

A Solution for Product Pricing in Densely Packed Scenes

Jun Yu¹, Liwen Zhang¹, Zeyu Cui¹, Haonian Xie¹, Zhong Zhang²,
Ye Yu³, Wen Su⁴, Fang Gao⁵, Feng Shuang⁵

¹University of Science and Technology of China, ²Hefei Zhanda Intelligence Technology Co., Ltd.,

³Hefei University of Technology, ⁴Zhejiang Sci-Tech University, ⁵Guangxi University

¹{zlw1113, mg980806, xie233}@mail.ustc.edu.cn, ¹harryjun@ustc.edu.cn, ²zhangzhong@zalend.com,

³yuye@hfut.edu.cn, ⁴wensu@zstu.edu.cn, ⁵{fgao, fshuang}@gxu.edu.cn

1. Task Introduction

The task of CVPR2021 Product Pricing Challenge is to obtain the price of the product in the product box through the price tag near the product box under the premise of giving the product box. This challenge is based on the new TraxPricing dataset. The TraxPricing dataset contains 651 images and 15063 products. Among them, 585 pictures are the training set, which contains 13376 products. The remaining 66 pictures are the validation set, which contains 1687 products. These images are randomly selected in order to avoid multiple appearances of shelves from the same supermarket in a small subset.

Finally, all methods will be tested on the new test set, and the test set is announced without annotation. The test method is provided by the evaluation code published by the organizer. Our final score is 0.922806662182, and we win the first place in this competition.

2. Method



Figure 1. Our product price tag recognition dataset.

Our method can be divided into four modules, which are object detection, product and price tag matching, text recognition, and get the confidence of each price. And our solution is shown in Figure (2).

Dataset: In order to improve the accuracy of the model’s recognition on price tags, we mainly use two datasets for training. One is the SVHN (Street View House Number) dataset [8], which is derived from Google Street View House Number. The pictures in this dataset are all printed house numbers, which are similar to the task scene, so we use them for training. The other is the dataset we make based on the TraxPricing dataset. We manually crop the images in the dataset, and crop the effective price tags for training (as shown in the Figure (1)). We name this dataset the product price tag recognition dataset. At the same time, we record the coordinates of the price tag as a training set for object detection. We name this dataset the product price tag detection dataset. Both datasets contain 12736 images. Experiments have proved that after training on the above two datasets, the accuracy of our model has been greatly improved.

Object detection: In the object detection part, we use Cascade R-CNN [1] as the baseline, we replace different backbones, and find that when ResNeXt-64x4d [3] is used as the backbone, the detection accuracy of the price tag is the highest. At the same time, we replace FPN [7] with PAFPN [9] as the neck. Experiments have proved that this is effective.

The matching algorithm: When designing the matching algorithm, we naturally think that the price tag closest to the product has the greatest possibility. At the same time, in order to make the candidate price label obtained by the detection better, we select the sample with $score_{detection} > 0.1$ during object detection as the candidate price label. Suppose the coordinates of the commodity box are $(x_{min}, y_{min}, x_{max}, y_{max})$, we use the center coordinates of the bottom of the product coordinate $p = (\frac{x_{min} + x_{max}}{2}, y_{max})$ to represent this product. In the same way, for each candidate price tag obtained through object detection, we can know its coordinates as $(x_{min}, y_{min}, x_{max}, y_{max})$. Different from the product box, we use the top center coordinate $T = (\frac{x_{min} + x_{max}}{2}, y_{min})$ to

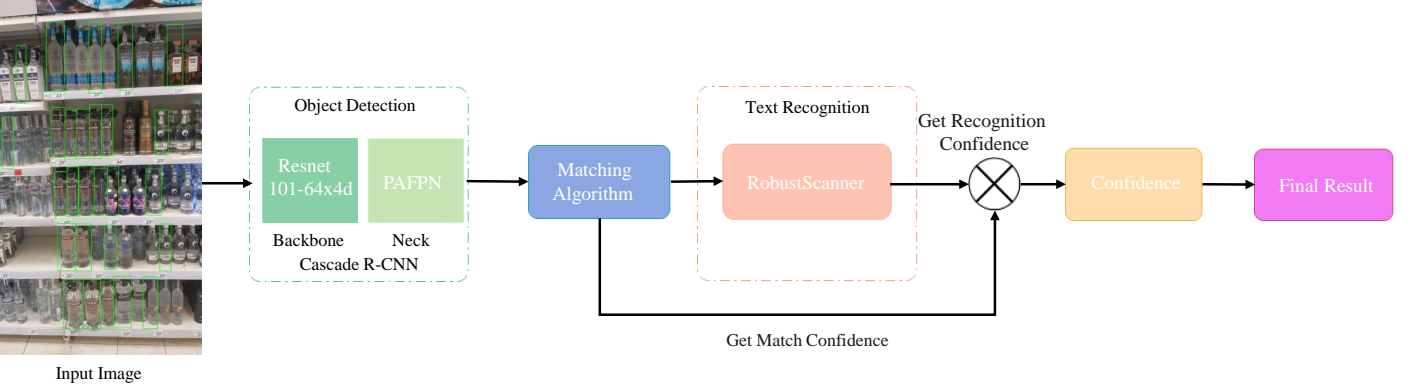


Figure 2. An overview of our solution.

Table 1. Comparisons with different methods trained on our price tag detection validation set.

Method	Backbone	Input	AP(valset)
Faster R-CNN [10]	ResNet-50 [6]	(1333,800)	71.5
Faster R-CNN	ResNet-50	(1800,900)	72.2
Faster R-CNN+PAFPN	ResNet-50	(1800,900)	72.3
DetectoRS [11]	ResNet-50	(1800,900)	73.5
Cascade R-CNN	ResNet-50	(1800,900)	74.7
Cascade R-CNN	ResNet-101	(1800,900)	75.0
Cascade R-CNN	Res2Net-101 [5]	(1800,900)	76.2
Cascade R-CNN	ResNeXt-64x4d	(1800,900)	76.3

represent the price tag.

So the problem of matching the product and the price tag is transformed into calculating the Euclidean distance from coordinate p to T_1, T_2, \dots, T_n (n is the number of price tags to be matched in a picture). If the Euclidean distance between T_i and p is the smallest, then the product matches the price tag corresponding to T_i .

Text recognition: In fact, in the text recognition part, we try to add SHVN, an external dataset containing printed numbers, but experiments show that adding the dataset did not work well. In the end, our text recognition model only use our product price tag recognition dataset. Product price tag recognition dataset, the dataset we manually produce contains the price labels corresponding to the marked product boxes and does not additionally mark the price labels corresponding to unmarked products. We selected RobustScanner [12] as our baseline, as this method proves to be very effective in identifying irregular texts such as price texts.

Confidence: The selection of confidence is very important to the result of the competition. Our selection of confidence is related to the three links of object detection, matching and text recognition. Firstly considering the impact of the matching process on confidence. We believe that the closer the price tag is to the product, the higher the probability that the price tag will match correctly. We set the distance from product coordinate p to the price tag

(T_1, T_2, \dots, T_n) as (D_1, D_2, \dots, D_n) , and the smallest in (D_1, D_2, \dots, D_n) is D_i , then the confidence of product P is

$$C = \frac{(\sum_{k=1}^n D_k^2) - D_i^2}{\sum_{k=1}^n D_k^2} \quad (1)$$

Confidence is obviously not only related to matching. In fact, it is easy to match correctly but wrong text recognition in the actual process. Considering this situation, we multiply C with the score obtained from text recognition to get the final confidence

$$C_{final} = C \cdot score_{text-recog} \quad (2)$$

3. Experiments

Table 2. The effect of selecting different object detection scores on the results of the validation set.

Score	Precision	Average Precision
0	0.764	0.767
0.1	0.765	0.768
0.3	0.764	0.769
0.5	0.764	0.771
0.7	0.763	0.772
0.9	0.753	0.779
0.99	0.617	0.802

Table 3. The effect of selecting different object detection scores and confidence on the results of the validation set.

Method	Score	Precision	Average Precision
A	(0,0)	0.764	0.767
B	(0.99,0.99)	0.617	0.802
C	(0,0.99)	0.764	0.835

Table 4. Comparisons with different methods trained on our price tag recognition validation set.

Confidence Method	Score	Precision	Average Precision
C	(0,0)	0.764	0.767
C	(0.1,0.1)	0.765	0.768
C	(0.1,0.99)	0.765	0.835
C_{final}	(0.1,0.99)	0.765	0.888

All of our experiments are implemented on PyTorch, and our object detection experiments are based on MMDetection [2]. Besides, our text recognition experiments are based on MMOCR [4].

Object detection: In the part of detecting product price tags, we train different models, as Table 1 shows. We divide product price tag detection dataset into training set and validation set according to the ratio of 9 to 1. We resize the input images to keep their shorter side being 900 and their longer side less or equal to 1800. And through experiments, PAFPN can achieve higher accuracy. The whole network is trained using the Stochastic Gradient Descent (SGD) algorithm with 0.9 momentum and 0.0001 weight decay. We train detectors with 1 GPU (2 images per GPU) for 20 epochs with an initial learning rate of 0.0025, and reduce to 1/10 after 16 and 19 epochs respectively. All other hyper-parameters follow the settings in Cascade R-CNN.

Text recognition: We divide product price tag recognition dataset into training set and validation set according to the ratio of 9 to 1. We train a text recognition model using RobustScanner. We resize the input image during training, we keep the aspect ratio of the input image unchanged, and at the same time set the height to 78 and the width to between 56 and 210, and at the same time, our optimizer chooses Adam during the training process. We train detectors with 1 GPU (192 images per GPU) for 5 epochs with an initial learning rate of 0.001, and reduce to 1/10 after 3 and 4 epochs respectively. During the test, we adjust the height to 88, and the aspect ratio remain unchanged when the width is within the range of 80 and 210.

Confidence: Good confidence can greatly improve the final result. As Table 2 shows, because the detection score is more related to matching, we set the confidence to Equation (1) instead of Equation (2). When the score is 0.1, the precision is the highest, which means that most of the false detection boxes are excluded. When the score is higher than 0.1, the precision gradually decreases as the score increases, indicating that some correct candidate price tags are excluded at this time, resulting in a match the probability of success decreases. Interestingly, although the precision has dropped

drastically, the final result has improved a lot. We believe that this is related to our confidence formula. When a price tag is matched with the closest distance to the product and the detection score is high, then the confidence corresponding to this price tag will be very high. And if the detection score of this price tag is lower than the threshold we set, it will only be found in the remaining price tags that meet the requirements. Because the minimum distance between the price tag and the product will be larger than before, the confidence obtained will be lower than the previous situation. And the result of the Table 3 shows that our idea is correct. Among them, method A and B use the image and confidence when score is equal to 0 and 0.99 to test, and method C uses the price tag image obtained when score=0 and the confidence obtained when score=0.99. It can be seen from the experimental results that this method is very effective. Finally, the experimental results are shown in the Table 4. It can be seen that C_{final} is useful, at the same time, we select the image obtained when score=0.1 and the matching confidence obtained when score=0.99 can achieve the best results.

References

- [1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.
- [2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.
- [3] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. *arXiv preprint arXiv:1707.01629*, 2017.
- [4] MMOCR Contributors. Mmocr: A comprehensive toolbox for text detection, recognition and understanding. <https://github.com/open-mmlab/mmocr>, 2021.
- [5] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip HS Torr. Res2net: A new multi-scale backbone architecture. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

- [6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [7] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
- [8] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bisacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.
- [9] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 821–830, 2019.
- [10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.
- [11] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of computer vision*, 37(2):151–172, 2000.
- [12] Xiaoyu Yue, Zhanghui Kuang, Chenhao Lin, Hongbin Sun, and Wayne Zhang. Robustscanner: Dynamically enhancing positional clues for robust text recognition. In *European Conference on Computer Vision*, pages 135–151. Springer, 2020.