Innovate the Retail Industry with Automatic Checkout System

Qian Zian 20412089
Ren Xuanchi 20493435
Wang Zhiwei 20412807

Abstract

Automatic checkout system (ACO) has been gaining increasing interest in recent years due to its practical value. Different from traditional object detection and recognition problem, ACO is faced the problem of domain shifting. The model learns the pattern from single object images but predicts on multi-object images which usually have totally different distribution than the training images. Existing work tried to solve the problem by data augmentation. We further develop the data augmentation solution by using a more efficient salient object detection algorithm and image inpainting to improve the realism of synthesized images. We first remove the background by using salient object detection algorithm, and combine the objects together on a plain white background. After this, we use image inpainting to learn the image background from ground-truth images and use the model to improve the realism of synthesized images. We use the images to train a FPN detector and experiments on large scale Retail Product Checkout (RPC) dataset shows that we achieve 63.32% checkout accuracy, which is higher than the 56.68% baseline accuracy.

1. Introduction

With the advent of no-man store in the retail industry, checkout automation has been gaining increasing interest in recent years. As one of the most time-consuming experience for customers in stores, checkout was improved throughout history by technologies such as barcodes and QR code. In recent years with the rapid development of computer vision technique, it is possible to further improve checkout efficiency by using automatic checkout system. (ACO) Customers no longer need to wait in long lines in the counter. They can simply take a photo of their shopping items in their basket and ACO should be able to generate an exact shopping list for the customer. ACO system can bring brand-new shopping experience for customers in retail stores.

ACO is technically an object detection and recognition problem. With a shopping image as the input, ACO should detect all the items in the image and count the number of each category. Different from traditional object detection problem, the main challenge of ACO is related to domain shifting. The training data are normally single object images with multiple views located at the center. In checkout setting, predictions are normally performed on images with multiple objects with totally different distribution from the training data. Thus, successful ACO should be able to identify the object from the source domain (single object image) to the target domain. (multiple object image)

Existing work [12] tried to solve the problem by data augmentation. They used a salient object detection algorithm and cycle-GAN to synthesize images which imitate the real-world checkout settings. However, this method for data augmentation is computational expensive and requires long time to synthesize large number of images.

In our project, we improve the data augmentation phase by using a deep learning based salient object detection algorithm proposed in [3]. This algorithm is much faster than the original one and produces better results. For rendering, we use image inpainting [7] to render high quality real-world images instead of cycle-GAN method which we found very hard to train. We used FPN [6] with ROI align as our network structure to train the detector. Experiments on the large-scale Retail Product Checkout (RPC) dataset [12] shows that we have achieved 64.32% checkout accuracy which is higher than the 56.68% baseline accuracy.

2. Related Work

In this section, we review previous work that also focus on ACO problem.

2.1. RPC: A Large-Scale Retail Product Checkout Dataset

This work [12] is published by face++, which is the team that provided this dataset. They also proposed a method for ACO problem and we treat it as a baseline.

They use a salient object detection algorithm proposed in [4], which is based on traditional image processing algorithm. This method is very complicated and computational expensive. Instead, we use a deep learning based salient object detection algorithm proposed in [3] to make an im-

Figure 1. All the objects are top-viewed, arranged evenly with no overlapping.

provement. For rendering, instead of cycle-GAN, we use image inpainting. The network we used are more or less the same. We modified the ROI pooling to ROI align to make a little improvement than their work.

2.2. Data Priming Network

[5] is a paper published online on April 10th, which works on exactly the same task.

The training of ACO system is challenged by the domain adaptation problem, in which the training data are images from isolated items while the testing images are for collections of items. To solve this problem, [5] propose to use method proposed in [3] as a data argumentation method which is exactly the same with us to generate synthesized images. However, image synthesis leads to unreal images that affect the training process. In our paper, we use image inpainting to make the synthesized images more realistic and then use FPN improved with RoI align for detection. Instead of image inpainting and FPN used by our paper, [5] propose a data priming network using detection and counting collaborative learning, and select more reliable images from testing data to train the final visual item tallying network.

3. Data

In this section, we will introduce what dataset we use and the detailed usages of the dataset.

The dataset we use are provided by face++ [12], which contains SKUs of 200 categories. In total, there are 53739 single-product exemplar images with multi-perspective view of the objects, and 30000 real world checkout images with multi-objects.

3.1. Usage of the Dataset

The usage of the dataset are divided into two parts: synthesizing images and using real world checkout images provided in the dataset.

1. Single Object Images

   Main component of this dataset are single object images. However, we decide not to use them for training because of the following reasons. Firstly, As we are using Faster RCNN and FPN in this phase, using this kind of training data will cause training failure of RPN in Faster RCNN because all those objects are in the center of the image. Secondly, using images with single object to train our network cannot help much on predicting multiple objects in an image. Thirdly, the background of those training images are not like the realistic checkout environment, which may lead to low accuracy when applying the network to reality. Thus, we use single-object images to synthesize our own multi-object checkout images.

2. Multi-Object images

   For 30000 real world multi-object checkout images, we divide them into training set with 24000 images and test set with 6000 images. However, the problem here is that all the checkout images are biased. As shown in figure.1, all the checkout images are top-viewed with no overlapping, which is not the case in real world. So we only take the testing accuracy of these data as a target, we will finally use our rendered data for training.

3.2. Plan of using the Data

1. Use synthesized multi-object images for training.

2. Use multi-object real world checkout images for training.

3. Concatenate the synthesized data and rendered data together for training.

4. Methods

In this section, we will explain in detail about the technical approach we use for data augmentations and network we use for object detection.

Figure 2. Original Image Using Salient Object Detection, and finally matting the object.

4.1. Data Augmentation

As we mentioned in previous section, the goal of our data augmentation is to synthesize large number of multi-object training images which are realistic. The multi-object image synthesize pipeline we propose consists of three parts: matting, synthesize and render.

1. Matting

   The first step of our image synthesis is matting the object in the single-object images. Since our dataset does not contain the information of segmentation, we have to find another way to mat them out.

   We first try a naive way which is using bounding boxes of each object to mat them out from the images. However, as shown in figure3, the result is totally not realistic. And this problem becomes even worse if the object is placed in other views. Our network may memorize this pattern during the training phase, and finally causes the failure of this task.

   Our second approach is using KNN matting proposed in [1], as we using this method, we find two serious problem. The first problem is that KNN matting requires trimap, so we have to label all the images by our self. The second problem is that it takes 3-4 minutes to matting a single images, which is long if we want to mat all 53739 images. Thus, we finally decided not to use it.

   Our final approach is to use salient object detection. We've compare the different method of salient object detection in [3] and [4]. Method proposed in [4] is based on image processing algorithm, it is very complicated and computational expensive. We find that the method proposed in [3] has both the high level semantic information and low level textual information, which is suitable for this task. However, the network proposed in this work is very sensitive to dark background. So we set the background to grey to turn off the background noise, and make the object darker. Then we feed our pre-processed images into the network to get the salient map.The result are shown in figure.11.

Figure 3. Our Matting Using Bounding Box, which is not that realistic.

2. Synthesize

   After obtaining the salient map, we multiply the salient map with the original images to get the object. Our synthesize method is similar to the method proposed in [1]. The objects are randomly selected and freely placed on a grey background image such that the occlusion rate of each instance less than 50 percent. Thus, our synthesized images will have multi-perspective view of the objects instead of top-viewed object. And the result is shown in figure.12.

3. Render

   After the synthesis step, domain gap still exists between the synthesized images and checkout images.

   It is easy to tell the difference between the images from these two domains by observing lighting conditions or shadow patterns. In order to render the synthesized images similar to real-world checkout setting, we employ Cycle-GAN [13] to translate these images into the checkout image domain.

   However, it is really hard to train a powerful Cycle-GAN model and after 4-day-training, the result, shown in figure.4. is quietly terrible. It seems quite difficult for the model to learn the object pattern and the back-

Figure 4. Result of CycleGAN after 4 days of training

ground at the same time. So we propose to use image inpainting to render the synthesized images.

For the inpainting method, [7] propose the use of partial convolution, where the convolution is masked and re-normalized to be conditioned on only valid pixels. They further include a mechanism to automatically generate an updated mask for the next layer as part of the forward pass. Nevertheless, in our case, we only need to inpainting the background, so we use pairwise data of real checkout image, shown in figure.6, generated by using the bounding box of the image to feed in the model during the training time. And for the testing time, we feed in pairwise data of our synthesized images.

A key element in this implementation is the partial convolution layer. Basically, given the convolution filter W and the corresponding bias b, the following partial convolution is applied instead of a normal convolution:

$$x^{'} = \begin{cases} W^T(X \odot M)\frac{1}{sum(M)} + b, & \text{if sum(M)>0} \\ 0, & \text{otherwise} \end{cases}$$

where $\odot$ is element-wise multiplication and M is a binary mask of 0s and 1s. Importantly, after each partial convolution, the mask is also updated, so that if the convolution was able to condition its output on at least one valid input, then the mask is removed at that location, i.e.

$$m^{'} = \begin{cases} 1, & \text{if sum(M)>0} \\ 0, & \text{otherwise} \end{cases}$$

The result of this is that with a sufficiently deep network, the mask will eventually be all ones. Essentially the inpainting network, shown in Fig.7 is based on a UNet-like structure, where all normal convolution layers are replace with partial convolution layers, such that in all cases the image is passed through the network alongside the mask.

4.2. Network Structure and Algorithms

1. Choice of main framework

   There are many kinds of mainstream frameworks for object detection, and these network architectures can be classified as one-stage network and two-stage network. It is generally concluded that one-stage networks like SSD [8] and YOLO [10] run faster and two-stage networks like Faster R-CNN [11] have higher prediction accuracy. Since ACO task requires high prediction accuracy, we chose Faster R-CNN as the main framework.

2. Feature Pyramid Networks

   Due to the lack of segmentation information of our dataset, we use Faster R-CNN [11] rather than Mask R-CNN [2]. To improve the prediction accuracy of the model, FPN [6] and RoI-align are added to the Faster R-CNN framework, shown in figure.8. FPN [6] , combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections. This feature pyramid has rich semantics at all levels and is built quickly from a single input image scale, which contributes to higher prediction accuracy. Unlike the RoI pooling layer that adjusts the input proposal from RPN to fit the feature map correctly, RoI-align, proposed by He et al, simply takes the object proposal and divides it into a certain number of bins. In each bin, a certain number of points are sampled and value at those points is determined using the bi-linear interpolation, which helps avoid the misaligned problem which occurs in RoI pooling.

5. Experiments

5.1. Implementation Details

The detection part is implemented by PyTorch [9]. Each mini-batch consists of 2 image on each GPU and we set the number of detections to be 256 for each image. We use the SGD optimization algorithm to train the network, and set the weight decay to be 0.0001 and momentum is set to be 0.9. For the detection part, the initial learning rate is 0.01 for the first 120k iterations, which decays by a factor of 10

Figure 5. Our result of prediction.The color of the bounding box represent the category of the object.

Figure 6. Our input images and its mask(real world checkout images) as the input of the partial convolution neuron network

for the next 40k iterations. The training of detection part is conducted on a desktop with 1 Nvidia TITAN X GPU.

The setting for the inpainting network model is similar to that of [7]. And this network is implemented by Keras. As the instructions from the paper, Holes present a problem for Batch Normalization because the mean and variance will be computed for hole pixels, and so it would make sense to disregard them at masked locations. However, holes are gradually filled with each application and usually completely gone by the decoder stage. In order to use Batch Normalization in the presence of holes, we first turn on Batch Normalization for the initial training using a learning rate of 0.0002. Then, we fine-tune using a learning rate of 0.00005 and freeze the Batch Normalization parameters in the encoder part of the network. The training of inpainting part are conducted on a server with 2 Nvidia 1080Ti GPUs with a batch size of 2.

5.2. Evaluation Protocol

Checkout accuracy (cACC) [12] is introduced as the main evaluation metric we use for the ACO task. Given i images and k different product category, ($CD_{i,k}$) represents the counting error for a specific product category in an image, where Pi,k is the predicted label and $GT_{i,k}$ is the ground truth label.

$$CD_{i,k} = |P_{i,k} - GT_{i,k}|,$$

Then, CD_i is defined to represent the total counting error for the i-th image. CD_i = 0 suggests a fully correct predic-

Method	Our Result	Face++	University of Chinese Academy of Sciences
Synthesize cAcc	0.0008	0.0927	0.0927
Synthesize + Render cAcc	0.6332	0.5668	0.8051

Table 1. cAcc comparison with related work.

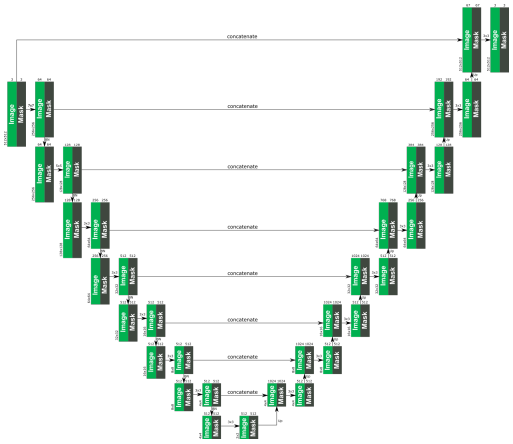

Figure 7. Network of image inpainting

Method	cAcc	threshold
Synthesize	0.0008	0.50
	0.0000	0.75
	0.0000	0.90
Render + Synthesize	0.6245	0.50
	0.6332	0.75
	0.4910	0.90
Checkout	0.9263	0.50
	0.9458	0.75
	0.9430	0.90

Table 2. cAcc of Our Result.

tion on the image.

$$CD_{i,k} = \sum_{k=1}^{K} CD_{i,k},$$

Checkout accuracy is then defined to evaluate the prediction accuracy of the model. $\delta(\cdot)$ returns 1 if and only if $\sum_{k=1}^{K} CD_{i,k} = 0$; This means that a prediction list is considered successful only if its exactly the same with the ground truth shopping list.

$$cAcc = \frac{\sum_{i=1}^{N} \delta(\sum_{k=1}^{K} CD_{i,k}, 0)}{N},$$

5.3. Inpainting Result and Analysis

The performance, shown in figure.8 of PConv network is much better than Cycle GAN. It can make up not only the background of the masked image but also the shadow and light condition of the objects by partial convolution. Such changes render the synthesized images and overcome the domain adaptation problem to some extent. However, it could be seen that there are some small black points on the predicted image. Though it has been tested that such noise does not influence the detection result, there are still much room of improvements for Inpainting Network. Due to the limit memory of GPU, we have to resize our image from a high-resolution (1840,1840) to a low-resolution (512,512). As a result, the predicted image will also have a low-resolution, which contribute to the noise on our resulted image due to partial convolution.

5.4. Detection Result and Analysis

As we mentioned in data section, we first use the checkout image to train our network as a target result. We divide the 30000 checkout images into two part: 24000 train images and 6000 validation images. After one week of training, the prediction accuracy (denoted by No Augmentation) is pretty high. As we mentioned in data section, both the training data and testing data are in same pattern: all the objects are placed in the same way, no overlapping and all the object are top-viewed, and the light condition are almost the same. The result is biased in a way that the real-world checkout settings would be much more complex.

Thus, training with synthesized images is necessary. We synthesis 25000 checkout images by ourselves using the 60000 single-object images(haven't been rendered). However, network training by such synthesized data (denoted by Syn) fails in every level because of the huge gap between the exemplars and the checkout images, confirming the problem of domain difference. It is easy to tell the difference between the images from these two domains by observing lighting conditions or shadow patterns.

As we running our test, we find that the failure is mainly because the network cannot locate the position of the object. Once the network locate the object, the accuracy of classification of this object is pretty high. So we think that the background and the shadow of objects are also very important for the network to locate the position of the object. So we further use image inpainting to learn the background and the shadow of the object.

We employ Inpainting method [7] to translate these images into the checkout image domain and get rendered images. Then. we train detectors with both the rendered images and the synthesized ones, denoted by Syn+Rendered

Figure 8. The result of our synthesized image after image inpainting..

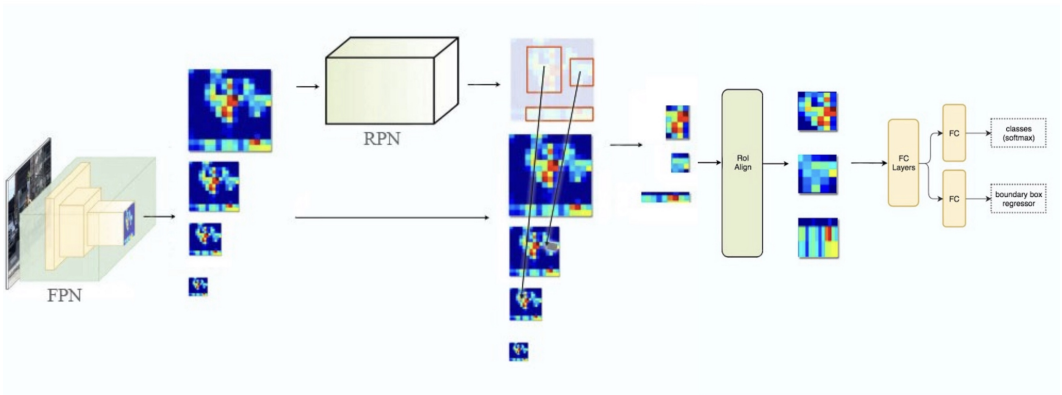

Figure 9. Network of FPN

More visual examples with different training data are shown in figure.5.

Our result are shown in Table2. and Table3., where AR is average recall, AP is average precision, IoU... area is the size of the bounding box we want to detect, where 'small' means the size of an object is less than 322, 'medium means the size of an object is less than 962 and greater than 322, 'large' means the size of an object is greater than 962.There's no objects with size less than 322, so the precision and recall are all 0.MaxDets means the threshold of max number of detection. And the result comparison are shown in Table1.

For the inference time, it takes about 0.96s on GPU and 5s on CPU for one image.

6. Conclusion

In this project, we have synthesized real-world checkout images using salient object detection algorithm and image inpainting. Multi-object checkout images were first used to train a FPN detector and have achieved 94.58% checkout accuracy. To further address the problem of domain shifting, we use synthesized checkout images as training data but the checkout accuracy was extremely. Further improvement was made by using both synthesized and rendered images as training data. The checkout accuracy increased to

Method	AR	AP	IoU	Area	maxDets
synthesize	0.056	0.030	0.50:0.95	all	1
	0.096	0.098	0.50	all	10
	0.096	0.013	0.75	all	100
	0.000	0.000	0.50:0.95	small	100
	0.040	0.000	0.50:0.95	medium	100
	0.096	0.030	0.50:0.95	large	100
Render + Synthesize	0.440	0.792	0.50:0.95	all	1
	0.834	0.986	0.50	all	10
	0.834	0.944	0.75	all	100
	0.000	0.000	0.50:0.95	small	100
	0.040	0.007	0.50:0.95	medium	100
	0.835	0.792	0.50:0.95	large	100
Checkout	0.457	0.831	0.50:0.95	all	1
	0.878	0.994	0.50	all	10
	0.878	0.960	0.75	all	100
	0.000	0.000	0.50:0.95	small	100
	0.040	0.040	0.50:0.95	medium	100
	0.878	0.831	0.50:0.95	large	100

Table 3. Average Precision and Recall of Our Result.

Figure 10. Failure Case.Two glues on the left is classified as one.

63.32%, which is higher than the 56.68% baseline accuracy. The main thing we learned from the project is how to use data augmentation technique to overcome domain shifting problem. In the future, we would like to further investigate other data augmentation techniques to improve the realism of the synthesized images and research on transfer learning which allows the model to learn new class easily.

References

[1] Q. Chen, D. Li, and C.-K. Tang. Knn matting. *IEEE transactions on pattern analysis and machine intelligence*, 35(9):2175–2188, 2013.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[3] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. H. Torr. Deeply supervised salient object detection with short connections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3203–3212, 2017.

[4] P. Hu, W. Wang, C. Zhang, and K. Lu. Detecting salient objects via color and texture compactness hypotheses. *IEEE Transactions on Image Processing*, 25(10):4653–4664, 2016.

[5] C. Li, D. Du, L. Zhang, T. Luo, Y. Wu, Q. Tian, L. Wen, and S. Lyu. Data priming network for automatic check-out. *arXiv preprint arXiv:1904.04978*, 2019.

[6] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[7] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

Figure 11. The Image on the Left is the Original Single Object Images, The Image on the Right is the Salient Map we Generate

Figure 12. Our Synthesis Images Using Single Object Training Images

[9] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[11] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[12] X.-S. Wei, Q. Cui, L. Yang, P. Wang, and L. Liu. Rpc: A large-scale retail product checkout dataset. *arXiv preprint arXiv:1901.07249*, 2019.

[13] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.