

SeeFood

A CS231N Final Report

Wai Tong Chung
Stanford University

wtchung@stanford.edu

Mahesh Saboo
Stanford University

mssaboo@stanford.edu

Chiin-Zhe Tan
SCPD

chiin@stanford.edu

Abstract

Food is important to agriculture, hospitality, and retail sectors. To explore automation in these areas, we trained a deep learning model to recognize 25 different classes of food. Here, we employ a Mask R-CNN architecture from Detectron 2. Our training dataset consists of food data largely extracted from the LVIS dataset. Compared to the baseline model (pre-trained on MS-COCO), our model can detect 17 additional food classes, and ignore non-food objects. However, quantitative results measured by the average precision show that improvements can be made in future work by dealing with challenges in class imbalance and similar classes.

1. Introduction

Food is without a doubt one of the most important aspects to human survival, culture, and pleasure. It is no surprise then that food automation poses as one of the most exciting applications of machine learning. Specifically, the ability to detect food via computer vision can potentially reduce costs for agriculture and food processing [5], and revolutionize the way we shop for our groceries [8].

Decisions regarding food are often made via a manual process, necessitating time and human effort: checking an inventory by hand, to confirm which items are present (and in what quantities), or examining a market stall to see what items are available (and at what prices), and deciding on a course of action based on this information. Holding a large number of food items in short term memory, whilst retrieving recipes from long term memory, is at best onerous and demanding, and at worst downright fallible.

By automating the food detection step, we can streamline this process, enabling a user to take a photo of their fridge with their mobile phone, immediately see which ingredients are present, and which recipes could be made. Cooking and shopping are universal activities, and there is no culture in the world where people would not benefit from this service.

We are developing a tool that will enable users to receive cooking recipes and ingredient recommendations from images captured from their phone. This involves providing ingredient recommendations for ready-made foods (tacos, cakes, etc.) as well as recipe recommendations for raw ingredients (tomato, flour, etc.). In our proposal, we narrowed our application towards images of food in grocery baskets. Due to difficulties in finding a sufficiently large database of grocery basket images, we now expand the scope of project to encompass any images with food in them.

We have divided our project into two main stages. Stage 1 consists of developing a food produce/grocery detector using state-of-the-art computer vision techniques. Stage 2 will use predictions from the object detector developed in stage 1 to build a recommendation system that provides recipes and missing ingredients.

The input to Stage 1 is a photographic image, taken of items (food and otherwise). We use a deep learning model to output a predicted image segmentation mask, describing food items present in the image. This is trivially converted into a list of food items which can be passed as input into Stage 2.

2. Related Work

These applications have been largely empowered by powerful object detection and semantic segmentation models, many of which are open-source. Frameworks such as Tensorflow and PyTorch has implementations for models such as RCNN, SSD, RPN, RetinaNet, etc.. These open-source models can be further fine-tuned at a later stage with smaller and tailored datasets for more specific ML problems. For our project we have decided to use the Detectron2 [12] framework by Meta. Detectron2 is implemented in PyTorch and includes high quality implementations of state of the art object detection and segmentation algorithms.

Another essential component to food detection involves the availability of massive object detection/semantic seg-

mentation datasets. MS COCO [7] contains ~ 80 object labels (both bounding boxes and semantic segmentation masks) and 200K labeled images resulting in 1.5M labeled objects in 200K images. LVIS [3] builds on the same image database with additional improvements to the annotations (~ 1200 object labels). The review article by [11] provides object detection/semantic segmentation datasets that are tailored to groceries and retail checkout. MVTec D2S [1] contains 60 object labels (semantic segmentation masks) and 21K labeled images, where each image contains objects of a single class on a homogeneous background (resembling an automated checkout). Retail Product Checkout (RPC) dataset [10] contains 200 product categories and 83,739 images including single-product and multi-product images both.

Thus the purpose of this project is to integrate existing open-source object detection/semantic segmentation models with existing databases of food products.

To scope out the details of this project: we first assemble a common dataset of food objects from LVIS dataset [3], Densely-segmented supermarket dataset [1], and RPC: A large scale retail product checkout dataset [10]. Then, we will iteratively fine-tune the models provided by Meta’s detectron 2 [12] to detect and classify only food products. This may involve replacing the base model in detectron2 with our own trained model. To build a recommendation system in stage 2, we will create a weak API to feed our predictions to Supercook [9].

Our aim is to generate more accurate food predictions than pretrained models from the detectron model zoo. To evaluate our food detector/classifier over a range of images, we will use mask average precision, as described in the COCO dataset [7]. We choose a range of Intersection-over-union (IoU) thresholds (0.5 to 0.95 in 0.05 increments) and categories: for each (threshold, category) pair, we will compute the average precision (performing the IoU computation on segmentation masks, labeling detections based on the IoU threshold). Our performance metric is the average over all (threshold, category) pairs of this metric.

3. Method

3.1. Detectron2

Detectron2 is a popular PyTorch based modular computer vision model library. It allows us to use pretrained state of the art computer vision algorithms to train our custom dataset for the purpose of object detection and segmentation. It is based on the maskrcnn benchmark. It supports multiple tasks such as bounding box detection, instance segmentation, keypoint detection, densepose detection, and so on. Much of our code is based off the Detectron2 tutorial.

3.2. Model Architecture

Instance segmentation requires the correct detection of all objects in an image along with the precise segmentation of each of the instance of the object. It involves object detection and semantic segmentation both. In object detection, the main goal is to classify individual objects and localize them using bounding boxes while in semantic segmentation, the goal is to classify each pixel into a fixed set of categories without differentiating object instances.

We used **Mask R-CNN** [4] framework for instance segmentation. Mask R-CNN can be considered as an extension to the Faster R-CNN. Faster R-CNN consists of two stages. The first stage proposes the bounding boxes for the candidate object, called the Region Proposal Network (RPN). The second stage, extracts the features using ROI Pool from each candidate and performs classification and bounding box regression. Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest, in parallel with the existing branch for classification and bounding box regression. The additional mask output is distinct from the class and box outputs and requires extraction of much finer spatial layout of an object. Figure 1 explains the higher level idea of Mask R-CNN.

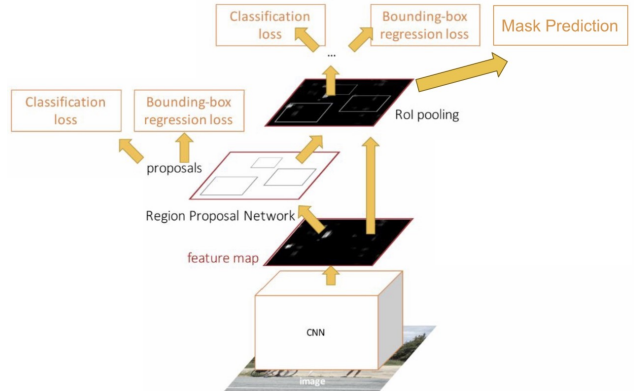


Figure 1: Architecture of Mask R-CNN [2]

During training, we can define a multi-task loss on each sampled RoI as $L = L_{cls} + L_{box} + L_{mask}$. Here, the L_{cls} represents the classification loss, L_{box} represents the bounding box loss and L_{mask} represents the mask prediction loss.

For instance segmentation using Mask R-CNN, the pixel-to-pixel behaviour requires that the RoI features are well aligned in order to preserve the per pixel spatial correspondence. Hence, there is a RoIAlign layer that plays a key role in mask prediction. The RoIAlign layer removes the harsh quantization of RoIPool and aligns the extracted features properly with the input. Figure 2 shows the overall framework for instance segmentation using Mask R-CNN, including the RoIAlign.

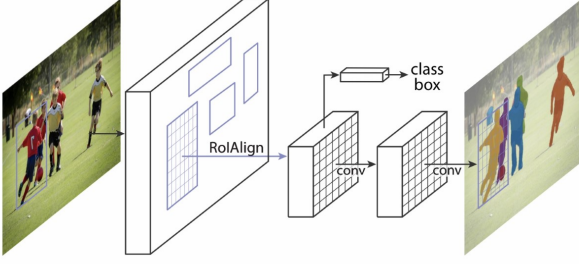


Figure 2: Mask R-CNN framework [4]

The network architecture of Mask R-CNN can be divided into two parts: (i) the convolution backbone architecture used for feature extraction over the entire image and (ii) the network head for classification, regression and mask prediction applied separately to each RoI. We have used the model **mask_rcnn_R_50_FPN_3x** from the Detectron2 for training our dataset. This model has a ResNet-50-FPN backbone.

ResNet is one of the most popular CNN architecture. It consists of very deep networks that uses residual connections. The ResNet has stacks of residual blocks each of which consists of two 3 X 3 convolution layers. Figure 3 shows a ResNet residual block. Feature pyramid networks (FPN) [6] leverages the ConvNet’s pyramidal feature hierarchy, which has semantics from low to high levels and build a feature pyramid with high-level semantics through out. It uses a top-down architecture with lateral connections to build an in-network feature pyramid from a single-scale input. ResNet-FPN backbones for feature extraction with Mask R-CNN have been known to give excellent gains in both accuracy and speed.

3.3. Baseline Model: MS COCO

Detectron2 consists of several pre-trained models for instance segmentation. We plan to start with the models trained on MS COCO dataset and compare them. We believe that MS COCO based models will be a good point to start for the purpose of instance segmentation. For each model, we can fine tune the hyperparameters such as learning rates, optimizer momentum, weight decay, batch size, number of iterations, *etc.* These hyperparameters can be changed using the config instance we create for training the data.

3.4. Training our Model

We modify the aforementioned **mask_rcnn_R_50_FPN_3x** pre-trained on MS-COCO towards detecting 186 food classes from our dataset. Training is performed via minibatch stochastic gradient

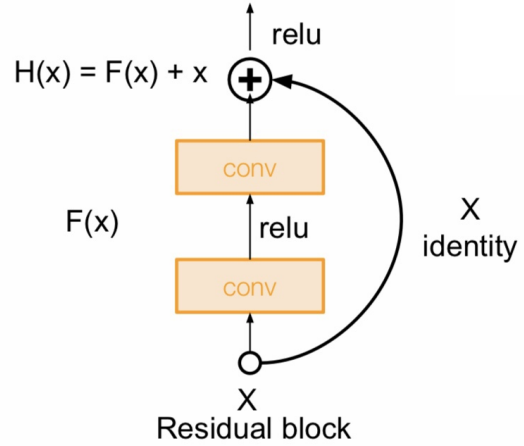


Figure 3: A residual block in ResNet [2]

descent at learning rates of 1E-4, 1E-5, and 1E-6 for 12,000, 3,000, and 7,000 iterations, respectively resulting in the converged loss plot shown in Figure 4. A batch size of 8 is used, which represents the upper limit of the GPU memory. Since the network converges quickly before 2 epochs, a piece-wise approach is employed for training on the 3 different datasets. In particular the model is first trained on the food dataset extracted from LVIS, and is then trained on labels with classes that are not well-predicted on the D2S and RPC datasets. This paper only discusses our results from training the network on the LVIS dataset, and we leave improvements from D2S and MVTEC as future work. Prior the training, the data is split in an 80:10:10 fashion for train, validation, and test, respectively, and data augmentation in the form of resizing the figure for the network’s input layer and random flipping. Training is performed on a single T4 Tensor Core GPU for 24 hours of wall-clock-time. Note that good agreement between the train and validation loss curves are seen in Figure 4, indicating an absence of overfitting.

3.5. Metrics

For sets A, B , the Intersection over Union (IoU) is $\frac{|A \cap B|}{|A \cup B|}$; this can be applied to both bounding boxes and segmentation masks.

When applying an object detection model to an image: let the ground truth objects be $A_1 \dots A_k$, and assume our model has made predictions (either boundary boxes or segmentation masks) $P_1 \dots P_n$, each with confidence C_i . Assuming that each prediction P_i overlaps at most one ground truth object, A_{k_i} , we compute the IoU for prediction P_i and

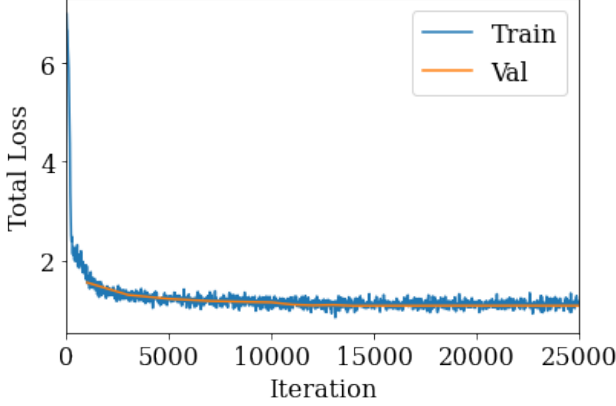


Figure 4: Training and validation curve. Note that validation is performed on the entire validation set every 1,000 steps due to limitations in the Detectron2 interface.

$$A_{k_i} \text{ as: } IoU_i = \frac{|P_i \cap A_{k_i}|}{|P_i \cup A_{k_i}|}.$$

For a threshold T : we classify prediction P_i as a **true positive** ($X_i(T) = 1$) when $IoU_i \geq T$, and as a **false positive** ($X_i(T) = 0$) when $IoU_i < T$. Without loss of generality, let C_i be a non-increasing series. Then we can compute precision and recall for each $i \leq n$:

$$Precision_i = \frac{\sum_{j=0}^i X_j(T)}{i}$$

$$Recall_i = \frac{\sum_{j=0}^i X_j(T)}{k}$$

Plotting the points $(Recall_i, Precision_i)$ gives the **precision-recall curve**. This can be made monotonically increasing by applying:

$$Precision(r) = \max_{\hat{r} \geq r} Precision(\hat{r})$$

Then we define the average precision for this threshold as:

$$AP(T) = \int_0^1 Precision(r) dr$$

(more generally, this is computed numerically at a number of equally spaced points on the $[0, 1]$ interval, e.g. COCO uses 101 points)

Our primary evaluation metric is mask average precision (AP), as described in the COCO dataset [7]: for a class C and threshold T , $AP_C(T)$ is the average precision over class C (performing the IoU computation on segmentation masks, labelling detections based on the threshold T). AP_C is the average of $AP_C(T)$ where T ranges from 0.5 to 0.95 in 0.05 increments. AP is the average of AP_C over all classes C .

4. Dataset and Features

We have a combined dataset that includes the LVIS dataset [3], D2S, dataset [1], and RPC: A large scale retail product checkout dataset [10]. The LVIS dataset contains ~ 1000 labels, many of which include non-food items

such as lamps, kettle, etc. Hence, we preprocessed the data by pruning the dataset to only involve 186 food classes. This reduced nearly 120,000 images to nearly 19,000 images. These 19,000 images possess a total of nearly 260,000 instances of food objects, as detailed in Appendix A. For the D2S and RPC datasets, the only processing that was required was to ensure that the category IDs from this dataset matched the LVIS dataset. Figure 5 demonstrates six snapshots from our test set, with box and segmentation labels, where all non-food categories are no longer considered as labels.

5. Results

We first evaluate quantitative results, in the form of food class AP, of the baseline MS-COCO model on the test set, with results summarized in Table 1. It can be seen that the baseline model box and segmentation APs ranging from 20 to 60 are seen for all 10 different food classes in the MS-COCO dataset.

Table 2 shows the class AP of our model on the test set. Compared to the baseline results in Table 1, the AP is about half as low as the baseline model. This drop in performance can be largely attributed to the more challenging training problem. In particular, our new model is trained to perform predictions on 186 food classes, with many labels containing similarities with each other, *i.e.*, apples and pears, cakes and cupcakes, bread and pizza, *etc.* In addition, the comparison of two different models with two different annotations can be difficult since the labels in the MSCOCO dataset are not completely identical to the LVIS dataset. Nevertheless, we note that our new model is more successful than the baseline model in its ability to detect 17 new food classes, with non-zero AP. All other food classes which cannot be predicted well can be seen in Appendix A, which is largely caused by class imbalances in the present LVIS dataset that we plan to address in future studies by supplementing the training set with the RPC and D2S datasets.

Figure 6 visualizes six model predictions from the baseline model. Note that while food classes from Table 1 are predicted by this model, the baseline model also recognizes many non-food items such as people, bowls, knives, *etc.* This makes the baseline model unsuitable for the purposes of our food detection application.

Figure 7 visualizes the predictions from our network. Here, the model has been trained solely to predict food classes. The model is successful at recognizing the presence of more food items compared to the ground truth in Figure 5. This is clearly observable by the recognition of salad components and carrots (upper right), more broccoli (lower left), the more bananas (lower center), and vegetable toppings on the pizza (lower right). However, the model confuses between bread and pizza, wine bottles and water bottles, and tomato and salami. This is a completely under-

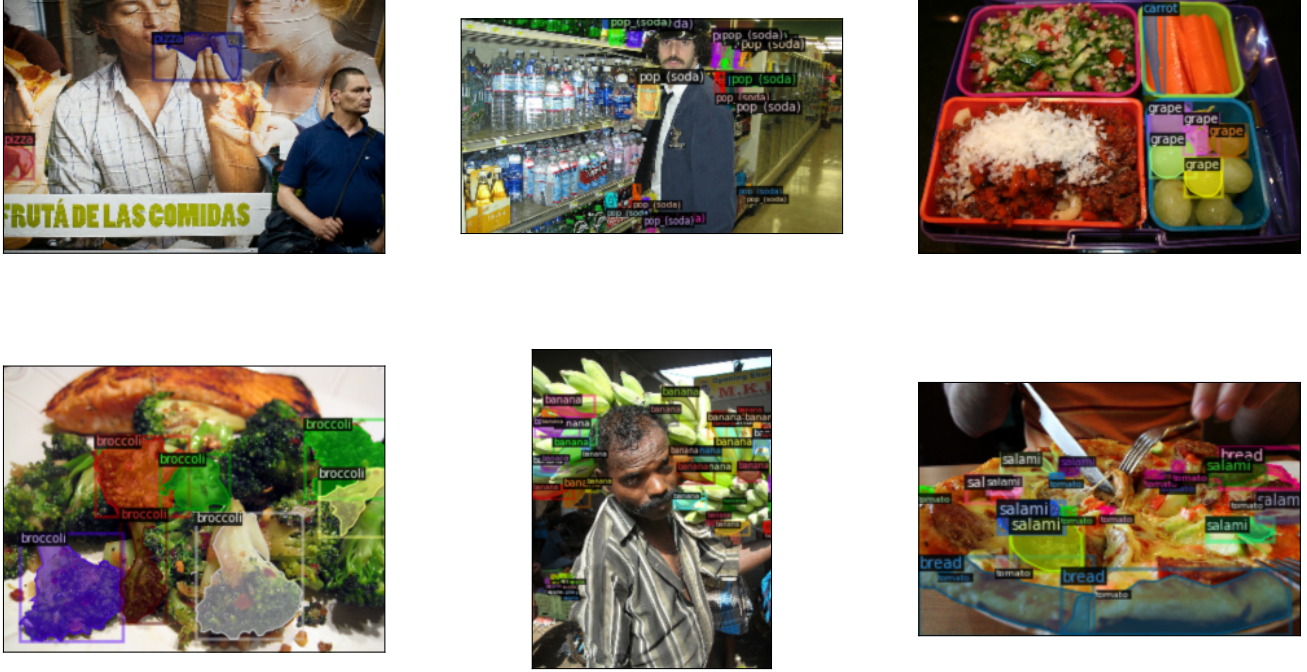


Figure 5: Six ground truth snapshots from the present food dataset.

Class	banana	apple	sandwich	orange	broccoli	carrot	pizza	donut	cake	hotdog
seg AP	25.6	24.3	42.3	32.5	28.9	24.7	56.9	55.1	37.5	28.6
box AP	29.7	25.1	42.2	32.9	30.7	27.9	58.6	55.1	37.6	31.7

Table 1: AP of all food classes from the baseline model.

Class	banana	apple	sandwich	orange	broccoli	carrot	pizza	donut	cake	hotdog
seg AP	11.1	13.3	11.0	14.6	15.2	11.5	4.1	28.6	22.5	7.8
box AP	12.7	12.8	10.8	14.0	11.2	15.3	8.6	28.4	21.7	7.5

Class	lettuce	mushroom	nut	onion	pastry	potato	strawberry	tomato	wine
seg AP	1.8	1.0	0.6	0.5	0.6	0.4	1.3	4.7	13.4
box AP	1.8	1.0	0.9	0.5	0.6	0.3	1.1	4.7	13.8

Class	beer	bell pepper	blueberry	bread	crumb	cupcake	grape	lemon
seg AP	2.7	0.1	0.6	4.7	0.3	9.6	1.4	0.2
box AP	2.1	0.1	0.5	4.9	0.3	8.5	1.4	0.2

Table 2: Non-zero AP of food classes from our model.

standable mode of failure since the listed foods have very similar shapes and colors to their mistaken counterparts. In addition, the current model cannot predict the cans of soda seen in the upper center figure, which is due to class imbalance seen in Appendix A. These mispredictions can be rectified in future work by including more samples of these imbalanced classes and similar food objects into the training set. However, this can be challenging as manual annotation of additional food samples can be costly.

6. Conclusions and Future Work

Our work on SeeFood was focused on training a food detector. To do this, we assembled a comprehensive food database, and used Detectron2, a state-of-the-art framework based on PyTorch. Hence we were able to perform transfer learning (with a pre-trained model) to train a food detector which can detect 17 new food classes.

The qualitative results are promising: the detector can detect food objects reasonably well. We see failures with

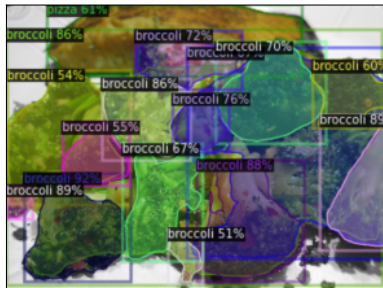


Figure 6: Predictions from the baseline model.

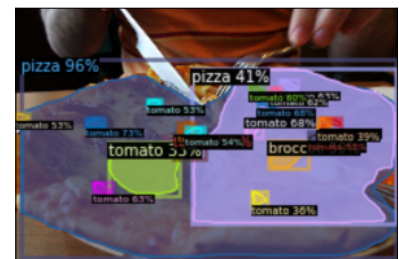
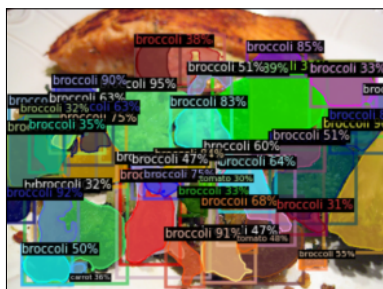
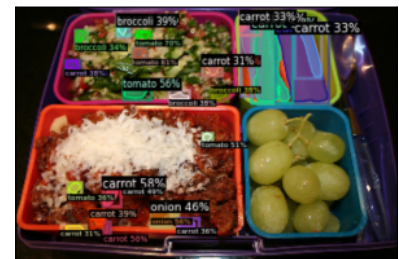


Figure 7: Predictions from our model.

respect to similar food objects, e.g. (salami, tomato). Quantitative results, specifically AP scores, are less promising.

Future work would focus on increasing the training data,

specifically focusing on examples for classes which are most commonly mispredicted. Investing resources (time, money, human effort) in further annotated sets would ad-

dress this issue. We have created food-related subsets of the RPC and MVTEC-D2S datasets, which are labelled consistently with LVIS; however, owing to lack of resources, we have not yet augmented the training data with these. This augmentation would be a natural next step.

7. Contributions and Acknowledgements

W.T.C implemented the training for the food detector and cleaned LVIS data. M.S. cleaned and labelled RPC data. C.Z.T. cleaned and labelled MVTEC-D2S data. All authors were involved in managing the project and writing the report.

Supplementary Material

All code for training, testing, and generating labels are provided in <https://github.com/waitong94/seefood>.

References

- [1] P. Follmann, T. Böttger, P. Härtinger, R. König, and M. Ulrich. Mvtec d2s: Densely segmented supermarket dataset. In *European Conference on Computer Vision (ECCV)*, 2018.
- [2] R. Gao, J. Wu, and L. Fei-Fei. CS231N Stanford. <http://cs231n.stanford.edu/>.
- [3] A. Gupta, P. Dollar, and R. Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [4] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn, 2017.
- [5] V. Kakani, V. H. Nguyen, B. P. Kumar, H. Kim, and V. R. Pasupuleti. A critical review on computer vision and artificial intelligence in food industry. *Journal of Agriculture and Food Research*, 2:100033, 2020.
- [6] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection, 2016.
- [7] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.
- [8] A. Polacco and K. Backes. The amazon go concept: Implications, applications, and sustainability. *Journal of Business and Management*, 24(1):79–92, 2018.
- [9] A. Rozenblatt, K. Cooperman, and S. Khan. Supercook. <https://www.supercook.com/>, 2015.
- [10] X.-S. Wei, Q. Cui, L. Yang, P. Wang, and L. Liu. Rpc: A large-scale retail product checkout dataset, 2019.
- [11] Y. Wei, S. Tran, S. Xu, B. Kang, and M. Springer. Deep learning for retail product recognition: Challenges and techniques. *Computational Intelligence and Neuroscience*, page 8875910, 2020.
- [12] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.

A. Food labels and instances

category	#inst.	category	#inst.	category	#inst.
alcohol	207	almond	1700	apple	17451
applesauce	7	apricot	62	artichoke	293
asparagus	969	avocado	1048	bagel	372
banana	50552	batter_(food)	26	beef_(food)	1242
beer_bottle	1227	beer_can	203	bell_pepper	4369
blackberry	406	blueberry	2114	boiled_egg	125
bread	6550	broccoli	12166	brownie	217
brussels_sp..	590	bubble_gum	4	bun	1780
burrito	14	butter	308	cake	2297
candy_bar	29	candy_cane	107	cantaloup	193
carrot	18049	casserole	12	cauliflower	1035
cayenne_(sp..	49	celery	911	cherry	903
chickpea	265	chili_(vege..	354	crisp_(pota..	541
chocolate_bar	179	chocolate_c..	80	chocolate_m..	7
chocolate_m..	1	cider	38	clementine	108
cocoa_(beve..	4	coconut	273	coleslaw	72
cookie	1920	edible_corn	1883	cornbread	10
cornmeal	1	crab_(animal)	50	crabmeat	5
cracker	510	crape	12	cream_pitcher	10
crescent_roll	152	crouton	140	crumb	3021
cucumber	1533	cupcake	1628	date_(fruit)	103
doughnut	11911	egg	813	egg_roll	15
egg_yolk	90	eggplant	337	escargot	5
fig_(fruit)	147	fish	525	fish_(food)	96
French_toast	41	fruit_juice	37	garlic	487
gelatin	248	ginger	93	gourd	101
grape	6377	green_bean	2571	green_onion	1618
grits	3	ham	1765	hamburger	126
honey	90	hot_sauce	70	hummus	9
icecream	180	popsicle	1	ice_pack	4
jam	29	jelly_bean	116	keg	6
kiwi_fruit	702	lamb-chop	8	lasagna	7
legume	333	lemon	2168	lemonade	2
lettuce	5500	lime	1134	liquor	66
lollipop	59	mandarin_or..	401	martini	3
mashed_potato	58	meatball	174	melon	167
milk	227	milk_can	8	milkshake	1
mint_candy	27	muffin	352	mushroom	6257
nut	790	octopus_(fo..	5	olive_oil	36
omelet	10	onion	9779	orange_(fru..	13034
orange_juice	223	pancake	295	papaya	206
pastry	4972	patty_(food)	20	pea_(food)	1869
peach	1041	peanut_butter	50	pear	1069
pepper	697	pepper_mill	91	persimmon	22
pickle	632	pie	228	pineapple	1636
pita_(bread)	28	pizza	4103	pop_(soda)	951
potato	4393	prawn	779	pretzel	179
prune	8	pudding	2	quesadilla	6
quiche	33	radish	519	raspberry	778
rib_(food)	32	root_beer	3	salad	171

	salami	290		salmon_(food)	14		salsa	22	
	saltshaker	543		sandwich	2315		smoothie	53	
	soup	193		sour_cream	49		soya_milk	2	
	crawfish	5		squid_(food)	6		steak_(food)	139	
	stew	7		strawberry	4386		string_cheese	1	
	sugar_bowl	10		sugarcane_(..	31		sushi	337	
	sweet_potato	137		Tabasco_sauce	5		taco	21	
	tea_bag	42		tequila	2		toast_(food)	125	
	tomato	12338		truffle_(ch..	4		turkey_(food)	120	
	turnip	109		vinegar	1		vodka	3	
	waffle	61		watermelon	814		whipped_cream	201	
	wine_bottle	4449		yogurt	116		zucchini	798	
	total	262491							