Deep Learning Based Goods Management in Supermarkets

Huu-Thieu Do and Viet-Cuong Pham
Ho Chi Minh City University of Technology, VNU-HCM, Vietnam
Email: huuthieucqt@gmail.com, pvcuong@hcmut.edu.vn

***Abstract*—Goods management in supermarkets is a simple yet time and effort consuming task. To mitigate the problem, we propose a computer vision based goods management system for supermarkets using Tiny YOLOv3 model, Convolutional Neural Network and Keras. The system, with a moving camera, can detect supermarket products such as cans, bottles, bags, etc. as well as their corresponding prices and barcodes thus allowing inventory management, empty shelves and mispricing detection, etc. The system was trained on our own dataset (object detection, 16 classes, using transfer training) and the SVHN dataset (digit recognition). Experiment results show that the system works well with 92.81% mAP (mean Average Precision) for object detection and 92.28% of accuracy for digit recognition.**

***Index Terms*—object detection, digit recognition, Tiny YOLOv3, CNN, goods management**

I. Introduction

As one of fundamental computer vision problems, object detection is able to provide valuable information about images and videos, and is commonly used in many systems such as image classification [1], [2], human behavior analysis [3], face recognition [4] and self-driving car [5].

In daily life, there are many simple but repeated tasks that lead to a lot of effort to perform. Managing goods in the supermarket is one of them, with a large number of products on the shelves, it takes a long time to check and arrange goods. It is necessary to devise an automated system to do this. Robots with visual systems are good options. The problem is that robots must have good visual systems to be able to perform monitoring and inspection tasks with fast response time (almost real-time) and high accuracy. In this paper, we focus on building a visual algorithm that meets the above requirements to cater to robots. The project consists of two main parts: First products in the supermarket will be identified by Tiny-YOLOv3 network, then the price and barcode of the products will be extracted from the price tags.

II. Background

A. Object Detection Methods

Modern methods of object detection can be divided into two types: One inherits traditional object detection pipeline, generating region proposals at first and then classifying each proposal into different object classes using Deep Learning networks. Meanwhile, the other considers the object detection problem as a regression or classification problem, through a single network and a unified training function to achieve the final result directly (object categories and object locations).

The region proposal based methods include R-CNN [6], SPP-net [7], Fast R-CNN [8], Faster R-CNN [9] and Mask R-CNN [10], some of which are correlated with each other (e.g. SPP-net modifies RCNN with a SPP layer). The regression/classification based methods mainly include MultiBox [11], AttentionNet [12], G-CNN [13], YOLO [14], SSD [15], YOLOv2 [16], YOLOv3 [17], [18]. Especially, both types are related when approaching the use of anchor boxes introduced in Faster R-CNN. Comparing the above methods about mAP (mean Average Precision) and fps (frame per second) on the MS-COCO dataset [19], [20], we can see that YOLOv3 model has a high mAP while the frame rate of model is the highest of the methods (90fps). Therefore this model is very suitable with real-time applications that require fast response speed.

B. YOLOv3

J. Redmon and A. Farhadi proposed a new version of YOLO [14] and YOLOv2 [16] called YOLOv3 [17], using all of the high level features of the feature maps to predict both scores of object categories and the bounding boxes. YOLOv3 divides the input image into an S × S grid and each grid cell is responsible for predicting the object if the object bounding box center is in that grid. Each grid cell predicts B bounding boxes and their corresponding confidence scores. In the same time, C- - conditional class probabilities is predicted in each bounding box. It should be noted that only the contribution from the grid containing the object is calculated.

Comparing to YOLO, YOLOv3 has additional techniques to improve mAP and fps as well as make training process more stable. Such techniques are Batch Normalization, anchor boxes, dimension clusters, direct location prediction and Feature Pyramid Networks (FPN) [21].

C. Tiny-YOLOv3

The backbone network of YOLOv3 uses the Darknet-53 network [17]. The network is too complex and

Manuscript received August 28, 2020; revised February 5, 2021.

doi: 10.12720/jait.12.2.164-168

requires very strong computational power of hardware. This result in the detection speed is greatly affected. Tiny-YOLOv3 is a simplified version of YOLOv3. The backbone network of Tiny-YOLOv3 has only 13 convolutional layers and 6 pooling layers. The pooling layers in Tiny-YOLOv3 are used to replace YOLOv3's stride 2 convolutional layers, which aim to reduce the size of feature maps and not to make the network too complicated (because pooling has no learning weights). The simplified network improves the detection speed, but it also loses some of the detection accuracy. However, with its compact network and acceptable accuracy, Tiny-YOLOv3 is very suitable with real-time applications.

III. Products Detection in Supermarkets Using Tiny-YOLOv3 Model

A. Data Preparation

Images were collected from the internet as well as photographed by the authors. The training set has 5352 images, including 16 classes. The test set consists of 1077 images divided into 16 classes. The classes include Coca Cola, Mirinda, Sting, Fanta, Redbull, Saigon Special, Tiger, Highland Coffee, Aquafina, Romano bottles, Oishi Pillow bags, Hao Hao instant noodles, TH milk bags, product price area and barcode area. All data are self-labeled by the author. The training set is divided into two sets: training set and validation set with a 80:20 ratio.

B. Data Augmentation

To avoid overfitting, images will be augmented by changing the color values in the HSV color space. Call *H', S', V'* as the values in the color space after augmentation, these values are determined as follows:

$$H' = random(0.9*H, 1.1*H) \quad (1)$$

$$S' = random(S, 1.5*S) \text{ or } S' = S/random(1, 1.5) \quad (2)$$

$$V' = random(V, 1.5*V) \text{ or } V' = V/random(1, 1.5) \quad (3)$$

Images are also augmented by flipping horizontally, cropping in the information area and shifting. The images are resized to size of 416 pixels × 416 pixels.

C. Training

In training process, we use these techniques:

Parallel training: Using CPU to process data (data augmentation and normalization), GPU to train data. Training in mini-batch mode with batch_size = 32.

Transfer learning: Using pre-trained model Tiny-YOLOv3 for COCO dataset. Using transfer learning helps to save training time.

Bottleneck-training: Freezing the whole network and training only the last two layers. Firstly, the data will be passed through the network until they arrive the last two layers. Then these feature maps will be saved and taken as inputs for the last two layers. Finally, we just need to train with two saved inputs. If the training process is not good, we will gradually unfreeze and train the upper layers.

Optimizer: Using Adam optimization algorithm. The learning rate is adjusted by Plateau function with base 10.

Regularization: Using L2 regularization to avoid overfitting.

Using *Early Stopping* to stop training model with patient epochs = 10. It means if the loss value on validation set doesn't decrease after 10 epochs, training process will be stopped. We also use checkpoint to save the model weights at the lowest validation loss.

IV. Detecting Product Price and Barcode in Price Tag

A. Detecting Price Area and Barcode Area on Price Tag

We still use Tiny-YOLOv3 model for detecting product price areas because of general characteristics as well as algorithm optimization. We will treat price and barcode areas as training classes, then all objects will be detect only once when we feed input image through network.

B. Extracting the Numeric Area in Price Area

Product price area after being extracted is as follows (Fig. 1):

Figure 1. Price area after being cropped.

Using classical image processing techniques such as thresholding and filtering to enhance the image quality after cropping. Then the contours will be got from this image. The numeric areas will be chosen from these contours as follows:

- Rearrange these contours from left to right position.
- Select contours whose height is 0.3 times greater than the height of the image.
- Select contours whose height is 1.2 times greater than the width of this contour.

C. Digit Recognition Using CNN

We use CNN for building a digit recognition model. The model includes 5 convolutional layers, 4 max pooling layers and 2 Fully Connected layers. Dropout is used to avoid overfitting with a dropout rate of 0.25. Model was trained on SVHN dataset includes 33404 training images and 13070 test images. After training, the accuracy of model on validation set is 95.02%. Training our model on SVHN dataset make it more general and able to recognize the digit with different fonts.

D. Detecting and Reading the Barcode on Price Tag

Similar to the way to detect price areas, barcode areas are also cropped from the price tags by using Tiny-YOLOv3. The barcode after cropping out is pre-processed using median filter, Otsu thresholding [22] and de-skewed to make barcode area straight. The code will then be read by pyzbar library.

After detecting information of products, product prices and barcode information from shelves, we combine all of

them into a goods management algorithm. The algorithm includes the following tasks:

- Pairing the price tags with its products.
- Determining whether the price and barcode information match the product and whether the product is in shelves.
- Provide right shelf information by reading barcode.

The algorithm works as follows: First, the center coordinates of the price areas on price tags will be collected. Next, the algorithm will scan the coordinates of the products in the image (under the price tags). If the position of the product compared to a price area is less than 200 pixels: the product information, the price area and the barcode area under it will be paired. From the matching pairs, we will first check if there is a product in the shelf. Next, the barcode will be read to get information about the product, this information will be checked with the existing product. If the barcode is too small or dim and the system cannot read, we will continue to check by considering whether the product matches the price. A shelf has wrong arrangement if there are no products on shelf or products do not match with the barcode information and the product prices. Error shelves will be notified and right information of these shelves will be read and saved.

V. Experiment Results

The entire experimental platform configuration in this paper is shown in Table I.

TABLE I. Experimental Platform Configuration

Names	Related configuration
Operating system	Windows
CPU/GHz	Intel Core i5-5200/2.2
RAM/GB	8
GPU	NVIDIA GeForce 830M
GPU acceleration library	CUDA 10.0, CUDNN 7.4

A. Object Detection

Result of classification (Fig. 2):

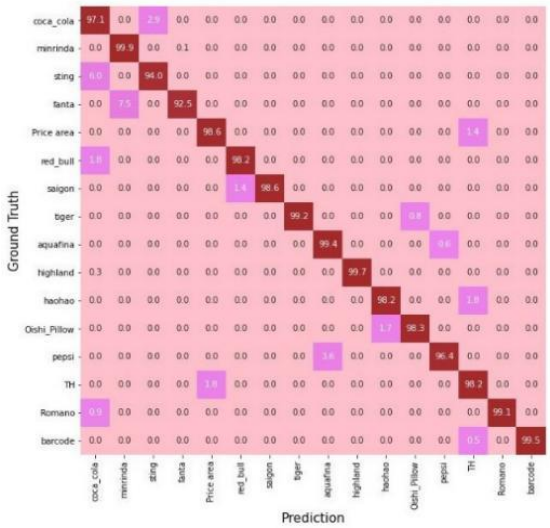

Figure 2. The accuracy of classes prediction.

Result of bounding boxes detection and mean average precision with threshold of 0.7 are shown in Table II.

TABLE II. Results of IOU and MAP

Class	IoU	Average Precision
Coca Cola (can)	86.81%	97.68%
Mirinda (can)	85.88%	98.02%
Sting (can)	87.43%	94.03%
Fanta (can)	86.68%	92.53%
Product Price area	80.09%	82.54%
Redbull (can)	89.15%	97.13%
Saigon Special (can)	87.76%	97.53%
Tiger (can)	89.20%	98.23%
Aquafina (bottle)	80.05%	80.16%
Highland Coffee (can)	88.65%	97.14%
Hao Hao instant noodle (bag)	85.92%	95.27%
Oishi Pillow (bag)	86.04%	97.26%
Pepsi (can)	83.45%	93.22%
TH true milk (bag)	84.49%	84.62%
Romano (bottle)	85.50%	87.76%
Barcode area	82.91%	91.90%

Mean IoU = 85.63%.
Mean Average Precision: 92.81%
Result of frame speed: 8-9 fps when running on GPU

B. Digit Recognition

Accuracy of price tags recognition: 92.28%
Accuracy of each digit recognition (Fig. 3):

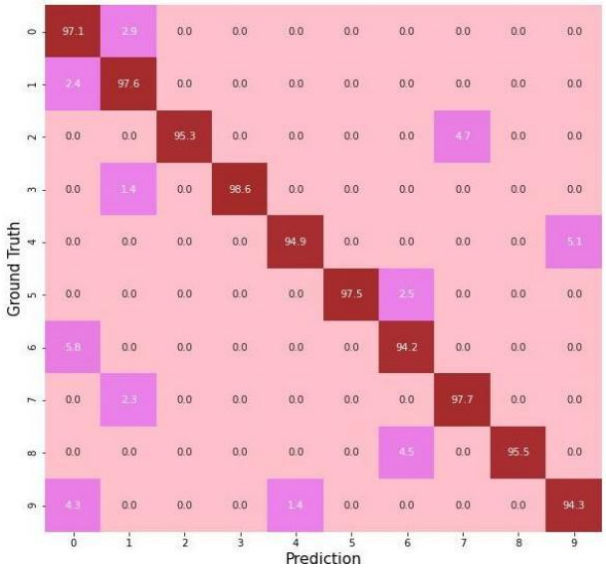

Figure 3. Result of digit recognition.

Speed of prediction: 0.02 - 0.04s per image.

Remark: The model works well in many different environmental conditions (in term of brightness, image quality, noise, etc.). Model is highly effective when the digit size is large enough (nearly 60 pixels × 30 pixels). It has a high speed of prediction.

C. Barcode Detection

Accuracy: 90.02%

Speed of prediction: 0.012s per image
Remark:

- Reading barcode is sensitive to environmental conditions such as image brightness, noise,…
- The speed of prediction is high and suitable with real-time system.

D. Result of Pairing Products and Price Tags

Case 1: There are no products on shelves. When there are no products on the shelf, the algorithm will notify *No object* at this shelf. In the Fig. 4(a) below, there are no products on shelves (shelves 0 and 1), so the No Object notification is at position 0,1.

Case 2: Product does not match price or barcode information. When the product do not match the barcode information (or do not match the price of the product), the system will notify *Not Match* at that location. In the Fig. 4(b) below, Sting costs VND 11,000 equivalent to the price and matches the information on the barcode so the system will not report an error.

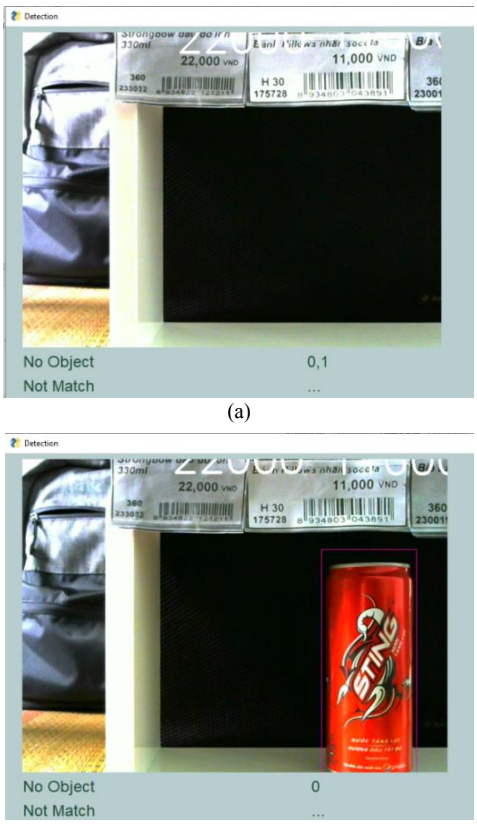

Figure 4. Result of pairing algorithm (a) case 1 (b) case 2

The information about the date, time, the error shelf and the correct product of that shelf will be saved to an Excel file for monitoring.

VI. Conclusion

We proposed a deep learning based computer vision system for goods management in supermarkets which is capable of detecting goods (14 classes) on shelves as well as their corresponding prices and barcodes. The system, using Tiny YOLOv3 model, Convolutional Neural Network and Keras, can efficiently support inventory management, detection of empty shelves and mispricing, etc. Experiment results show that the proposed system works well with a high accuracy and an acceptable running time. Further investigations are in progress implementing an end-to-end system combining the whole process, improving the system performance, and incorporating a mobile system to obtain a truly autonomous operation.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Viet-Cuong Pham designed and directed the project. Huu-Thieu Do conducted the experiment and analyzed the data. Both authors discussed the results and contributed to the final manuscript.

References

[1] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. the 22nd ACM International Conference on Multimedia*, 2014.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097-1105, 2012.

[3] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[4] Z. Yang and R. Nevatia, "A multi-scale cascade fully convolutional network face detector," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[5] C. Chen, A. Seff, A. L. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proc. the IEEE International Conference on Computer Vision*, 2015.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2014

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904-1916, 2015.

[8] R. Girshick, "Fast R-CNN," in *Proc. the IEEE International Conference on Computer Vision*, 2015.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards realtime object detection with region proposal networks," in *Proc. the 28th International Conference on Neural Information Processing Systems*, 2015, pp. 91-99.

[10] K. He, G. Gkioxari, P. Dollar, and R. B. Girshick, "Mask R-CNN," in *Proc. the IEEE International Conference on Computer Vision*, 2017.

[11] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[12] D. Yoo, S. Park, J. Y. Lee, A. S. Paek, and I. So Kweon, "Attentionnet: Aggregating weak directions for accurate object detection," in *Proc. the IEEE International Conference on Computer Vision*, 2015.

[13] M. Najibi, M. Rastegari, and L. S. Davis, "G-CNN: An iterative grid based object detector," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. European Conference on Computer Vision*, 2016.
[16] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," arXiv:1612.08242, 2016.
[17] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.
[18] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," arXiv:1807.05511, 2019.
[19] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. European Conference on Computer Vision*, 2014.
[20] J. Hui. (2018). Object detection: Speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3). [Online]. Available: http://medium.com
[21] T. Y. Lin, P. Dollar, R. B. Girshick, K. He, B. Hariharan, and S. J.Belongie, "Feature pyramid networks for object detection," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
[22] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Sys. Man. Cyber*, vol. 9, no. 1, pp. 62-66, 1979.

Huu-Thieu Do received the B.Eng in Automation and Control Engineering from Ho Chi Minh City University of Technology, VNU-HCM, Vietnam in 2020. He currently works as AI Engineer at Emage Development VietNam. His research interests include computer vision, machine learning, robotics and their applications.

Viet-Cuong Pham received his B.E. and M.E. degrees from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM, Vietnam, in 2001 and 2003, respectively, and Ph.D. degree in electrical engineering from National Cheng Kung University, Taiwan, in 2013. From 2013 to 2015 he worked as postdoctoral researcher at National Cheng Kung University, Taiwan. Since 2015 he has been a faculty member of Department of Control Engineering and Automation, HCMUT, VNU-HCM, Vietnam. His research interests include computer vision, machine learning, deep learning, mobile robot exploration, localization and mapping.