This article has been accepted for publication in IEEE Transactions on Instrumentation and Measurement. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIM.2024.3370750

A Practical Unified Network for Localization and Recognition of Arbitrary-oriented Container code and type

Jian Zhao, Ning Jia, Xianhui Liu, Gang Wang, and Weidong Zhao

***Abstract*—The fast and accurate recognition of the container code and type is very important to speed up the passage of trucks through the gate and improve the efficiency of port. In this paper, a Practical Unified Network (PUN) is proposed for the first time to recognize arbitrary-oriented container code and type. The model is based on the encoder-decoder network, which integrates the detection of the second part of container code, the semantic segmentation of the three parts of container code and type, and the classification of single character features of container code and type. The ablation experiments verified the performance of each part of PUN, and obtained a stable and reliable model. The comparison experiments compare PUN with state-of-the-art text detection and end-to-end text recognition algorithms, and the results show that PUN achieves almost the best performance in both container code and type detection and end-to-end recognition. The proposed algorithm has only 15.53M parameters, with a *FPS* of 4.86 with a input size of 768×768, which meets the company's needs. Our model has been deployed and running in multi smart gates of Shanghai Port for nearly a year, showed certain practical application capabilities.**

***Index Terms*—container code and type, end-to-end recognition, detection, semantic segmentation, classification**.

I. Introduction

In recent years, with the increasing requirements of logistics market on container terminal handling efficiency, terminal operator safety and cost, a surge of automated terminals has been set off in ports around the world. In this boom, automatic identification technology is the top priority of automated terminals. The smart gate is the earliest field in the process of port intelligence, and its main role is to maximize the passage efficiency of container trucks through automatic identification technology to recognize and compare the license plate number of container trucks and the container code and type. Compared with the traditional one person one gate mode, smart gate managers can perform one-to-many efficient management remotely, reducing labor costs, while being safe and efficient.

The gate working scenario is shown in Fig. 1. Four high-speed cameras are installed at the gate to take pictures of the truck passing through the gate from the front, back, left and right angles, and identify the relevant information in the pictures in real time to achieve the goal of smart gate.

Fig. 1. Gate working scenario. For the convenience of observation, the container code and type in the picture has been enlarged.

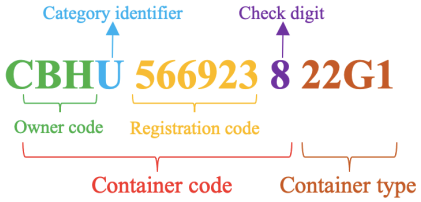


Fig. 2. Meaning of the container code and type.

This paper carries on the recognition of container code and type in gate working scenario. According to ISO 6346, container code and type are composed of 11 and 4 characters, respectively. The character set includes 26 uppercase English letters and 10 Arabic numerals, a total of 36 categories. As shown in Fig. 2, the 11 characters of container code is divided into three parts, the first part consists of four English letters, the first three represent the Owner code, and the fourth represents the Category identifier, which can only be U, J or Z, and U is the most common. The next 6 Arabic digits in the second part represents the Registration code, and the third part is the Check digit, which is an Arabic digit. The 4 characters in container type are composed of letters or digits.

Previous work on container code recognition is divided into two stages: character region detection and recognition. Traditional character detection methods distinguish character

Jian Zhao, Ning Jia (*Corresponding author*), Xianhui Liu, and Weidong Zhao are all with the College of Electronics and Information Engineering, Department of Computer Science and Technology, Tongji University, Shanghai 201804, China. e-mail: zjtju1919@gmail.com; 1510501@tongji.edu.cn; 1910678@tongji.edu.cn; wd@tongji.edu.cn. Gang Wang is with the Institute of Data Science and Statistics, School of Statistics and Management, Shanghai University of Finance and Economics, Shanghai 200433, China. e-mail: gwang.cv@gmail.com.

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on February 28,2024 at 04:16:57 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Instrumentation and Measurement. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIM.2024.3370750

and non-character regions by extracting pixel-level features of images. These features generally include contours, gray levels [1], masks [2], histogram information from grayscale images, and distribution of wavelet or discrete cosine transform coefficients [3]. Because of high density of edges and contours, low gradients above and below text, these pixel-level feature differences help to distinguish character regions from non-character regions. Before character recognition, the detected text needs to be separated into single characters. With the help of high-pass filter [4] or connected components analysis [5], the single character area in the text can be obtained. Further, single character recognition is carried out based on SVM [8] classifier or template matching [6],[7] or other methods.

With the development of deep learning, many text detection and recognition methods have emerged. EAST [9] can detect text regions of different angles, PSENet [11] based on region extension can detect arbitrary shaped text, DB [10] was designed for fast detection, DETR [28] performs text detection by extracting CNN features, and then transformer was used for text encoding and decoding, achieving very good text detection results. For text recognition, the classical CRNN [12] model can recognize strings of arbitrary length. MTv3 [13], ABCv1 [14], ABCv2 [15], TESTR [29] and DeepSolo [30] are end-to-end methods that integrate text region detection and recognition. With the help of these deep learning methods, [16] conducted medical records named entity recognition, [17] conducted integrated circuit markings recognition. For container code recognition, [18] and [19] designed a framework that first detects and then recognizes, while [20] designed an end-to-end recognition model. The drawback is [18] and [19] can only recognize horizontal container code, and [20] can only identify vertical ones, and the images are manually cropped with the container code area, so these methods cannot identify container code with complex background noise in real application scenarios.

It is very difficult to recognize the container code and type in real working scene. The first influence is the background noise. The background of the photos taken by the camera is complex, including roads, street lamps, trees and workers. Moreover, there are also many other texts on the container surface, which affect the detection of container code and type. The second is the effect of shooting angle. Since the camera does not shoot the container surface vertically, the container code and type will be deformed, as shown in the bottom left and bottom right of Fig.1. Finally, the positions of the three parts of container code and type are changeable, some are vertical, some are horizontal, some are 1 row/column, some are 2 rows/columns or 3 rows/columns, as shown in Fig. 4. In addition, changes in weather and illumination in the working scene will be troubles too.

Our algorithm is based on the Shanghai Port Smart Gate Project, and the working scenario is shown in Fig. 1. We designed a Practical Unified Network for localization and recognition of arbitrary-oriented container code and type. In the localization step, the text detection method [18], [19], [20] will detect all the texts in the image, and then judge whether they are container codes, this operation is very cumbersome. We adopt the method of object detection to perform the coarse location of the container code, and further use the method of semantic segmentation to accurately segment the container code and type. In the recognition step, we abandon the traditional CRNN [12] text recognition model. Considering that the length of container code and type are fixed, we directly perform average cutting of CNNs features, and then use classification model to classify single characters. This method is simpler, and can identify container code and type in any direction. In addition, we design a very efficient RoIResize module to connect the detection, semantic segmentation and classification modules, to achieve end-to-end container code and type recognition.

The contributions are summarized as follows.

1) We design an end-to-end network, whose core structure is an encoder-decoder for input feature learning. We also design the rough detection module for detection of the second part of contain code, the accurate location module for contain code and type localization, and a classification module for recognition. In addition, an efficient RoIResize module that does not require learning is introduced and used to bridge the three modules to realize the error backward propagation and construct an end-to-end PUN model.

2) Compared with other text detection and recognition models, our PUN has obvious advantages. Parameter size of our model is only 15.53M, which takes up little GPU memory and is suitable for deployment in industrial scenarios. At the same time, when the size of input image is 768 × 768, the *FPS* reaches 4.86, which meets the requirements of smart gate of Shanghai Port. In addition, our model can recognize both horizontal and vertical container code and type. Our model has been deployed and running in multi smart gates of Shanghai Port for nearly a year, showed certain practical application capabilities.

3) We design a dataset for smart gate container code and type recognition. The dataset contains 5877 images of the gate working scene with a size of 1080×1920 including horizontal and vertical container code and type. And we also have annotated all the images, the annotations include 4 coordinates and string content of the three parts of container code and container type.

The rest of this article is organized as follows. Section II introduces the details of the proposed framework. Section III conducts ablation experiments, as well as comparative experiments with other text detection and recognition models, and Section IV concludes this article.

II. Proposed Method

The core structure of the proposed PUN is an encoder-decoder as shown in Fig. 3, the Rough detection and Accurate location head are designed for container code and type localization, the Classification head is used for container code and type recognition. Accurate location head contains a Segmentation head and an Isolation operation. Through semantic segmentation, the three parts of the container code and type can be classified at the pixel level, and the position information of the single character can be obtained through the isolation module which only be used in the inference phase. The

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on February 28,2024 at 04:16:57 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Instrumentation and Measurement. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIM.2024.3370750

RoIResize module is designed to connect the encoder and decoder, as well as the decoder and classification head, to obtain an end-to-end trainable container code and type recognition model.

Fig. 3 shows the process of forward propagation and error back propagation in the PUN training phase. The Encoder-Decoder, Rough detection head, Segmentation head and Classification head in the PUN model are trainable, and the RoIResize and Isolation module do not need to be learned. The Rough locations and Accurate locations in the training phase are provided by the input annotations. The forward propagation process with the information pass process in Fig. 3 are the inference process. The Rough locations and Accurate locations in the inference stage are provided by the Rough detection head and Accurate location head respectively.

In the following, we will introduce the Rough detection in Section II-A, the RoIResize operation in Section II-B, the Segmentation in Section II-C, Isolation in Section II-D, and the Classification in Section II-E.

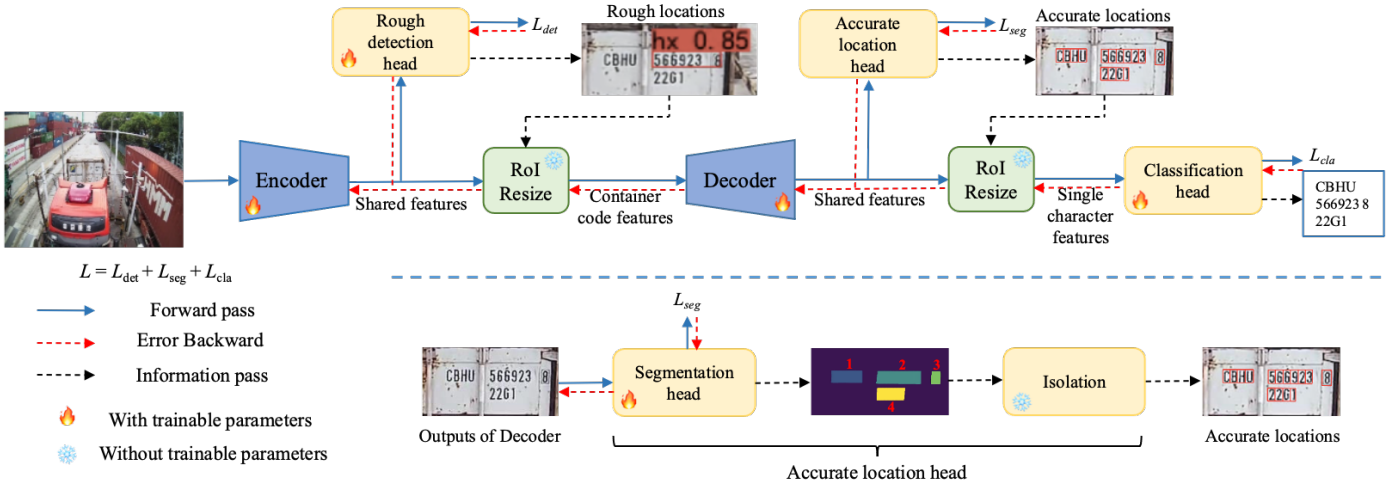


Fig. 3. Structure of the proposed PUN. Encoder and Decoder are used for input feature learning, Rough detection head and Accurate location head are for container code and type localization. Segmentation head and Isolation operation are contained in Accurate location head, used for the three parts of container code and type segmentation and single character isolation respectively. RoIResize is introduced to bridge them to an end-to-end model.

A. Rough detection

It is difficult to directly detect the container code and type from the images taken at the working gates. We adopt a step-by-step strategy, first making a rough detection of the container code, and then using semantic segmentation to accurately locate the three sub-parts of container code and type.

The ability of the detection module is to detect the rough position of container code in the input image, but there are many different arrangements of container code and type, including one row or one column, two rows or two columns, three rows, etc., so choosing the appropriate detection objects is very important. By observing the positional relationship between three parts of container code and container type of a large number of containers, we found that there are only 10 relative arrangements as shown in Fig. 4. Since the sub-parts of container codes are relatively close, the container code and type can be obtained by detecting the most significant area among the three parts of the container code and then expanding the area. We select the second part of the container code as this rough detection object, therefore, we divide the detection objects into two categories, the second part of horizontal container code was marked as "hx", and the second part of vertical container code was marked as "sx", as shown in Fig. 4.

Rough detection module consists of an Encoder and a rough detection head as shown in Fig. 5. Encoder first encodes an RGB image of size H × W to obtain the 4-stage CNN features. On the one hand, these 4-stage features are input into the rough detection head for rough localization of the second part of container code, and on the other hand, they are used as shared features for subsequent accurate location and recognition of container code and type. In the rough detection head, the dimensions and feature size of the 4-stage CNN features of encoder are first adjusted through convolution and up-sampling operation. After this adjustment, the dimensions of the 4-stage CNN features will be the same as the dimension of stage 2 of the encoder, and the feature size will be equal to the size of stage 1 of the encoder. Then the adjusted 4-stage CNN features are spliced through Concat operation, and finally the output $\boldsymbol{O}_{det}$ sized of ¼H × ¼W with 2 feature dimensions are obtained through convolution operation. We use the first dimension of $\boldsymbol{O}_{det}$ to detect the second part of the container code, and the second dimension to classify the horizontal and vertical container code, so we design the following loss function

$$l_{obj} = \mathrm{MSE}(\boldsymbol{O}_{det}[0], \boldsymbol{mask}_{obj}) \quad (1)$$

$$l_{cate} = \mathrm{MSE}(\boldsymbol{O}_{det}[1], \boldsymbol{mask}_{cate}) \quad (2)$$

In formula (1), MSE(·) represents the mean square error, $\boldsymbol{O}_{det}[0]$ represents the first dimension of $\boldsymbol{O}_{det}$, and $\boldsymbol{mask}_{obj}$ is the ground truth for the second part of the container code, where the value of the second part of the container code is 1, and the rest are 0. In Formula 2, $\boldsymbol{O}_{det}[1]$ represents the second dimension of $\boldsymbol{O}_{det}$, $\boldsymbol{mask}_{cate}$ is the category label of "hx" and "sx" container code, where the value of the second part of the horizontal and vertical container code is 1 and 2, respectively, and the rest are 0.

So, the loss for the training of rough detection module is

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on February 28,2024 at 04:16:57 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Instrumentation and Measurement. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIM.2024.3370750

$$L_{det} = l_{obj} + l_{cate} \quad (3)$$

It should be noted that the rough detection head we designed only obtains the pixel blocks of the second part of the container code. During the test process, we need to further obtain the specific coordinates of the second part of the container code based on post-processing steps such as cv2.minAreaRect and cv2.boxPoints.

Fig. 4. 10 kinds of positions of container code and type. The chosen detection objects are shown in the light blue area for horizontal container codes and light yellow for vertical ones.

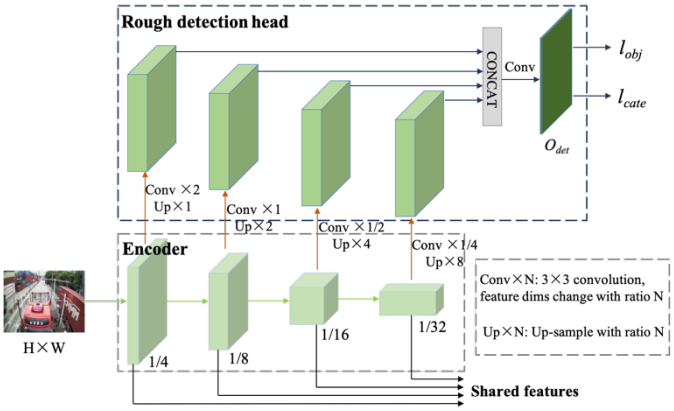


Fig. 5. Structure of Rough detection module in Fig. 3.

B. RoIResize

The RoIResize module is a bridge connecting rough detection module, accurate location module and classification module. It returns fixed size feature maps by performing crop and resize operations on feature maps of a specific area, which facilitates the end-to-end training of the network. Next, we will introduce how RoIResize is used in connecting encoder and decoder in Fig. 3.

The detection result of rough detection module is the second part of container code, which we called initial RoI. We need to expand the initial RoI to get the feature area that contains the whole container code and type, and then obtain the fixed size feature maps through RoIResize module. As shown in Fig. 6, we set the height of the initial RoI to h and width to w. The position coordinates of the initial RoI are obtained through ground truth during training process while through the rough detection module during inference. We expand the initial RoI by observing the 10 positional relationships of container code and type in Fig. 4. For horizontal initial RoI, we extend it by w to the left, $0.4w$ to the right, and $2h$ to the bottom and top. For vertical cases, we extend it by w to the left, $3w$ to the right, $1.0h$ to the top, and $0.3h$ to the bottom to obtain the RoI.

We will up-sample the shared features of the 4 stages in encoder to 384 × 384, 192 × 192, 96 × 96 and 48 × 48, respectively, and perform RoIResize operation on the 4 up-sampled features as shown in Fig. 6. Assuming the size of the output of semantic segmentation in accurate location module is $H_s \times W_s$, the size of the 4 up-sampled features after RoIResize will be $^1/_2H_s \times {}^1/_2W_s$, $^1/_4H_s \times {}^1/_4W_s$, $^1/_8H_s \times {}^1/_8W_s$ and $^1/_{16}H_s \times {}^1/_{16}W_s$, respectively. Fig. 6. shows the operation on the shared features of the first stages.

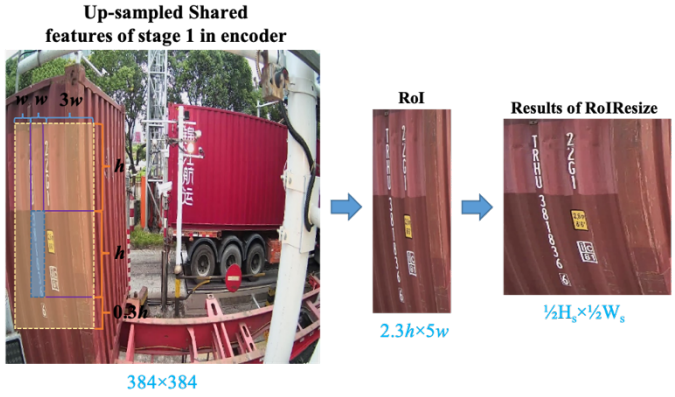


Fig. 6. Operations of RoIResize module between encoder and decoder. The initial RoI and RoI are the light blue and yellow background areas in the left CNN features, respectively. Note that the above operations are all on feature maps, but images are used here for the convenience of expression.

Our RoIResize designed with reference to RoIRotate [23], but more concise. According to the actual tilt angle of the text, RoIRotate crops a fixed size area from feature maps, and further uses CRNN to recognize horizontal text. In our model, the semantic segmentation and classification module can handle a certain inclined text area. Moreover, the container code and type with an inclined angle can increase the diversity of the training dataset, making the model more robust. Therefore, we don't need to correct the tilt angle of container code and type.

There are two steps involved in RoIResize. First, calculate the affine transformation matrix between the RoI area coordinates (x, y) and coordinates (x', y') after RoIResize. Assumes that the RoI area coordinates as (x_{min}, y_{min}), (x_{max}, y_{min}), (x_{max}, y_{max}), (x_{min}, y_{max}), we need to map these 4 points into (0, 0), (s_x, 0), (s_x, s_y), (0, s_y). For the shared CNN features of the 4 stages after RoIResize, (s_x, s_y) is ($^1/_2H_s$, $^1/_2W_s$), ($^1/_4H_s$, $^1/_4W_s$), ($^1/_8H_s$, $^1/_8W_s$) and ($^1/_{16}H_s$, $^1/_{16}W_s$), respectively. This process can be computed by an affine transformation as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 & b \\ 0 & c & d \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4)$$

Where $\begin{bmatrix} a & 0 & b \\ 0 & c & d \end{bmatrix}$ is the affine transformation matrix, which can be calculated by the corresponding coordinates between (x_{min}, y_{min}), (x_{max}, y_{min}), (x_{max}, y_{max}), (x_{min}, y_{max}) and (0, 0), (s_x, 0),

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on February 28,2024 at 04:16:57 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Instrumentation and Measurement. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIM.2024.3370750

(s_x, s_y), (0, s_y). Then, other transformated coordinates of the points in RoI can be obtained based on formula (4).

Second, the results of RoIResize could obtained by bilinear sampling from RoI based on the corresponding relationship between (x', y') and (x,y).

C. Segmentation

The string length of the three parts of container code and container type is fixed, so the individual characters can be obtained through average cutting of responding regions, and the container code and type could be recognized by classification module, but the precondition is that the parts of container code and type could be identified. Semantic segmentation can classify the 9 different objects, we assign the three parts of the horizontal container code and container type to categories 1-4, and categories 5-8 to vertical ones, with category 0 as the background. In this way, we can identify the three parts of container code and type by semantic segmentation.

Unet [25] is a very well-known semantic segmentation network. It is an encoder-decoder network structure. The first half is used for feature extraction, and the second half is spliced to realize the fusion of the underlying location information and deep semantic information, and further achieve the same size as the input through layer-by-layer up-sampling. The encoder of our semantic segmentation module is shared with the backbone of rough detection module, and the decoder is implemented with reference to Unet++ [26]. The decoder uses "up-sampling + concat + CBL" ("CBL" is a block with one convolution layer, one BatchNorm and one Leaky Relu activation layer) to form the basic decoding unit. It should be noted that the Segmentation Head structure in Fig. 3 is very simple, we only use a CBL to match the dimension 32 of the outputs of decoder and classification category number 9 of semantic segmentation module.

The losses of the Segmentation module consist of two parts, the pixel-level cross-entropy loss l_{CE} and Dice loss [24] that measures the degree of overlap between prediction and ground truth, namely l_{Dice}, as shown below:

$$L_{seg} = l_{CE} + l_{Dice} \tag{5}$$

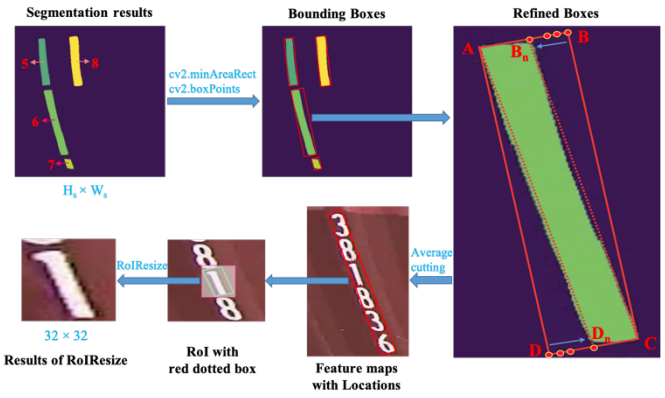


Fig. 7. Operation steps of isolation module. The above operations are all on feature maps, but images are used here for the convenience of expression.

D. Isolation

Semantic segmentation is a pixel-level classification method, it will set each pixel a value from 0 to 8. As shown in Fig. 7, the semantic segmentation can distinguish the three parts of container code and container type very well, but the coordinates of these parts cannot be directly obtained. The isolation module is designed to obtain these coordinates from the semantic segmentation results.

Fig. 7 shows the operation steps of isolation module. The input of segmentation module is the CNN features after RoIResize from Fig. 6. Segmentation module has the ability to classify pixels, so the values of the three parts of container code and container type are 5, 6, 7, and 8, respectively. First, we need to calculate the coordinates of each bounding boxes. These coordinates can be obtained by the ground truth during training. For inference, the minimum bounding boxes of each pixel blocks with value of 5, 6, 7, 8 can be obtained respectively through the cv2.minAreaRect algorithm in Opencv. Further, coordinates of four point of each minimum bounding rectangle can be obtained through cv2.boxPoints. Due to the large inclination angle of some container codes, the minimum bounding rectangle obtained cannot reflect the most appropriate container code area. Therefore, we need to adjust these coordinates obtained from Opencv. We first calculate the ratio of the number of pixels in the text area to the area of ABCD, if it is less than the threshold of 0.95, the coordinates need to be improved, otherwise there is no need to adjust the coordinates. For the situation that needs to be adjusted, we first calculate whether the sum of the 8 neighborhood pixels of the four points A, B, C and D is greater than 0. If it is greater than 0, it means that this point is within the text area, otherwise this point needs to be moved along the short side of the rectangle. For vertical container code as shown in Figure 7, it is obvious that the sum of the 8 neighborhood pixels of point B and D is 0, while the sum is greater than 0 for point A and C, so we need to move B toward A along BA, and D toward C along DC. The movement of B and D is carried out at the same time, and the movement interval is 3 pixels. After each movement, it is necessary to check whether the sum of the 8 neighborhood pixels of B_i and D_i is greater than 0. If the sum of Bi or Di are greater than 0, then treat Bi and Di as the new text boundary points. As shown in Fig. 7, after adjustment, the text area becomes AB_nCD_n.

Next, we equally divide the 4 blocks into 4, 6, 1, and 4 pieces according to actual length of the parts to obtain single character locations, and with these locations, we could get the corresponding single character feature maps. Fig. 7 shows the operations on character '1' from the second part of container code. We take the minimum (x_{min}, y_{min}) and maximum (x_{max}, y_{max}) values of the coordinates of four points to obtain the RoI with red dotted box area, and finally through RoIResize module to get the CNN features of the character '1' with a size of 32 × 32.

E. Classification

After the operations of isolation module, all single character features can be obtained, so single character classification can be performed through the Classification Head. The encoder and decoder in Fig. 3 have already extracted the features of container code and type very well, so the structure of Classification Head is very simple, consisting of "CBL + Pooling(32) + fc(16,36)". The Pooling layer realizes the dimensionality reduction of single character features, the fully connected layer fc maps the 16 dimensions features to the 36 categories of the single character, and we use *CE* loss here for the feature classification module, that is

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on February 28,2024 at 04:16:57 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Instrumentation and Measurement. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIM.2024.3370750

$$L_{cla} = l_{CE} \tag{6}$$

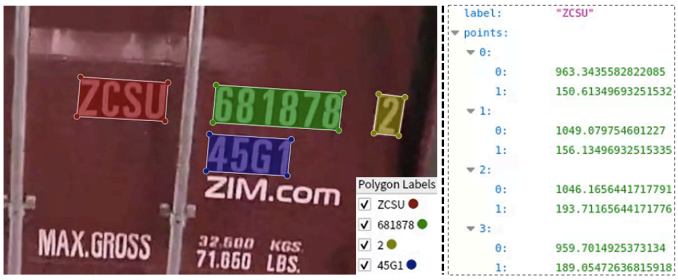
The loss for the whole PUN model for training is as follows:

$$L = L_{det} + L_{seg} + L_{cla} \tag{7}$$

III. Experimental Results

A. Dataset and Experiments Setup

Our experimental data contains 5877 RGB photos of size 1080×1920 taken by the cameras of the Shanghai Port smart gate working scene. We use labelme software to annotate all images. The annotations are divided into two levels. The first level is the classification of horizontal and vertical container code, which does not contain coordinate information. The horizontal and vertical container code are marked as "hx" and "sx", respectively. The second level is used to mark the three parts of container code and container type, including the coordinates of four points and the text contents. The second level annotations are shown in Fig. 8. It should be noted that the labeling is done on the 1080×1920 image, and only the container code and type area is provided here for convenience of display.



(a) (b)

Fig. 8. (a) The second level annotations of horizontal container code and type. The bottom right corner of figure (a) shows the text of the four parts; (b) The four coordinates of "ZCSU".

We divided 5877 pictures into a training set and a testing set at a ratio of 9:1, which are 5286 and 591 respectively. During the training process, all images will be resized to 768 × 768, so the width W and height H of input images are both set to 768. We have used a variety of data augmentation methods based on albumentations, including position transformation augmentation methods HorizontalFlip, VerticalFlip, Transpose, and color gamut augmentation methods RandomBrightness, HueSaturationValue, CoarseDropout, and Blur.

We implement the proposed algorithm using Torch 1.10.2, and all experiments are conducted on a server consists of two 2080Ti GPUs with 11 GB memory each. We train our PUN model from scratch using Adam with a weight decay of 5×10^{-4}. The maximum training epochs is 80. Since the existing project cannot perform parallel training on multiple GPUs, we can only set a small batch size of 2, and in the following experiments, the metric of *FPS* reflecting the speed of the model is obtained on a single 2080Ti GPU.

B. Ablation Study for the backbone of detection module

The purpose of the rough detection module is to roughly locate the second part of the container code and provide features for the decoder. We selected ResNet [21], ResNeXt [22] and EfficientNet [31] series as backbone for comparative experiments. The evaluation metrics are consistent with [20], which are *mAP*, Precision (abbreviated as Pr_d), Recall (abbreviated as Re_d) and *F*1. Considering the industrial application scenario, it is necessary to ensure that the backbone has a smaller parameter amount and a faster inference speed, so we also calculated the model parameter size and *FPS*. From the experimental results in Table I, we can see that the detection performance of ResNet is better than both ResNeXt and EfficientNet, and ResNeXt is better than EfficientNet. For ResNet18 and ResNet50, the latter performs better than the former on *mAP*, Pr_d, Re_d and *F*1, with 2.12%, 0.83%, 1,23% and 1.04% higher, respectively. For *#Params* and *FPS*, the EfficientNet has the smallest number of model parameters, but the speed is not the fastest. ResNeXt18 also has very few model parameters, only 0.33M more than EfficientNet0, and the *FPS* is the highest, reaching 42.11, but the detection performance is not that much good. After comprehensive consideration, we choose the ResNet series as the backbone. For ResNet18 and ResNet50, although the latter has better detection results than the former, the number of model parameters is about twice that of the former, and the inference speed is about 70% of the former. Later, we will further select the final backbone through comparative experiments between ResNet18 and ResNet50.

TABLE I
Results Of Ablation Study For Backbone Used In Encoder

Backbone	*#Params* (*M*)	*mAP* (%)	Pr_d (%)	Re_d (%)	*F*1 (%)	*FPS*
ResNet18	11.32	91.36	96.96	94.22	95.57	36.01
ResNet50	23.63	**93.48**	**97.79**	**95.45**	**96.61**	25.41
ResNeXt18	5.62	90.06	96.51	93.01	94.73	**42.11**
ResNeXt50	25.02	90.06	96.03	93.59	94.79	24.65
EfficientNet0	**5.29**	87.00	94.68	91.77	93.20	34.34
EfficientNet3	12.23	84.89	94.38	89.62	91.94	25.34

C. Ablation Study for the methods of segmentation module

In Section II-C, we have introduced Unet++ as the segmentation method for the three parts of container code and type. Here we conduct an ablation experiment of the segmentation method to further show the advantages of Unet++ over others, such as PAN [32], FPN [33] and Unet. As different segmentation methods have different model structure and parameter amount, so besides *mIoU* for evaluating the segmentation performance, we also consider the metrics of *#Params* and *PFS*. It should be noted that the model parameter includes not only the parameters of the decoder and the accurate segmentation head, but also the parameter of the convolution operation between the decoder and the decoder to adjust the feature dimension. Because the dimension of the 4-stage features of Resnet18 and ResNet50 are different, for the same segmentation method, when the encoder uses different backbone, the number of parameters is also different. In addition, since the CNN features of the decoder are shared features of the segmentation head and classification head, which will affect the performance of the classification module, we also consider the metric of *Acc*, which is the ratio of the number of correctly recognized single characters to the total number of characters. We set H_s and W_s both to 512 here for experiments.

As shown in Table II, for the same segmentation method, since the decoder remains unchanged, even if the encoder uses different backbones, the segmentation performance does not

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on February 28,2024 at 04:16:57 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Instrumentation and Measurement. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIM.2024.3370750

change significantly. This is particularly obvious in PAN, where *mIoU* of ResNet50 only increased by 0.01% compared to ResNet18. For Acc, since ResNet50 learns input images more comprehensively than ResNet18, the classification ability has improved, which could be seen from FPN, the *Acc* increased from 98.37% to 98.79%. In terms of *#Params* and *FPS*, ResNet50 has approximately 2 times the number of parameters as ResNet18, and the inference speed is slower. For different segmentation methods, when the backbone of encoder selects ResNet18, the *mIoU* and *Acc* of Unet++ are the best, reaching 90.48% and 98.70% respectively. Although PAN has the smallest number of parameters, only 1.24M, its segmentation performance and *Acc* are the worst. Unet is the fastest, but *mIoU* and *#Params* have no advantages over Unet++.

TABLE II
RESULTS OF ABLATION STUDY FOR MODEL USED IN SEGMENTATION MODULE

Model	*#Params* (M)		*mIoU* (%)		*FPS*		*Acc* (%)	
	Res18	*Res50*	*Res18*	*Res50*	*Res18*	*Res50*	*Res18*	*Res50*
PAN	**1.24**	**4.74**	90.22	90.23	180.98	141.75	97.58	97.97
FPN	2.89	6.38	90.34	**90.60**	207.91	170.01	98.37	**98.79**
Unet	3.05	6.54	90.24	90.39	**238.16**	**187.30**	98.23	98.47
Unet++	3.39	6.89	**90.48**	90.41	198.52	161.62	**98.70**	98.73

Considering the practicality of the algorithm, we need to choose a better segmentation method with fewer parameters and faster speed, which can greatly reduce the consumption of GPU memory and the demand for high performance hardware in industrial scenarios, so we finally select ResNet18 as the backbone of encoder, and Unet++ as the segmentation method.

D. Ablation Study for the output size of segmentation module

Since the size of the decoder output will affect the performance of the segmentation module and classification module, ablation experiments are performed here on H_s and W_s. As shown in Table III, we set $H_s = W_s$, when the size increases from 128 to 640, the *mIoU* of segmentation model first increases and then decreases, and the *Acc* of classification model basically shows an increasing trend. When the size is 416, *mIoU* reaches the maximum 91.45%, and *Acc* also reaches the ideal 98.57%, so we set the decoder output $H_s \times W_s$ to 416 × 416.

TABLE III
RESULTS OF ABLATION STUDY FOR OUTPUT SIZE OF DECODER

size	*mIoU* (%)	*Acc* (%)
128	90.92	97.82
224	91.27	98.14
320	91.42	98.46
416	**91.45**	98.57
512	90.48	**98.70**
640	90.54	98.55

TABLE IV
RESULTS OF THE PROPOSED FRAMEWORK AND THE STATE-OF-THE-ART MODELS

Method	Backbone	#Params (*M*)	Detection			End-to-End Recognition			*FPS*
			Pr_d (%)	Re_d (%)	*F*1 (%)	Pr_r (%)	Re_r (%)	*H-mean* (%)	
EAST (CVPR2017)	ResNet50	26.21	87.88	92.51	89.13	-	-	-	4.69
DB (AAAI2020)	ResNet18	**12.27**	91.77	95.04	92.29	-	-	-	4.41
PSENet (CVPR2019)	ResNet18	15.56	95.87	96.80	95.74	-	-	-	1.87
DETR (AAAI2023)	ResNet50-ViT	45.75	95.79	**99.39**	97.06	-	-	-	7.18
MTv3 (ECCV2020)	ResNet50	45.47	94.43	74.94	80.00	63.39	59.56	61.69	2.20
ABCv1 (CVPR2020)	ResNet50	36.88	93.91	97.88	95.21	58.46	93.42	69.12	**9.26**
ABCv2 (TPAMI2022)	ResNet50	47.97	96.56	94.95	95.25	86.90	89.72	88.07	4.65
TESTR (CVPR2022)	ResNet50-ViT	49.48	95.44	99.06	96.57	90.46	93.47	91.72	6.81
DeepSolo (CVPR2023)	ViTAEv2-S	33.75	94.66	97.92	95.47	93.19	96.70	94.68	8.13
Our PUN	ResNet18	15.53	**98.51**	96.98	**97.43**	**97.01**	**97.05**	**96.99**	4.86

E. Comparison with the State-of-the-Art methods

In this section we compare the PUN model with state-of-the-art text detection algorithms and end-to-end text recognition algorithms. The compared text detection algorithms include CNN-based methods EAST [9], DB [10], PSENet [11] and transformer-based DETR [28], the compared end-to-end text recognition algorithms include CNN-based methods, such as MTv3 [13], ABCv1 [14] and ABCv2 [15], and transformer-based methods, such as TESTR [29] and DeepSolo [30]. All comparative experiments are conducted on the source code corresponding to the paper.

CNN-based EAST, DB, PSENet are trained from scratch on 5286 training sets, but transformer-based DETR are pretrained on multi very large text detection datasets and then finetuned on our container dataset, and all comparison algorithms are tested on 591 testing sets for text detection. Since there are currently few end-to-end algorithms that can perform horizontal and vertical text recognition at the same time, and MTv3, ABCv1 and ABCv2 can only handle horizontal strings, so we found out all images containing only horizontal container code and type from original dataset to obtain a new training set and testing set with 2207 and 275 images. The training process of CNN-based MTv3, ABCv1, ABCv2 and transformer-based TESTR, DeepSolo will first import the model parameters trained on multi large-scale text recognition datasets mentioned in the corresponding papers, then fine-tune on the 2207 images with horizontal container code and type and test on the 275 images. Our model is directly trained from scratch on the training set with 5286 images and test with 591 images.

The detection objects here are the three parts of container code and container type, and further through the isolation operation, the end-to-end recognition can be achieved, so container code and type that has not been detected will not be recognized. The metrics of detection module are Pr_d, Re_d and *F*1. Since MTv3, ABCv1 and ABCv2 models use CRNN network as the main structure of the recognition module, the

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on February 28,2024 at 04:16:57 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Instrumentation and Measurement. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIM.2024.3370750

length of the recognition results is not fixed, we didn't use *Acc* here, but the edit distance to evaluate the substitution, insertion and deletion errors to obtains the same evaluation metrics as [17], namely Precision (Pr_r), Recall (Re_r) and *H-mean*.

As shown in Table IV, our PUN has certain advantages over existing methods in container code and type detection and end-to-end recognition. For detection, the transformer-based methods are better than the CNN-based methods in Re_d and $F1$, and Pr_d is almost the same as the best CNN method. Although our PUN is also a CNN-based method, the starting point of PUN is not text detection, but the detection of the second part of container code, and then the area is expanded to the entire container code, and through segmentation to obtain the accurate location of container code and type, so the proposed PUN can better locate container code and type. The Pr_d and $F1$ of our PUN are better than all text detection methods and end-to-end text recognition algorithms, Pr_d is 1.95% higher than the second best ABCv2, and $F1$ is 0.37% higher than the second best DETR, but Re_d is not the best.

For end-to-end recognition, PUN performs better than other CNN-based and transformer-based methods in Pr_r, Re_r and *H-mean*. The Pr_r, Re_r and *H-mean* of our PUN are 3.82%, 0.35% and 2.31% higher than the second best DeepSolo. MTv3, ABCv1, ABCv2 and DeepSolo all use CTC-based decoding to recognize strings. CTC decoding assumes that each feature patch is independent of each other, which will cause difficulty in identifying adjacent characters with the same label. Methods such as DeepSolo use a dictionary containing the real text of the test set for decoding during the recognition process, our PUN does not require such a dictionary. TESTR uses the method of classifying CNN features to realize character recognition, which is similar to our classification model, but TESTR does not have the isolation module to obtain the accurate CNN features of a single character, so its classification performance is worse than ours.

Our PUN is also very competitive for #Params and *FPS*. Under the working conditions of the smart gate of Shanghai Port, our algorithm can only be deployed on one computer with one 1660Ti NVIDIA GPU, which requires our algorithm to be lightweight. Considering that more model parameters need more GPU memory consumption, we chose ResNet18 as the backbone of the encoder, Unet++ as the segmentation method, and only one fully-connected layer for classification module, so our PUN has only 15.53M parameters, which is about twice less than DeepSolo, the end-to-end recognition model with the second smallest number of parameters. For *FPS*, although our PUN is slower than ABCv1, TESTR and DeepSolo, it is faster than ABCv2 and MTv3. As shown in Tables I and II, detection and segmentation process of PUN is very fast, but the final *FPS* is only 4.86, which is mainly due to the isolation module. The premise of single character classification is that the model can extract CNN features well, but the encoding and decoding network with limited parameters cannot fully learn the characteristics of single characters, so we designed the isolation module to accurately obtain the area of single characters, which avoids the interference of surrounding characters on the classification model, and improves the recognition accuracy of container code and type. However, the isolation module is not calculated on GPU, so the speed of PUN is slower than ABCv1, TESTR and DeepSolo, but 4.86 FPS meets the needs of smart gate of Shanghai Port.

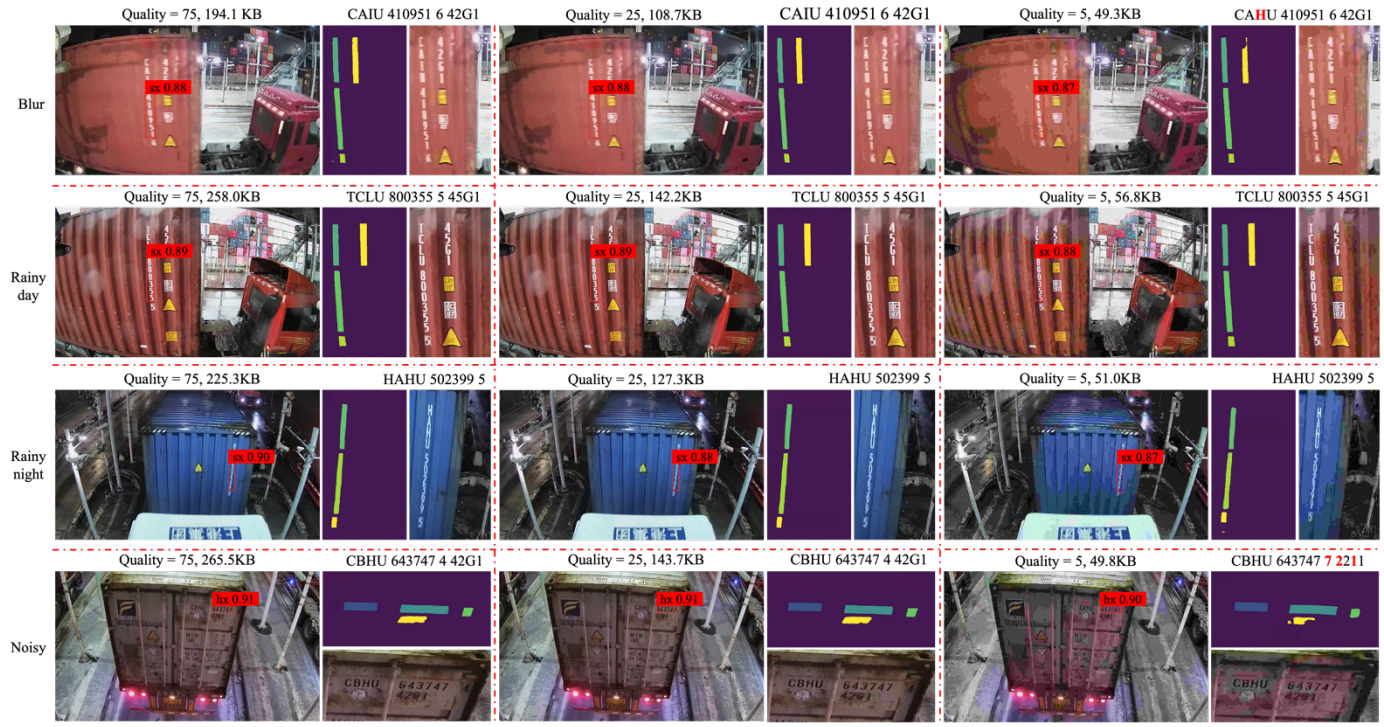


Fig. 9. The results of Robustness experiments. 4 images of blur, rainy day, rainy night and noisy conditions with image quality of 75, 25 and 5 are tested. Rough detection results of the second part of container code, segmentation results of the three parts of container code and type, and end-to-end recognition of container code and type are showed.

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on February 28,2024 at 04:16:57 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Instrumentation and Measurement. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIM.2024.3370750

F. Robustness experiments of PUN

Due to the complexity and variety of smart gate working scenarios, in order to verify the stability of our model, this part will conduct robustness tests. From the test set, we select blurry images caused by inaccurate camera focus to simulate fog weather, images of rainy day and rainy night environments to verify different lighting and rainy weather, and images with severe pollution on the surface of containers for verification of the noisy situation. In addition, considering that the quality of pictures taken by different cameras is different, we process each chosen image through the PIL.Image library to obtain lower quality pictures for testing. We selected four pictures, Blur, Rainy day, Rainy night and Noisy, with quality value of 75 (the picture quality value in PIL.Image ranges from 1 to 95, the larger the value, the clearer the picture, the quality value of the original picture is 75), and obtained two images corresponding to four images with quality of 25 and 5 respectively. As shown in Fig. 9, the quality value and kbyte are placed above each picture. We show the detection results of the rough detection module in the picture, "sx 0.88" means that this area is judged to be vertical container code with a probability of 0.88. The semantic segmentation results of the RoI are placed on the right of the rough detection results, and we also place the end-to-end recognition results on the upper right. As can be seen from Fig. 9, for the original image and the image with quality of 25, our PUN can perform very good localization and recognition of the three parts of container code and type in blur, day, night, rain or noisy environments. When the image quality is further reduced to 5, low-quality images do not affect the rough detection module, but the semantic segmentation model is slightly affected, which can be seen from the container type segmentation of blur and noisy images, and low-quality images also affect the single-character classification. The first part of the container code "CAIU" in the blur image is recognized as "CAHU", the third part of the container code in the noisy image is "4", but we recognized as "7", the container type is "42G1", we identified it as "2211". It should be noted that the image quality will not be so bad under actual working conditions, but this shows that our model still has space for improvement in low-quality image recognition.

IV. Conclusion

We propose an encoder-decoder network structure, which integrates detection, semantic segmentation and classification modules to achieve end-to-end container code and type recognition. Our model has been deployed and running in multi smart gates of Shanghai Port for nearly a year, showed certain practical application capabilities. The recognition rate and inference time consumed have both met the requirements of the port, and it is planned to be deployed on a large scale. In fact, our algorithm has also carried out on the recognition of the license plate of trucks collected in the smart gate, and it has also achieved success. Although the inference speed of our algorithm meets the requirements of the project, there is still space for improvement compared to end-to-end text recognition methods such as ABCv1, which will be one of our follow-up research directions.

Acknowledgment

This work was supported by the National High Technology Research, Development Program of China (2022YFB330570), the Shanghai Innovation Action Project of Science and Technology (21511104302).

References

[1] M. Goccia, M. Bruzzo, C. Scagliola, and S. Dellepiane, "Recognition of container code characters through gray-level feature extraction and gradient based classifier optimization," in *Proc*. 7th *Int. Conf. Document Anal. Recognit.*, Edinburgh, U.K., Aug. 2003, pp. 973-977. doi: 10.1109/TIM.2021.3115211.

[2] W. Al-Khawand, S. Kadry, R. Bozzo, and S. Khaled, "8-neighborhood variant for a better container code extraction and recognition," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 4, pp. 182-186, Apr. 2016.

[3] Q. Ye, Q. Huang, W. Gao, and D. Zhao, "Fast and robust text detection in images and video frames," *Image and Vision Computing*, vol. 23, no. 6, pp. 565-576, 2005.

[4] W. Wu, Z. Liu, M. Chen, X. Yang, and X. He, "An automated vision system for container-code recognition," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 2842-2855, Feb. 2012.

[5] T. Szabo and G. Horvath, "Finite word length computational effects of the principal component analysis networks," *IEEE Trans. Instrum. Meas.*, vol. 47, no. 5, pp. 1218-1222, Oct. 1998.

[6] Z. He, J. Liu, H. Ma, and P. Li, "A new automatic extraction method of container identity codes," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 1, pp. 72-78, Mar. 2005.

[7] X. Zhao, Y. Wang, C. Xiao, Q. Zhu, X. Lu, H. Zhang, J. Ge, and H. Zhao, "Automated visual inspection of class bottle bottom with saliency detection and template matching," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 11, pp. 4253-4267, Nov. 2019.

[8] Z. Ma, L. T. Yang, and Q. Zhang, "Support multimode tensor machine for multiple classification on industrial big data," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3382-3390, May. 2021.

[9] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: An efficient and accurate scene text detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2642-2651.

[10] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang,M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differential binarization," in *Proc. AAAI Conf. Artif. Intell., 2020,* pp. 11474-11481.

[11] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao, "Shape robust text detection with progressive scale expansion networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9336-9345.

[12] 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 Trans. Pattern Anal. Mach. Intell., vol. 39, no. 11, pp. 2298-2304, Nov. 2017.

[13] M. Liao, G. Pang, J. Huang, T. Hassner, and X. Bai, "Mask TextSpotter V3: Segmentation proposal network for robust scene text spotting," In Proc. ECCV, vol. 12356, Aug. 2020, pp. 706-722.

[14] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, "ABCNet: Realtime scene text spotting with adaptive bezier-curve network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9806-9815.

[15] M. Liao, P. Lyu, M. He, C. Yao, W. Wu, and X. Bai,Y. Liu, C. Shen, L. Jin, P. Chen, C. Liu and H. Chen, "ABCNet v2: Adaptiive Bezier-curve network for real-time endto-end text spotting," IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 11, pp. 8048-8064, Nov. 2022.

[16] N. Liu, Q. Hu, H. Xu, X. Xu, and M. Chen, "Med-BERT: A pretraining framework for medical records named entity recognition," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5600-5608, Aug. 2022.

[17] Z. Chen, C. Zhang, L. Zuo, T. Xiahou, and Y. Liu, "An adaptive deep learning framework for fast recognition of integrated circuit markings,"

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on February 28,2024 at 04:16:57 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Instrumentation and Measurement. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIM.2024.3370750

IEEE Trans. Ind. Informat., vol. 18, no. 4, pp. 2486-2496, Apr. 2022.
[18]R. Zhang, Z. Bahrami, T. Wang, and Z. Liu, "An Adaptive Deep Learning Framework for Shipping Container Code Localization and Recognition," *IEEE Trans. Instrumentation and Measurement*, vol. 70, pp., 2021, doi: 10.1109/TIM.2020.3016108.
[19]Y. Liu, T. Li, L. Jiang, and X. Liang, "Container-code recognition system based on computer vision and deep neural networks," Proceedings of the 2nd International Conference on Advances in Materials, Machinery, Electronics. 2018.
[20]R. Zhang, Z. Bahrami, and Z. Liu, "A Vertical Text Spotting Model for Trailer and Container Codes," IEEE Trans. Instrumentation and Measurement, vol. 70, pp.–, 2021, doi: 10.1109/TIM.2021.3115211.
[21]K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770-778.
[22]S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5987-5995.
[23]X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan, "FOTS: Fast oriented text spotting with a unified network," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5676-5685.
[24]F. Milletari, N. Navab, and S. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *Fourth International Confernece on 3D Vision*, 2016, pp. 565-.571.
[25]O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput. Assisted Intervention*, 2015, pp. 234-241.
[26]Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A nested U-Net architecture for medical image segmentation," *4th Deep Learning in Medical Image Analysis Workshop*, 2018, pp. 3-11.
[27]A. G. Roy, N. Navab, and C. Wachinger, "Concurrent spatial and channel 'Squeeze & Excitation' in fully convolutional netwroks," 2018, *arXiv:1803.02579*.
[28]M. Ye, J Zhang, S. Zhao, J Liu, B. Du, and D. Tao, "DPText-DETR: towards better scene text detection with dynamic points in transformer," in *Proc. AAAI Conf. Artif. Intell., 2023,* pp. 3241-3249.
[29]X. Zhang, Y. Su, S. Tripathi, and Z. Tu, "Text Spotting Transformers," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 9519-9528.
[30]M. Ye, J. Zhang, S. Zhao, J. Liu, T. Liu, B. Du, and D. Tao, "DeepSolo: Let Transformer Decoder with Explicit Points Solo for Text Spotting," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 19348-19357.
[31]M. Tan, Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019.
[32]H. Li, P. Xiong, J. An, and L. Wang, "Pyramid Attention Network for Semantic Segmentation," arXiv preprint arXiv:1805.10180, 2018.
[33]Kirirllov, R. Girshick, K, He, and P. Dollar, "Panoptic Feature Pyramid Netgworks," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6399-6408.

Ning Jia received the B.S. degree in information management and information systems and the M.S. degree in computer technology from the Shandong University of Science and Technology in 2012 and 2014, respectively, and the Ph.D. degree in computer science and technology from Tongji University, Shanghai, China, in 2019. He is currently a Post-Doctoral Researcher with the College of Electronic and Information Engineering, Tongji University. His current research interests include computer vision and pattern recognition.

Xianhui Liu is an associate professor. He received his PhD from Tongji University in 2014. He is currently an associate professor with the College of Electronic and Information Engineering of Tongji University, Shanghai, China. And he is current deputy director of CAD Research Center of Tongji University. He is also a member of Artificial Intelligence Committee of Shanghai Computer Association. His research topics include machine learning, data mining, big data, and networked manufacturing.

Jian Zhao received his MS degree from the College of Engineering in Huazhong Agricultural University in 2019, Wuhan, China. He is currently pursuing the PhD at College of Electronic and Information Engineering, Department of Computer Science in Tongji University, Shanghai, China. His research interests include image processing, deep learning, and big data.

Gang Wang received the Ph.D. degree in computer science from Tongji University, Shanghai, China, in 2016. He is currently an Associate Professor with the School of Statistics and Management, Shanghai University of Finance and Economics, Shanghai. His current research interests include computer vision, pattern recognition, and data analysis.

Weidong Zhao is a professor. He is a member of China National Technical Committee for Industrial Automation Systems and Integration Standardization, the chief expert on Information Technology in Sci-Tech Engineering of Manufacturing Industry during the eleventh five-year project of the Ministry of Science and Technology of China, and the team leader of Information Technology Manufacturing Engineering of Shanghai Municipal Science and Technology Commission.

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on February 28,2024 at 04:16:57 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.