applied sciences

MDPI

Article

A Two-Stage Automatic Container Code Recognition Method Considering Environmental Interference

Meng Yu [1], Shanglei Zhu [1], Bao Lu [2], Qiang Chen [3] and Tengfei Wang [1,*]

[1] School of Transportation and Logistics Engineering, Wuhan University of Technology, Wuhan 430063, China; ymmona@whut.edu.cn (M.Y.); zslwh@whut.edu.cn (S.Z.)
[2] China Ocean Shipping Agency Tianjin Co., Ltd., Tianjin 300456, China; lubao@penavicotj.com
[3] Qingdao New Qianwan Container Terminal Co., Ltd., Qingdao 266000, China; chen.q@qqctn.com.cn
* Correspondence: wangtengfei@whut.edu.cn

Abstract: Automatic Container Code Recognition (ACCR) is critical for enhancing the efficiency of container terminals. However, existing ACCR methods frequently fail to achieve satisfactory performance in complex environments at port gates. In this paper, we propose an approach for accurate, fast, and compact container code recognition by utilizing YOLOv4 for container region localization and Deeplabv3+ for character recognition. To enhance the recognition speed and accuracy of YOLOv4 and Deeplabv3+, and to facilitate their deployment at gate entrances, we introduce several improvements. First, we optimize the feature-extraction process of YOLOv4 and Deeplabv3+ to reduce their computational complexity. Second, we enhance the multi-scale recognition and loss functions of YOLOv4 to improve the accuracy and speed of container region localization. Furthermore, we adjust the dilated convolution rates of the ASPP module in Deeplabv3+. Finally, we replace two upsampling structures in the decoder of Deeplabv3+ with transposed convolution upsampling and sub-pixel convolution upsampling. Experimental results on our custom dataset demonstrate that our proposed method, C-YOLOv4, achieves a container region localization accuracy of 99.76% at a speed of 56.7 frames per second (FPS), while C-Deeplabv3+ achieves an average pixel classification accuracy (MPA) of 99.88% and an FPS of 11.4. The overall recognition success rate and recognition speed of our approach are 99.51% and 2.3 ms per frame, respectively. Moreover, C-YOLOv4 and C-Deeplabv3+ outperform existing methods in complex scenarios.

Keywords: maritime management; ship monitoring; video image recognition; convolutional neural network; feature fusion

check for updates

Citation: Yu, M.; Zhu, S.; Lu, B.; Chen, Q.; Wang, T. A Two-Stage Automatic Container Code Recognition Method Considering Environmental Interference. *Appl. Sci.* **2024**, *14*, 4779. https://doi.org/10.3390/app14114779

Academic Editors: Carlos Enrique Palau Salvador and David Sarabia-Jácome

Received: 4 March 2024
Revised: 22 May 2024
Accepted: 22 May 2024
Published: 31 May 2024

1. Introduction

Containers serve as the primary mode of transportation for international trade [1]. According to the United Nations Conference on Trade and Development (UNCTAD) report, 90% of international trade volume is carried out through maritime transport, with over 60% of maritime trade being conducted via container shipping [2]. In 2020, the global container transport trade volume reached 140 million TEUs (twenty-foot equivalent units). The container code serves as a unique identifier for containers and is essential for enhancing control and management of cargo within containers during various stages of Container Multimodal Transportation (CMT).

Container terminals, particularly at gate entrances, are crucial nodes in CMT, responsible for container code recognition and recording [3]. The speed and accuracy of container code recognition at gate entrances directly impact the overall efficiency of CMT. With the rapid growth of global container transport volume, port throughput has increased from 485 million TEUs in 2007 to 820 million TEUs in 2020 [2]. However, the existing methods for container code recognition at gate entrances are unable to cope with the rapid growth of port throughput. The slow recognition speed results in long waiting times for containers at gate entrances, leading to traffic congestion and environmental pollution [4]. Consequently, container terminal gate entrances have become a bottleneck that hinders the development of CMT [5].

The application of Internet of Things (IoT) technology in smart ports is rapidly advancing. Smart ports leverage IoT technology to achieve real-time monitoring and management of cargo, equipment, and infrastructure, enhancing operational efficiency and reducing costs. Existing IoT solutions primarily rely on barcode scanners and RFID devices for data collection. Although these methods are mature, they have certain limitations, such as the need for additional hardware and high maintenance costs. This manuscript proposes a video-based data collection solution that utilizes existing surveillance cameras, eliminating the need for extra hardware and enabling more efficient data collection. Moreover, the algorithm presented in this paper takes into account different angles and lighting conditions, making it adaptable to various image capture devices. As a result, it offers a more flexible and cost-effective option for IoT systems in smart ports.

Due to the slow speed and high cost associated with manual container code recognition and recording [6], Automatic Container Code Recognition (ACCR) systems are primarily used at port gate entrances. Among them, barcode-based ACCR systems have been phased out due to their poor reliability. Radio Frequency Identification (RFID)-based ACCR systems offer high detection accuracy but come with high deployment and maintenance costs [7]. On the other hand, Optical Character Recognition (OCR)-based ACCR systems have low installation costs [8] and are the mainstream approach in modern container terminals due to their simple system integration. However, their recognition accuracy is generally below 95%. This is mainly attributed to the high requirements of methods such as container code localization (CCL) [9], character segmentation, and container code recognition (CCR) [10] in terms of recognition conditions. In situations with uneven lighting, image blurring, or skewed container codes, errors in container code localization or character recognition are prone to occur.

Deep learning algorithms can automatically extract features directly from raw images, showing strong resistance to interference, and have achieved significant results in various areas such as traffic signal detection [11], license plate recognition [12], and medical image segmentation [13]. Based on deep learning algorithms, the development of ACCR can be divided into two types: (1) using a single neural network to complete the entire container code recognition process [14] and (2) stacking two neural networks to separately accomplish CCL and CCR [15]. However, the former requires complex post-processing, does not improve detection speed, and may result in decreased recognition accuracy. Although the latter has improved recognition accuracy, the model parameters are too large, leading to longer computation time. Existing algorithms fail to achieve a balance between container code recognition speed and accuracy, and they also have high requirements for hardware facilities, making it difficult to deploy them at port entrances.

To address these issues, this paper presents a two-stage automatic container number recognition method based on deep learning algorithms. This method offers high recognition accuracy, fast speed, and easy deployment on devices with limited computational power. The first stage utilizes YOLOv4 for container number region localization, while the second stage employs Deeplabv3+ for container number character recognition on the localized region images. Additionally, improvements are made to the network structures of YOLOv4 and Deeplabv3+ to enhance the speed and accuracy of both models in the CCL and CCR tasks, resulting in the modified algorithms named C-YOLOv4 and C-Deeplabv3+. The method is applied to the data input segment of the port Internet of Things system. The main contributions of this paper are as follows:

(1) A dataset of container number images is constructed, primarily collected from container terminal gates. The dataset includes various complex scenarios such as tilted containers and rusty or contaminated container number characters;
(2) The paper proposes the CCL algorithm based on C-YOLOv4. By compressing the backbone feature-extraction network, redesigning the multi-scale recognition module,

and improving the loss function of YOLOv4, the algorithm achieves higher accuracy and speed in container number region localization in complex scenarios;

(3) The paper introduces the CCR algorithm based on C-Deeplabv3+. A decoding structure combining sub-pixel convolutional upsampling and transpose convolutional upsampling is designed and applied to the DeepLabv3+ network. Additionally, a new dilated convolution branch is introduced to preserve more character details in the image without increasing the parameter count.

The remaining parts of this paper are organized as follows: Section 2 provides a brief review of existing methods for CCL and CCR. Section 3 describes the framework of the container number recognition method based on C-YOLOv4 and C-Deeplabv3+. In Section 4, experiments on container number region localization and character recognition are conducted using a self-made dataset, followed by an analysis of the experimental results. Section 5 discusses some conclusions.

2. Literature Review

This section reviews the ACCR methods in the previous literature, mainly comparing the recognition success rate and recognition time, as shown in Table 1. As can be seen from Table 1, the previous ACCR methods only meet one aspect of the recognition success rate or recognition speed.

Table 1. Comparison of related research.

References	CCL	CCR	Success Rate	Time
Koo and Cha 2013 [6]	Edge-based	Back Propagation Neural Network (BPNN)	98.39%	240 ms
Yoon et al., 2016 [16]	Appearance-based	Support Vector Machine (SVM)	96.00%	580 ms
Mei et al., 2016 [17]	Edge-based	LeNet-5 and Template Matching (TM)	94.70%	920 ms
Roeksukrungrueang et al., 2018 [18]	Connected components	LeNet-5	95.41%	10,000 ms
Koo and Cha 2013 [6]	Maximally Stable External Regions (MSER) and Connectionist Text Proposal Network (CTPN)	Convolutional Recurrent Neural Network (CRNN)	93%	-
Han et al., 2018 [7]	-	Broad Learning System (BLS)	97.85%	20 ms
Cao et al., 2019 [9]	MSER-based	Convolutional Neural Networks (CNN)	96.30%	-
Wu et al., 2020 [19]	MSER-based	LeNet-5 and TM	99.30%	1800 ms
Zhang et al., 2020 [20]	ResNet and U-net	CRNN	93.33%	885 ms
Mi et al., 2020 [21]	Edge-based	BPNN	96.80%	2000 ms
Chun-ming 2020 [10]	Single Shot MultiBox Detector (SSD)	SVM	94.60%	630 ms
X. Feng et al., 2020 [15]	You Only Look Once V3 (YOLOv3)	CRNN	96%	159 ms
X. Q. Feng et al., 2020 [22]	Efficient and Accuracy Scene Text (EAST)	-	96%	52 ms
Wan et al., 2021 [23]	Differentiable Binarization (DBNet)	Show, Attend and Read (SAR)	95%	200 ms
Li et al., 2022 [24]	Edge-based	SVM	97.30%	100 ms

2.1. CCL and ACCR

CCL generates a bounding box for the container code area in raw image. The output of this step has a great impact on the accuracy of CCR. Existing CCL algorithms can be divided into three categories: edge-based, MSER-based and DL-based (deep learning) [25].

Container code characters have strong vertical edges, so vertical edge detection can be used to locate container code area. Mei et al. [17] and Li et al. [24] use the filter to detect vertical edges on the image and then use connection component analysis to remove noisy edges and locate the container code area. Koo and Cha [6] use the edge detection operators to detect the vertical edge and then use the sliding windows to find the distinguishing edge features to complete the location of the container code area. Keserwani et al. [26] proposed a centroid-based vector regression method for generating text region quadrilateral bounding boxes, which exhibits high generalization for text regions of different shapes and aspect ratios. Edge-based CCL is easy to implement and computationally fast, but it cannot be used in polluted or inclined containers. Further, this method is very sensitive to illumination changes.

Some researchers use MSER to locate container code area. Cao et al. [9] use MSER to detect areas where characters may exist then use the geometric clustering algorithm and the spatial structure template to identify the container code area. Liu et al. [27] combine Connectionist Text Proposal Network (CTPN) with MSER to improve the locating accuracy, but the algorithm is complex. Wu et al. [19] propose a CCL method combining MSER and edge, which achieved good results. Although MSER-based CCL has a good locating effect on images with a rusty container number, the reliability of this algorithm is insufficient in an uneven lighting environment.

Recently, the DL-based CCL has gradually become the mainstream. Deep learning algorithms can learn features directly from the raw images and have good detection results in a variety of complex situations, such as inclined containers and illumination changes. However, the existing object detection algorithms have the disadvantages of a large number of parameters and long calculation time, so researchers have proposed lightweight networks to improve it. X. Q. Feng et al. [22] present a CCL method based on the ShuffleNet-EAST network. Chun-ming [10] combined lightweight MobileNet and SSD algorithms to perform CCL with 97% accuracy. X. Feng et al. [15] train MobileNetV2-YOLOv3 network for CCL.

2.2. CCR

Previous work on CCR typically needs to segment characters in the container code image firstly and then recognizes each segmented character [28]. This paper focuses on summarizing previous CCR. The existing CCR algorithms include TM-based, SVM-based, and DL-based (deep learning).

TM-based methods recognize each character by measuring the similarity between characters and templates. Mei et al. [17] and Wu et al. [19] use the LeNet algorithm and TM algorithm to recognize container characters, but the recognition accuracy of the former is only 91.9%, and the recognition time of the latter is as long as 1800 ms. TM-based methods are simple and fast, but they can only be used to recognize characters of single font and fixed size, with no rotation or broken areas, which has a poor recognition effect for blurred, skewed, and incomplete container characters.

Some researchers regard container characters as different classes, and train SVM to classify the characters and identify the characters. Yoon et al. [16] use a single SVM for CCR. Li et al. [24] and Chun-ming [10] train two SVMs to identify the numbers and letters of the container code, respectively. In summary, using the SVM-based method, the accuracy rate can be about 95%, but its recognition time is long, and SVM requires a large amount of training data.

DL-based CCR is more robust, and it can recognize container code characters in a variety of complex environments, including tilt and deformation. Roeksukrungrueang et al. [18] use two LeNet networks to identify letters and numbers, respectively, with an accuracy of 95.41%. Mi et al. [21] train a BP neural network to recognize the segmented characters. Zhang et al. [20] and Liu et al. [27] propose a CCR model based on CRNN (Convolutional Recurrent Neural Network).

However, these DL-based CCR methods have slow recognition speed and unsatisfactory recognition accuracy. On the one hand, the existing target detection model is complex, and the number of algorithm parameters is large. On the other hand, the output of CCL has a great influence on the recognition results of CCR. The poor locating effect in the CCL stage eventually leads to the low accuracy of CCR.

To address these deficiencies, some researchers choose to train a single deep learning model to directly identify container characters to avoid errors and missed detections caused by inaccurate area locating and thereby improve CCR accuracy [29]. Han et al. [7] present the BLS (Broad Learning System) for CCR, in which the recognition time only needs 20 ms for one frame, and it has a certain resistance to various complex situations.

In order to further improve the speed of the CCR algorithm, researchers have made lightweight improvements to neural networks. X. Feng et al. [15] use a lightweight CRNN network for character recognition. Wan et al. [23] modified the SAR feature-extraction module to MobileNetv3 for CCR. Experimental results show that this lightweight neural network can greatly improve the recognition speed under the premise of ensuring recognition accuracy.

3. Methodology

3.1. Overall Structure

This paper proposes a two-stage ACCR method including a CCL stage and CCR stage. The overall structure is shown in Figure 1. The first stage takes the full raw image as input. It detects the container code area in the image using the C-YOLOv4 algorithm and outputs the cropped container images to the second stage. The CCR stage takes the cropped image as input. It recognizes container characters using a C-Deeplabv3+ algorithm module and outputs the final result. The method does not need the segmentation stage, which can avoid the decline of recognition accuracy caused by the failure of character segmentation.

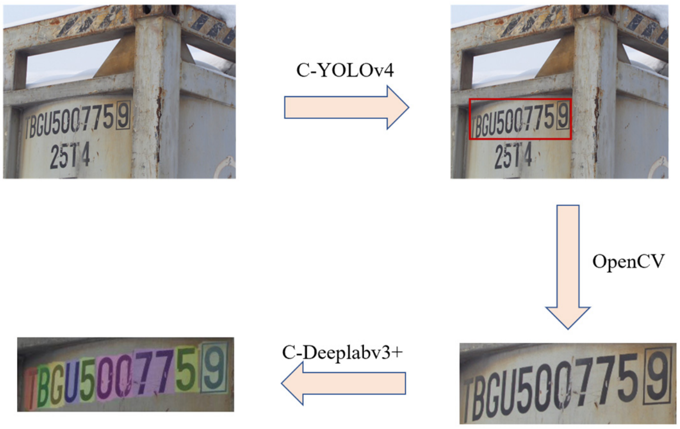


Figure 1. The structure of proposed ACCR framework.

3.2. Step1: The Framework of CCL: C-YOLOv4

3.2.1. C-YOLOv4

YOLO is a typical single-stage target detection algorithm with high accuracy and speed. The complex and sizable architecture of YOLOv4 necessitates a highly configured hardware platform to operate effectively, achieving the requisite speed and accuracy. This requirement for substantial computational resources constrains the algorithm's application within this study. Researchers have improved the original model using lightweight networks. For example, Singh et al. [29] proposed a lightweight model, Latent Graph Attention, which achieves efficient image transformation with lower computational costs. Yar et al. [28] enhanced the YOLOv5 algorithm using MobileNetV3, which achieved good results with lower complexity and a smaller model size. Additionally, Parez et al. [25] utilized MobileNetV3 to improve deep learning models for plant disease detection. It is not proper to deploy at the port gate. This paper proposes an improved lightweight YOLOv4 algorithm (C-YOLOv4), and its structure is shown in Figure 2. Specific improvements are as follows:

(1) The algorithm uses a lightweight network MobileNetV3 as the backbone network and removes the Spatial Pyramid Pooling (SPP) network to lighten the network. And the Feature Pyramid Network (FPN) is adopted as the neck network to increase the speed of network;
(2) The algorithm cancels the 13 × 13 scale feature map and reduces the scale of the feature-extraction network. Then, a K-means++ algorithm was used to recalculate the six more appropriate prior boxes to improve the detection accuracy;
(3) The algorithm uses Enhanced Intersection Over Union (EIOU) to improve the loss function of the yolo4 algorithm, which enhances the training effect.

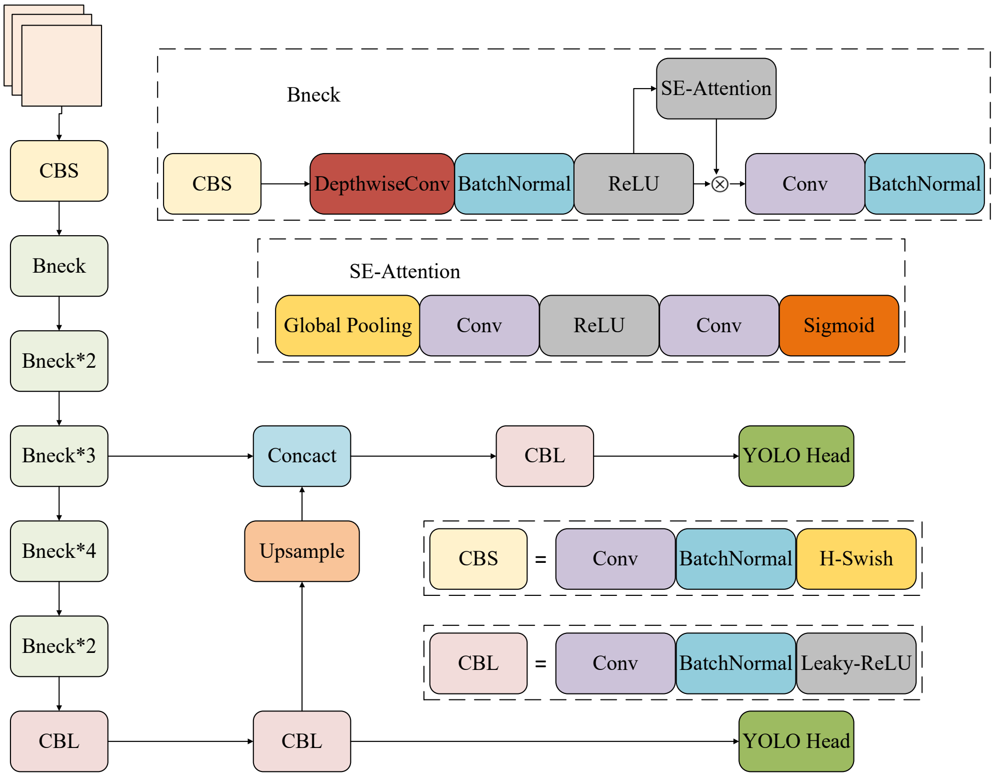


Figure 2. The structure of improved CCL framework.

The output prediction network still uses Yolo Head, while predicting and regressing the class and location of the object at the same time.

3.2.2. Improvement of Backbone Network

C -YOLOv4 takes CSPDarknet53 as the backbone network, which has a strong feature-extraction capability. But it contains five residual blocks and 104 layers of convolutional network. The parameter number and the computational effort of this network are very large. In addition, the container code area is a small or medium single-detection target. CSPDarknet53 has excessive performance in CCL, which leads to poor detection effect.

MobileNetV3 is a lightweight network. It applies depthwise separable convolution, mobile inverted bottleneck convolution, and the SE (Squeezed and Excitation) attention mechanism. Depthwise separable convolution is a special type of convolution, compared with conventional convolution, which can obtain the same output results while significantly reducing the computation and parameter number.

Therefore, MobileNetV3 is adopted as the backbone network of C-YOLOv4, which can reduce the hardware requirements of the algorithm and improve the recognition speed under the condition of ensuring the recognition accuracy.

The MobileNetV3 of C-YOLO consists of one common convolutional layer Conv and several bottleneck structures. Its parameters are shown in Table 2.

Table 2. Parameters of MobileNetV3.

Input	Operator	Exp Size	Out	SE	NL	S
$416^2 \times 3$	Conv2d	-	16	-	HS	2
$208^2 \times 16$	bneck, 3 × 3	16	16	-	RE	1
$108^2 \times 16$	bneck, 3 × 3	64	24	-	RE	2
$104^2 \times 24$	bneck, 3 × 3	72	24	-	RE	1
$104^2 \times 24$	bneck, 5 × 5	72	40	✓	RE	2
$52^2 \times 40$	bneck, 5 × 5	120	40	✓	RE	1
$52^2 \times 40$	bneck, 5 × 5	120	40	✓	RE	1
$52^2 \times 40$	bneck, 3 × 3	240	80	-	HS	2
$26^2 \times 80$	bneck, 3 × 3	200	80	-	HS	1
$26^2 \times 80$	bneck, 3 × 3	184	80	-	HS	1
$26^2 \times 80$	bneck, 3 × 3	184	80	-	HS	1
$26^2 \times 80$	bneck, 3 × 3	480	112	✓	HS	1
$26^2 \times 112$	bneck, 3 × 3	672	112	✓	HS	1
$26^2 \times 112$	bneck, 5 × 5	672	160	✓	HS	2
$26^2 \times 160$	Conv2d	-	256	-	HS	1

Input denotes the shape of the feature map. Operator denotes the type of the feature-extraction structure. Exp size denotes the number of channels after feature map ascending dimension. Out denotes the final number of output channels of this layer. SE denotes whether there is a Squeeze-And-Excite in that block. NL denotes the type of nonlinearity used. HS denotes h-swish activation function, and RE denotes ReLU activation function. S denotes stride; Bneck is a bottleneck convolution module.

3.2.3. Improvement of Multi-Scale Identification Structure

After the input image passes through the backbone network, C-YOLOv4 will output feature maps at scales 13 × 13, 26 × 26, and 52 × 52. These feature maps are sent to YOLO Head for decoding and prediction after stacking convolution and other operations by the feature fusion module.

However, in the CCL task, the container code area occupies a relatively small proportion in the whole picture. The container image corresponding to each grid in the 13 × 13 scale feature map is too large, which makes it difficult to accurately locate the position of the container code area boundary container. Therefore, the C-YOLO v4 algorithm deletes the 13 × 13 scale feature map and detection head.

C-YOLO v4 uses the k-means clustering algorithm to obtain nine prior bounding boxes of different scales. The scales of these prior bounding boxes are generated based on the COCO or VOC dataset, which are not applicable to the container dataset.

The k-means clustering algorithm randomly selects the initial clustering center point, leading to a certain degree of randomness in the clustering results. It is easy to fall into a local minimum. Compared with the k-means clustering algorithm, the k-means++ algorithm improves the selection of the initial clustering center point, resulting in clustering results that are more stable and reasonable. Therefore, we selected the k-means++ clustering algorithm as the prior bounding boxes clustering method. The C-YOLO v4 algorithm recalculates six greater prior bounding boxes. The recalculated prior container size is shown in Table 3.

Table 3. Prior container size.

Layer	Dimensions of Prior Bounding Boxes	Target
26 × 26	(46, 14) (86, 39) (143, 75)	Middle target
52 × 52	(188, 50) (221, 94) (314, 158)	Small target

3.2.4. Improvement of Loss Function

The loss function of C-YOLO v4 is composed of confidence loss, prediction container regression loss, and classification loss, where the prediction container regression loss function is Complete Intersection-Over-Union (*CIOU*), and its specific formula is as follows:

$$L_{CIOU} = 1 - IOU + \frac{\rho^2\left(b, b^{gt}\right)}{c^2} + \alpha v \tag{1}$$

$$\alpha = \frac{v}{(1 - IOU) + v} \tag{2}$$

$$v = \frac{4}{\pi^2}\left(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h}\right)^2 \tag{3}$$

Intersection-Over-Union (*IOU*) is the intersection ratio between the prediction bounding box and the ground truth. $\rho^2\left(b, b^{gt}\right)$ is the center of the Euclidean distance. c is a true container that predicts the container and the diagonal distance of minimum closure area. α is weight function; v, width to height ratio, is used to measure the weight function; b is said to predict bounding box center coordinates; and w, h are the width and height of the prediction bounding box. b^{gt} is the center point coordinate of the ground truth, and w^{gt}, h^{gt} are the width and height of the ground truth.

Although *CIOU* considers the overlapping area, aspect ratio, and center point distance of bounding box regression, its weight function only reflects the weight of aspect ratio and does not consider the difference between width and height, respectively, and its own confidence, which will hinder further optimization of the model. Therefore, we introduce a new loss function Efficient Intersection-Over-Union *EIOU* as the prediction container regression loss function of C-YOLOv4, as shown in Equation (4):

$$L_{EIOU} = 1 - IOU + \frac{\rho^2\left(b, b^{gt}\right)}{c^2} + \frac{\rho^2\left(w, w^{gt}\right)}{c_w^2} + \frac{\rho^2\left(h, h^{gt}\right)}{c_h^2} \tag{4}$$

where c_w, c_h are the width and height of the minimum outer bounding boxes covering the two rectangular closures. *EIOU* takes side length as the penalty term, which alleviates the problem that width and height cannot increase or decrease at the same time in *CIOU* to some extent, and this improves the prediction accuracy of the model.

3.3. Step 2: The Framework of CCR: C-DeepLabv3

3.3.1. C-Deeplabv3+

Deeplabv3+ has high segmentation accuracy, but it has a huge number of parameters, high model complexity, and slow segmentation speed. In addition, although the ASPP module improves the ability of the DeepLabv3+ to extract the target, it leads to the hollow phenomenon of target segmentation. Therefore, this paper proposes an improved DeeplabV3+ algorithm (C-DeeplabV3+); its structure is shown in Figure 3. The specific improvements are as follows:

(1) The algorithm uses MobileNetV3 instead of the Xception network. The principle is similar to the content in Section 3.2.2 and will not be described again;

(2) The algorithm introduces a new atrous convolution branch into the ASPP module, and it changes the dilation rate to enhance the segmentation ability for objects of different sizes;

(3) The algorithm uses upsampling with transposed convolution and upsampling with sub-pixel convolution to replace the two upsampling structures in the decoding network.

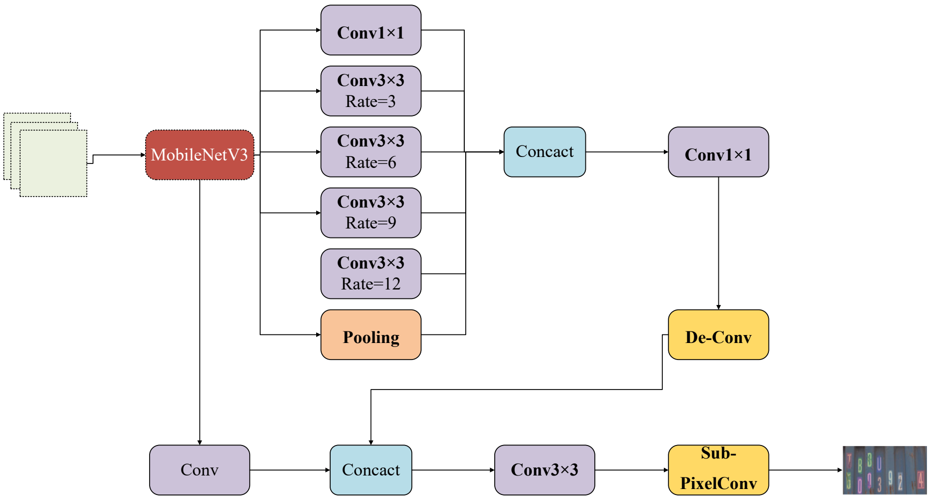


Figure 3. The overall structure of improved C-DeepLabv3.

3.3.2. Improved ASPP

The ASPP module is composed of multiple atrous convolutions with different dilation rates and Spatial Pyramid Pooling (SPP). The atrous convolution is used to extract multi-scale features from the feature map output by the backbone network, and the features are fused and processed through the SPP module to obtain richer contextual semantic information.

Compared with standard convolution, atrous convolution has an additional hyperparameter called dilation rate, which can adjust the receptive field of the atrous convolution, so that the receptive field of the atrous convolution is increased and more details are retained without increasing the computational burden. The atrous combination rate of DeepLabv3+ is (6, 12, 18), which can perform fine segmentation of large-scale targets. However, the single container number character belongs to a small target, so the original DeepLabv3+ has an incomplete effect in the CCR task.

Therefore, the algorithm introduces a new atrous convolutional branch. Moreover, the atrous combination rate is adjusted to (3, 6, 9, 12), so that the network can alleviate the problem of information loss and rough detection of category boundary and can recognize and retain more details of the container number character. Figure 4 shows the comparison of the ASPP module before and after modification.

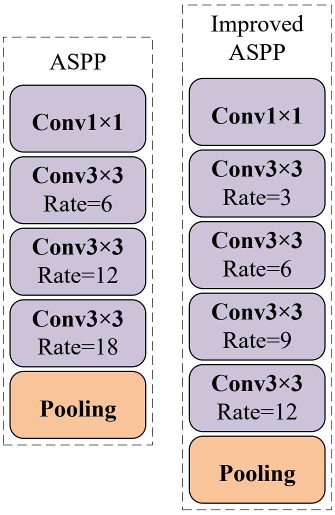


Figure 4. Comparison of ASPP (the ASPP before modification is displayed on the left, and the ASPP after modification is displayed on the right).

3.3.3. Improved Decoder Module

The task of the Deeplabv3+ decoder is to classify each pixel in the feature map transmitted and finally complete the segmentation task. And the upsampling structure in the decoder reduces the computational complexity by interpolating between the original feature maps. But these interpolation methods need more manual operation, which makes the network not have strong plasticity. In addition, traditionally, sampling needs to prefill 0 in the middle or around the low-resolution image, and this information is invalid, which will lead to difficult convergence of the model in the process of gradient optimization. Therefore, in this paper, we use, respectively, upsampling based on transpose convolution and upsampling based on sub-pixel convolution to replace the two upsampling structures.

The upsampling with transposed convolution has learnable parameters, which makes the decoding network have learning ability, and it is convenient to train the model. Upsampling with sub-pixel convolution directly obtains pixel information from the feature map and performs Pixel Shuffle, without the need to complement the 0 operation, which can effectively reduce the gradient optimization problem caused by invalid information transmission. In addition, the receptive field of upsampling with sub-pixel convolutional is larger, which can collect more contextual semantic information to generate richer pixel details, shorten the image reconstruction time, and improve the segmentation accuracy.

4. Experiment

The experiments of CCL, CCR, and complete container number recognition, respectively, are proposed in this section to verify the advantages of the proposed ACCR method. The operating system used in this experiment is Windows10. The GPU is GeForce RTX3090 and is manufactured by the Nvidia company based in Santa Clara, CA, USA. And the processor is Intel Xeon Silver 4210R@2.40 GHz and is manufactured by Intel Corporation, which is based in the Santa Clara, CA, USA.

4.1. Dataset

The main collection scene of this experimental dataset is the container terminal gate. In addition, in order to improve the versatility of the model, the dataset also contains the container images collected by cameras and mobile phones at the container terminal yard. After data cleaning, images that cannot be recognized by human eyes, such as severely rusty image characters and seriously occluded container number character areas, were removed, leaving 1721 images in the dataset. The dataset is mainly composed of dry bulk containers and oil tanks, including container images when the container number area is partially obscured by snow, the container is tilted, and the container number character is rusted or polluted. Some of the images are shown in Figure 5.

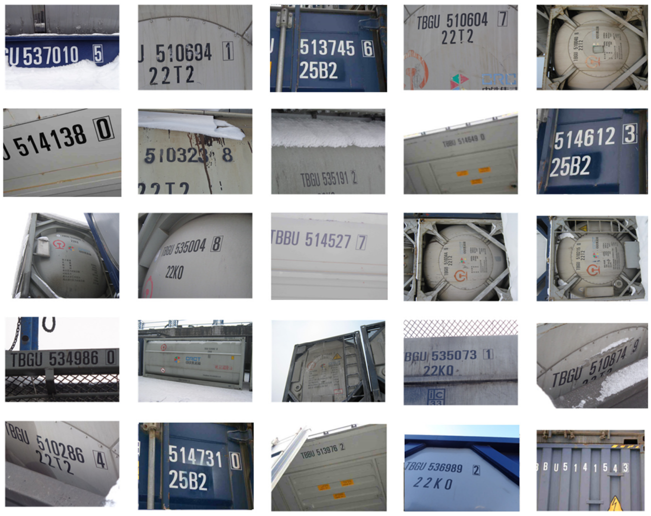


Figure 5. Examples of the dataset in this paper.

4.2. Evaluation Indicators

In this paper, precision–recall (P–R) curves, F1 scores, and accuracy are used to evaluate the results of CCL experiments. Precision refers to the proportion of truly positive samples in all data predicted to be positive, as shown in Equation (5). Recall is the ratio of the number of all correctly predicted positive images to the total number of total positive images, as shown in Equation (6). Accuracy refers to the proportion of all image data that is correctly predicted, as shown in Equation (7). The F_1 score is a weighted harmonic average of precision and recall, as shown in Equation (8):

$$Precision = \frac{\sum TP}{\sum TP + \sum FP} \tag{5}$$

$$Recall = \frac{\sum TP}{\sum TP + \sum FN} \tag{6}$$

$$Accuracy = \frac{\sum TP + \sum TN}{\sum TP + \sum TN + \sum FP + \sum FN} \times 100\% \tag{7}$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{8}$$

where *TP* means true positive, *FP* means false positive, *FN* means false negative, and *TN* means true negative. In addition, *FPS* (frames per second) is also used as an evaluation index for the real-time performance of the model.

Mean Pixel Accuracy (*MPA*) and *FPS* are evaluation indicators of the CCR experiment. *MPA* refers to the result of averaging the accuracy of category pixels in pixel classification, and the larger the *MPA* value, the higher the prediction accuracy, as shown in Equation (9):

$$MPA = \frac{CPA_1 + \cdots CPA_n}{n} \tag{9}$$

CPA (Class Pixel Accuracy) refers to the proportion of pixels in the classification that are correctly classified for each class, as shown in Equation (10). *IoU* (Intersection Over Union) represents the proportion of the intersection of a certain type of prediction region and the actual region to the union of the predicted region and the actual region of the class, as shown in Equation (11):

$$CPA = \frac{\sum TP}{\sum TP + \sum FP} \tag{10}$$

$$IoU = \frac{\sum TP}{\sum TP + \sum FP + \sum FN} \tag{11}$$

4.3. Experimental Localization of Container Numbers

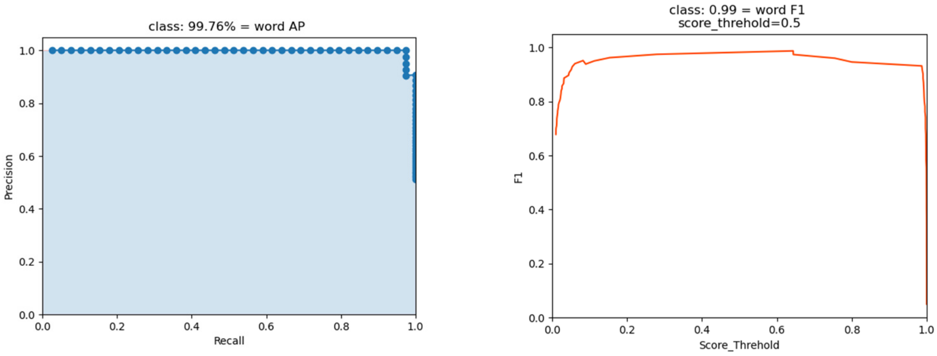
The input image size is set to (416, 416, 3), with a ratio of 1:9 for the validation set and training set. Predicted boxes with an Intersection Over Union (*IoU*) greater than 0.4 with the ground truth boxes are retained, and image distortion adjustment is not performed. The batch size for image processing is set to 8 images per iteration, with a base learning rate of 0.001. Training is terminated when the learning rate remains below 1e-6 for 10 epochs.

First, we compare the model parameters and detection speed of C-YOLO v4 and YOLOv4, as shown in Table 4. YOLOv4 has 39 MB parameters, while C-YOLO v4 only has 13.6 MB parameters, reducing it by approximately 78%.

Table 4. Number of model parameters and real-time computation speed.

Model	Number of Parameters	Model Size
YOLOv4	64,363,101	39 MB
C-YOLOv4 (Ours)	14,265,246	13.6 MB

In this paper, the detection accuracy of the proposed C-YOLOv4 model is also verified by P–R curve and F_1 curve, and the results are shown in Figure 6. Compared with the single recall and precision, the F_1 value can be a more comprehensive measure of the model performance. From the P–R curve, it can be seen that the precision of the model has been kept around 1 during the recall from 0 to 0.9, which indicates that C-YOLOv4 is very accurate in predicting the positive samples. And the F_1 value, after removing the two threshold poles of 0 and 1, is basically above 0.95, which proves that the model's detection effect is very stable.



(**a**) (**b**)

Figure 6. Analysis of C-YOLOv4 model results. (**a**) P-R; (**b**) F_1.

In summary, the number of C-YOLOv4 parameters is reduced, and the detection speed is substantially increased, but the detection accuracy of the model is not negatively affected. Figure 7 shows some cases of successfully localized and segmented box number regions using C-YOLOv4.

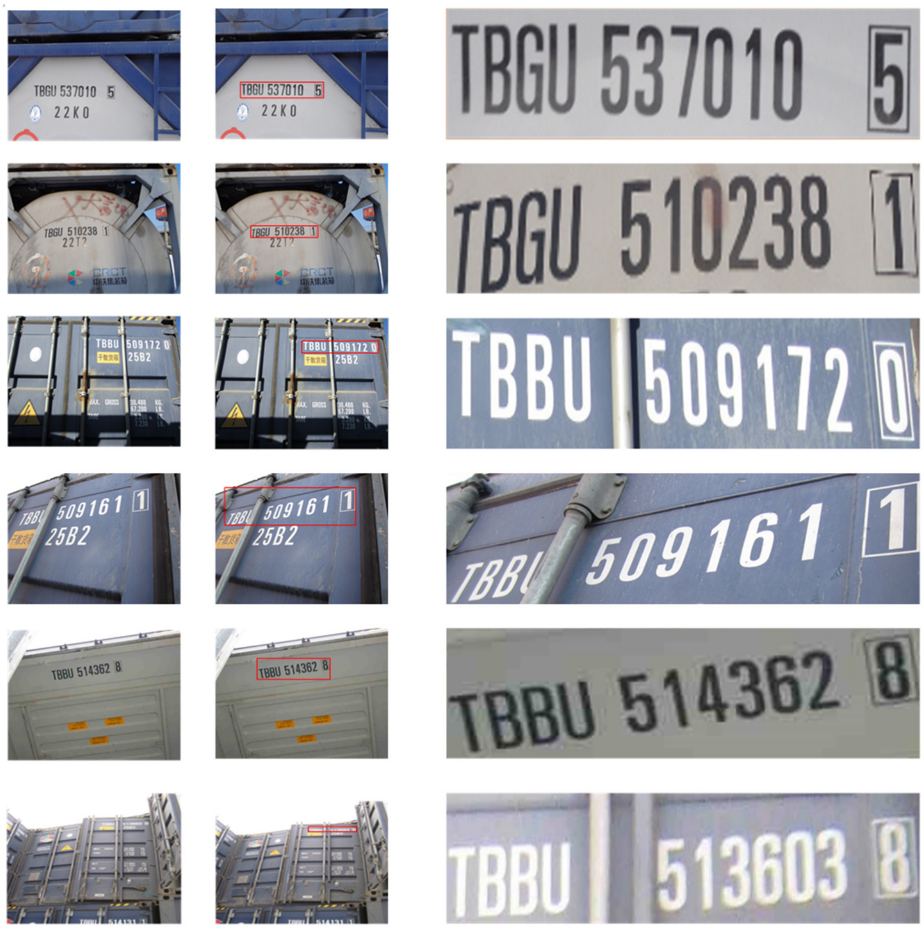


Figure 7. Cargo number areas successfully localized by C-YOLO.

In this paper, YOLOv3, YOLOv4, YOLOv5s, and C-YOLOv4 are also selected for the comparison of accuracy (Accuracy) and speed (FPS) for the localization of box number region, and the results are shown in Table 5.

Table 5. CCL Model recognition accuracy and recognition rate.

Model	Accuracy (%)	FPS (ms)
YOLOv3	94.70	23.5
YOLOv4	98.81	23.0
YOLOv5s	98.53	18.7
C-YOLOv4	99.76	17.6

As can be seen from Table 5, YOLOv3 has the lowest accuracy of 94.50%, while YOLOv5s also has an accuracy of 98.43%, which is slightly lower than YOLOv4's 98.81%, because it sacrifices some of the accuracy for improving the recognition speed; C-YOLO's model has the highest accuracy of 99.76%, which is 1.06% higher than YOLOv4. In addition, YOLOv3 has the slowest recognition speed of 23.5 ms a frame, and YOLOv5s recognition speed is higher than YOLOv4, but still slower than C-YOLO's 17.6 ms a frame. In summary, C-YOLO achieves an excellent balance in recognition accuracy and recognition speed.

The box number region localization accuracies of YOLOv3, YOLOv4, and YOLOv5s are lower than that of C-YOLOv4 because we have collected a large number of box number images in our dataset that are tilted, with missing characters or with contaminated box number regions, whereas YOLOv3, YOLOv4, and YOLOv5s are not able to adapt to these complexities, and they often suffer from omissions and have poor localization results, and our proposed C-YOLOv4, for the box number recognition task, optimizes loss function, multi-scale recognition, and other modules and can always locate the box number region accurately. Some comparison results are shown in Figures 8–10.

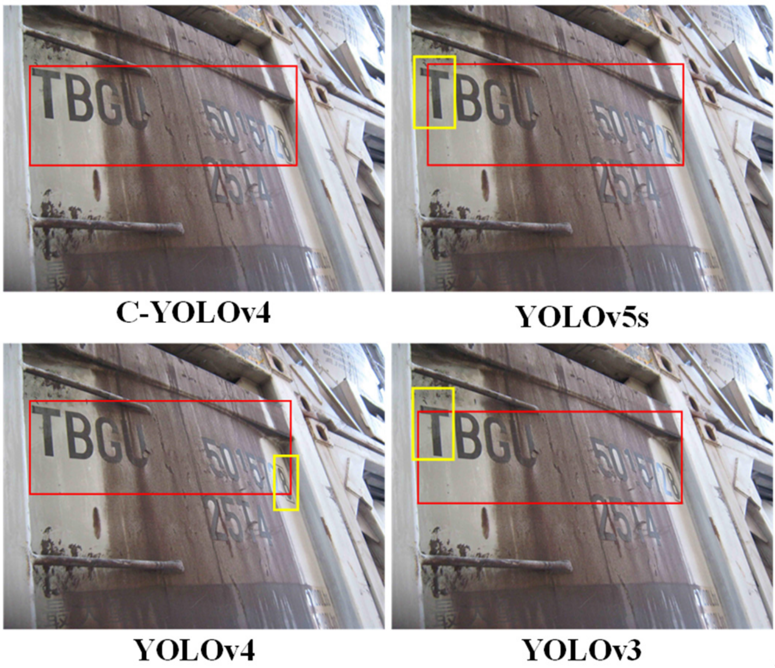
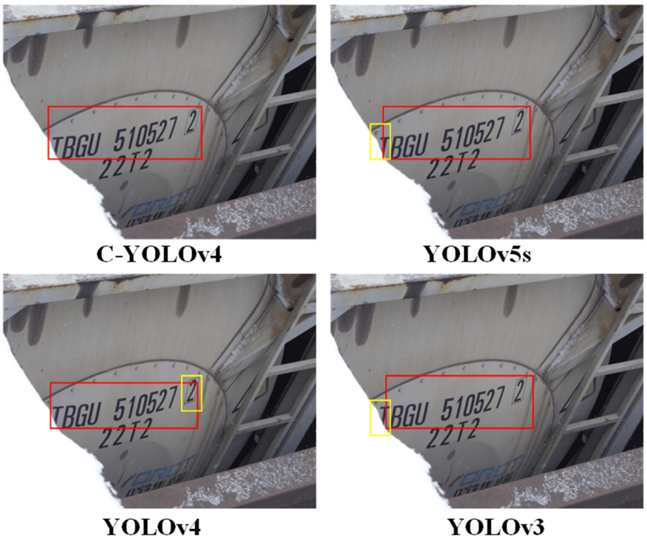


Figure 8. Schematic diagram of the recognition effect of oil pollution scene.

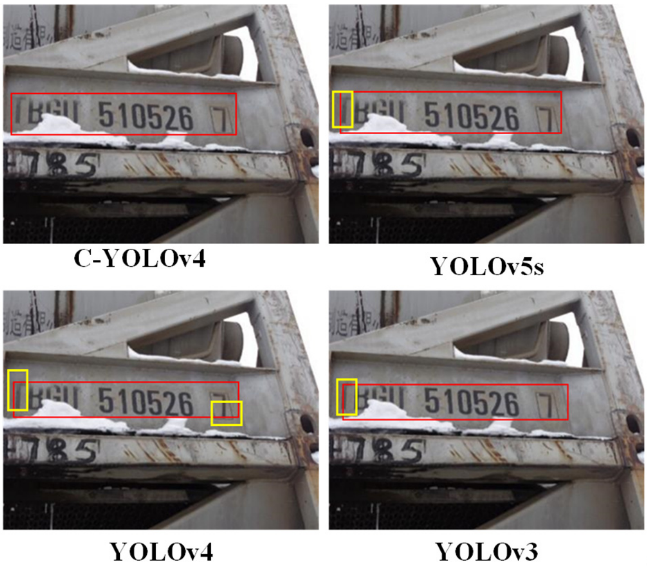

C-YOLOv4

YOLOv5s

YOLOv4

YOLOv3

Figure 9. Schematic diagram of the recognition effect of severe tilt in the box number area.


C-YOLOv4

YOLOv5s

YOLOv4

YOLOv3

Figure 10. Box number area partially obscured with unclear characters.

In terms of localization effect, C-YOLOv4 can accurately locate the area where the box number is located in the image when facing the problems of oil contamination, serious tilting of the box, and blocking of the box number area, while YOLOv3, YOLOv4, and YOLOv5s are sometimes missing in the edge portion, and YOLOv3 and YOLOv5s are more serious, with poorer localization accuracy. Figures 8–10, the red box is the result of the object detection frame of different models, and the yellow box is the difference of the results of different object detection frames.

4.4. Experiment of Box Number Character Recognition

Before the box number character recognition experiment, we recreated the dataset and manually localized and segmented the box number region in the self-constructed dataset; the effect is similar to the segmented box number region image in Section 4.3, but the manual localization and segmentation is more accurate, which can avoid the failure of character recognition due to localization errors. We use Labelme as an annotation tool to annotate the box number characters, and the annotated image is shown in Figure 11, where rectangular boxes of different colors represent different English letters or numbers.

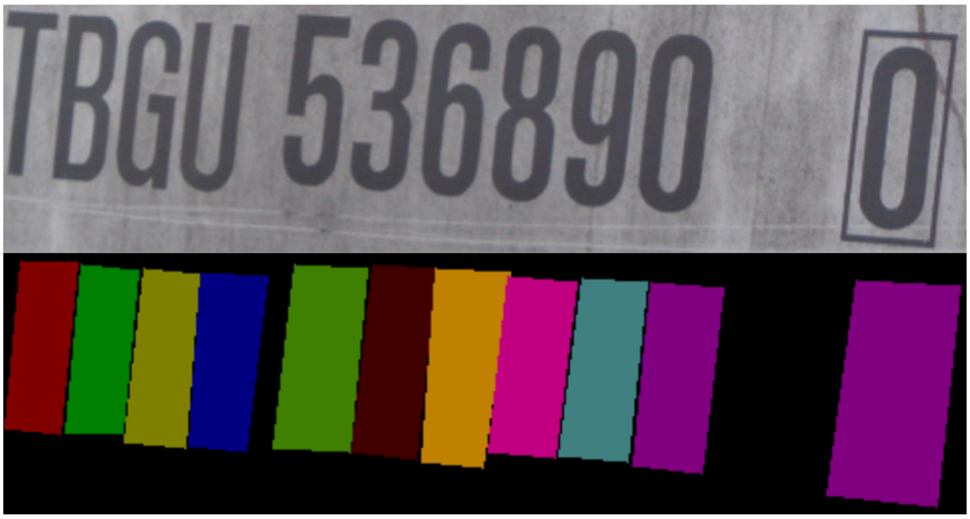


Figure 11. Image of container number after labeling.

In addition, because the English part of the container box number characters in the self-built dataset are mainly two kinds of TBBU and TBGU, in order to ensure that the model recognizes other English characters to improve the generalizability, we also cut down the images of other parts of the characters on the containers, which are used for model training, and some of the images are shown in Figure 12.

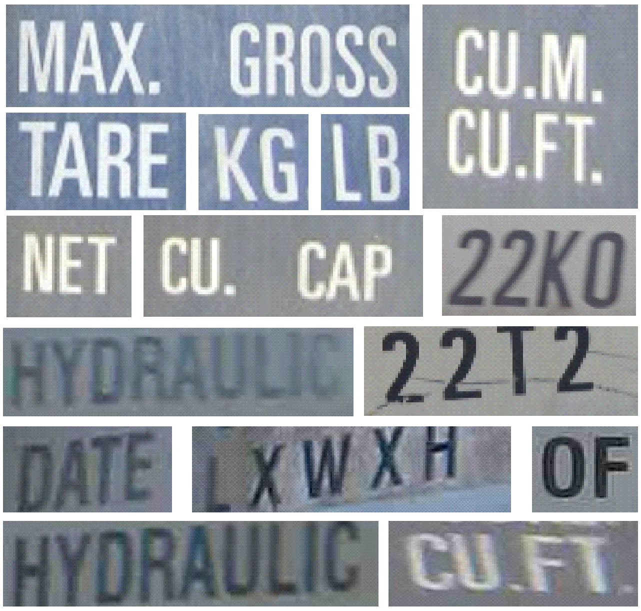


Figure 12. Other categories of characters on containers considered in the dataset.

The image downsampling multiplicity is set to 16, and the ratio of validation set to training set is 1:9. The image batch size is 8 images each time, the base learning rate is set to 0.0001, and the training is finished when the learning rate is less than 1×10^{-6} in all 10 rounds.

Firstly, the number of model parameters and recognition speed of DeepLabv3+ and C-DeepLabv3+ are compared, and the results are shown in Table 6. The number of parameters of DeepLabv3+ is 39.3 MB while the number of parameters of C-DeepLabv3+ is only 2.63 MB, which is reduced by about 95%.

Table 6. Comparison of model performance.

Model	Number of Model Parameters	Model Size
DeepLabv3+	41,253,330	39.3 MB
C- DeepLabv3+ (Ours)	2,758,597	2.63 MB

We choose MPA to evaluate the accuracy of C-DeepLabv3+ in character recognition. Since the target of box number character recognition is 0–9 and A–Z with 36 categories, we obtained the recognition results of 36 characters in total, as shown in Figure 13, and the mean value of its MPA is 99.22%, which shows that the overall recognition accuracy of C-DeepLabv3+ is very high. More specifically, although recognizing “0”, “1”, “6”, “8”, “B”, “I”, “J”, “O”, “Q “, “U”, “V”, and other similarly shaped characters has lower MPA values, the MPA index for each character is still above 97%.

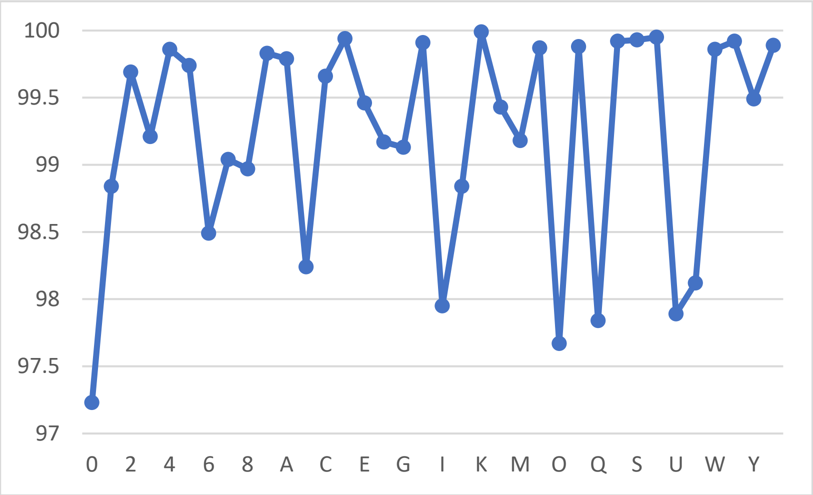


Figure 13. Recognition accuracy of different characters.

In this paper, Segnet, PSPNet, DeepLabv3+, and C- DeepLabv3+ are also selected for the comparison of mean pixel classification accuracy (MPA) and recognition rate (FPS), and the results are shown in Table 7.

Table 7. Model recognition accuracy and recognition rate.

Model	MPA (%)	FPS
PSPNet	95.68	10.2
Segnet	94.36	6.3
DeepLabv3+	97.43	9.2
C-DeepLabv3+	99.88	11.4

C-DeepLabv3+ has the highest MPA and recognition speed, which are 2.51% and 24% higher than Deeplabv3+, respectively, while Segnet has the lowest MPA of 91.23%, which cannot satisfy the requirement of the accuracy of the box number character recognition, and the Segnet network has a complex structure while the model has a slow recognition speed of only 6.3 images per second. Taken together, the C-DeepLabv3+ network has the best character recognition effect.

C-DeepLabv3+ integrates MobileNetv3 as its core architecture, which confers a substantially increased recognition speed over Segnet, PSPNet, and DeepLabv3+. Additionally, the model augments the Atrous Spatial Pyramid Pooling (ASPP) module and the decoder to enhance the accuracy of box number character recognition, achieving the highest accuracy reported. The methodology outlined in Table 1 was adhered to for comprehensive box number recognition. The localization outcomes from C-YOLOv4, processed by OpenCV, were subsequently fed into C-DeepLabv3+ for character recognition. The resultant recognition rates and speeds are detailed in Table 8.

Table 8. Overall success rate and speed of the recognition algorithm.

Overall Success Rate (%)	Recognition Speed (ms)
99.51	115

We combine Tables 1 and 8 into the same table to compare the box number recognition accuracy and recognition speed for different documents, and the results are shown in Table 9.

Table 9. Comparison of metrics for box number identification in different models.

Reference	Number Area Localization Method	Character Recognition	Success Rate (%)	Recognition Speed (ms)
Koo and Cha 2013 [6]	Edge-based and sliding windows	BP Neural Network	98.39%	240 ms
Yoon et al., 2016 [16]	Based on appearance	SVM	96.00%	580 ms
Mei et al., 2016 [17]	Edge-based	LeNet-5 and Template Matching Method	94.70%	920 ms
Roeksukrungrueang et al., 2018 [18]	Grouping of connected components	LeNet-5	95.41%	10,000 ms
Liu et al., 2018 [27]	MSER and CTPN	CRNN	93%	-
Han et al., 2018 [7]	-	Width learning system	97.85%	20 ms
Wu et al., 2020 [19]	MSER	LeNet-5 and template matching method	99.30%	1800 ms
Zhang et al., 2020 [20]	ResNet and U-net	CRNN	93.33%	885 ms
Chun-ming 2020 [10]	SSD	SVM	94.60%	630 ms
X. Q. Feng et al., 2020 [22]	YOLOv3	CRNN	96%	159 ms
X. Q. Feng et al., 2020 [22]	EAST	-	96%	52 ms
Wang et al., 2021 [11]	DBNet	SAR	95%	200 ms
Li et al., 2022 [24]	Edge-based recognition	SVM	97.30%	100 ms
Ours	C-YOLO	C-Deeplabv3+	99.51%	115 ms

As can be seen from Table 9, our proposed ACCR, in terms of recognition success rate and recognition speed, outperforms all previous methods, reaching 99.51% and 115 ms, respectively. The ACCR methods proposed in the previous literature are either too slow or have insufficient success rates. The recognition success rate of the method proposed by Wu et al. is the highest, reaching 99.3%, which is only slightly lower than that of our method, but its recognition speed is 15.6 times slower than our method, and it cannot realize real-time detection. Li et al.'s method is slightly faster than our method, but its recognition success rate is low, and its robustness is poor in complex environments, and the application at the dock gates is poor; Feng et al.'s method has a more balanced recognition speed and success rate and also has a certain degree of resilience to the complex environments, but it is still not as excellent as our method. The other methods in the table are not good enough in both recognition success rate and recognition speed, and they will not be repeated.

In summary, our method achieves the highest recognition success rate and speed, and it can accurately and quickly obtain recognition results in various complex scenes. Figure 14 shows some of the box number character recognition results.

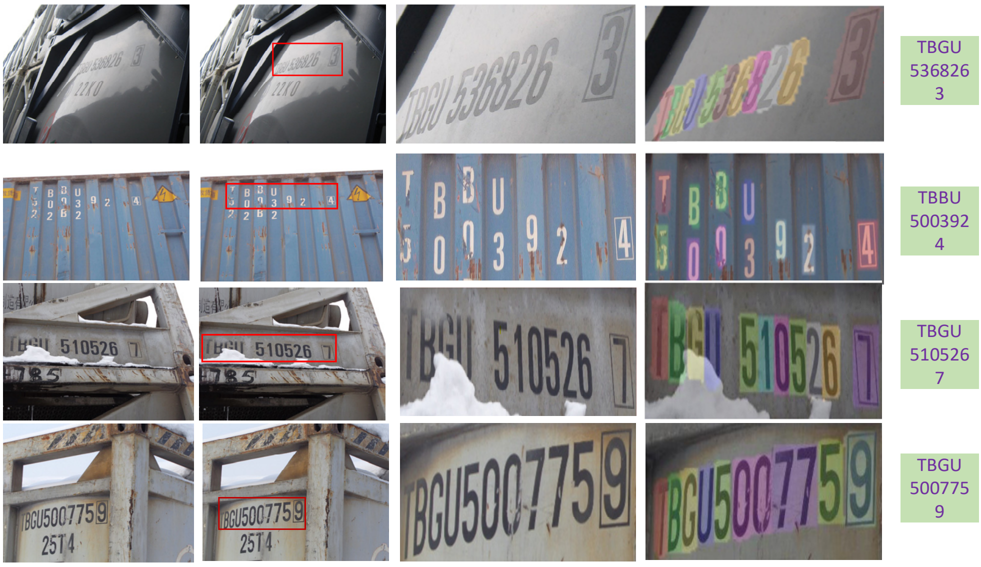


Figure 14. General demonstration of the recognition effect of the algorithm.

5. Conclusions

This study introduces a dual-stage container number recognition method leveraging deep learning techniques, specifically designed to swiftly and accurately identify container numbers at terminal gates. The proposed approach incorporates the enhanced algorithms C-YOLOv4 for detecting container number areas and C-Deeplabv3+ for character recognition. This method not only maintains high accuracy but also significantly accelerates recognition speed and reduces the complexity of the model, making it more practical for deployment at terminal facilities.

Significant enhancements were applied to YOLOv4, including replacing the CSPDarknet53 network with MobileNetv3 and substituting PANet with FPN to reduce parameters. The removal of 13 × 13 scale feature maps and the recalibration of prior box values have tailored the model to better fit the container number localization task. Furthermore, the adoption of an Enhanced Intersection Over Union (EIOU) loss function has optimized the speed of model training. The modified C-YOLOv4 demonstrated a 30% improvement in speed and a 1.06% increase in accuracy over the original model, outperforming other conventional models like YOLOv3 and YOLOv5s in complex scenarios. Modifications to Deeplabv3+ were also pivotal. Replacing Xception with MobileNetv3 as the backbone feature-extraction network reduced parameter size. The introduction of a new atrous convolution branch and a redesigned Atrous Spatial Pyramid Pooling (ASPP) module have improved the model's precision in detecting small targets. Adjustments in the decoder, specifically the shift to transposed and sub-pixel convolution upsampling, have effectively preserved image details, boosting the character recognition capabilities of C-Deeplabv3+. Compared to its predecessor, the updated model has shown a 23.9% faster recognition speed and a 1.58% higher accuracy rate. However, the slower operation of C-Deeplabv3+ compared to C-YOLOv4 suggests the need for further optimization.

In future work, we will explore how to adapt our algorithms and models to various IoT device platforms and optimize their performance to ensure efficient operation under diverse hardware and resource constraints. Additionally, we will focus on diversifying the dataset to enhance the algorithm's generalization ability. Through these efforts, we aim to extend the reach of our container number recognition system to a broader range of IoT devices and have a greater impact on actual port operations. Future efforts will focus on these areas to advance the system's application in real-world port operations.

This research provides a solid foundation for robust, efficient container number recognition systems, with potential for further enhancement and application in real-world port operations.

Author Contributions: Conceptualization, M.Y. and S.Z.; methodology, M.Y. and S.Z.; validation, M.Y., S.Z., B.L. and T.W.; formal analysis, B.L.; resources, Q.C.; data curation, S.Z.; writing—original draft preparation, S.Z. and B.L.; writing—review and editing, M.Y. and Q.C.; visualization, S.Z.; supervision, T.W.; project administration, M.Y.; funding acquisition, M.Y. and T.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China (2020YFB1712400) and the National Natural Science Foundation of China (No. 52272423, No. 71672137).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: Author Bao Lu was employed by the company China Ocean Shipping Agency Tianjin Co., Ltd. Author Qiang Chen was employed by the company Qingdao New Qianwan Container Terminal Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Hao, C.; Yue, Y. Optimization on Combination of Transport Routes and Modes on Dynamic Programming for a Container Multimodal Transport System. *Procedia Eng.* **2016**, *137*, 382–390. [CrossRef]
2. United Nations Conference on Trade and Development (UNCTAD). *Review of Maritime Transport 2021*; UNCTAD: Geneva, Switzerland, 2021; 177p.
3. Chao, S.-L.; Lin, Y.-L. Gate Automation System Evaluation: A Case of a Container Number Recognition System in Port Terminals. *Marit. Bus. Rev.* **2017**, *2*, 21–35. [CrossRef]
4. Wu, L.; Gao, G.; Yu, J.; Zhou, F.; Yang, Y.; Wang, T. PDD: Partitioning DAG-Topology DNNs for Streaming Tasks. *IEEE Internet Things J.* **2024**, *11*, 9258–9268. [CrossRef]
5. Li, X.; Peng, Y.; Huang, J.; Wang, W.; Song, X. Simulation Study on Terminal Layout in Automated Container Terminals from Efficiency, Economic and Environment Perspectives. *Ocean Coast. Manag.* **2021**, *213*, 105882. [CrossRef]
6. Koo, K.; Cha, E. A Novel Container ISO-Code Recognition Method Using Texture Clustering with a Spatial Structure Window. *Int. J. Softw. Eng. Its Appl.* **2013**, *7*, 12.
7. Han, Y.; Li, T.; Zuo, Y.; Tian, Y.; Cao, Y.; Philip Chen, C.L. Application of Broad Learning System for Container Number Identification. In Proceedings of the 2018 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), Jinan, China, 14–17 December 2018; IEEE: Jinan, China, 2018; pp. 332–336.
8. Panahi, R.; Gholampour, I. Accurate Detection and Recognition of Dirty Vehicle Plate Numbers for High-Speed Applications. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 767–779. [CrossRef]
9. Cao, L.; Gai, Z.; Liu, E.; Gao, H.; Li, H.; Yang, L.; Li, H. Automatic Container Code Recognition System Based on Geometrical Clustering and Spatial Structure Template Matching. In *Communications, Signal Processing, and Systems*; Liang, Q., Mu, J., Jia, M., Wang, W., Feng, X., Zhang, B., Eds.; Lecture Notes in Electrical Engineering; Springer: Singapore, 2019; Volume 463, pp. 2198–2204. ISBN 978-981-10-6570-5.
10. Chun-ming, T.; Peng, C. Container Number Recognition Method Based on SSD_MobileNet and SVM. *Am. Sci. Res. J. Eng. Technol. Sci.* **2020**, *74*, 12.
11. Wang, Q.; Zhang, Q.; Liang, X.; Wang, Y.; Zhou, C.; Mikulovich, V.I. Traffic Lights Detection and Recognition Method Based on the Improved YOLOv4 Algorithm. *Sensors* **2021**, *22*, 200. [CrossRef]
12. Silva, S.M.; Jung, C.R. Real-Time License Plate Detection and Recognition Using Deep Convolutional Neural Networks. *J. Vis. Commun. Image Represent.* **2020**, *71*, 102773. [CrossRef]
13. Chen, S.; Qiu, C.; Yang, W.; Zhang, Z. Combining Edge Guidance and Feature Pyramid for Medical Image Segmentation. *Biomed. Signal Process. Control* **2022**, *78*, 103960. [CrossRef]
14. Zhiming, W.; Wuxi, W.; Yuxiang, X. Automatic Container Code Recognition via Faster-RCNN. In Proceedings of the 2019 5th International Conference on Control, Automation and Robotics (ICCAR), Beijing, China, 19–22 April 2019; IEEE: Beijing, China, 2019; pp. 870–874.
15. Feng, X.; Wang, Z.; Liu, T. Port Container Number Recognition System Based on Improved YOLO and CRNN Algorithm. In Proceedings of the 2020 International Conference on Artificial Intelligence and Electromechanical Automation (AIEA), Tianjin, China, 26–28 June 2020; IEEE: Tianjin, China, 2020; pp. 72–77.
16. Yoon, Y.; Ban, K.-D.; Yoon, H.; Kim, J. Automatic Container Code Recognition from Multiple Views. *Etri J.* **2016**, *38*, 767–775. [CrossRef]

17. Mei, L.; Guo, J.; Liu, Q.; Lu, P. A Novel Framework for Container Code-Character Recognition Based on Deep Learning and Template Matching. In Proceedings of the 2016 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII), Wuhan, China, 3–4 December 2016; IEEE: Wuhan, China, 2016; pp. 78–82.
18. Roeksukrungrueang, C.; Kusonthammrat, T.; Kunapronsujarit, N.; Aruwong, T.N.; Chivapreecha, S. An Implementation of Automatic Container Number Recognition System. In Proceedings of the 2018 International Workshop on Advanced Image Technology (IWAIT), Chiang Mai, Thailand, 7–9 January 2018; IEEE: Chiang Mai, Thailand, 2018; pp. 1–4.
19. Wu, J.; Guo, J.; Li, X. Key Methods of Recognizing Container Number Automatically Using Video Stream in Intelligent Tally. In Proceedings of the 2019 Chinese Intelligent Systems Conference, Nanchang, China, 3–6 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; Volume II, pp. 10–17.
20. Zhang, R.; Bahrami, Z.; Wang, T.; Liu, Z. An Adaptive Deep Learning Framework for Shipping Container Code Localization and Recognition. *IEEE Trans. Instrum. Meas.* **2020**, *70*, 1–13. [CrossRef]
21. Mi, C.; Cao, L.; Zhang, Z.; Feng, Y.; Yao, L.; Wu, Y. A Port Container Code Recognition Algorithm under Natural Conditions. *J. Coast. Res.* **2020**, *103*, 822–829. [CrossRef]
22. Feng, X.Q.; Liu, Q.; Wang, Z.W. Port Container Number Detection Based on Improved EAST Algorithm. *J. Phys. Conf. Ser.* **2020**, *1651*, 012088. [CrossRef]
23. Wan, Z.; Liu, Q.; Liu, T. Multichannel Real-Time Video Container Numbers Recogntion in Container Yard. In Proceedings of the 2021 China Automation Congress (CAC), Beijing, China, 22–24 October 2021; IEEE: Beijing, China, 2021; pp. 583–588.
24. Li, Y.; Li, H.; Gao, G. Towards End-to-End Container Code Recognition. *Multimed. Tools Appl.* **2022**, *81*, 15901–15918. [CrossRef]
25. Parez, S.; Dilshad, N.; Alanazi, T.M.; Weon Lee, J. Towards Sustainable Agricultural Systems: A Lightweight Deep Learning Model for Plant Disease Detection. *Comput. Syst. Sci. Eng.* **2023**, *47*, 515–536. [CrossRef]
26. Keserwani, P.; Dhankhar, A.; Saini, R.; Roy, P.P. Quadbox: Quadrilateral Bounding Box Based Scene Text Detection Using Vector Regression. *IEEE Access* **2021**, *9*, 36802–36818. [CrossRef]
27. Liu, Y.; Li, T.; Jiang, L.; Liang, X. Container-Code Recognition System Based on Computer Vision and Deep Neural Networks. *AIP Conf. Proc.* **2018**, *1955*, 040118.
28. Yar, H.; Khan, Z.A.; Ullah, F.U.M.; Ullah, W.; Baik, S.W. A Modified YOLOv5 Architecture for Efficient Fire Detection in Smart Cities. *Expert Syst. Appl.* **2023**, *231*, 120465. [CrossRef]
29. Singh, A.; Bhambhu, Y.; Buckchash, H.; Gupta, D.K.; Prasad, D.K. Latent Graph Attention for Enhanced Spatial Context. *arXiv* **2023**, arXiv:2307.04149.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.