A Hybrid Model for Container-code Detection

Cai Sun, Kuikun Liu, Haoyuan Chi, Mesoume Zareapoor
School of Electrical Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China

Abstract— **In this paper, we propose a container code detection algorithm that combines PSENet and CRNN in a manner that ensures that the obtained images are greatly affected by the other containers as well as text information in the picture. The proposed algorithm divided into three parts: object detection module, text detection module and text recognition module. At the initial step, the object detection module is used to calculate the position of those areas in container code that needs to be predicted, we then use the text detection module based on pixel segmentation, and finally obtain the container code through the end-to-end text recognition module. This algorithm is able to detect different container code for a vertical and horizontal scaling. We show that, in a complex multi-container scenario, the detection performance is good, and highly stable when trained with multi-angle container, achieving relative improvement of up to 95%.**

Keywords-Container-code; Object Detection; Text Detection; Text Recognition; Deep Neural Networks; Computer Vision.

I. INTRODUCTION

With the rapid progress of science and technology and also a significant growth for exports and imports, the development of the logistics and transportation industry is necessary and accordingly the containers which are used in this industry play an important role in maritime transportation. With the vigorous development of maritime transport, a more intelligent and faster algorithm is needed to obtain container code information, so as to realize the informatization, intelligence and modernization of large-scale containers.

The object detection algorithms based on deep neural networks are divided into two categories: region proposal and regression-based. Region proposal such as Faster R-CNN [1] and R-CNN [2] has higher accuracy than regression-based algorithms such as SSD [4] and YOLO [5]. For the container text detection, traditional scene text detection algorithms are mainly use [6]-[12], such as SWT [11] and MSER [12]. There are also scene text detection algorithms based on deep neural networks [13]-[18]. The text box generated by the traditional scene text detection algorithm is accurate, but the anti-interference and robustness is not as high as the deep neural networks models, e.g., for example, traditional algorithms not able to properly detect the incomplete characters. At the same time, the text box generated by the deep neural networks algorithm will have varying degrees of missing or limited to the positioning of the horizontal container code. To detect a container code segmenting the characters is another method [19],[20] that widely used. These models have their own limitations. Most of the current code detection methods [21]-[23], are not effective in detecting vertical codes or complex text background. Therefore, a robust algorithm for detecting container codes is highly required. A sample of container code is shown in Fig.1. This container code is composed of four letters (CAIU), seven digits (7075488), and four letters (45G1). The first four letters are the abbreviation of the company, and the next seven digits are the identity of container. The seventh digit is the container check code, which can be calculated from the first four letters and next six digits to verify the identity of container code. The last four digits represent the size and type of the container.

Figure 1. Illustration of container code

The current container code detection algorithm mainly contains three parts: object detection module, text detection module and text recognition module. The object detection module and text detection module are algorithms based on computer vision and deep neural networks, which have high robustness and accuracy. The text recognition module uses an end-to-end recognition algorithm for characters. The combinations of the three modules show high accuracy in identify the container codes.

II. TECHNICAL DETAILS

A. Architecture Overview of Algorithm

The proposed model consists of, Faster R-CNN, modified PSENet and CRNN approaches. These relationships are shown in Fig. 2. The object detection module adopts Faster R-CNN to effectively capture the container code area, and eliminate the interferences in the context of the multiple containers and complex text background. Then, the container images are given as input to the container code area to obtain the bounding box of the container code. The container code positioning is achieved using the bounding box. Moreover, for the text detection, we modified the PSENet to detect the tight and slant texts. However, our modification does not restrict by the type of container, and hence the detection results can be obtained for both horizontal and vertical scaling. The detections thus obtained are combined into the text recognition module. The text recognition module used the CRNN, to get the container

978-0-7381-0545-1/20/$31.00 ©2020 IEEE

code recognition. Finally, the container code information is obtained through the rules of container code. These three learning steps are unified as an end-to-end framework, to be mutually-beneficial to each other.

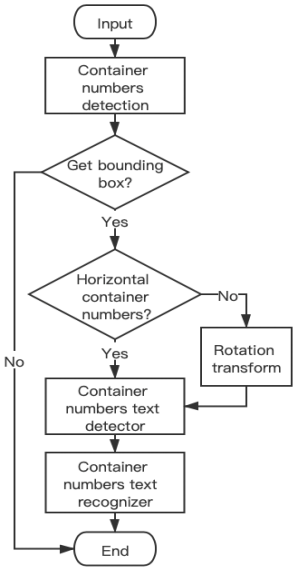


Figure 2. Flow chart of algorithm

The combination can eliminate the interference of the other containers, reduce the difficulty in detection and recognition, and thereby improve the accuracy of the container code recognition.

B. *Object Detection Module*

The images of having many containers in background such as stacking yards have the characteristics of "multiple containers and multi-text information interference". The existing container code recognition approaches do not eliminate these interferences, and also are difficult to detect container code. They are often inaccurate or need manual intervention. We integrated the object detection into the proposed architecture to facilitate container code detection and recognition.

Region-based object detection algorithms are generally represented by R-CNN, Fast R-CNN, Faster R-CNN, etc., and they are composed of two parts: Region Proposal and Target Classification. In our work, we use Faster R-CNN to extract the container code area, and then detect the particular text in the extracted area to significantly reduce the interference of other containers and text information. The overall framework of Faster R-CNN is shown in Fig. 3. As it observes, it uses a convolutional layer to extract the meaningful features from the container image, and then passed the obtained features through the region proposal network (region proposal network, RPN) and the fully connected layer.

The RPN network is based on the anchor box that obtained by softmax screening, to classify whether the regional target of the candidate box belongs to the foreground or the background of the container. Hence, with this classification results, it can directly generate the container candidate box. Since, the candidate frames have different scales, thereby the Region of Interest pooling layer (RoI Pooling) is used to resize the candidate frames and make it applicable with different input sizes. Finally, the regression network is used to obtain the position of each container candidate frame. In addition, we used L1 loss function for the regression network, and the Cross Entropy Loss function for the classification performance.

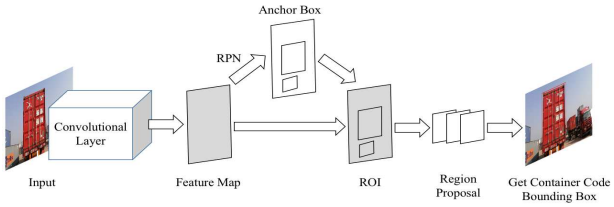


Figure 3. Faster R-CNN object detection algorithm framework

Through the trained Faster R-CNN model, the container code area can be obtained. The text information and container information can be excluded by obtaining the bounding box of the container code and the model achieved the role of acquiring the main area.

C. *Text Detection Module*

Regression-based and pixel-based are two main ways to detect container code. The regression detection algorithm is not applicable for the detecting the slant container code, while the pixel-based approaches perform better than regression approaches on the slant container code, also they can detect the horizontal and vertical recognition without being restricted by the recognition scene and container type.

The PSENet (Shape Robust Text Detection with Progressive Scale Expansion Network) algorithm is a pixel-based text detection algorithm that can detect any shape of container code. The container code is used to detect the slant container images; also the container code of the adjacent texts is used to better segment the meaningful features. The network framework is shown in Fig. 3 below. By down-sampling the container images, 4 container thumbnails are generated, subsequently, four 256-channel feature maps are extracted using ResNet. Based on the FPN network [25], the low-level features and the high-level features are then fused, to obtain four-layer feature maps. These fused-features with the down-sampling, will be up-sampled to the scale of p_2. At the end, we can get 1024-channel output image F, which are having same sizes as the original image. The objective result is as:

$$F = C(p_2, p_3, p_4, p_5) = p_2 \| U_{p*2}(p_3) \| U_{p*2}(p_4) \| U_{p*8}(p_5) \quad (1)$$

Then through the (3*3) convolutional layers followed by ReLU and n (1*1) upsampling sigmoid layers, the projection of F in the n branches is used to obtain n container code

segmentation results. The obtained results are the prediction of the container text S_1 -S_n images. Among these components, S_i is the segmentation mask of container code instances with different proportions, S_1 represents the smallest container code segmentation mask, and S_n represents the original container segmentation mask. After obtaining n container segmentation masks, the S_1 is gradually expanded into the other sets (S_2, …, Sn), through a progressive algorithm, and finally the container code connected domains are obtained.

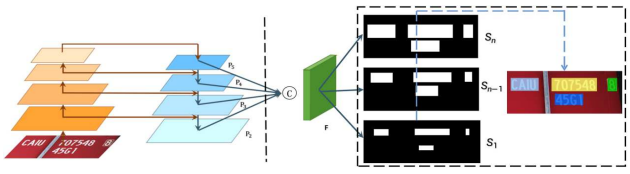


Figure 4. Illustration of PSENet

- Progressive scale expansion algorithm

As shown in Figure 5, CC stands for the function of finding connected components and EX stands for scale expansion algorithm. In Figure 5(a), S_1 represents the detection result of the smallest container code. Based on these, there are four connected areas as, C= {c1, c2, c3, c4} which are embedded into the model. In Figure 5(b) each color corresponds to the one of the connected regions (i.e., there is four colors that corresponds to the four connected areas). Subsequently, we determine if the pixel adjacent C is placed in the S_2, and if the response is yes, we then merge it into Figure 5(b). We do the same process for all S. At the end, the connected areas which are spotted with different colors (in Figure 5(d)) are used as the final container code detection result.

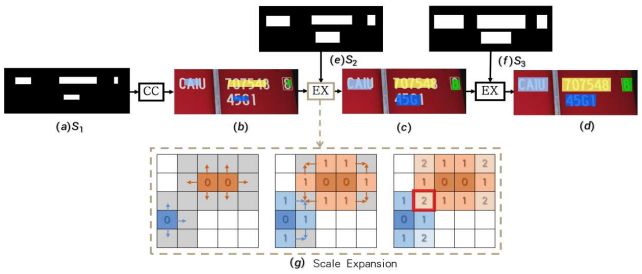


Figure 5. Illustration of Progressive scale expansion algorithm

- Improvement of PSENet

As we mentioned earlier, the PSENet consists of three parts: feature extraction layer, feature fusion layer and output layer. One of the most time-consuming steps is computing the box-number area. To this end, we modified the PSENet to make the training easier. The original network uses ResNet_152 to extract features. Due to the simple structure of the box number area, we do not need more numbers of convolutional layers. We simplify the model by using the PVANet [24] network structure to extract the characteristics of the container code. PVANet is a lightweight network with a few channels. PVANet uses the ReLU function to speed up the computation; it also uses the inception structure to have large and small receptive fields, and it uses Hypernet to detect multi-scale container codes. The basic network in this article is trained on the ImageNet dataset.

- Loss of PSENet

The loss function of PSENet is shown as follows:

$$L = \lambda L_C + (1-\lambda) L_S \tag{2}$$

The loss function has two parts: L_C represents the loss of the entire container code instance (S_n), L_S represents the loss of the container code instance ($S_{1,\,2\ldots,\,n-1}$) after scaling, where λ is balance parameters between L_C and L_S. However, a large proportion area in the picture is fully occupied by the constructed container code; therefore, using binary cross entropy loss make the prediction result more biased towards the non-container-code area. The dice coefficient loss function (3) is used here:

$$D(S_i, G_i) = \frac{2\sum_{x,y}(S_{i,x,y} * G_{i,x,y})}{\sum_{x,y} S_{i,x,y}^2 + \sum_{x,y} G_{i,x,y}^2} \tag{3}$$

Here, $S_{i,x,y}$ represents the pixel value of the segmentation result S_i at (x, y), and $G_{i,x,y}$ represents the pixel value of ground truth G_i at (x, y). Considering that many container code areas are similar to container railings, thus in order to avoid false detections, OHEM is used to accurately distinguish the container code areas. L_C is the loss of segmenting container text and non-text regions, M are the training mask of OHEM, and the L_C is computed as:

$$L_C = 1 - D(S_n \cdot M, G_n \cdot M) \tag{4}$$

L_S loss is used to reduce the text instances. The original text area surrounds the reduced text area, so the pixels in the non-text area could be ignored to avoid of pixel redundancy. We calculate the L_S as:

$$L_S = 1 - \frac{\sum_{i=1}^{n-1} D(S_i \cdot W, G_i \cdot W)}{n-1} \tag{5}$$

where W represents the mask of the text area in S_n, and $S_{n,x,y}$ represents the pixel value of (x, y) in S_n (6).

$$W_{x,y} = \begin{cases} 1, if\ S_{n,x,y} \geq 0.5 \\ 0, otherwise \end{cases} \tag{6}$$

D. Text Recognition Module

The outputs of the container code and the text recognition model are having different sizes. The boxes obtained by the container code detection model are required for preprocessing task. First, the vertical container text box is rotated 90° angles to perform a horizontal conversion, so that all the container code boxes will be obtained in a horizontal state. We then use perspective transformation to change the irregular quadrilateral text boxes into a horizontal rectangular box, in order to smooth the CRNN training for the text recognition processing. Here we normalize the container text box, and scale the height of the container code image sequence to 32 pixels.

The network structure of CRNN is shown in the table I.

TABLE I. CRNN NETWORK STRUCTURE

Layers	Parameters	Layers	Parameters
Input	W×32×1（W×H×C）	Conv	3×3×256, s=1, padding=same
Data Augmentation	-	BN	-
Conv	3×3×64, s=1, padding=valid	MaxPool	2×2, s= [2,1]
Conv	3×3×64, s=1, padding=same	Conv	3×3×512, s=1, padding=same
BN	-	Conv	3×3×512, s=1, padding=same
MaxPool	2×2, s= [2,2]	BN	-
Conv	3×3×128, s=1, padding=same	MaxPool	3×1, s= [3,1]
Conv	3×3×128, s=1, padding=same	Map-to-Sequence	-
BN	-	BLSTM	512
MaxPool	2×2, s= [2,1]	BLSTM	512
Conv	3×3×256, s=1, padding=same	CTC	-

CRNN consists of convolutional layer (CNN [26][27]), recurrent layer (RNN) and transcription layer (CTC). Among them, the convolutional layer is used for image feature extraction, and the convolutional layer is composed of four Conv*2-BN-MaxPool. The double convolution is used for feature extraction on the container code image. BN can solve the problem of gradient disappearance and gradient explosion. MaxPool can Speed up training and extract deep features. The Map-to-Sequence layer connects the convolutional layer and the recurrent layer, which vectorizes the features. The recurrent layer is composed of two-way LSTM, which can identify feature vectors to obtain the probability distribution of each feature. The transcription layer uses forward and backward algorithms and CTC to calculate the highest probability label sequence, so that the input sequence and the output sequence correspond. The container code is only composed of numbers and letters. The identification result of the container code is obtained by the value of the tag in the dictionary index containing only numbers and letters.

- CRNN loss function

The recurrent layer trains the convolutional layer and LSTM through CTC. We assume $\chi \in \{IMG_i, label_i\}$ as the training set and y_i as the output of the CRNN network, where IMG_i is the container code text image, $label_i$ is the container code text label, and the model loss function is defined as (7):

$$CTCLoss = -\sum_{IMG_i, label_{i\in\chi}} \log_p (label_i \mid y_i) \qquad (7)$$

III. EXPERIMENTAL RESULTS

A. Experimental Environment and Data

Our experiments are implemented based on Pytorch deep learning framework and using Python. All of our experiments are running on an NVIDIA GTX 2080Ti GPU and Intel (R) i7-6850 K CPU (3.60 GHz).

Experiment with container pictures obtained from the camera at the exit of the yard vehicle. The container code object detection data set uses rectangular frames to mark the container code areas. The data format is COCO data set. Then, for the labeling of the object detection dataset, the image of the location of the container code is intercepted. The vertical container code image is rotated 90° counterclockwise. The container text detection data set adopts the ICDAR2017 label format. Then, on the basis of the text detection labeled data, via perspective transformation, the four-point perspective of the irregular quadrilateral is transformed into the smallest bounding rectangle to obtain the container code. The obtained images of the container codes are all horizontal, and manually labeled. Overall, there are 1000 horizontal and 1000 vertical container code images.

The text detection dataset contains 1000 horizontal container code images, and 3000 text boxes. There are also 1000 vertical container code images and 3000 text boxes in total. The training set and the test set are divided at a ratio of 9:1.

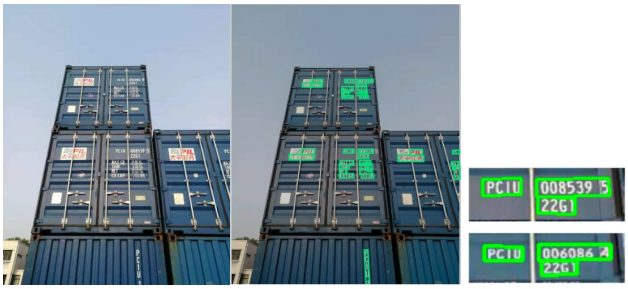
B. Analysis of Results

The results of our proposed model are compared with Algorithm 1 which is the PSENet recognition algorithm combined with the object detection, Algorithm 2, is the PSENet detection algorithm with no using object detection, Algorithm 3, is the traditional text detection MSER, and the Algorithm 4 is the MSER combined with the object detection. We used precision, recall, and F-measure to evaluate the performance of the models. For each container code, the correct correction of text area considers as TP, the incorrectly predicted container text considers as FP, and the missed container text area as FN. Where, correct means that the intersection ratio of the container code preselected box and the container box number is greater than 0.7. The results are shown in Table II.

TABLE II. COMPARISON OF THREE CONTAINER CODE DETECTION

Algorithm	Precision (%)	Recall (%)	F-measure (%)
Faster R-CNN Improved PSENet	96.6	97.1	96.8
Faster R-CNN PSENet	95.8	96.0	95.9
PSENet	91.2	88.2	89.7
MSER	71.3	59.2	64.7
Faster R-CNN MSER	82.9	81.1	82.0

It can be seen that the F-measure of algorithm combined with object detection is higher than that without the object detection, and deep learning algorithm is better than traditional algorithm. Low recall rate indicates that a text detection model without the object detection module show miss-detection. Therefore, our proposed model not only meets the real-time requirements, but also has higher detection performance. Some sample test results are shown in Fig. 6.



(a) (b) (c)

Figure 6. The effect of horizontal contaienr code detection(a):origin container image;(b):PSENet without Faster R-CNN;(c):PSENet with Faster R-CNN.

In this paper, an end-to-end container text recognition model is compared with the traditional text recognition algorithms. Specifically, the container code characters are segmented using MSER, and then the characters are recognized one by one through the neural network based on ResNet [28]. We present the results of proposed model on horizontal and vertical container codes in the table III.

TABLE III. COMPARISON OF TWO CONTAINER CODE RECOGNITION

Algorithm	Horizontal Container Code Precision (%)	Vertical Container Code Precision (%)	Average Precision (%)
CRNN	98.3	98.5	98.4
MSER ResNet	93.5	92.9	93.2

As the results show, the proposed end-to-end text recognition model highly performances on recognizing the horizontal and vertical container codes.

IV. DISCUSSION AND CONCLUSION

In this paper we proposed a container code detection algorithm based on object detection which is combined with improved PSENet and CRNN. The proposed model highly performs in multiple containers code detection, complex environment or detecting horizontal and vertical container codes.

By analysis the experimental results, the following conclusions and insights can be drawn:

- Through the object detection of the container code area, the interference of other containers and other text information can be ignored to improve the detection accuracy of container code detection;
- By improving the PSENet text detection network, the horizontal and vertical container code can be detected, by changing the network structure. The improved PSENet increases the detection speed without losing accuracy;
- Using CRNN the end-to-end container text recognition model can recognize horizontal and vertical container code efficiently.

The algorithm has high performance in container code detection in complex situations, but there are still some problems, such as missing text. This paper has the following optimization directions in the future:

- Improve the number and quality of special samples;
- Enhance the generalization and accuracy of learning;
- Optimize the network structure and the weight of the loss algorithm;
- Adjust the internal parameters of the PSENet algorithm;
- Extend the algorithm for text detection.

REFERENCES

[1] Ren S , He K , Girshick R , et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 39(6).

[2] GIRSHICK R, DONAHUE J, DARRELL T, et al. Region-based convolutional networks for accurate object detection and segmentation[J].IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016, 38(1), 142–158.

[3] GIRSHICK, R. Fast R-CNN[J]. The IEEE International Conference on Computer Vision, 2015, 1440-1448.

[4] LIU W, ANGUELOV D, ERHAN D, et al. SSD: single shot multibox detector[J]. European

[5] Conference on Computer Vision, 2016, 9905: 21-37.REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: Unified, real-time object detection[C]. The IEEE Conference on Computer Vision and Pattern Recognition, 2016, 779-788.

[6] Chen X , Yuille A L . Detecting and reading text in natural scenes[C]// Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. IEEE, 2004.

[7] Liu C , Wang C , Dai R . Text detection in images based on unsupervised classification of edge-based features[C]// Eighth International Conference on Document Analysis & Recognition. IEEE Computer Society, 2005.

[8] Epshtein B , Ofek E , Wexler Y . Detecting Text in Natural Scenes with Stroke Width Transform[C]// Computer Vision & Pattern Recognition. IEEE, 2010.

[9] W. Huang, Z. Lin, J. Yang and J. Wang, "Text Localization in Natural Images Using Stroke Feature Transform and Text Covariance Descriptors," 2013 IEEE International Conference on Computer Vision, Sydney, NSW, 2013, pp. 1241-1248, doi: 10.1109/ICCV.2013.157.

[10] Neumann L., Matas J. (2011) A Algorithm for Text Localization and Recognition in Real-World Images. In: Kimmel R., Klette R., Sugimoto A. (eds) Computer Vision – ACCV 2010. ACCV 2010. Lecture Notes in Computer Science, vol 6494. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-19318-7_60

[11] B. Epshtein, E. Ofek, and Y. Wexler. "Detecting text in natural scenes with stroke width transform." In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 2963-2970. IEEE, 2010.

[12] H. Chen, S.S. Tsai, G. Schroth, D.M. Chen, R. Grzeszczuk, and B. Girod. "Robust text detection in natural images with edge-enhanced maximally stable extremal regions." In Image Processing (ICIP), 2011 18th IEEE International Conference on, pp. 2609-2612. IEEE, 2011.

[13] Tian Z , Huang W , He T , et al. Detecting Text in Natural Image with Connectionist Text Proposal Network[C]// European Conference on Computer Vision. Springer, Cham, 2016.

[14] Zhou X , Yao C , Wen H , et al. EAST: An Efficient and Accurate Scene Text Detector[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017.

[15] Shi B , Bai X , Belongie S . Detecting Oriented Text in Natural Images by Linking Segments[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017.

[16] Deng D , Liu H , Li X , et al. PixelLink: Detecting Scene Text via Instance Segmentation[J]. 2018.

[17] Long S , Ruan J , Zhang W , et al. TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes[J]. 2018.

[18] Li X , Wang W , Hou W , et al. Shape Robust Text Detection with Progressive Scale Expansion Network[J]. 2018.

[19] P. He, W. Huang, Y. Qiao, C.C. Loy, and X. Tang. "Reading Scene Text in Deep Convolutional Sequences." In AAAI, pp. 3501-3508. 2016.

[20] B. Shi, X. Bai, and C. Yao. "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition." IEEE transactions on pattern analysis and machine intelligence, 2016.

[21] S. Xu, Z.F. Ma, and W. Wu. "Container code Recognition System Based on Computer Vision." Video Engineering 5 pp. 035, 2010.

[22] W. Wu, Z. Liu, M. Chen, X. Yang, and X. He. "An automated vision system for container-code recognition." Expert Systems with Applications 39, no. 3, pp. 2842-2855, 2012.

[23] Liu Y , Li T , Jiang L , et al. Container-code recognition system based on computer vision and deep neural networks[C]// International Conference on Advances in Materials, Machinery. 2018.

[24] Sanghoon Hong , Byungseok Roh , Kye-Hyeon Kim , et al. PVANet:Lightweight Deep Neural Networks for Real-time Object Detection[C].arXiv preprint:1611.08588, 2016.

[25] Lin T Y , Dollar P , Girshick R , et al. Feature Pyramid Networks for Object Detection[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, 2017.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, pp. 2278-2324, 1998.

[27] Kim Y . Convolutional Neural Networks for Sentence Classification[J]. Eprint Arxiv, 2014.

[28] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.