CS230

Simplifying Grocery Checkout with Deep Learning

Jing Ning
Department of Computer Science
Stanford University
annaning@stanford.edu

Yongfeng Li
Department of Computer Science
Stanford University
liyongfe@stanford.edu

Ajay. Ramesh
Department of Computer Science
Stanford University
ajay_ramesh@stanford.edu

Abstract

Self-checkout are becoming increasingly popular in grocery stores. Most stores stick plastic bar-code tags onto products, especially for fruits and vegetable, to help customer checkout. It presents several problems regarding maintenance cost and environment. In the paper, we propose an alternative Mask R-CNN based instance segmentation system to precisely classify and localize the items at the pixel level. We also study the effect of various hyper-parameters on the performance of the system and present an Average Precision matrix. It shows that the algorithm is able to perform well on both object detection and instance segmentation tasks with an overall mean Average Precision(mAP) of 84% at an IoU interval of [50:95]. Meanwhile, the detection time for the model is around 75 ms per image, which can achieve the near-real-time detection.

1 Introduction

Our goal is to build a model that can improve the efficiency of the supermarket checkout process. In most grocery chains, stores stick plastic tags with identification numbers onto the surface of checkout items. This presents several problems including those with environmental sustainability with generation of significant plastic waste, consumer health with leaving potentially harmful glue residue on sensitive items such as fruits or vegetables, manual labor intensity of labelling activities and the propensity of these tags to easy damage that can subsequently result in poor checkout chain efficiency.

The classic deep learning Convolutional Neural Network (CNN) algorithm can only classify the item at the image level [1]. More advanced semantic segmentation algorithms have limitations in separating different instances in the image. The state-of-art object detection algorithm such as YOLO can localize item but not at pixel level [2]. These problems inspire us to think of another alternate, a single Computer-Vision deep learning algorithm that can perform classification, object detection at a instance level. We apply a state-of-the-art object-classification and instance-segmentation technique called Mask R-CNN. More details are described in the method section.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

Classic CNN: Horea Muresan el.[3] applied a classic convolutional network for fruit detection. They used a four layer convolutional network with a filter size of (5,5), increasing the number of filters from 4 to 64, each followed by a max pooling layer and fully connected layers. They reported test set accuracy of 94% for 60 different fruit classes. Frida Femling el.[2] experimented with different architectures including the Inception v3 model and MobileNet. They reported a top three test accuracy with InceptionNet as 96% and a 97% with MobileNet. They were able to train/test the system with multiple instances represented in the same image.

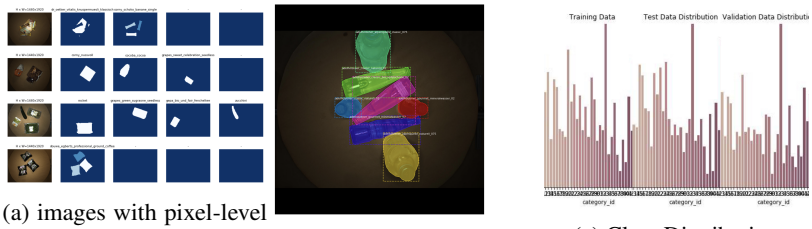
Faster Region-based CNN: The authors proposed a method to detect fruits using a Faster R-CNN model, achieving an F1 score of above 0.8 [4]. The network they proposed used a VGG-16 model with 13 convolutional layers followed by Region Proposal Network (RPN). Their network only detected the presence of a fruit, and did not have to classify the fruit itself.

While these approaches work well in their respective applications, we are uniquely challenged in that our system will not only have to detect grocery items, but also accurately classify them. Furthermore, we require that our system also handle the presence of multiple instances of each grocery item. With limited GPU resource to train our system, the mask R-CNN based system is able to achieve accurate detection and instance segmentation even with limited training iterations.

3 Dataset and Features

MvTec Densely Segmented Supermarket (D2S) dataset is a benchmark for instance-aware semantic segmentation in industrial domain [5]. It includes image with high resolution (1920,1440) in RGB color, and it contains 60 unique grocery product classes simulating a checkout belt. The dataset contains a total of 15,654 objects with 6 different backgrounds. Various lighting and rotation angles are captured to augment the dataset. Annotations are created with class information, bounding boxes and pixel-wise segmentation of all object instances in a format compliant with MS-COCO [6]. Four coordinate values are specified for bounding boxes and masks are generated by per-pixel boolean annotations [7].

Below is one example of the image and the pixel level annotation from MvTec D2S data set.



(a) images with pixel-level annotations and masks

(b) Images with annotations

(c) Class Distribution

Figure 1: Examples of MvTec D2S dataset and annotations

We shuffled and selected a subset of 3600 images from the data set and split the images into train/validation/test sets with a ratio of 80:10:10, resulting in a 2880/360/360 image count split, respectively. We normalized all our images to zero mean based on training set statistics. We set a mini-batch size to two images based on the single-GPU (NVIDIA K80) machine characteristics.

4 Methods

Here we describe how mask R-CNN performs image segmentation with high accuracy. The method was originally proposed by Kaiming He, from Facebook AI Research (FAIR) [8], We use a reference Github repository from Matterport in our implementation [9].

Mask R-CNN is an extension over Faster R-CNN [10]. Faster R-CNN predicts bounding boxes and Mask R-CNN essentially tacks on an additional branch to predict an object mask [11]. The additional mask output is distinct from the class and box outputs, requiring extraction of much finer spatial

layout of an object. Mask-RCNN is considered the state of the art for instance segmentation problems [10]. It has been proven to be useful across many industries and domains.

Below we show the high level architecture of the mask R-CNN architecture and describe its major components:

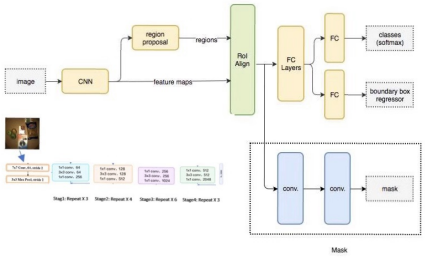


Figure 2: Mask-RCNN Architecture

- **Backbone model**: a standard convolutional neural network (such as MobileNet, ResNet50, or ResNet101) which work as a feature extractor over the entirety of the image. For example, it turns a 1024*1024*3 RGB image into a 32*32*2048 feature map that serves as input for the next layers.
- **Region Proposal Network (RPN)**: Using regions defined with as many as two hundred thousand anchor boxes, the RPN scans each region and predicts if an object is contained in it. To its advantage, the RPN does not scan the actual image, but instead the feature maps. Feature Pyramid Network(FPN) is introduced with multiple feature maps pathway including bottom-up, top-down and lateral connections. This allows network to output proportionally sized feature maps at multiple levels for a single-scale image input . This stage outputs several Regions of Interest (RoI).
- **Region of Interest Classification and Bounding Box**: This step of the algorithm takes at its inputs the RoIs proposed by the RPN and outputs a classification (softmax) and a bounding box (regressor). RoIAlign was introduced to align the difference in size in a way where the feature maps are cropped and resized.
- **Segmentation Masks**: In the final step, the algorithm takes the positive RoI regions and outputs 28x28 pixel masks with float values representing detected objects. During training, we scale down ground truth but at inference phase, these masks are then scaled to size of ROI bouding box.

Loss function on each sample RoI: For training iterations, the mask R-CNN method define a multi-task loss on each sampled RoI as follows:

$$L = L_{cls} + L_{box} + L_{mask}$$

The classification loss L_{cls} and bounding-box loss L_{box} are identical to those defined in Faster R-CNN. Specifically, the classification loss is defined as $L_{cls} = L_{cls}(p, u) = -logpu$, wherein, $p = (p0, ..., p_k)$ are the discrete probability distributions (per RoI) over $K + 1$ categories. p is computed by a softmax function over the $K + 1$ outputs with a fully connected layer. The bounding-box loss L_{box} is defined over a tuple of true bounding-box regression targets for the class u, $v = (v_x, v_y, v_w, v_h)$, and its predicted tuple $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$ as below[4],

$$L_{loc}(t^u, v) = \sum_{i \in x,y,w,h} smooth_{Li}(t^u_i - v_i)$$

where

$$smooth_{Li}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise, \end{cases}$$

is a robust L_1 loss that is less sensitive to outliers than the L_2 loss. When regression targets are unbounded, training with L_2 loss can require careful tuning of learning rates in order to prevent exploding gradients.

$$L_{mask} = -\frac{1}{m^2}\sum_{i\in i,j}[y_{ij}log\hat{y}_{ij}^{k} + (1-y_{ij})log(1-\hat{y}_{ij}^{k})]$$

The mask branch has a Km^2-dimensional output for each RoI, which encodes K binary masks of resolution $m * m$, one for each of the K classes. To this mask R-CNN applies a per-pixel sigmoid, and defines L_{mask} as the average binary cross-entropy loss. For a RoI associated with ground-truth class k, L_{mask} is only defined on the k-th mask, and other mask outputs do not contribute to the loss. This definition of L_{mask} allows the network to generate masks for every class without competition amongst classes.

5 Experiments/Results/Discussion

We evaluate our system with substantial experimentation and report its performance using the standard MS-COCO metrics (including mean Average Precision (mAP) at various Intersection over Union (IoU) thresholds and object scales) [6]. We use a single metric for validation iterations: the Average Precision over the test dataset at IoU intervals in range [50:90]. All our validation models are trained over just 10 epochs due to limited GPU time budgets.

To measure baseline performance, we set up a model pre-trained with MS-COCO weights and further train all its layers over 10-epochs. We achieved a baseline mAP= 0.239 over the IoU interval [50:90].

In order to improve model performance, we then perform an extensive hyper-parameter exploration using a coarse to fine search method. To begin with, we searched across different optimization algorithms such as Adam, RMSprop and Stocastic Gradient Descent with Momentum (SGDm). We found that Adam required tuning at much lower learning rates, consequently elongating training times. Based purely on running times < 2s/Step as satisfying metric, we picked SGDm.

With the optimization algorithm set, we then tuned the learning rate, sampling our system's performance at a log scale using 10^r where r sample between $(-1, -6)$. After a few such iterations, we narrowed down the learning rate to range within $(0.001, 0.01)$ where after a random search was performed.

With a final learning rate of 0.002, we were able to improve on our baseline by 28%, using a weight decay rate of 0.0001. We also analyzed our systems performance with different backbone architectures, and found *resnet50* worked very well for our application, providing good training performance with little accuracy degradation.

Below we show some important details on the parameter tuning process, including model architecture, hyper-parameters in transfer learning and image size.

Table 1: Results for backbone architecture

backbone	$AP_{50:95}$	AP_{50}	AP_{75}	AP_M	AP_L
ResNet-50	0.753	0.962	0.962	0.314	0.756
ResNet-101	**0.788**	**0.957**	**0.924**	**0.291**	**0.791**

Table 2: Results for image size

Image Size	$AP_{50:95}$	AP_{50}	AP_{75}	AP_M	AP_L
800*1024	**0.837**	**0.979**	**0.979**	**0.61**	**0.839**
512*512	0.753	0.962	0.962	0.314	0.756
256*256	0.04	0.072	0.042	0	0.058

Table 3: Results for layers to tune

layers to tune	$AP_{50:95}$	AP_{50}	AP_{75}	AP_M	AP_L
heads	0.625	0.946	0.764	0.147	0.628
3+	**0.84**	**0.98**	**0.97**	**0.451**	**0.843**
4+	0.795	0.976	0.958	0.441	0.798
all	0.753	0.962	0.962	0.314	0.756

- **Architecture**: After full length exploration of the hyper-parameters, we picked best ones and then babysat the learning process. The network performance improved slightly with architecture depth as shown in Table 1.
- **Image Size**: We explored the effect of image resolution on training accuracy. We observe that mAP increases with increased image resolution as shown in Table 2. To balance training time and

accuracy, we choose image resolution of 512 X 512 which produced good result with reasonable training time.

- **Layers to re-train**: Due to the limited labelled supermarket dataset for training, we decide to test different architectural training depths and apply transfer learning to frozen layers (Table 3). We show that we can predict supermarket item class at instance level, the pixel level mask works even with a certain level of occlusion. The algorithm also works with different background setting and various lighting conditions. The following show some of the results from our test set.

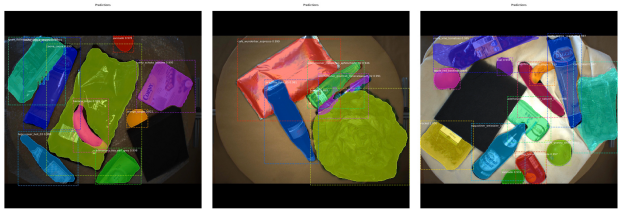


Figure 3: Prediction on test set

We were able to achieve best results by tuning '3+' layers resulting in a mAP of 0.84 over IoU 50:90. As can be seen from Figure 3, it works well even with some level of occlusion. In order to visualize the accuracy of the model's training, we inspect some intermediate convolution layers' feature maps, from which see hot areas gathered at uniquely identifying areas of each object.Error analysis were conducted on mAP of each class against number of object count as shown from Figure 5, Carrot, Adelholzener water, and Kamillentee tee have lowest mAP. Error appear to relates to difference in size of objects from same category and object occlusion with limited label displaying.

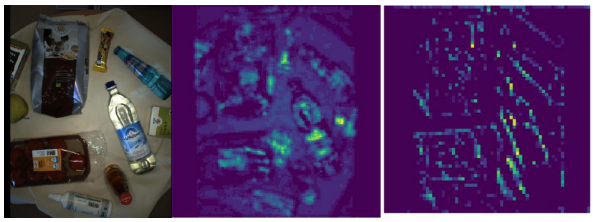
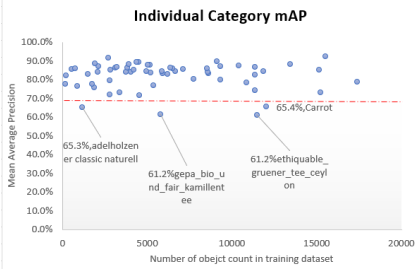
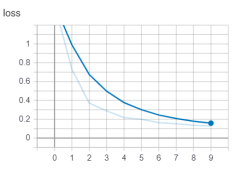


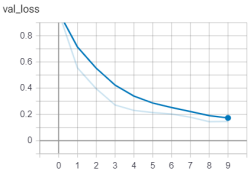
Figure 4: Feature Maps from Resnet layers



(a) Error analysis on each class



(b) training loss



(c) Validation loss

6 Conclusion/Future Work

In conclusion, we explore the hyper-parameter space within our computation budgets, and find optimal parameters to fit this dataset. The model was able to achieve a mAP of 84% on IoU 50:90 and mAR of a 81% on IoU interval [50:90] on the test set, surpassing our baseline model by a big margin.

For future work, we like to expand our model's application to other similar tasks such as fruits/vegetables sorting, quality control and other classification tasks. We could also build more robustness into the model by increasing the level of complexity of dataset (albeit at much increased computation cost) by collecting more data with images taken at different angles.

7 Contributions

Development was largely a collaborative effort with all team members working together and equally on research, coding, model parameter tuning and writing.

We sincerely thank the teaching staff of CS230: Deep Learning, with special thanks to Zahra Koochak for her immense help in every step of the project - from the project proposal to the final report.

Code has been made available on *Fruit Detection Github Repo*.

References

[1] Jiuxiang Gu et al. "Recent Advances in Convolutional Neural Networks". In: *CoRR* abs/1512.07108 (2015). arXiv: 1512.07108. URL: http://arxiv.org/abs/1512.07108.

[2] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *CoRR* abs/1804.02767 (2018). arXiv: 1804.02767. URL: http://arxiv.org/abs/1804.02767.

[3] Horea Muresan and Mihai Oltean. "Fruit recognition from images using deep learning". In: *CoRR* abs/1712.00580 (2017). arXiv: 1712.00580. URL: http://arxiv.org/abs/1712.00580.

[4] Ross B. Girshick. "Fast R-CNN". In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[5] Patrick Follmann et al. "MVTec D2S: Densely Segmented Supermarket Dataset". In: *CoRR* abs/1804.08292 (2018). arXiv: 1804.08292. URL: http://arxiv.org/abs/1804.08292.

[6] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[7] Patrick Follmann et al. "Learning to See the Invisible: End-to-End Trainable Amodal Instance Segmentation". In: *CoRR* abs/1804.08864 (2018). arXiv: 1804.08864. URL: http://arxiv.org/abs/1804.08864.

[8] Kaiming He et al. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[9] Waleed Abdulla. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. https://github.com/matterport/Mask_RCNN. 2017.

[10] Francisco Massa and Ross Girshick. *maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch*. https://github.com/facebookresearch/maskrcnn-benchmark. Accessed: [Fall 2019]. 2018.

[11] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.