference. Thus, began the need to simplify this task using machine learning algorithms in computer vision to help people determine the calorific value in the food they eat.

Our project aims to simplify this task to just clicking a picture for the users to find the number of calories present in the food being consumed. The project leverages deep learning models for computer vision to achieve this task. **The input to our algorithm is an image of the food. We then use Mask R-CNN deep learning model [1] to output a predicted calorific value of the food item.** While a lot of off-the-shelf calorie counters exist today, there is still a lot of scope in improving accuracy when it comes to food iden-

tification and quantity estimation from the image provided.

2. Past Related Work

Food and its classification has gauged interest and has been a topic of re- search in the Deep learning community over an extended period of time. There has been significant work done on food detection by images, the preliminary work being recognition of food wherein the items were uncooked raw staples. An advancement to this in [2], [3] and [4] was the food dish recognition wherein recipes or dishes with multiple ingredients were identified in comparison to single ingredients. A wide range of works and even commercialised systems promise such food recognition systems based on images. However, most of them have very less dishes/classes used for training and they end up identifying many common dishes as just Food/No Food rather than their original dish name. While most of these services are based on sophisticated neural networks [1], only two professional services Clarifai and Watson had results which outperformed others by a huge margin. While most of the earlier work has been on classification of food items from the images, working on calorie estimation using dish and its portion recognition is relatively newer. Therefore, the next logical step towards this direction was identification of food portions from images. Combining both the efforts of dish recognition and portion estimation led to the idea of calorie counters. Although a lot of commercial calorie counters till now like Calorify and HealthifyMe relied on manual entries of the dishes and their quantities, there have been some updates and sophisticated apps in the market which allow the user to take a picture and upload it for calorie estimation. While, the earlier ones just recognized the images and gave possible suggestions, there have been few works which identify the dish as well as the quantity from the image making the task of calorie assessment a lot more easier than before.[5] presents a deep learning algorithm - a faster R-CNN followed by a GrabCut algorithm which analyzes the composition of the food in the picture along with its contours and can tell the amount of calories in the picture. The chal-

lenge lies in the need for top and side view images of the food being analyzed for an accurate volume and mass centric estimation. [food-cnn-3], [food-cnn-4] talks about other such techniques using finger in the image and a Graph Cut Segmentation for portion estimation.

3. Methods

For this project we have experimented with 2 different models and setups. The first step of our project was to use ResNet-50 (as a baseline model for classification) and then we used Mask-RCNN for enhanced image and instance segmentation.

3.1. Algorithms Used

3.1.1 ResNet-50

Resnet-50 was chosen as our initial baseline model to explore our performance on the dataset with object detection. Moreover, Resnet-50 was chosen primarily because the Resnet model serves as a backbone of our M-RCNN model (specifically Resnet-101). For simpler datasets, where the classes are easily distinguishable, or where the features are more easy to identify and classify, a ResNet-50 model is often a better fit; it's smaller, faster to train and easier to use and deploy. Since the ResNet-101, is slower to train and easier to overfit considering our smaller dataset (10,000 images out of the original 101,000 images in Food-101), we used the simpler model as our baseline model. The ResNet model also handles the issues arising in deep neural networks such as the increasing depth of the network and the problem of vanishing gradients extremely well. The ResNet architecture includes residual blocks within it's network architecture to solve the issue of vanishing gradients and slower learning in deep neural networks.

Residual blocks simply add the activations from previous layers to the currents layer value prior to activation:

$$a^{[l]} = g^{[l]} \left(\left(W^{[l]} a^{[l-1]} + b^{[l]} \right) + a^{[l-2]} \right) \quad (1)$$

where $a^{[l]}$ is activation of layer l.

For our smaller subset of Food-101 (10 categories with 1000 images each), we observed accuracy greater than 95% within reasonable number of epochs (45 epochs). More deeper network architecture (such as ResNet-101, etc.) can be explored when using the entire dataset of Food-101, which contains about 101,000 images.

3.1.2 Mask-RCNN

The project follows a sequential two-step architecture approach for image-based food calorie estimation. Mask-RCNN is primarily used as the first step - i.e. as the food

item classifier. Detecting food items in an image is an object detection task, for which using CNNs, is a popular approach. Some of the popular CNN designs for object detection are Mask R-CNN [6], Faster RCNN, and YOLO [7]. We chose Mask-RCNN primarily because of its ease of use and its ability to perform instance segmentation and labelling efficiently, thereby enabling us to keep a track of instances and its labels needed for calorie calculations. We built our project by adopting **transfer learning on the Matterport's trained model of Mask R-CNN for object detection and instance segmentation using Keras and TensorFlow** [8]. The Matterport implementation is trained on the MS COCO dataset. We **manually annotated images for our own dataset** of food images, **extended the implementation to our own custom dataset by transfer learning** [9] and **fine-tuned the model by training it further on our own dataset** using our experiments.

Mask RCNN, is a state-of-the-art Convolutional Neural Network (CNN) for image and instance segmentation and it was developed on top of the Region-Based Convolutional Neural Network, Faster R-CNN. Given an image, it results in detecting different classes of objects along with different instances of each class, therefore providing a count of instances. This is extremely helpful for calorific estimation where we roughly equate the **instance detected to a serving size** for the particular food item (or count for food items where a obtaining a count is possible).

Mask RCNN is a combination of two different types of CNNs namely Faster R-CNN and Fully Convolutional Net (FCN). Here, the Faster R-CNN component of the architecture does the object detection, i.e. the image is scanned and proposals are generated. The FCN performs the semantic segmentation, i.e. classifies the proposals created in first step and generates bounding boxes and masks. Mask R-CNN outputs a mask which are the actual shape of the objects unlike square bounding boxes. with each detected object instance.

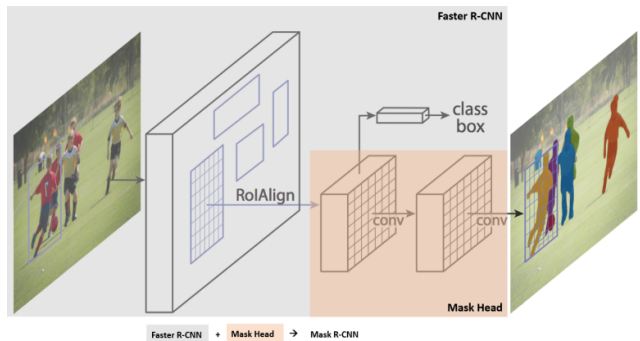


Figure 1. The Mask R-CNN framework for instance segmentation

3.2. Calorie Prediction

We predict calories using a simple CSV that we have created for the categories of food items in our subset of the Food-101 dataset. The CSV essentially maps each food item to its calorific value per serving. We roughly equate the instance detected to a serving size for the particular food item (or count for food items where obtaining a count is possible). We then use the outputs from our Mask-RCNN model and implement a simple algorithm that uses these as inputs (the food labels detected, and their counts) to determine the calorific value of the entire image using a simple lookup on the CSV file.

4. Dataset and Features

4.1. Dataset Analysis

Available datasets for food images that are categorized include the Food-101 dataset [10], the Recipe1M+ dataset [11] among others. These datasets are open-source and although they are not an exhaustive collection of food items and dishes around the world, these datasets would serve the needs of our current project.

For our project, we are using the Food-101 dataset, that contains a 101 categories of food items and 1000 images per food category, totalling a 101,000 images. The dataset was used in the Paper "Food-101 – Mining Discriminative Components with Random Forests" by Lukas Bossard, Matthieu Guillaumin and Luc Van Gool [10]. It's a sufficiently large dataset for testing computer vision techniques. Since, we could not find annotated images of the Food-101 dataset for the Mask-RCNN model **we had to manually annotate images with the ROI of food items for training**. This impacted our ability to run our model on the whole Food-101 dataset. Therefore, we restricted our calorie estimator to run on 10 categories of food items (omelette, macaroons, macaroni and cheese, samosa, cup cakes, pizza, donuts, prime rib, hot dog, caesar salad) and **extracted 40 random images from the Food-101 for each category** as seen in 2



Figure 2. Sample images for all food labels

The Food-101 dataset contains metadata for splitting the dataset into 250 test images and 750 training images per

class. The training images were not cleaned. The dataset contains images of 101 categories of food items in the jpeg format.

4.2. Data Preprocessing

4.2.1 Data Preprocessing/Augmentation for ResNet-50 model

For the ResNet-50 model, we ran our model on the 10 categories that we subsequently ran our Mask-RCNN model on. The dataset used thus contained **1000 (750 train, 250 val) images from the Food-101 for each of the 10 categories**.

To ensure all images had uniform dimensions for the neural network and to ensure that our model generalizes better (as the object(s) of interest we want our models to learn are not always wholly visible in the image or the same scale in our training data), we performed **random cropping and resizing** on all the images and resized them to 224 x 224 pixels. The original dataset had a maximum side length of 512 pixels for any image. When we experimented with trying to crop the images to smaller sizes, we found that the training loss significantly increased. Thus each image could be represented as matrix of shape (3*224*224). **Random horizontal flipping** (where we horizontally flip given images randomly with a given probability), **random rotation** (-45° to +45°), **addition of color jitter** and **normalization** were also performed on the images as additional data augmentation techniques along with their transformation to tensors. This was done to ensure that the model generalizes well to images of a food item that would vary due to illumination, color to an extent and orientation in the training image.

4.2.2 Data Preprocessing/Augmentation for Mask-RCNN model

For the Mask-RCNN model, because we were constrained by the manual annotations required on the image data, we created a smaller custom dataset that contained **40 images from the Food-101 for each category**, where we sourced **30 images from the training images and 10 images from the images designated as validation**, for the respective purposes.

The annotations were performed using the **VGG Image Annotator Tool** [12] (as seen in image 4). Care was taken to **use polygonal bounding boxes** (to improve training accuracy as polygonal bounding boxes describe the ROI more accurately than rectangular bounding boxes) while annotating. The annotations were obtained in the "json" format for both the training and validation subset. We tried to ensure a variety of backgrounds for images of a food in a category to make the model more robust. Sample images and their corresponding masks are shown in 3.

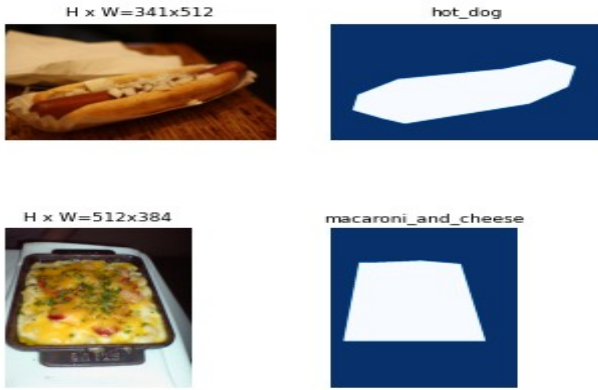


Figure 3. Sample masks for images after manual annotation

We also performed a testing accuracy pass by using the data designated as validation and performing 5-crop data augmentation on it.

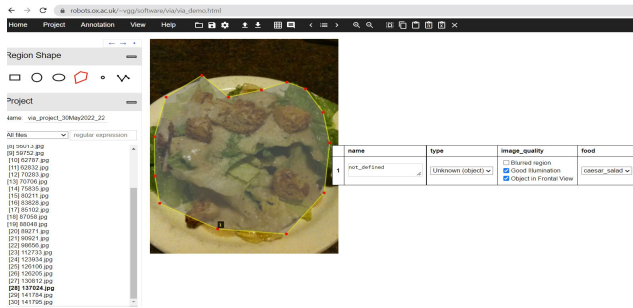


Figure 4. Manual Annotation of ROIs for images using VGG IA Tool

For the Mask-RCNN model we experimented with several data augmentation techniques. We initially faced problems of overfitting on the training data as we observed that our training loss decreased, but validation loss shot up with increasing training epochs. The current model for the project augments the data by applying the following augmentation transformations to only 30% of all images - **flip images horizontally** (50% of the 30 %), **gaussian blur** (each image with a gaussian kernel with a sigma of 0.5), **affine transformations** (scaling, rotating, shearing) to ensure good generalization of the model.

5. Experiments, Results and Discussion

5.1. ResNet-50 model- Baseline Experiment & Results

The baseline experiments were done using a ResNet-50 architecture for food items classification on Food-101

dataset, as it is a classic neural network that is used as a fundamental model for many computer vision problems.

To achieve higher accuracy with the subset of dataset (10 categories with 1000 images in each), deeper networks are used. ResNet is best choice for this training as it solves problems arising due to increased depth of models i.e. vanishing gradient. ResNet resolves this problem by employing "skip-connections".

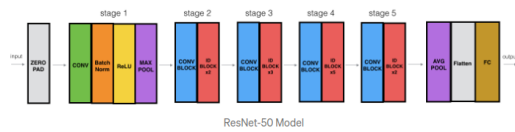


Figure 5. Resnet-50 Model Architecture

We started by using the pretrained ResNet-50 model with the Imagenet weights available through the Keras library for all pre-final layers. For the final layer of the model we replace the final fully connected layer with a new fully connected layer (with random weight initialization) that takes in 2048 features and produces 10 class outputs for our dataset(a subset of Food-101 with 10 categories).

We initially freeze all pre-final layers and train only for the final layer (for our experiments - we ran the training for 15 epochs). We freeze the weights so that during future training rounds we will avoid disturbing any of the information the model contains. After 15 epochs, the new added final layer learns to turn the old features into predictions on our dataset. Subsequently, we use the weights obtained from the best saved model saved during this training, to then train for all the layers of the network (we ran the training for this part for a further 15 epochs) with a very low learning rate to adapt pretrained features to new data.

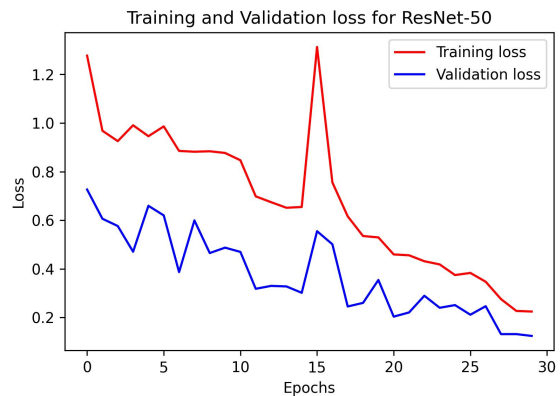


Figure 6. Training and Validation Loss for ResNet-50

We experimented with both adam and SGD with mo-

momentum for optimization where we observed Adam as an optimizer achieved lower accuracy for 30 epochs, hence we selected SGD with momentum as the optimizer. We used a learning rate of 0.02 for first training of 15 epochs to obtain fine-tuned weights which resulted to a validation accuracy of 89.65% and validation loss of 0.3017. As the weights were fine-tuned, we used 0.01 learning rate for next training of 15 epochs to achieve higher accuracy. When we unfreeze the model parameters after 15 epochs we see a spike in loss as seen in figure 6, which eventually decreases during next 15 epochs. This spike is seen because at this stage entire model starts learning. We achieved a validation accuracy of 96.28% and validation loss of 0.1243.

Table 1. Test Accuracy of 10 classes with ResNet-50 architecture

Class	Test Accuracy
caesar_salad	96.80%
cup_cakes	97.20%
donuts	92.00%
hot_dog	96.80%
macaroni_and_cheese	97.20%
macarons	96.80%
omelette	94.00%
pizza	98.00%
prime_rib	97.20%
samosa	96.80%

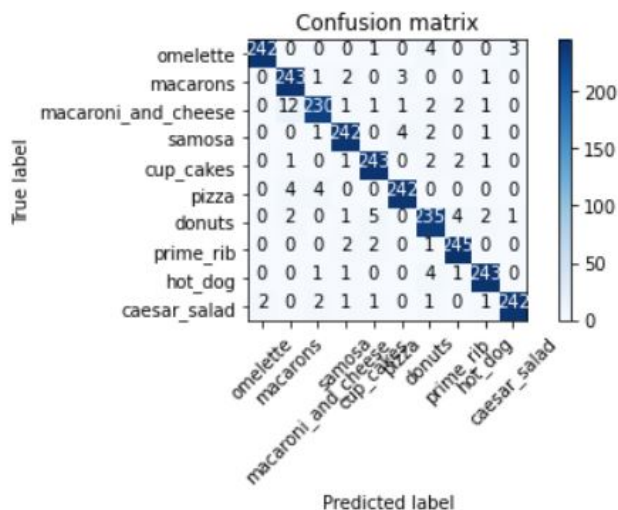


Figure 7. Confusion matrix plot for predictions from each class

To better visualize the results from ResNet-50 architecture model for the above 10 classes, we plot the confusion matrix 7 with the data of predictions for each class in table

1. We also plotted the saliency maps, the figure 8 represents the saliency map for class cupcakes.

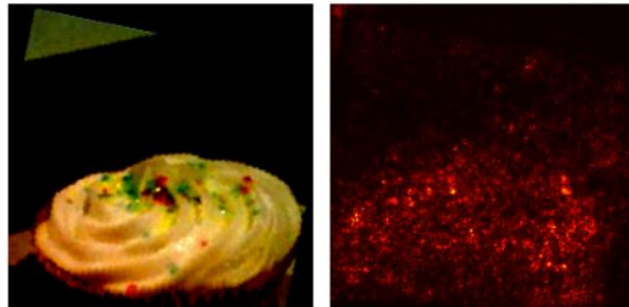


Figure 8. Saliency map (R) for Cupcakes class on a transformed image (L)

5.2. Mask R-CNN - Experiments & Results

To estimate the calorific content of the food in the image, we simplify the processing by having the data for calorie content of single servings for each of the categories (the calorific values of each single serving are present in the lookup table created).

We conducted several experiments to tune our Mask-RCNN model implementation (based off the original "mat-terport" implementation). Our initial experiments were with learning rates for gradient descent. We found the **learning rate of 0.001** to be optimal for our project. Lower learning rates (such as 0.0001) led the model to train very slowly and moreover, worsened the loss. Higher learning rates of 0.01 led the loss to reach undefined values. For our **optimizer, we chose SGD with a momentum value of 0.9**, because of faster training time when compared to choosing Adam. Our batch sizes were constrained to 2 by the GPU. However, to support calculation for the mean-Average Precision Loss (mAP loss), we had to further restrict our **batch size to 1, setting the steps per epoch to 100**. At this stage, the only data augmentation that was performed was flipping of images. We initially witnessed overfitting on the training dataset as after >40 epochs we viewed that that even though the training loss decreased, the validation losses tended to shoot up. Therefore, our next approach was to perform greater data augmentation (as described in section 4.2.2). This improved our results and we were able to achieve decrease in validation losses with higher epochs. The validation loss stabilized at about 80-100 epochs. Although we tried several approaches (running experiments upto 220 epochs, lowering learning rates at local minima points observed for validation losses at these higher epochs), we could not view any further decrease in validation losses. The graph for comparing the validation loss against the training loss vs. epochs is as seen in 9

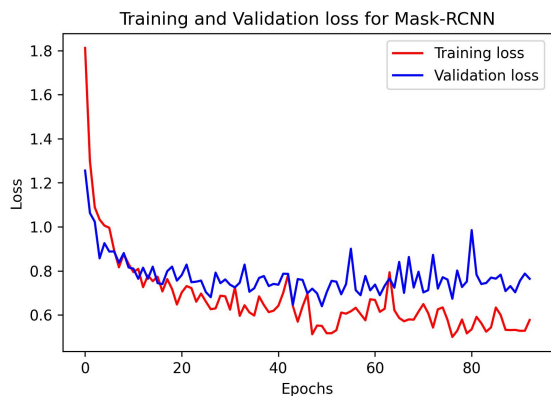


Figure 9. Plot for Val. Loss and Training Loss vs Epochs - Mask-RCNN

Mask-RCNN starts by implementing a Region Proposal Network (RPN) and runs a lightweight binary classifier on a lot of boxes (anchors) over the image and returns object/no-object scores. Anchors with high objectness score (positive anchors) are passed to the stage two to be classified. Often, even positive anchors don't cover objects fully. So the RPN also regresses a refinement (a delta in location and size) to be applied to the anchors to shift it and resize it a bit to the correct boundaries of the object [8], as seen in 13



Figure 10. Sample image of a cupcake from the dataset

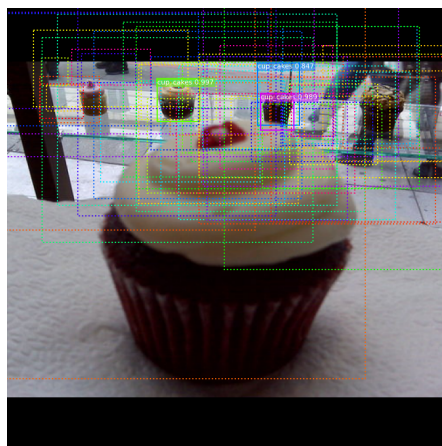


Figure 11. Top Anchors by score before refinement for image in Fig 10 - Mask-RCNN

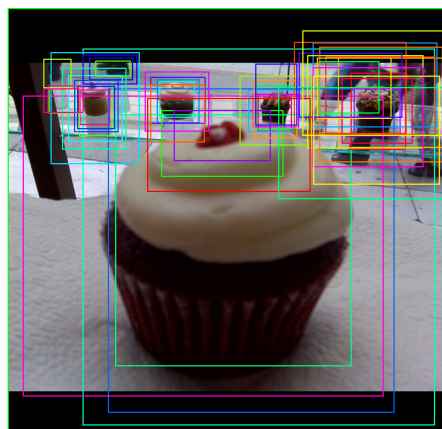


Figure 12. Top Anchors by score after refinement and NMS for image in Fig 10 - Mask-RCNN

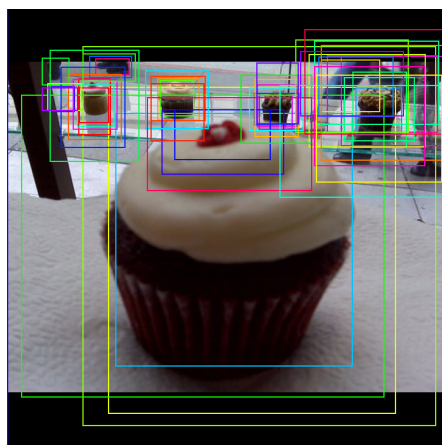


Figure 13. Final proposals for image in Fig 10 by the RPN - Mask-RCNN

The second stage takes the region proposals from the RPN and classifies them. Here we run the classifier heads on proposals to generate class probabilities and bounding box regressions.

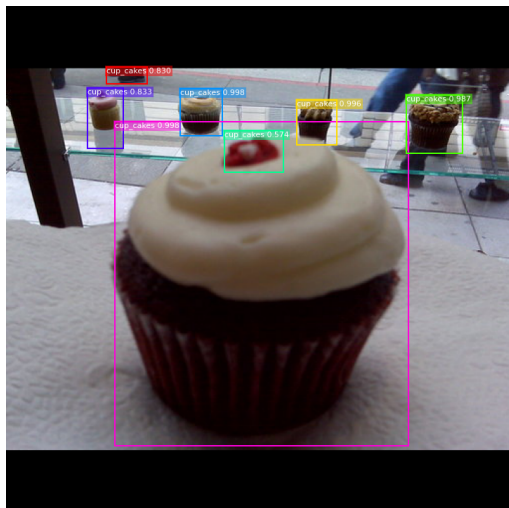


Figure 14. Object Detections in the image in Fig 10 - Mask-RCNN

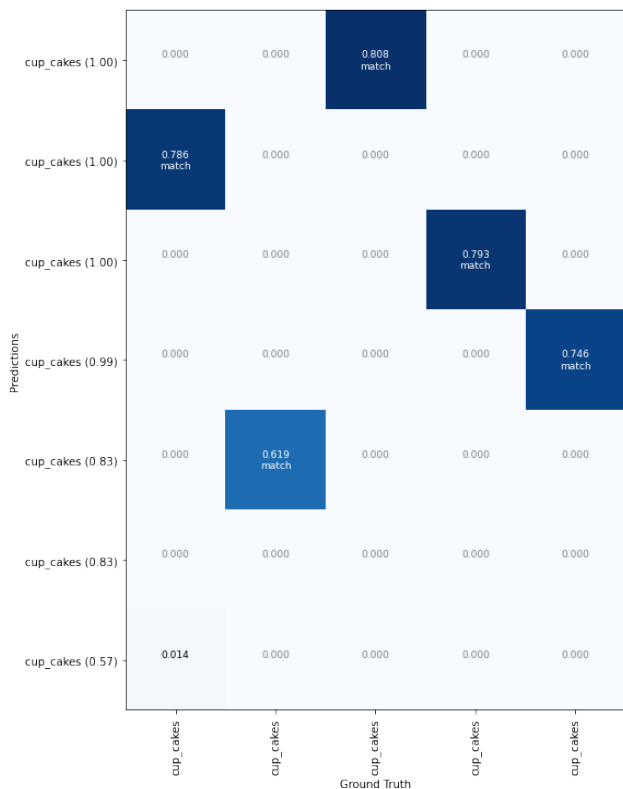


Figure 15. Confusion Matrix for image in Fig 10 - Mask-RCNN

To help us infer the results and visualize the workings of

the model, we generated confusion matrices and precision recall curves (as seen in 15 and 16 respectively) for each image. As expected, we observed that certain food categories that had similar features - for instance donuts and macarons that have similar shapes - tended to be mis-classified with a higher probability. As we had used the COCO pre-trained weights on our model, we also observed that the couple of food items that intersected between COCO and our dataset, for instance "pizza" get better results and had higher APs.

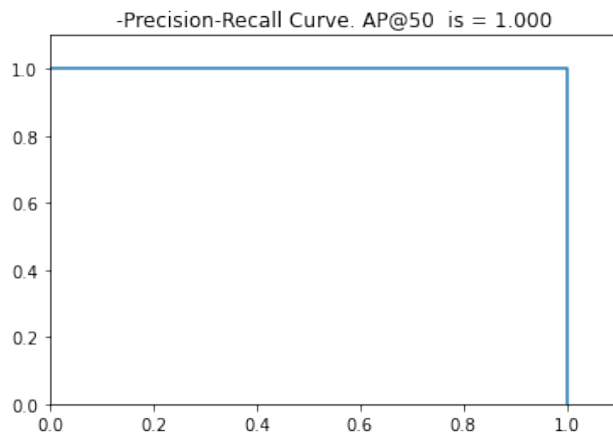


Figure 16. Precision Recall Curve for image in Fig 10 - Mask-RCNN

For our experiments with the Mask-RCNN model - we used 2 metrics to track our performance. The first metric is the mean average precision score (mAP) which is based on sub-metrics such as the confusion matrix, intersection over union(IoU), recall, and precision. This metric is often used to evaluate object detection models such as Fast R-CNN, YOLO, including Mask R-CNN too. The second metric is the mCA or mean Calorific Accuracy(in percent). This metric simply tracks the mean of the accuracy of the calorific value predictions made over all the images in the validation set. The results are summarized in table. As expected, the numbers for mAP and mCA are close to each other which indicates that the number of instances as well as their predicted calorie values are similar. Certain misses which are expected can be found from analysis of final object detections. Figure 14 shows the number of cupcakes it has detected in the image. As we can see, it has mis-classified the cherry and the background object as a cupcake although their confidence scores are lower than the correct detections. Keeping checks on confidence scores can help mitigate this mis-detection problem. 2

Table 2. Results of Mask-RCNN

Metric	Value
Mean Average Precision (mAP@IOU=50)	0.807
Mean Calorific Accuracy	76.03%

6. Conclusion and Future Work

Looking at the results from especially the Mask-RCNN model, it becomes clear that there is a lot of scope of improvement when it comes to instance segmentation and improving calorific value predicted. The model can be scaled to include more food categories, and experiments could be performed to ensure robustness against miss-classification between food items that are identical looking. However, the results achieved by a model like Mask-RCNN for the complicated task of predicting food items, which is an inherently tougher task when compared to instance segmentation for a lot of other objects, are still remarkable. We hope to continue extending to the project, as there are a lot of interesting optimizations and areas, that have the promise of significantly improving the utility and performance of the project.

A lot of the progress on this project was also slowed by the labor-intensive task of manually annotating the images. More efficient tools could be explored/developed for such purposes, as it would greatly benefit not only this, but many such projects that rely on instance segmentation. This would allow the model to train and validate on far many more images.

Another area could be exploring the possibility of changing the backbone model of Mask-RCNN (with Inception networks or DenseNet-101 instead of ResNet-101) and observing the difference it creates on the performance. Our team wanted to primarily try two things as next steps for improving results - increasing data augmentation, and exploring the use of pretrained weights from our ResNet-50 model trained on a far greater number of images per-category for better accuracy. Also, we feel the utility of this project will greatly improve with the integration of an end to end UI where the user can simply upload the image, for instance in a mobile app, and get the calorie value.

Volumetric analysis of the food item is another challenging extension to this project. Since, food items such as mac and cheese and caesar salad don't have a count, it may be better to estimate the serving using some form of volumetric analysis with such food items. Our team did come across some interesting articles for this too.

7. Contributions & Acknowledgements

Authors equally contributed to the work beginning with problem identification, literature surveying, datasets search and comparisons, data preprocessing, the code for implementing the models and the full write-up of the project report. The code for our project is available at <https://github.com/Jagriti-dixit/EatRight-Food-Calorie-Estimation>.

References

- [1] Kaiming He et al. "Mask R-CNN". In: (2017). URL: <https://arxiv.org/abs/1703.06870>.
- [2] Björn Hempel. *Investigation of strategies for image classification*.
- [3] Malina Jiang. "Food Image Classification with Convolutional Neural Networks". In: (2019).
- [4] Md Tohidul Islam; B.M. Nafiz Karim Siddique; Sagidur Rahman; Taskeed Jabid. *Food Image Classification with Convolutional Neural Network*.
- [5] Rework Deep Learning Summit in Boston 2015. *Talk by Kevin Murphy*.
- [6] viso.ai. *Mask R-CNN: A Beginner's Guide*. URL: <https://viso.ai/deep-learning/mask-r-cnn/>.
- [7] Rohith Gandhi. *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms*. URL: <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>.
- [8] Matterport. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. URL: https://github.com/matterport/Mask_RCNN.
- [9] Waleed Abdulla. *Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow*. URL: <https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46>.
- [10] ETH Zurich. *The Food 101 Dataset*. URL: https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/.
- [11] MIT CSAIL. *The Recipe 1M Dataset*. URL: <http://pic2recipe.csail.mit.edu/>.
- [12] Ankush Gupta Abhishek Dutta and Andrew Zisserman. *VGG Image Annotation Tool*. URL: <https://www.robots.ox.ac.uk/~vgg/software/via/>.