Enhancing Container Number Recognition Accuracy through Multi-Model OCR Comparison and Post-Processing

Anakorn Roumpattana[1], Chalothon Chootong[1], Boonchoo Jitnupong[1], Sarut Serarom[2], Jirawan Charoensuk[1,*]

[1]Department of Computer Science and Information, Faculty of Science at Sriracha, Kasetsart University, Chonburi, Thailand

[2]KLN Seaport Limited, Chonburi, Thailand

Received 02 October 2025; revised 08 May 2026; accepted 14 May 2026

DOI: https://doi.org/10.46604/aiti.2026.15750

Abstract

Extracting container numbers from moving trucks in seaports remains challenging. This study aims to evaluate and compare the performance of optical character recognition (OCR) technologies, including EasyOCR, PaddleOCR, TesseractOCR, and TrOCR, for container number recognition in CCTV-based surveillance systems. The evaluation considers both horizontal and vertical text orientations with three-character categories: alphanumeric, alphabetic, and numeric. The results indicate that horizontal text recognition significantly outperforms vertical text recognition across all models and character categories. In experiments with alphanumeric container number formats, PaddleOCR and TrOCR achieve initial character error rates (CER) of 3.82% and 1.64% within 5.19 and 12.32 seconds, respectively. After applying post-processing rules, the CER is reduced to 0.36% for both models. PaddleOCR obtains comparable accuracy to TrOCR while offering a faster processing speed (7.78 seconds). Considering both recognition accuracy and processing time, PaddleOCR demonstrates an efficient performance for horizontal container number detection in seaport environments.

Keywords: optical character recognition, PaddleOCR, container number recognition, horizontal text alignment

1. Introduction

Container number recognition (CNR) is a fundamental component of modern logistics infrastructure, particularly for port operations and intelligent transportation systems. Accurate identification of container numbers from images is challenging, primarily due to varying environmental conditions, motion blur, and strict processing time and accuracy requirements [1]. Nevertheless, accurate CNR is essential for automated cargo delivery and supply chain optimization [2]. Standard manual inspection is inherently labor-inefficient and prone to errors, especially in large-volume port operations processing thousands of containers on a daily basis [2]. As a result, manual inspection can become a significant operational bottleneck that impacts overall system efficiency and precision. Container numbers also follow a standardized 11-character format consisting of an owner code, product group code, serial number, and check digits [3]. Although the standardized format promotes consistency, it also introduces specific challenges for automated recognition systems.

OCR technology has rapidly advanced from traditional rule-based systems to architectures based on deep learning and large language models, underscoring the need for comprehensive performance comparisons across various technological approaches. A thorough understanding of the strengths and limitations of each OCR model is therefore essential for selecting

* Corresponding author. E-mail address:jirawan.charo@ku.th

appropriate solutions for industrial logistics applications. However, comparative studies evaluating different OCR models for container number recognition under real-world port conditions remain limited. Several widely used OCR frameworks, including EasyOCR, Tesseract, and PaddleOCR, are based on deep learning.

EasyOCR, an open-source OCR framework developed by Jaided AI, is widely used for its ease of deployment and multilingual support capabilities [4]. It uses convolutional neural networks (CNNs) to detect and recognize text features in images. Coching et al. [5] developed a license plate recognition (LPR) system for traffic management using EasyOCR to extract license plate information. This is accomplished by drawing a bounding box around the license plate region and extracting the alphanumeric text within. Similarly, Ngo et al. [6] presented a vision-based container code inspection system for international port terminals, in which EasyOCR was used to precisely identify container code areas under complex operational conditions.

PaddleOCR, an open-source OCR system developed by Baidu, offers flexibility and high precision [7]. Its modular design separates text detection, recognition, and classification into distinct stages to enhance performance. The recognition module adopts a single-visual-transformer with a lightweight convolutional network (SVTR-LCNet), which combines a transformer-based architecture with lightweight CNNs [8] to balance recognition accuracy and computational efficiency. Prajapati et al. [9] presented a system combining YOLOv7 with PaddleOCR that achieved 97% accuracy in license plate detection and 95% in character recognition on a vehicle dataset of 1,000 images. In another study, Fabijanic et al. [10] integrated PaddleOCR and EasyOCR with YOLOv8 to detect vessel registration numbers from a monocular camera in 2023. Their results showed that PaddleOCR achieved 22.63% higher character-level text recognition accuracy than EasyOCR.

Tesseract OCR, initially developed by Hewlett-Packard and subsequently maintained by Google, is a widely adopted traditional OCR engine [11]. It has incorporated long short-term memory (LSTM) networks to improve sequence-based text recognition since 2018. Lestari and Mulyana [12] used Tesseract OCR to recognize text on packaging boxes during carton handling and found that preprocessing significantly improves performance, achieving 81.03% accuracy with a precision of 82.14%, recall of 90.79%, and an F1-score of 86.26%. Shithil et al. [13] developed an ISO container code recognition system using LSTM Tesseract OCR with images taken from multiple angles to improve recognition accuracy. Overall, these studies indicate that although Tesseract remains a reliable baseline OCR method, its performance is highly dependent on preprocessing and imaging conditions.

TrOCR is an OCR model developed using the transformer architecture to recognize text from both printed and handwritten images [14]. The encoder utilizes a vision transformer to convert input images into deep feature representations and captures their spatial relationships. The decoder is an autoregressive transformer that generates text tokens based on the previous context. It is pre-trained on large-scale synthetic datasets and fine-tuned with human-labeled datasets.

TrOCR has been applied to receipt-based OCR and information extraction tasks, where images are resized to different dimensions [15], enabling end-to-end recognition without explicit text localization. The fine-tuned model achieved a maximum F1-score of 87.8 and a character error rate (CER) of 4.98%, while preserving the correct reading order. Zhang [16] further evaluated TrOCR under varying image conditions. The results showed 97% character recognition accuracy with a black font on a white background, whereas the accuracy reached 99.7% under degraded image conditions, such as blur, noise, and low illumination. Therefore, the TrOCR model is robust at recognizing diverse text sources and is resilient to noise and rotation.

Traditionally, manual verification of container numbers has often led to human errors, delays, and inefficiencies, especially in high-throughput environments such as seaports and container yards. With recent developments in artificial intelligence and computer vision, OCR has emerged as a promising solution for automating container code identification. However, the real-world deployment of CNR systems using OCR-based methods presents a range of challenges. In practical

applications, container numbers may appear in either horizontal or vertical orientations depending on the container's positioning. Moreover, environmental variables such as poor lighting, adverse weather conditions, low image resolution, and viewing angles can further degrade recognition performance.

In addition, OCR often faces challenges when dealing with blurred, partially hidden, or degraded fonts, which can reduce the recognition accuracy. Variations in font styles, environmental weathering conditions, and inconsistent number placement on container surfaces increase the difficulty of accurate CNR. Therefore, modern CNR systems must balance accuracy and speed while ensuring compatibility and seamless integration with diverse port management, customs, and supply chain tracking systems within complex global logistics infrastructures.

To address these issues, object detection techniques have been integrated with OCR to enhance the accuracy of location identification and text reading. Among them, you only look once (YOLO) has been employed in vision tasks due to its balance of speed and precision. In this study, YOLOv11 [17] was used to detect the region and character sequences that appear in the container number. Specifically, a hierarchical approach comprising two processes—(1) container region detection and (2) character segmentation and alignment—is developed to detect both regions and characters, incorporating a transformation stage to normalize vertically oriented text into horizontal text alignment.

Subsequently, this study evaluates the CNR performance of four modern OCR models: EasyOCR, PaddleOCR, TesseractOCR, and TrOCR, for container numbers with both vertical and horizontal orientations. Furthermore, post-processing replacement rules were applied to resolve OCR errors. This dataset was collected from operational CCTV systems deployed at KLN Seaport Limited's logistics facilities. Performance evaluation was conducted by standard metrics, including precision, recall, mean average precision at 50 (mAP@50), and CER.

However, limited studies have compared OCR performance for container number recognition under different text orientations in real-world port environments. Therefore, the main contributions of this study are as follows: First, this study proposes a hierarchical detection and transformation pipeline using YOLOv11, which effectively addresses the challenge of recognizing both horizontal and vertical container numbers in automated port operations. Second, a rigorous comparative analysis of four OCR models (EasyOCR, PaddleOCR, TesseractOCR, and TrOCR) is presented using real-world CCTV data. Third, a specialized post-processing strategy based on ISO 6346 standards is implemented to minimize recognition errors. These contributions offer a practical and high-accuracy solution for automated logistics operations using real-world CCTV data from seaport environments.

The rest of this paper is structured as follows. Section 2 outlines the system design and methodology employed. Section 3 presents the dataset and experimental results along with a detailed discussion. Finally, Section 4 provides the conclusion and outline for future work.

2. System Design and Methodology

The proposed framework consists of six main stages, including system architecture overview, container region detection, character segmentation and alignment, OCR scanning, post-processing, and performance evaluation. Each stage was systematically designed to enhance the accuracy, reliability, and robustness of container number recognition in a seaport environment.

2.1 System Architecture Overview

The proposed system aims to extract container numbers from CCTV camera images captured at a seaport. As illustrated in Fig. 1, the CNR system utilizes a processing pipeline for automatic container number identification in a port environment. First, side-view images of trucks are captured from a CCTV camera installed at the entrance of a seaport facility. The camera is equipped with a motorized lens (2.7–13.5 mm) and records images at a resolution of 3840 × 2160 pixels with a frame rate

of 25 frames per second. The images were collected during daytime under real-world outdoor operational conditions, including varying lighting and weather conditions. The dataset primarily contains side-view container images in which the container identification numbers are vertically arranged on the container surfaces.

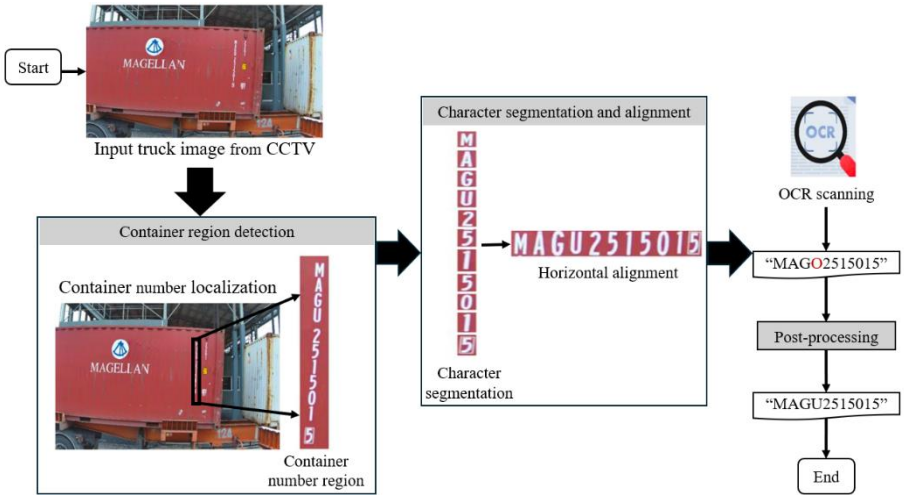


Fig. 1 The CNR architecture pipeline

These images are then subjected to container region detection, character segmentation and alignment, OCR scanning, and post-processing. The container region detection process crops the container number block using YOLOv11 nano (YOLOv11n). The character segmentation and alignment process also applies YOLOv11n to identify individual characters. The detected characters are sorted by their original sequence, resized, and then rotated to align with the character region. An OCR model is applied to scan the processed image and convert it to text. Finally, a post-processing step is implemented to enhance recognition accuracy using a rule-based approach. This process returns the container number (e.g., "MAGU2515015") in text format.

2.2 Container Region Detection

Container region detection is used to identify container number regions in backgrounds such as trucks, containers, and port facilities. This process extracts relevant image regions and prepares them for subsequent processing. A pre-trained YOLOv11n model is employed to detect regions of interest by predicting bounding boxes. The algorithm begins by loading input I_{img} and determining its dimensions (W, H). The detection model is then applied using a predefined confidence threshold (θ_{conf} = 0.10) and intersection over union (IoU) threshold (θ_{iou}= 0.10). For each predicted bounding box $b_i = (x_1, y_1, x_2, y_2)$, the detected region is adjusted to capture the relevant parts of the container number. These coordinates (x'_1, y'_1, x'_2, y'_2) are then used to crop the corresponding image region I_i. Each cropped region is validated and saved as a separate image file. The container region detection pseudocode is shown in Fig. 2.

Algorithm: Container region detection

Input:
I : Input image containing the side view of a truck
Region_detection: Pre-trained YOLOv11n model for container region detection
θ_{conf} : Confidence_threshold, θ_{iou} : IOU_threshold, p : padding size
Output: Cropped and saved images of detected regions containing container numbers

Begin
1. I_{img} ← load_Image (I)
2. (W, H) ← get_image_dimension (I_{img})
3. β ← *Region_detection.detect* (I_{img}, θ_{conf} = 0.10, θ_{iou} = 0.10)
4. **For each** $b_i = (x_1, y_1, x_2, y_2) \in$ β **do**
5. (x'_1, y'_1, x'_2, y'_2) ← padding (x_1, y_1, x_2, y_2, p = 1.00, W, H)
6. I_i ← crop (I_{img}, x'_1, y'_1, x'_2, y'_2)
7. **If** valid (I_i) = True **then**
8. f_i ← generate_filename ("RegionCrop" + i)
9. save_image (I_i, f_i)
10. **End If**
11. **End For**

End

Fig. 2 Pseudocode for container region detection

2.3 Character Segmentation and Alignment

The goal of the character segmentation and alignment process is to extract and organize detected characters into a consistent horizontal sequence in the container number from an image. This is achieved by converting vertically stacked container number images into a horizontally aligned format ($I_{concatenated}$) that is more suited for machine interpretation and subsequent text recognition tasks. For OCR systems, which often function better with horizontally arranged inputs, this modification increases readability and uniformity. Fig. 3 displays the character segmentation and alignment pseudocode.

Algorithm: Character segmentation and alignment

Input:
I : regions containing container numbers from Container region detection
Character_segmentation : Pre-trained YOLOv11n model for character segmentation
θ_{conf} : Confidence_threshold, θ_{iou} : IOU_threshold
Output: A horizontally concatenated and saved image of detected characters

Begin
1. $I_{img} \leftarrow$ load_image (I)
2. $\beta \leftarrow$ *Character_segmentation.detect* (I_{img}, θ_{conf} = 0.10, θ_{iou} = 0.10)
3. $\beta_{sorted} \leftarrow$ *sort_by_y_ascending* (β)
4. $S \rightarrow \emptyset$
5. **For each** $b_j = (s_{x1}, s_{y1}, s_{x2}, s_{y2}) \in \beta_{sorted}$ **do**
6. $I_j \rightarrow$ crop_image (I_{img}, s_{x1}, s_{y1}, s_{x2}, s_{y2})
7. **If** not_empty (I_j) = True **then**
8. $S \rightarrow S \cup \{I_j\}$
9. **End If**
10. **End For**
11. **If** $|S| > 0$ **then**
12. h_{max} = max{*height* (I_j) | $I_j \in S$}
13. $S_{resized} \rightarrow$ resize_all (S, $to_{height} = h_{max}$)
14. $I_{concatenated} \rightarrow$ *horizontal_concatenate* ($S_{resized}$)
15. f = generate_filename ("*CharacterConcatenated_*" + *timestamp* ())
16. save_image ($I_{concatenated}$, f)
17. **End if**
End

Fig. 3 Pseudocode for character segmentation and alignment

The algorithm begins by loading a cropped region of interest (ROI) I_{img} using $load\_image(I)$, which is generated from the container region detection stage. A pre-trained YOLOv11n model for character segmentation is applied to this image to detect individual characters, producing small-scale bounding boxes β with corresponding confidence threshold (θ_{conf} = 0.10) and IoU threshold (θ_{iou} = 0.10). Subsequently, the detected bounding boxes are sorted along the y-axis to create β_{sorted}, establishing a consistent reading order. Each detected box $b_j = (s_{x1}, s_{y1}, s_{x2}, s_{y2})$ is then used to crop the corresponding sub-image (I_j), and non-empty crops are collected in set *S*. This algorithm computes the maximum height h_{max} and resizes all sub-images accordingly to ensure vertical alignment, creating $S_{resized}$. These resized regions are then concatenated horizontally to produce a single image ($I_{concatenated}$). The final output is saved to a designated directory with a timestamped filename.

2.4 OCR Scanning

This study directly compared the recognition accuracy performance of four OCR models: EasyOCR, PaddleOCR, TesseractOCR, and TrOCR by adopting the RGB image format to maintain consistency and satisfy the input specifications of the models. All experiments were conducted on a machine equipped with an Intel Core i7-9750H CPU, 32 GB RAM, and an NVIDIA GeForce RTX 2070 GPU. Moreover, GPU acceleration was applied to compatible models. To preserve visual integrity, input photos were scaled in accordance with each engine's specifications while maintaining the aspect ratio. The setup for each model is summarized in Table 1.

Table 1 Configuration details of the OCR models

Model	Engine/Architecture	Configuration	GPU Accelerated
EasyOCR	Deep learning-based framework	English-language recognition, detailed output disabled	Yes
PaddleOCR	SVTR-LCNet (PP-OCRv4)	Detection module disabled, recognition component active	Yes
Tesseract OCR	LSTM-based engine (hybrid legacy system)	OCR engine mode: 3, page segmentation mode: 6	No (CPU only)
TrOCR	Transformer-based model (Microsoft)	Pre-trained model: microsoft/trocr-large-printed	Yes

2.5 Post-Processing

In this process, incorrect characters in the owner code, category identification, and serial number region were corrected using replacement rules to reduce the number of errors in the OCR engine outputs, according to ISO 6346. A summary of three replacement rules is shown in Table 2.

(1) Owner code replacement: The first three characters of the container number representing the owner code comprise letters A–Z, which must be capitalized. When working with partially buried or deteriorated fonts, alphanumeric characters are often confused by an OCR model due to their similarity. These replacements were selected based on their frequency of appearance in the dataset. For example, common OCR mistakes are corrected by replacing characters such as "0" with "O", "1" with "I", "5" with "S", and "3" with "E". These replacements help to ensure that the text is more accurate and matches the required format.

(2) Category identifier replacement: The category identifier is in the fourth position, in accordance with ISO-6346. It can be "J" for related equipment, "R" for refrigerated units, "U" for standard containers, and "Z" for trailers/chassis. ISO 6346 regulations dictate that the fourth character should be transformed as follows: 0, O, V, U → U; 2, 7, S, Z → Z; I, 1, L, J → J, while others are defaulted to the most prevalent type "U". The "R" category was excluded from this study.

(3) Serial number replacement: To improve the accuracy of the OCR output for the seven-digit serial number at the end of the container code, a character replacement algorithm was applied to address common misrecognitions. The following substitutions were implemented based on visual similarity between alphanumeric characters, e.g., O → 0, B → 8, U → 0, Q → 0, and T → 7. These mappings help correct frequently occurring OCR errors and align the output with the expected numeric format.

Table 2 Replacement rules

Post-processing	Replacement rules
Owner Code (1st–3rd chars)	0 → O; 1 → I; 8 → B; 6 → G; 2 → Z; 5 → S; 3 → E; 4 → A; 7 → T;
Category Identifier (4th char)	{0, O, V, U} → U; {2, 7, S, Z} → Z; {I, 1, L, J} → J; Others → U (default);
Serial Number (last 7 digits)	O, U, Q → 0; I, J, L → 1; Z → 2; B → 8; S → 5; G → 6; T → 7

2.6 Evaluation Metrics

For quantitative assessment of performance, this study employed multiple complementary metrics for both detection and recognition tasks.

(1) Detection Performance Metrics

To evaluate the detection and transformation processes using the YOLOv11n-based region detection model and the character segmentation model, this study employed standard object detection metrics: precision, recall, and mAP@50.

Precision and recall collectively assess the accuracy and completeness in detecting relevant instances. A detection is considered a true positive if the IoU between the predicted bounding box and the corresponding ground-truth bounding box exceeds a predefined threshold. Precision (the proportion of correctly predicted container number regions or each character region relative to all regions identified by the model) is defined as:

$$Precision = \frac{Ture\ positives}{True\ positives + False\ positives} \tag{1}$$

where recall (the proportion of actual container number regions that were correctly identified) is defined as:

$$Recall = \frac{True\ positives}{True\ positives + False\ Negatives} \tag{2}$$

where true positives represent correctly detected regions and false negatives refer to ground truth regions that the model missed.

mAP@50, which represents the mean average precision calculated at an IoU threshold of 0.5, is defined as:

$$mAP@50 = \frac{1}{|C|}\sum_{c \in C} AP_c \tag{3}$$

where |C| denotes the total number of classes. In this study, for both the container region detection and character segmentation and alignment process, |C| is set to 1, as both YOLOv11n models are trained to detect a single target class—either a container number region or an alphanumeric character. A prediction is considered a true positive only if its IoU with the ground truth bounding box is greater than or equal to 0.5.

(2) The OCR Evaluation Metric

CER was employed to evaluate OCR performance. It measures the proportion of character-level errors between the OCR-generated text and the ground truth text, and is defined as follows:

$$CER(\%) = \frac{S + D + I}{N} \cdot 100 \tag{4}$$

where *S* represents the number of substituted characters, *D* denotes the number of deleted characters, *I* serves as the number of inserted characters, and *N* denotes the total number of characters in the ground-truth text.

3. Results and Discussion

This section presents the experimental results and discussion of the proposed container number recognition framework. The performance of each stage, including container region detection, character segmentation and alignment, OCR scanning, and post-processing, is evaluated using CCTV data collected from a seaport environment. In addition, comparative analyses of the four OCR models are conducted using standard evaluation metrics and processing time measurements to assess both recognition accuracy and operational efficiency.

3.1 Container Region Detection Training Results

The container region detection dataset consists of 242 container images, divided into three subsets: 144 images for training, 48 for validation, and 50 for testing. All images were resized to a fixed resolution of 640 × 640 pixels to meet the input requirements of the YOLOv11 model. The annotation process focused on the container number region as a single class. Each image was annotated using Roboflow, with bounding boxes drawn to identify the container number regions. The training spanned 100 epochs with a YOLOv11n architecture (yolo11n.pt) at 640 × 640 resolution and a batch size of 16. An example of the dataset and the corresponding detection results are shown in Fig. 4.

Fig. 4 The results of using YOLOv11n with the container region detection algorithm

Fig. 5 presents the training results for container region detection using the YOLOv11n model for detecting container number regions. The classification loss showed the most significant reduction (84.6%; 3.05–0.47), followed by box loss (56.8%) and distribution focal loss (DFL) (40.2%). These results show that the first model learned to identify container number regions before refining their precise boundaries. The final model achieved precision and recall values of 0.98 and 0.99, respectively, and mAP@50 of 0.99. The latter metric value, surpassing 0.77, indicates effective localization precision across the IoU threshold.

The model improved quickly at first but showed less progress after 60 epochs and became nearly stable before reaching 100 epochs. Validation losses show that this model obtains good generalization performance without overfitting while maintaining precise boundary localization. These results confirm that the container region detection model achieved reliable container number localization performance suitable for logistics applications.

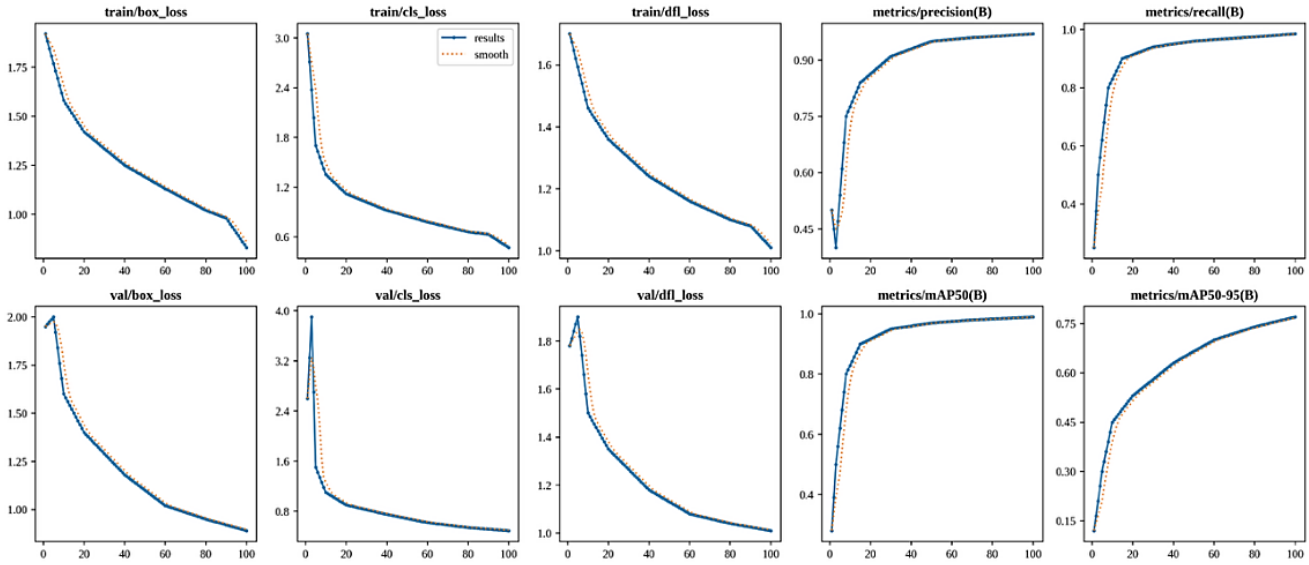


Fig. 5 Training results for YOLOv11n for annotating bounding container number regions

3.2 Character Segmentation and Alignment Training Results

The character segmentation and alignment algorithm is designed to extract and structurally organize fine-grained information, including both characters and digits, from input images. Similar to the container region detection process, all images were manually annotated using the Roboflow platform to ensure consistent and precise labeling. Accordingly, this model was trained to detect character regions, including digits (0–9) and uppercase letters (A–Z), as a single class. The dataset consisted of 3,050 characters collected from the training set (144 images) and validation set (48 images) used for this model.

The implementation of YOLOv11 demonstrated significant effectiveness for fine-grained character recognition within the container identification numbers. The training protocol encompassed 100 epochs utilizing a batch size of 16. The training and validation loss graphs in Fig. 6 show consistent convergence over 100 epochs. The box loss decreased from approximately 1.5 to 1.05, while the classification loss improved to 0.5, and the DFL decreased to 0.95.

The experimental results demonstrate that the model can accurately distinguish characters from background images, with precision and recall values of approximately 95% and mAP@50 values approaching 97%, indicating the model's ability to identify and locate individual characters effectively. These results confirm that the proposed model is robust for fine-grained character-level detection in complex container image environments. The consistent performance across training and validation sets further indicates good generalization ability without significant overfitting.

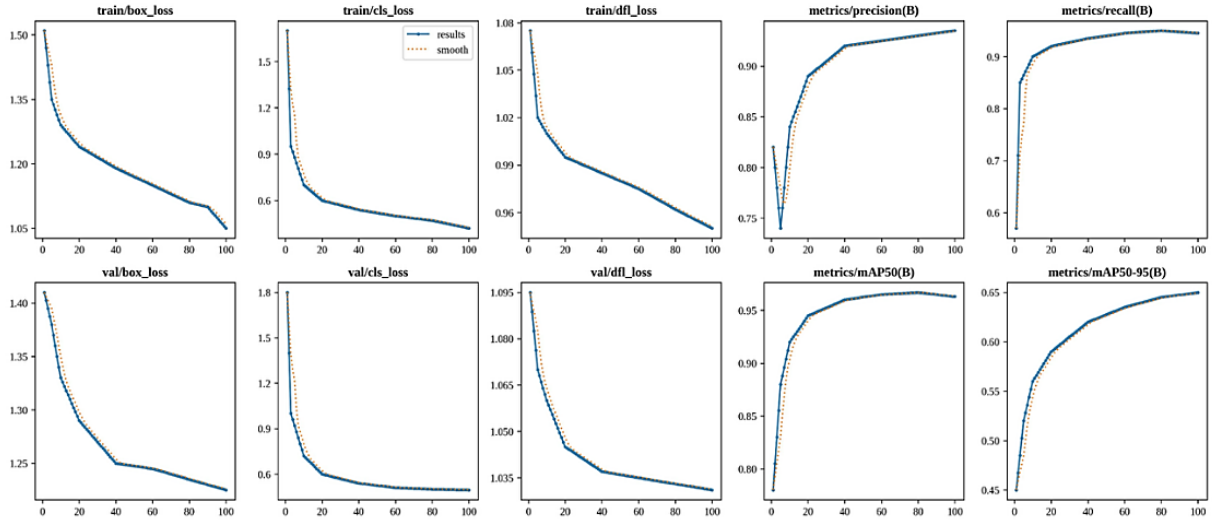


Fig. 6 Training results for YOLOv11n bounding each character of the container number

To further evaluate the performance of the proposed model, experiments were conducted using a test set of 50 side-view container images containing 550 individual character instances. The evaluation was formulated as a binary classification task consisting of two classes: character and background. The classification performance is shown through the confusion matrix presented in Fig. 7. Based on the matrix, the model achieved a precision of 0.954 and a recall of 0.950. These findings demonstrate that the YOLOv11n model effectively identifies characters on container images and accurately determines the bounding boxes for each character, providing a robust foundation for the subsequent OCR process.

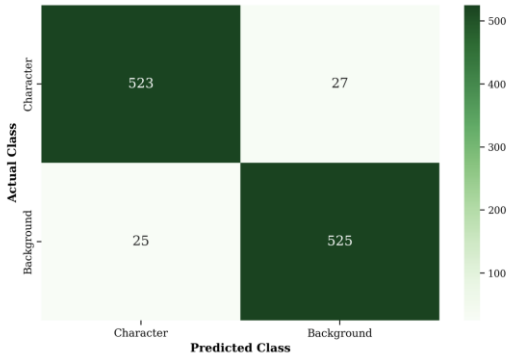


Fig. 7 Confusion matrix of the character detection performance

To demonstrate the character segmentation and alignment process, Fig. 8 presents example results of the vertical-to-horizontal transformation process, which takes a vertically arranged container number region from the previous stage as input. Fig. 8(a) shows the character segmentation regions in a vertical layout, where 11 individual characters are detected and localized. Fig. 8(b) presents the final output, where the detected characters are cropped, aligned, and concatenated into a horizontally aligned format. This transformation normalizes character orientation and facilitates the subsequent OCR process.

(a) Character segmentation regions (vertical layout)

(b) Concatenated characters in a horizontally aligned format

Fig. 8 Example results of the vertical-to-horizontal transformation

3.3 OCR Scanning Results

Beforehand, the dataset was labeled with ground-truth container numbers following the ISO 6346 format (e.g., "MAGU2515015"). To evaluate OCR performance, each image was tested in both vertical and horizontal orientations. In addition, each image was analyzed in three distinct configurations to isolate performance across character types: (1) the alphanumeric category, which encompasses the full 11-character code; (2) the alphabetic category, represented by the initial four-letter prefix indicating the owner and equipment category; and (3) the numeric category, comprising the final seven-digit sequence used for serial identification and checksum validation. This structured approach enabled a fine-grained analysis of OCR accuracy with respect to both layout orientation and character composition.

Table 3 CER comparison using the four OCR models

Container Number Orientation	Model	CER		
		Alphanumeric	Alphabetic	Numeric
Vertical	TesseractOCR	46.18%	30.50%	47.14%
	EasyOCR	89.82%	100.00%	87.71%
	PaddleOCR	100.00%	100.00%	100.00%
	TrOCR	106.00%	95.50%	99.43%
Horizontal	TesseractOCR	12.73%	6.50%	6.86%
	EasyOCR	24.73%	22.00%	20.86%
	PaddleOCR	3.82%	1.00%	0.00%
	TrOCR	1.64%	1.00%	0.29%

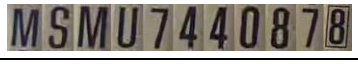
The experimental results of OCR scanning without post-processing are shown in Table 3. It can be seen that the four OCR models could read characters from horizontal images more accurately than from vertical ones. For the latter, TesseractOCR achieved the lowest CER values: 46.18%, 30.50%, and 47.14%, for all characters, letters, and numbers, respectively, significantly outperforming the other models. The other three models (EasyOCR, PaddleOCR, and TrOCR) had CER values close to or greater than 100%, which indicates more than one error per character. Moreover, these OCR methods tended to generate outputs with more characters than those in the actual container numbers, and letters and numbers were usually inserted in the wrong positions.

For horizontal container numbers, PaddleOCR registered the best performance in reading numeric data with a CER of 0.00%, followed by TrOCR (0.29%). For letters in the alphabet, PaddleOCR and TrOCR provided the best CER values (1.00%), followed by TesseractOCR (6.50%). For the alphanumeric category (the complete container numbers), all four models could read data from horizontal images more accurately than from vertical images, with TrOCR providing the lowest CER value (1.64%), followed by PaddleOCR, TesseractOCR, and EasyOCR (3.82%, 12.73%, and 24.73%, respectively).

3.4 Post-Processing Results

Table 4 presents representative examples of OCR error correction before and after post-processing using the proposed replacement rules. The examples illustrate how common character confusions in container numbers are corrected based on position-specific rules, including the owner code, category identifier, and serial number. The CER values for alphanumeric, alphabetic, and numeric characters used to evaluate the post-processing of both vertical and horizontal images are reported in Table 5.

Table 4 Examples of OCR output correction using the post-processing

Container image	Before post-processing	After post-processing	Error position	Replacement rule
	**7**G**8**U6522097	**T**G**B**U6522097	1st (7→T), 3rd (8→B)	Owner Code
	TSS**0**2228519	TSS**U**2228519	4th (0→U)	Category identifier
	MSMU744087**B**	MSMU744087**8**	Last digits (B→8)	Serial number

The experimental results for vertical container images in Tables 3 and 5, before and after post-processing, indicate that TesseractOCR consistently achieved the lowest CER values across all character types, outperforming EasyOCR, PaddleOCR, and TrOCR. These results suggest that the implemented post-processing procedure had minimal impact on error rates across all OCR models. However, slight reductions in CER were observed for the TrOCR model after post-processing, particularly for alphanumeric and alphabetic characters (with reductions of 2.36% and 9.00%, respectively). Despite these improvements, the overall impact of post-processing on vertical images remained limited.

Table 5 CER Comparison for the four OCR models after post-processing

Container Number Orientation	Model	CER		
		Alphanumeric	Alphabetic	Numeric
Vertical	TesseractOCR	46.18%	30.50%	47.14%
	EasyOCR	89.82%	100.00%	87.14%
	PaddleOCR	100.00%	100.00%	100.00%
	TrOCR	103.64%	86.50%	99.43%
Horizontal	TesseractOCR	10.18%	6.00%	6.86%
	EasyOCR	23.45%	21.50%	20.86%
	PaddleOCR	0.36%	0.50%	0.00%
	TrOCR	0.36%	0.50%	0.29%

For horizontal images, PaddleOCR and TrOCR achieved the lowest CER values for both alphanumeric and alphabetic characters, with CER values of 0.36% and 0.50%, respectively, significantly outperforming TesseractOCR and EasyOCR. For numeric characters, PaddleOCR achieved the lowest CER (0.00%), followed by TrOCR (0.29%). In addition, post-processing substantially improved the performance of PaddleOCR and TrOCR, reducing their CER values by 3.46% and 1.28%, respectively. These results indicate that PaddleOCR and TrOCR are more effective for recognizing container numbers in horizontal images, particularly when combined with the proposed post-processing method.

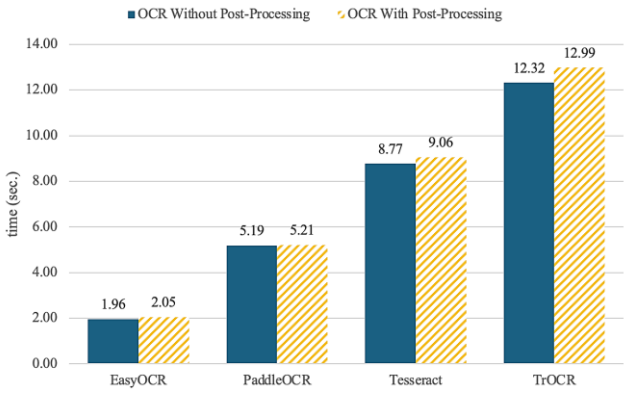


Fig. 9 Comparison of the processing times for the OCR models

When selecting an OCR model for extracting container numbers from trucks in a port environment, both efficiency and system throughput should be considered. Fig. 9 compares the time required to extract container numbers from 50 horizontal images with and without post-processing. The results indicate that although EasyOCR was the fastest, it also achieved the highest interpretation error rate. When considering PaddleOCR and TrOCR, which both achieved low CER values indicating superior accuracy over the other two models, the processing time for TrOCR with and without the post-processing procedure (12.32 and 12.99 seconds, respectively) was significantly longer than that of PaddleOCR (which was approximately 2.5 times faster than TrOCR). In addition, the processing time for the PaddleOCR with the post-processing procedure was only 3 seconds longer.

4. Conclusions

This study proposed and evaluated a CNR system for port operations and logistics. The system comprised a sequential processing pipeline starting with truck images captured from CCTV cameras. The proposed framework integrated YOLOv11-

based container region detection, character segmentation and alignment, OCR scanning, and post-processing based on ISO 6346 standards. In addition, a vertical-to-horizontal transformation process was applied to improve OCR readability for vertically arranged container numbers. Experimental evaluation was conducted using EasyOCR, PaddleOCR, TesseractOCR, and TrOCR under both vertical and horizontal text orientations. The results demonstrate that the proposed framework can effectively improve container number recognition accuracy and support logistics operations in seaport environments. The main conclusions of this study are as follows:

(1) The YOLOv11n-based container region detection model achieved reliable localization performance with precision, recall, and mAP@50 values of 0.98, 0.99, and 0.99, respectively.

(2) The YOLOv11n-based character segmentation and alignment model effectively identified and localized individual characters, achieving precision, recall, and mAP@50 values of approximately 0.95, 0.95, and 0.97, respectively.

(3) A clear performance gap was observed between vertical and horizontal text orientations, with horizontal images achieving lower CER values across all OCR models.

(4) PaddleOCR and TrOCR achieved the lowest CER values for horizontal images, while TesseractOCR showed comparatively better performance for vertical container numbers.

(5) The proposed ISO 6346-based post-processing method effectively reduced character recognition errors, particularly for PaddleOCR and TrOCR, improving the reliability of container number recognition results.

(6) PaddleOCR achieved the most balanced performance in terms of recognition accuracy and computational time, making it suitable for deployment in port environments, while TrOCR also provided high recognition accuracy but required longer processing time.

Future work will be focused on real-time deployment by integrating the proposed pipeline into live CCTV systems within a port environment. This includes developing monitoring dashboards, OCR logging capabilities, and alert mechanisms for detecting container mismatches. Furthermore, integrating CNR with LPR into a single pipeline could strengthen the association between vehicles and containers, thereby enhancing both efficiency and security in port operations.

Acknowledgments

This research was financially supported by the Faculty of Science at Sriracha and the Kasetsart University Research and Development Institute (KURDI), Kasetsart University.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] M. Yu, S. Zhu, B. Lu, Q. Chen, and T. Wang, “A Two-Stage Automatic Container Code Recognition Method Considering Environmental Interference,” Applied Sciences, vol. 14, no. 11, article no. 4779, 2024.

[2] R. Shetty, R. Cáceres, J. Pastrana, and L. Rabelo, “Optical Container Code Recognition and Its Impact on the Maritime Supply Chain,” Proceedings of the Industrial and Systems Engineering Research Conference, Orlando, Florida, USA, pp. 1535-1544, 2012.

[3] C. Tang and P. Chen, “Container Number Recognition Method Based on SSD_MobileNet and SVM,” American Scientific Research Journal for Engineering, Technology, and Sciences, vol. 74, no. 1, pp. 200-211, 2020.

[4] JaidedAI, “EasyOCR,” https://github.com/JaidedAI/EasyOCR, accessed in 2024.

[5] J. K. Coching, A. J. L. Pe, S. G. D. Yeung, C. M. L. Ang, R. S. Concepcion, and R. K. C. Billones, "License Plate Recognition System for Improved Logistics Delivery in a Supply Chain with Solution Validation through Digital Twin Modeling," IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), Coron, Palawan, Philippines, 2023.

[6] D.-D. Ngo, V.-H.-A. Phan, H.-T. Pham, T.-T. Be, V.-B. Nguyen, and M.-H. Le, "A Vision-Based Container-Code Checking System: Case Study at International Terminal," 2023 International Workshop on Intelligent Systems (IWIS), Ulsan, Republic of Korea, 2023.

[7] Y. Du, C. Li, R. Guo, X. Yin, W. Liu, J. Zhou, et al., "PP-OCR: A Practical Ultra Lightweight OCR System," arXiv preprint, arXiv:2009.09941, 2020.

[8] Y. Du, Z. Chen, C. Jia, X. Yin, T. Zheng, C. Li, et al., "SVTR: Scene Text Recognition with a Single Visual Model," arXiv preprint, arXiv:2205.00159, 2022.

[9] R. K. Prajapati, T. Nagar, S. Dangi, Y. Bhardwaj, P. R. S. Rao, and R. K. Jain, "Automatic Number Plate Recognition Using YOLOv7 and PaddleOCR," International Journal of Advanced Research in Science, Engineering and Technology, vol. 10, Special Issue 2, pp. 26-32, 2023.

[10] M. Fabijanić, M. Magdalenić, J. Obradović, N. Kapetanović, F. Ferreira, and N. Mišković, "Vessel Registration Number Detection and Recognition System," Proceedings of IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW), IEEE Press, pp. 1450-1456, 2025.

[11] R. Smith, "An Overview of the Tesseract OCR Engine," Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR), IEEE Press, pp. 629-633, 2007.

[12] I. N. T. Lestari and D. I. Mulyana, "Implementation of OCR (Optical Character Recognition) Using Tesseract in Detecting Character in Quotes Text Images," Journal of Applied Engineering in Technological and Science (JAETS), vol. 4, no. 1, pp. 58-63, 2022.

[13] S. M. Shithil, A. R. M. Kamil, S. Tasnim, and A. A. M. Faudzi, "Container ISO Code Recognition System Using Multiple View Based on Google LSTM Tesseract," Proceedings of Computational Intelligence in Machine Learning, Springer, pp. 433-440, 2022.

[14] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, et al., "TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models," Proceedings of the AAAI Conference on Artificial Intelligence, pp. 13094-13102, 2023.

[15] H. Zhang, E. Whittaker, and I. Kitagishi, "Extending TrOCR for Text Localization-Free OCR of Full-Page Scanned Receipt Images," Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), IEEE Press, pp. 1471-1477, 2023.

[16] R. L. Zhang, "A Comprehensive Evaluation of TrOCR with Varying Image Effects," National High School Journal of Science, pp. 1-10, 2024.

[17] R. Khanam and M. Hussain, "YoLov11: An Overview of the Key Architectural Enhancements," arXiv preprint, arXiv:2410.17725, 2024.