# Design and Implementation for BIC Code Recognition System of Containers using OCR and CRAFT in Smart Logistics

HANGSEO CHOI, JONGPIL JEONG, CHAEGYU LEE, SEOKWOO YUN, KYUNGA BANG, JAEBEOM BYUN
Department of Smart Factory Convergence,
Sungkyunkwan University,
Cheoncheon-dong, Jangan-gu Suwon-si, Gyeonggi-do,
REPUBLIC OF KOREA

*Abstract:* - The BIC (Bureau International des Containers et du Transport Intermodal) Code is the identification code for ocean shipping containers and is crucial for logistics, transportation, and security. Accurate recognition of container BIC Code is essential for efficient import and export processes, authorities' ability to intercept illegal goods and safe transportation. Nevertheless, the current practice of employees recognizing and manually entering container BIC codes is inefficient and prone to error. Although automated recognition efforts have been made, challenges remain due to the aging of containers, manufacturing differences between companies, and the mixing of letters and numbers in the 11-digit combination. In this paper, we propose the design and implementation of a BIC Code recognition system using an open source-based OCR engine, deep learning object detection algorithm, and text detector model. In the logistics industry, various attempts are being made to seamlessly link the data required at each stage of transportation between these systems. If we can secure the stability and consistency of BIC Code recognition that can be used in the field through our research, it will contribute to overcoming the instability caused by false positives.

*Key-Words:* - Smart Logistics, Container, BIC-code, Object Detection, Text Detection, OCR.

Received: June 6, 2022. Revised: April 17, 2023. Accepted: May 12, 2023. Published: June 16, 2023.

## 1 Introduction

The Container BIC Code can be written in different languages and fonts, so it is important to use robust algorithms and development logic when developing a container BIC Code recognition system, [1]. To achieve this, it is essential to conduct intensive studies on Object Detection models and build a model with good performance, [2]. Object detection is a technique that utilizes deep learning to recognize and locate specific objects in images or videos, [3]. There are various types of object detection models, which can be categorized into 1-Stage and 2-Stage models. In a two-stage model, the location of the object is first predicted and then the object is classified based on that location. To accomplish this, the system first utilizes an RPN(Region Proposal Network) to extract regions in the image that are likely to contain objects. These regions are then processed through a backbone network of CNNs to generate a feature map. The feature map is subsequently transformed into a fixed-size feature map using RoI(Region of Interest) Pooling. Finally, the RoI features are fed into the Classification and Bounding Box Regression Layer, consisting of a Fully Connected Layer and a Softmax Layer, to predict the object class and location. A well-known model in this category is Faster R-CNN, [4]. On the other hand, 1-Stage models, such as YOLO, are known for their faster processing speed. YOLO divides the image into grid cells to identify objects and predicts the probability and location of objects within each grid cell. Another algorithm called SSD extracts feature maps of different sizes to predict the location and class of objects, [5].

Since container BIC Codes are typically found in specific areas of the container, a Bounding Box can be created to identify and extract those regions, [6]. Object Detection employs a CNN (Convolutional Neural Network) based algorithm to generate bounding boxes and extract features for extracting the BIC Code from images, [7]. To read the text within the extracted region, OCR (Optical Character Recognition) is necessary, wherein the character recognition algorithm considers the object's size and position when processing the image, [8].

In the past, it was common practice to extract container BIC codes by separately conducting object

detection and OCR. However, this study evaluates object detection and OCR in an integrated manner, which can serve as a reference in real-world industrial applications. It is also worth noting that we carefully selected a model with an outstanding performance by comparing various open-source OCR engines, as well as the actual performance of object detection models such as Faster R-CNN, YOLOv5, and the CRAFT text detection model. This enables us to assess the strengths and weaknesses of each model and contribute to the methodology of selecting the optimal model for the field. The main idea of this study is to generate BIC Code coordinates extracted through object detection as custom input for OCR, thereby generating images for analysis. By comparing different object detection techniques and models to identify a model that can be customized for industrial applications, valuable insights, and practical solutions can be obtained. Ultimately, the extraction of container BIC codes plays a crucial role in the logistics and shipping industry, and this research can contribute to the development of more accurate and efficient logistics and shipping management systems. This paper is highly relevant in the field of computer vision and artificial intelligence, and it is expected to generate significant interest in logistics-related domains such as transportation and ports.

The paper is organized as follows: Section 2 provides an overview of the technology and the concept of OCR using deep learning, Section 3 presents the overall architecture of the system, Section 4 describes the implementation process, and Section 5 concludes with future research considerations.

# 2 Related Work

## 2.1 Object Detection

Object detection is a technique for detecting and recognizing specific objects in images or videos. The Object Detection model determines where a particular object exists in the input image and marks the region of the object as a bounding box, [9]. These bounding boxes represent the location and size of the object and are used to extract the object's location, which is a detail as well as the classification of the object. Some of the algorithms that can be used to perform object detection are YOLO, Faster R_CNN, and SSD, [10].

These algorithms divide the image into multiple grid cells and train a deep-learning model that predicts the presence of objects and bounding box information for each cell. These deep learning models are typically based on CNNs. Each algorithm works slightly differently, but they all preprocess the input image and then use a CNN to generate a feature map. This feature map is used to predict the presence of an object in each cell and its bounding box information. Bounding boxes are then generated based on the predicted information in each cell, and redundant bounding boxes are removed using the NMS(Non-Maximum Suppression) algorithm. The Object Detection model is trained using a large dataset, and the trained model can identify and recognize objects in new images. You can also use Transfer Learning techniques to fine-tune the pre-trained model to improve object detection accuracy. Thus, Object Detection can be used to identify the location of an object in an image containing a container BIC Code. The image at that location can then be passed to OCR to extract text, classify it, and recognize the container BIC Code.

### 2.1.1 Faster R_CNN

Faster R_CNN is a deep learning model that combines RPN and Fast R_CNN to perform object detection. The architecture of Faster R_CNN consists of RPN, RoI Pooling Layer, and Classifier. Fig. 1 explains the architecture and behavior of the described Faster R-CNN.

- RPN: It is responsible for finding regions in the input image where objects are likely to be present. To do this, RPN uses a CNN that performs a sliding window on the image and predicts the likelihood of an object being present in each window.
- RoI Pooling Layer: The candidate object regions found by the RPN are passed to the RoI Pooling Layer to create a fixed-size feature map. This allows it to effectively handle objects of different sizes.
- Classifier: Takes the feature map generated by the RoI Pooling Layer as input and is used to predict the class and bounding box of an object.

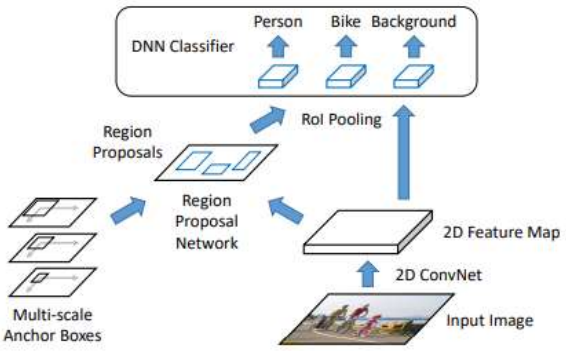


Fig. 1: Contrasting the Faster R-CNN architecture for object detection in images, [11].

### 2.1.2 YOLO5

YOLOv5 is based on the concept of YOLO (You Only Look Once) and provides faster processing speed and higher accuracy, [12]. YOLOv5 is mainly composed of Backbone Network, Neck, and Head.

The Backbone Network is responsible for extracting the feature map from the input image. YOLOv5 can use backbone networks such as EfficientNet and CSPDarknet53, and the Neck processes the feature map to predict the location and size of objects. YOLOv5 can use Necks such as SPP, PANet, etc. Finally, the Head takes the feature map generated by the Neck as input and predicts the class and bounding box of the object. Like YOLOv3, YOLOv5 uses feature maps of various scales to predict the location and size of objects and uses concepts such as anchors and grid cells to generate bounding boxes.

When you input an image containing a container BIC Code using a model trained with Object Detection, the model determines the location of the object in the input image and creates a bounding box around it. From this, the location of the container BIC Code can be determined, and the text can be extracted, classified, and recognized using the OCR model. The OCR network and architecture can be divided into three parts. First, there are the Convolutional Layers, which preprocess the input images and convert them into a format that can be used as input to the CNN. Then there are the Recurrent Layers, which use the output of the Convolutional Layers as input to the RNN. The Recurrent Layers process the features of the image as a continuous sequence, from which the textual information of the image is extracted. Finally, there are Fully Connected Layers to identify the extracted textual information. Through this process, the trained model can recognize patterns of container BIC Code in the input image, separate each pattern and combine the recognized text to recognize the entire container BIC Code. However, in the case of the container BIC Code, there is a problem in that the check digit needs to be recognized separately. In this case, you can extract the check digit by first recognizing 4 alphanumeric characters and 6 numeric characters, combining them to get a 10-digit number, and then applying an algorithm to calculate the check digit.

## 2.2 OCR

OCR is a technology that recognizes text in images and converts it into machine-readable text. OCR is used in a variety of fields and is widely used in areas such as document recognition, license plate recognition, and handwriting recognition. The general way OCR works starts with preprocessing. Before text can be recognized from an image, preprocessing is necessary and can include image resizing, denoising, contrast enhancement, and binarization, [13]. Next comes feature extraction. To extract text from a preprocessed image, various feature extraction algorithms can be used to extract features of the text, [14]. For example, algorithms such as HOG (Histogram of Oriented Gradients), SIFT (Scale-Invariant Feature Transform), SURF (Speeded Up Robust Feature), and CNN are used. Character recognition involves extracting features and then recognizing them in an image. Character recognition can be performed with a variety of algorithms, including machine learning algorithms (SVM, KNN, Neural Networks, etc.), statistical models (Hidden Markov Model, Conditional Random Field, etc.), and rule-based approaches.

Finally, the text generated by the character recognition process is subjected to post-processing to improve its accuracy. Post-processing can include error correction, character segmentation and merging, word segmentation, and application of language models, [15]. The performance of OCR is affected by a variety of factors. For example, image quality, character size, font, background, etc. affect the accuracy of OCR. OCR also performs differently in different languages, and models must be built for each language.

## 2.3 Text Classification

Text Classification is a technique for assigning a given piece of text to a predefined class or category. It is used for spam filtering, sentiment analysis, category categorization, etc., and trained and consists of five main steps. The first is data collection and preprocessing, where the collected data must be stored in text format and have a balanced distribution of all possible classes or categories. The preprocessing step normalizes the textual data, removes special characters, dead

words, etc., tokenizes words, and creates a vector representation for each word, [16]. Each text sample must then be converted to a vector, [17]. Several techniques can be used to generate these vector representations. Techniques such as BoW(Bag of Words), TF-IDF(Term Frequency-Inverse Document Frequency), Word2Vec, GloVe, etc. can be used in this step to convert the text into vectors that the model can understand. Finally, the model is selected and trained, and evaluated to tune the model and make predictions. There are many different text classification models. These models take the vectors generated from feature extraction as input and learn to assign every input vector to a corresponding class, [18]. Typical text classification algorithms include Naïve Bayes, SVMs, Random Forests, and Neural Networks. Recently, transformer-based models, such as BERT and GPT, have become popular. Model evaluation metrics include accuracy, precision, recall, and F1 score, and hyper-parameter tuning is performed to improve model performance, and the final model is generated.

## 2.4 Deep Learning Frameworks

Deep Learning Frameworks enable the implementation and training of large-scale complex models and provide high performance by utilizing hardware accelerators such as GPUs or TPUs, [19]. Major deep learning frameworks include TensorFlow, Pytorch, Keras, MXNet, and Caffe. All these frameworks, [20], are open-source and continuously evolved by the community. Of these, TensorFlow and PyTorch are among the most widely used frameworks. TensorFlow is an open-source library developed by Google that is very efficient for implementing large-scale neural network models. Pytorch is an open-source library developed by Facebook that supports native Python code for easy debugging and development and supports dynamic graph computation for fast model development and experimentation.

Pytorch provides various functions such as model design, data processing, model training, and deployment. Model design can be easily implemented by inheriting the nn. Module class and data processing can be easily implemented using Pytorch's Dataset and DataLoader. Model training provides various techniques such as learning rate schedule, weight initialization, and normalization, and supports GPU acceleration for high performance. Pytorch also allows you to deploy your models on a variety of platforms, including mobile and web.

Pytorch is based on the Torch library, a tensor computation library, which provides various modules, projects, and libraries to support various operations, making it easy to implement various deep learning models such as CNN, RNN, and Transformer. Due to these features, Pytorch has an active community and is used by many researchers and developers, making it easier and faster to implement deep learning models.

## 2.5 CRAFT

The CRAFT (Character-Region Awareness For Text Detection) model is based on the VGG16 architecture, and Basenet defines the structure of VGG16. Basenet is divided into five sections from Slice1 to Slice5, and each section consists of a CNN layer, Batch Normalization, ReLU activation function, and Max Pooling layer. The four slices from Upconv1 to Upconv4 use a U-Net structure, each using a Double Convolution layer to expand the feature map. Finally, Conv_cls processes the feature map using the Convolutional layer and ReLU activation function and finally generates an output to the Convolutional layer with two output channels, which is a two-dimensional binary map representing the character region detection result. Fig. 2 illustrates the CRAFT processing process.

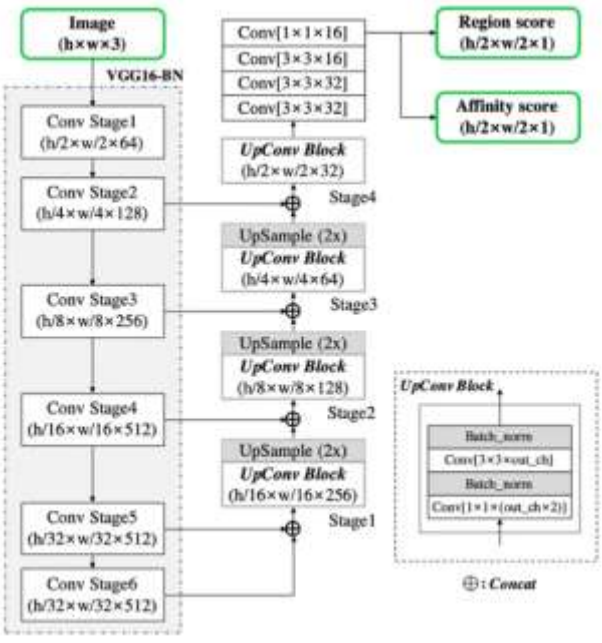


Fig. 2: CRAFT Schematic illustration of our network architecture, [21]

# 3 Proposed Method

## 3.1 Container BIC Code Recognition System

The container BIC code consists of 11 characters. The 11 characters consist of the Owner Code, Product Group Code, Serial Number, and Check Digit, as shown in Fig. 3.


Fig. 3: Container BIC Code Structure

Since the container code requires all 11 characters to be detected to be meaningful as a code, if any of these characters are missing, it is difficult to recognize the BIC Code. In addition to the difficulty of recognizing the BIC Code, noise, blur, deformation, etc. may occur due to varying image quality, making object recognition difficult. As shown in Fig. 4, the check digit at the end of the code, which is a number for accurate data delivery and verification of the container, is located at the end of the container BIC Code and cannot be easily detected by the OCR algorithm because it is a number in a box.


Fig. 4: Check Digit location and Shape

The idea of this paper is to use the Object Detection model to detect the image region containing the container BIC Code, identify the location of the object, and pass the location of the object to OCR to extract the text at that location, classify it, and recognize the container BIC Code. An additional task in this process is to check the Check Digit. It is used to self-verify the BIC Code without relying on preprocessing, as shown in the last step in Fig. 5.

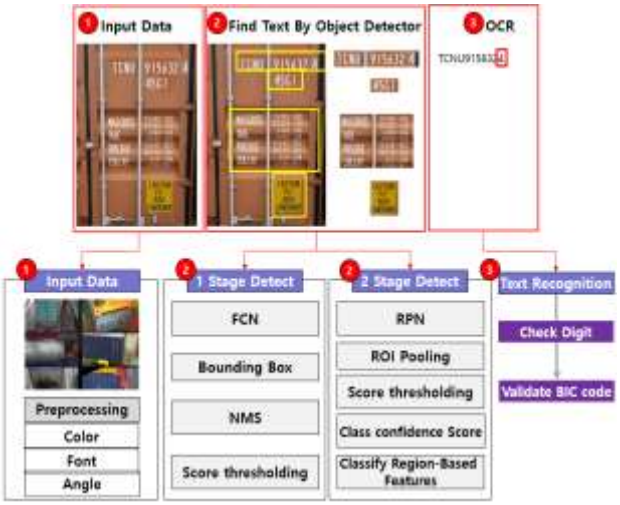


Fig. 5: System Concept

## 3.2 Deep Learning Object Detection Algorithms

To recognize the container BIC Code, an OCR model must be used, which is responsible for recognizing and converting text from a given image into text, [22]. Since the Container BIC Code consists of 4 alphanumeric characters, 6 numeric characters, and 1 check digit, the OCR model must be able to recognize and classify this pattern to extract the corresponding Container BIC Code, [23]. For this purpose, we selected Faster R_CNN and YOLOv5 as the Object Detection to use. Object detection models such as Faster R_CNN and YOLOv5 can be used to localize the container BIC Code in the input image and generate the bounding box of the object, [24]. However, the most important thing is the detection of Check Digit. This is because the container image may contain multiple texts, and if the Check Digit is not detected, the BIC Code cannot be matched even if the other texts are extracted normally. Fig. 6 shows the training structure using a table detection model to find Check Digit.

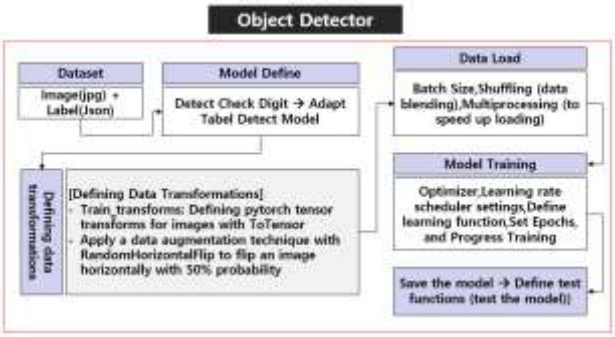


Fig. 6: Table detection model training structure for check digit recognition

### 3.3 Text Object

CRAFT is one of the deep learning models responsible for text regions. It can extract text regions because it works by segmenting text regions in detail to detect character boundaries and then predicting character boundaries. As a result, CRAFT is highly accurate in text recognition tasks and can be analyzed by object detection models and OCR engines because of this characteristic. However, CRAFT does not recognize the content of characters in text recognition tasks, only the boundaries of characters.

CRAFT, as well as other object detection algorithms, must locate the BIC Code location and integrate it with an OCR library based on the coordinates to extract, classify, and recognize the BIC Code. At this point, the OCR model will segment the image to find the exact location and boundaries of the recognizable characters in the image. A segment is a part of an image that contains a character. The OCR model finds and extracts segments containing characters from the input image and converts them into strings. To do this, the OCR model performs segmentation during image preprocessing. The segmentation process consists of two steps: first, finding the character regions, and second, separating the character regions. There are several ways to find character regions, including global thresholding and local thresholding. After finding the text area, there are methods to separate the actual character from the area, such as Connected Component Analysis, Split and Merge, Labeling and Segmentation, etc.

### 3.4 Find Text Area

To read the text in the container, it is necessary to preprocess the text clarity, boundaries, and size of the aging equipment so that text detection can proceed efficiently. The preprocessing process can be divided into image preprocessing and OCR preprocessing. As shown in Fig. 7, the container has a lot of text information as well as the BIC Code. To determine the BIC Code among them, the quality of the input data is important and needs to be preprocessed.

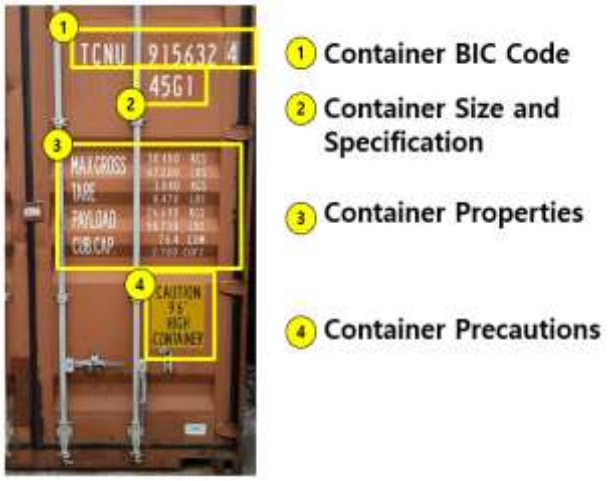


Fig. 7: Container Text Area

Image preprocessing is the process of preprocessing the images that are input to the Object Detection model. This can include image resizing, rotating and flipping, denoising, and color equalization. Image resizing and rotation first resize the image to fit the input size of the model, [25]. This is essential because the input size of the model must be constant. Image rotation and inversion increase the diversity of the image and allow the model to detect objects from different angles. Noise is then removed so that the model can identify the exact location of the object, and color equalization of the image is performed to make the object more visible.

### 3.5 Open-Source OCR

For open-source OCR, we chose EasyOCR and TesseractOCR. EasyOCR is based on deep learning technology, supports multilingual OCR recognition, and has a simple API and high recognition accuracy. It is also a robust OCR engine that works well with slightly noisy images.

TesseractOCR is an open-source OCR engine developed by Google that uses statistical-based techniques along with deep learning techniques like LSTM. Tesseract offers high accuracy and reliability and is known to have high recognition rates for several languages, including English and other European languages. EasyOCR and TesseractOCR were selected as representative models for the performance comparison because they have good performance in terms of recognition accuracy, multilingual recognition support, and options.

# 4 Experiment and Results

## 4.1 Experimental Environments

System and workload parameters are important for evaluating and optimizing the performance of an algorithm. These parameters include system and hardware configuration, image resolution, image format, processing speed, etc. The hardware and software environments were tested on Google Corel, with NVIDIA T4 as the graphics card and Python 3.8, CUDA 11.8, and torch2.0.0 as the development language. For the learning model, the open source-based OCR engine used the library in its pre-trained form without modification, the object detection algorithm used a pre-trained model with COCO data, and CRAFT used craft_mlt_25k.pth. Container images and label data were prepared separately for training, verification, and testing, and the image resolution, format, and label data structure were modified according to each pre-trained model.

## 4.2 Data Set

To recognize container BIC Code using Faster R_CNN or YOLOv5, we need a dataset to train on. Table. 1 shows the training and testing targets, organized after ingestion.

Table 1. Training Target

| Train | Test | Image | Label | Source |
|---|---|---|---|---|
| 100 | 20 | JPG | JSON | AI-Hub |
| 3,000 | 600 | JPG | JSON | AI-Hub |
| 6,000 | 1,200 | JPG | JSON | AI-Hub |

We used A.I. Hub's 2021 "Logistics Infrastructure Data for Connected Ports" dataset. This dataset is the result of securing source data to provide BIC Code, a method of identifying actual containers, and yard trailers, a means of transportation, and basic data to build realistic and business-applicable port logistics data, which is the basis for the development of smart port construction. The data source was collected from quay cranes (2 Terminals) and deck gates (3 Terminals, 12 Lanes). Images and labels are in pairs. Fig. 8 shows the JSON structure of the label data, which provides the text of the BIC Code and the coordinates of the area to help learn.


Fig. 8: Label Data Structure

## 4.3 Results

### 4.3.1 Faster R-CNN

We evaluated the performance (mAP, Precision, Recall) by applying Faster R-CNN, the most famous 2-Stage Detector model, and experimented with various hyperparameters to find the optimal results. The Learning_Rate, which is used to update the weights during training, was set to 0.005. The Optimizer was set to SGD, Momentum to 0.9, and Weight_Decay to 0.0005. Batch_Size is the number of data used for training at a time, and since we were running with Colab-pro, we set 16 and 32 to account for GPU memory, but there was no significant difference in training speed between the two settings. Finally, Num_Epochs, which determines the number of training times, was set to 40 depending on the amount of data and the possibility of overfitting. The results of the experiment showed that the values of mAP, Precision, and Recall increased as the amount of training data increased, indicating that it became more accurate as it learned more data. However, Recall has a lower detection rate for Check Digit than mAP, so We had tried to train with more training data and changed hyperparameters. Table. 2 shows that the performance of the Faster R-CNN model improves slightly as the number of training data increases.

Table 2. Check Digit Detection Experiment Results by Increasing Faster R_CNN Training Data

| Train | Test | mAP | Precision | Recall |
|---|---|---|---|---|
| 100 | 20 | 0.34 | 0.56 | 0.12 |
| 3,000 | 600 | 0.41 | 0.64 | 0.17 |
| 6,000 | 1,200 | 0.46 | 0.68 | 0.24 |

Table 3. Check Digit Detection Experiment Results by Increasing YOLOv5 Training Data

| Train | Test | mAP | Precision | Recall |
|---|---|---|---|---|
| 100 | 20 | 0.30 | 0.50 | 0.10 |
| 3,000 | 600 | 0.37 | 0.58 | 0.16 |
| 6,000 | 1,200 | 0.43 | 0.65 | 0.20 |

### 4.3.2 YOLOv5

To optimize the 1-Stage Detector model, YOLO, we used hyper-parameter optimization techniques, Grid Search and Bayesian Optimization, to derive the optimal hyper-parameters. Grid search involves trying all possible combinations of hyper-parameter values to find the optimal combination, while Bayesian optimization is an automatic optimization technique. We compared the performance of the model by applying two methods, Pre-defined Anchor Box and Anchor Box free. Pre-defined Anchor Box method detects objects using predefined anchor boxes, and Anchor Box free method detects objects without using anchor boxes.

To proceed with YOLOv5, image files and YOLO Text files containing label information must be configured as a Dataset. The set of image files and label files are divided into Train, Validation, and Test, and the label file consists of the class index and the coordinate information of the bounding box. The coordinate information is expressed as the center (x, y) coordinate and the width and height of the object, and the value is expressed as a value relative to the width and height of the image. Typically, coordinate values are normalized to a range between 0 and 1. The closer the object's Width and Height values are to 0, the smaller the image, and closer to 1, the larger the image, so that the same label format can be maintained regardless of the image size.

The YAML file is the configuration file used by YOLOv5, which sets the training datapath, validation datapath, batch size, and number of epochs according to the predefined format, and finally, the performance evaluation metric for YOLOv5 uses the same mAP for comparison with Faster R_CNN. Table. 3 shows that the performance of the YOLOv5 model improves slightly as the number of training data increases.

### 4.3.3 CRAFT

Each model recognizes text in a given image, finds the bounding box of the text, and outputs the coordinates in a result file, based on the data characteristics for training the model. Ideally, the four parameters should produce a four-sided rectangle with vertices, but the results can predict irregular shapes such as trapezoids or polygons.

Fig. 9 shows the output after running CRAFT, which returns regions with a polygonal structure, as opposed to the label data for training. Because the dimensions of the training and output data were different, post-processing was performed to match the same dimensions.

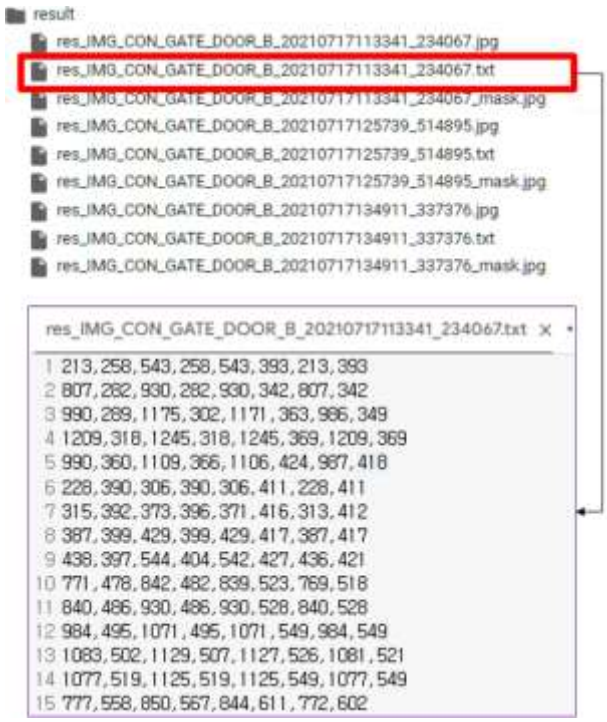

Fig. 9: CRAFT Results Data Structure

Many Object Detect algorithms are based on the following criteria: WA(Word Accuracy): The percentage of matches between the extracted text and the actual text, CA(Character Accuracy): The percentage of matches between extracted and real characters, and Precision, Recall, and F1-score: Precision, Recall, and F1-score values between extracted and real text as performance metrics. However, in the case of CRAFT, the accuracy measurement method for text recognition is

different from the commonly used method. In general, the IoU value between the ground truth and the predicted value is 0.5 or more, and the Precision, Recall, F1-score, etc. are calculated based on that, but unlike other algorithms, CRAFT aims to generate multiple candidate regions and miss as few real text regions as possible, rather than producing highly accurate predictions. In training and testing, CRAFT outperformed comparable OCR engines and object detection models and achieved a success rate of 74.5% in detecting check digits in the test data. Table. 4 shows that the CRAFT model performs well from the first training, and like other algorithms, the performance improves as the number of training data increases.

Table. 4. Check Digit Detection Experiment Results by Increasing CRAFT Training Data

| Train | Test | mAP | Precision | Recall |
|---|---|---|---|---|
| 100 | 20 | 0.66 | 0.70 | 0.62 |
| 3,000 | 600 | 0.69 | 0.74 | 0.63 |
| 6,000 | 1,200 | 0.75 | 0.78 | 0.71 |

### 4.4 Text Recognition

Since EasyOCR and TesseractOCR already include an Object Detection algorithm, we tested them without customization to handle detection and recognition as a 1-Stage method. As a result, the OCR engine had a good text detection rate for the total number of characters in the image, which was sufficient to recognize text after detecting the BIC Code. However, the function of recognizing BIC Code as a single string and recommending them as text by combining them, and the detection of check digits were more than 95% in error, so the evaluation method through index evaluation was meaningless.

## 5 Conclusions

In this paper, we compared and evaluated OCR engines, object detection algorithms, and text detection models to identify the location of container BIC Code objects, and then passed the location of the objects to OCR to extract and classify the text at the location to recognize container BIC Code. In general, container BIC codes are recorded manually in logistics, but this process is cumbersome and needs to be improved due to the possibility of human error. Therefore, we expect that if this result is developed and optimized by applying it to the field, it will lay the foundation for the automatic recognition and processing of BIC Code. As a result, high accuracy and fast processing speed can be expected compared to manual recognition, and it can be used not only in logistics but also in various industries to increase user satisfaction.

It is also an achievement that we have confirmed that there is a need for improvement in accuracy and consistency while conducting this experiment. In terms of accuracy, the detection rate for the entire text is satisfactory because the open source-based OCR does not have a fixed and partial purpose of recognizing container numbers. However, in the case of the object detection algorithm, a stronger preprocessing and postprocessing could have resulted in higher performance, but the lack of training data and various hyperparameter settings are disappointing. The results from all the tested models were inconsistent even on the same data. This can probably be overcome with technical improvements.

*Acknowledgment:*
This research was supported by the SungKyunKwan University and the BK21 FOUR (Graduate School Innovation) funded by the Ministry of Education (MOE, Korea) and the National Research Foundation of Korea(NRF). And, this work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A1060054). Corresponding authors: Professor Jongpil Jeong and Professor Chegyu Lee.

*References:*

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

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

[3] D. G Lee, CNN-based Image Rotation Correction Algorithm to Improve Image Recognition Rate, *The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)* Vol.20, No.1, 2022, pp.225-229.

[4] S. Ren, K. He, R.B. Girshick, and J. Sun, Faster R-CNN: towards real-time object

detection with region proposal networks. *CoRR*, 2015, abs.1506.01497.

[5] J. Redmon, A. Farhadi, YOLOv3: An Incremental Improvement, *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp.1-8.

[6] J. Song, N. Jung, H. Kang, Container BIC-code region extraction and recognition method using multiple thresholding, *Journal of the Korea Institute of Information and Communication Engineering*, Vol.19, No.6, 2015.

[7] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, K. Murphy, Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7310-731.

[8] T. Szabo, G. Horvath, Finite word length computational effects of the principal component analysis networks, *IEEE Trans. Instrum.Meas*, Vol.47, No.5, Oct. 1998, pp. 1218–1222.

[9] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, 580–587.

[10] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Member, IEEE, Yuhong Guo, and Jieping Ye, Fellow, Object Detection in 20 Years: A Survey, *IEEE Proceedings of the IEEE*, March 2023 Volume: 111, Issue: 3.

[11] Y. Chao, S. Vijayanarasimhan, B. Seybold, David A. Ross, J. Deng, R. Sukthankar, Rethinking the Faster R-CNN Architecture for Temporal Action Localization, *arXiv: 1804.07667v1 [cs.CV] 20*, Apr 2018.

[12] A. Bochkovskiy, C. Wang, A. Mykhailych, YOLOv5: Improved Real-Time Object Detection, *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 2290-2298.

[13] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, J. Liang, EAST: An efficient and accurate scene text detector, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5551-5560.

[14] N. Subramani, A. Matton, M. Greaves, A. Lam, A Survey of Deep Learning Approaches for OCR and Document Understanding, *arXiv:2011.13534*, 2021.

[15] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, *arXiv:1409.1556*, Apr. 2015.

[16] V. E. Bugayong, J. Flores Villaverde, N. B. Linsangan, Google Tesseract: Optical Character Recognition (OCR) on HDD / SSD Labels Using Machine Vision, *14th International Conference on Computer and Automation Engineering (ICCAE)*, 2022, pp. 56-60.

[17] J. Singh, B. Bhushan, Real Time Indian License Plate Detection using Deep Neural Networks and Optical Character Recognition using LSTM Tesseract, *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2019, pp.347-352.

[18] B. Shi, X. Bai, C. Yao, An end-to-end trainable neural network for image-based sequence recognition. Image-based sequence recognition and its application to scene text recognition, *IEEE Trans. Pattern Anal. Mach. Intell*, 2017, pp. 2298-2304.

[19] L. Mei, J. Guo, Q. Liu, and P. Lu, A new framework for containers. A new framework for code-to-character recognition based on deep learning and template matching, *Conf. Ind. Informat.-Comput. Technol, Ind. Inf. (ICIICII)*, Wuhan, Hubei, China, Dec. 2016, pp. 78-82.

[20] Y. Lee, P. Moon, A Comparison and Analysis of Deep Learning Framework, *Journal of the KIECS*, Vol.12, 2017, pp.115-122.

[21] Y. Baek, B. Lee, D. Han, S. Yun, H Lee, Clova AI Research, NAVER Corp. Character Region Awareness for Text Detection, *CVPR 2019 open access*

[22] C. Roeksukrungrueang, T. Kusonthammrat, N. Kunapronsujarit, T. N. Aruwong, and S. Chivapreecha, Automatic implementation of a container number recognition system, *Workshop Adv. Image Technol. (IWAIT)*, Chiang Mai, Thailand, Jan. 2018, pp.1-4.

[23] Y. Liu, T. Li, L. Jiang, X. Liang, Container-code recognition system. Container-code recognition system based on computer vision and deep neural network, *Int. Conf. Adv. Mater, Mach, Electron*, Xi'an, China, Jan. 2018, 20-21.

[24] U. Mittal, P. Chawla, R. Tiwari, EnsembleNet: a hybrid approach for vehicle detection and estimation of traffic density based on faster R-CNN and YOLO models,

*Neural Computing and Applications (2023)* 35:4755-4774

[25] K. Zhang, H. Wang, X. Li, An ensemble of YOLOv5 and Faster R-CNN for container BIC Code detection and recognition, *Robotics and Automation Sciences (ICRAS)*, pp.1-6.

**Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)**

Hangseo Choi Hangseo designed the research topic and goals, implemented, and experimented with CRAFT, and wrote the paper.
Prof. Jongpil Jeong conceptualized the idea, presented the methodology, and reviewed it.
Prof. Chaegyu Lee provided academic advice.
Seokwoo Yoon implemented and experimented with Faster R-CNN.
KyungAh Bang implemented and experimented with YOLOv5.
Jaebeom Yoon implemented and experimented with OCR.

**Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself**

This research was supported by the SungKyunKwan University and the BK21 FOUR(Graduate School Innovation) funded by the Ministry of Education(MOE, Korea) and the National Research Foundation of Korea(NRF). And this work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2021R1F1A1060054). Corresponding authors: Professor Jongpil Jeong.

**Conflict of Interest**

The authors have no conflict of interest to declare.