
# University of Science and Technology of Ha Noi

**Information and Communication Technology Department**


---

# COMPUTER VISION

## FINAL REPORT: VIETNAMESE LICENSE PLATE DETECTION

---

**Major: Data Science**


| Student Name | Student ID |
|---|---|
| 1. Nguyen Thi Van | 22BI13459 |



**Submission Date : 10/05/2024**

# Contents

# 1 Introduction

Automatic license plate detection has become a significant area of research and application in the fields of computer vision and intelligent transportation systems. As urban areas in Vietnam continue to grow and traffic density increases, there is a rising need for efficient, automated solutions to monitor and manage vehicles. In this part we introduce about our motivation and goal to conduct this project.

## 1.1 Motivation

Traditional manual methods of vehicle identification are time-consuming and prone to error, making automation a crucial step forward. In this context, license plate detection using image processing and AI technologies has emerged as a practical and scalable solution.

## 1.2 Goal

With the development of AI and Machine Learning algorithm, we aim to build a system, in which:

- Detecting the license plate with high accuracy and rapidly.
- Extracting the license plate information after capturing the image.

Below is the illustration of our input and output.


Input Output

Figure 1: Input and Output demo

# 2 Methodology

## 2.1 Workflow

To implement the project, we divided our workflow into 2 main phases - Object Detection and OCR. The workflow is illustrated below.

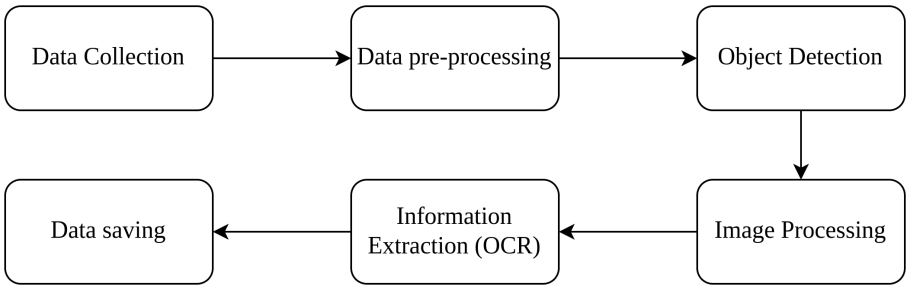



Figure 2: Project Workflow

## 2.2 Data Collection

We collecting manually 550 images contains license plate. All the image must satisfy the following conditions:

- There is only 1 license plate in an image with enough brightness for detection.
- No duplication of image is allowed.
- The angle and position of image is different between images.

Below is an example of our dataset and a comparison between qualified image and unqualified image.


Figure 3: Image example

## 2.3 Data pre-processing

After having a data set, we need to do some pre-processing before training the model. Below is the pipeline for this data pre-processing part.


Figure 4: Data pre-preprocessing pipeline

To satisfy the requirement from YoloV8, which is the main model used for object detection, we modify the data as below:

- **Resizing image**: we convert all the images into 640x640 dimensions and save in .jpg format to uniform the image and avoid bias or error while training causes by image size.YoloV8 require a dataset which is labeled by using bounding box to mark around the object and a corresponding label file in .txt format which contains the class's name and the information of the bounding box (center coordinates, width, height).
  To labeling the image, we using a computer vision tool recommended by YOLO authors: Roboflow Annotate for labeling and exporting the corresponding label file. Below is the illustration of labelling process and the label of the image.

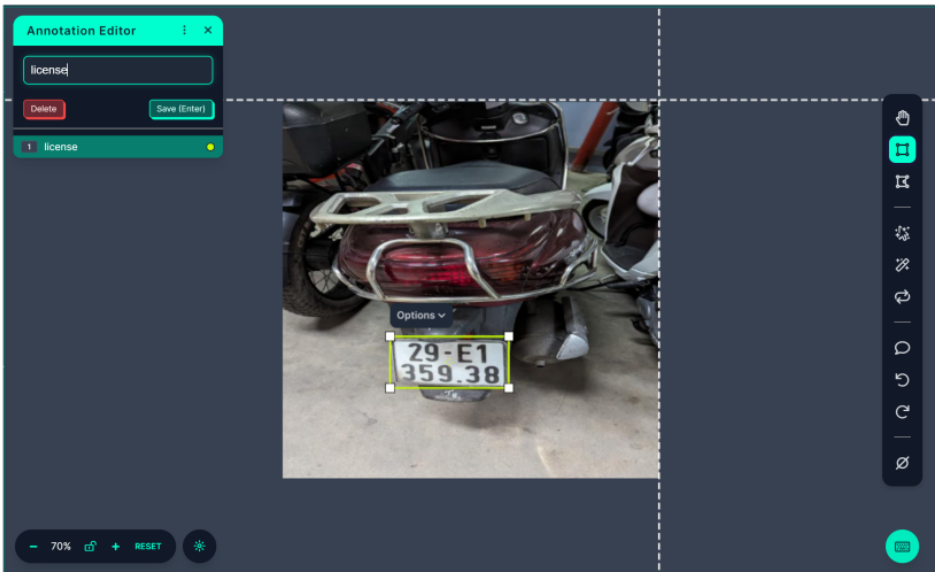



Figure 5: Labeling image using Roboflow

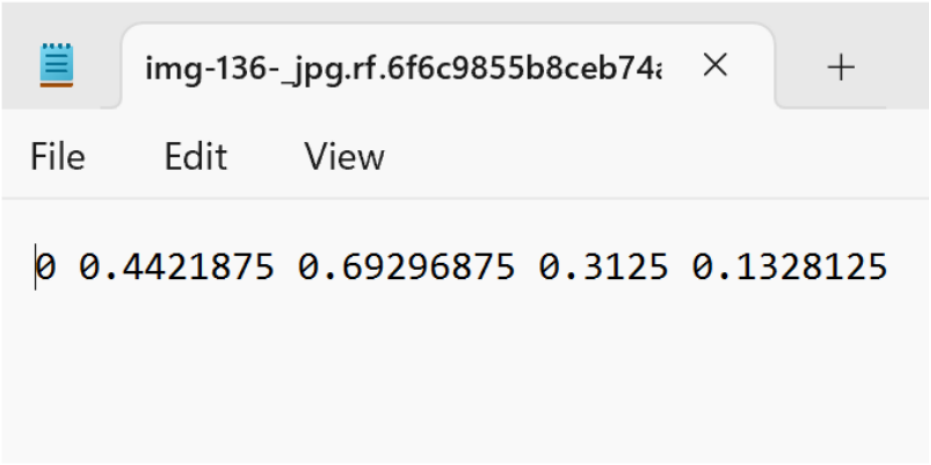



Figure 6: Label file format for the image

- **Constructing the dataset**: after labeling the data, we divided our dataset into 3 parts: train set, valid set and test set with the proportion 80%-10%-10% for each set. The data must be organized as the required structure in order to training with YoloV8, including a .yaml for storing paths to folders.

## 2.4 Object Detection

After pre-processing the data, we started with the object detection part. The object we need to detect is the license plate from an image. Below is the pipeline for training an object detection model.

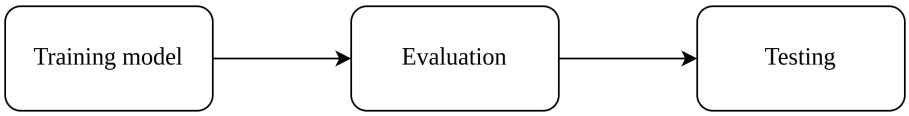



Figure 7: Object Detection Pipeline

1. **Training model**: We training the detection model using **YOLOv8** and the pre-trained parameters from **yolov8n.pt** file.
2. **Evaluating model**: Then we evaluate the model by using **val** method and trained parameters.
3. **Testing model**: Using the **predict** method from the model to testing with the test set.

### 2.4.1 YOLOv8

YOLOv8 is a model belonging to (You Only Look Once) family of object detection models, designed to deliver state-of-the-art accuracy while maintaining real-time speed and efficiency. Its architecture and training techniques incorporate several key innovations that improve upon previous versions, making it highly suitable for tasks like license plate detection.

YOLOv8 Model Architecture and Key Features

1. **Modular Structure**: Backbone, Neck, and Head YOLOv8's architecture is composed of three main components, each with distinct roles:
    - **Backbone:** The backbone acts as a feature extractor that processes the input image to capture hierarchical visual information. YOLOv8 uses a modified CSP-Darknet53 backbone, which integrates Cross Stage Partial (CSP) connections to enhance gradient flow and reduce computational cost. This backbone extracts rich features at multiple levels, from simple edges and textures in early layers to complex shapes in deeper layers.
    - **Neck:** The neck bridges the backbone and the detection head. It performs multi-scale feature fusion using a Path Aggregation Network (PANet) combined with a novel C2f (Cross Stage Partial with concatenation of all bottleneck outputs) module. This design effectively merges semantic information from different scales, improving detection of objects of varying sizes, including small or rotated license plates.
    - **Head:** The head generates final predictions, including bounding box coordinates, objectness confidence scores, and class probabilities. YOLOv8's head uses multiple detection modules that operate on different feature map scales, allowing precise localization and classification of objects in a single forward pass.

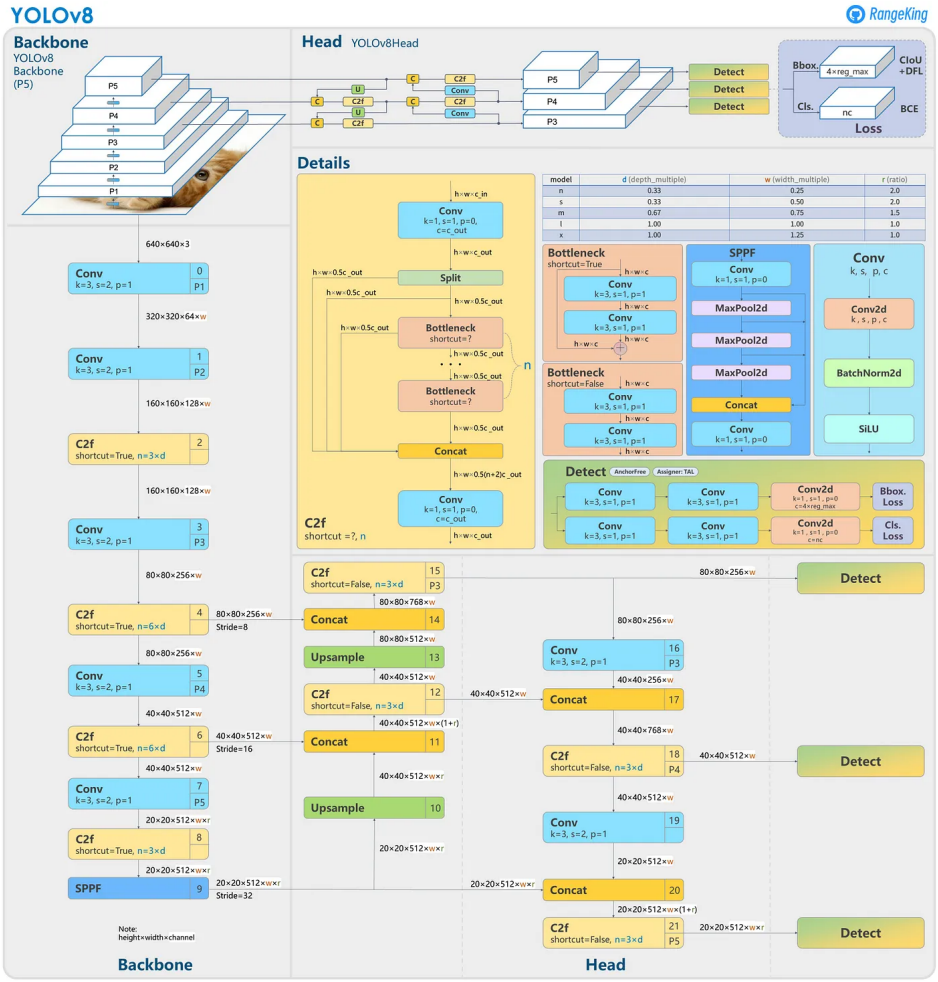



Figure 8: YOLOv8 architecture

2. **Anchor-Free Detection**: Unlike earlier YOLO versions that rely on predefined anchor boxes, YOLOv8 adopts an anchor-free detection approach. Instead of predicting bounding box offsets relative to fixed anchors, the model directly predicts the center points of objects and their dimensions. This approach:

    - Simplifies the training process by removing the need to tune anchor sizes for specific datasets.
    - Improves generalization on custom datasets like Vietnamese license plates.
    - Speeds up Non-Maximum Suppression (NMS) by reducing the number of bounding box candidates.

3. **Mosaic Data Augmentation with Scheduled Stopping**: YOLOv8 employs mosaic data augmentation, which combines four different images into one during training. This technique enriches the training data by providing varied context and object scales, enhancing the model's robustness. A key improvement in YOLOv8 is stopping mosaic augmentation in the last ten epochs of training to fine-tune detection performance and avoid overfitting.

4. **C2f Module in Backbone and Neck**: The C2f module replaces the earlier C3 module used in previous YOLO versions. It concatenates outputs from all bottleneck residual blocks rather than only the last one, which:

    - Enhances gradient flow throughout the network.

- Speeds up training convergence.
- Improves feature reuse and representation, leading to better detection accuracy

5. **Model Scaling and Variants**: YOLOv8 offers multiple model sizes-from lightweight YOLOv8n (nano) to larger YOLOv8x (extra-large)-allowing users to balance between inference speed and accuracy depending on hardware constraints and application needs.
6. **Training Enhancements**: YOLOv8 incorporates advanced training strategies, including:
   - Use of Rectified Adam (RAdam) optimizer for faster and more stable convergence.
   - A modified loss function combining box regression loss, classification loss, and distribution focal loss (DFL) to improve bounding box precision and class separation.
   - Scheduled learning rate adjustments and early stopping mechanisms.

## 2.5 Image Processing

Pre-processing and enhancing license plate images before OCR is critical, especially in outdoor, real-world conditions with noise, motion blur, or varying lighting. The following techniques were applied for this project:

- **Grayscale conversion**: Reduces computational complexity and focuses processing on luminance, which is more important for text-based tasks.
- **Histogram Equalization**: Enhances image contrast by redistributing intensity values, improving text visibility.
- **Gaussian Blur**: A smoothing filter that reduces image noise and detail; it's especially useful before edge detection or thresholding.
- **Non-local Means Denoising**: An advanced denoising algorithm that preserves important textures and edges by considering the similarity between image patches, not just local neighborhoods.

## 2.6 Information Extraction (OCR)

OCR is the process of converting text from images into machine-readable characters. The goal in this context is to extract Vietnamese license plate numbers accurately from cropped images.

### 2.6.1 OCR Challenges in License Plates

Vietnamese license plates pose several challenges:

- Multiple formats: Varying patterns across vehicles (e.g., two lines vs. one line).
- Character diversity: Combinations of digits and uppercase letters (excluding confusing characters like I, O, J, Q).
- Environmental noise: Dirt, reflections, lighting, or physical damage can obscure characters.

### 2.6.2 EasyOCR and PyTesseract

Two open-source OCR engines were used:

1. **EasyOCR**: A deep learning-based OCR system using convolutional neural networks (CNNs) and attention mechanisms. It supports over 80 languages and provides both bounding boxes and recognized text. Its architecture typically includes:
   - A CNN-based feature extractor.
   - A sequence model (e.g., BiLSTM) to capture character dependencies.
   - An attention-based decoder for sequence prediction.
2. **PyTesseract**: A Python wrapper for Google's Tesseract OCR engine. Tesseract uses a two-pass approach:
   - Line Finding and Layout Analysis: Identifies blocks, paragraphs, and lines of text.
   - Character Recognition: Applies a character classifier, often using LSTMs in modern versions.

## 2.7 Evaluation

### 2.7.1 Object Detection

For evaluating the performance of the YOLOv8 object detection model on license plate localization, we use standard object detection metrics based on the quality of predicted bounding boxes compared to ground truth annotations.

1. **Precision**: The proportion of predicted bounding boxes that are correct.

$$Precision = \frac{TP}{TP+FP}$$

2. **Recall**: The proportion of actual bounding boxes that are correctly predicted.

$$Recall = \frac{TP}{TP+FN}$$

3. **F1-score**: The proportion of actual bounding boxes that are correctly predicted.

$$F1 = 2.\frac{Precision.Recall}{Precision+Recall}$$

4. **Mean Average Precision (mAP)**: a standard metric used in object detection benchmarks. It is calculated as the mean of Average Precision (AP) values across all classes and IoU thresholds. For a single class (license plates), it is computed as.

$$mAP = \frac{1}{n}\sum_{k=1}^{n} AP_k$$

where $AP_k$: the average precision of class k and $n$ is the number of classes. We report $mAP$ at 0.5 and 0.5-0.95 threshold.

### 2.7.2 Information Extraction (OCR)

1. **Character Error Rate (CER)**: measures the rate in which an OCR model incorrectly recognize a characters from a text.

$$CER = \frac{\text{Number of incorrect characters}}{\text{Total number of characters in the reference text}} \times 100$$

2. **Accuracy**: measures the rate of an OCR model to recognize correctly the characters from a text.

$$Accuracy = 1 - CER$$

# 3 Experimental results

## 3.1 Object Detection

After around 2 hours training the data with 100 epochs, we received a best.pt file which contains trained model parameters.

### 3.1.1 Evaluation

The overall loss of model is decrease over time and reach the minimum at the last epochs.

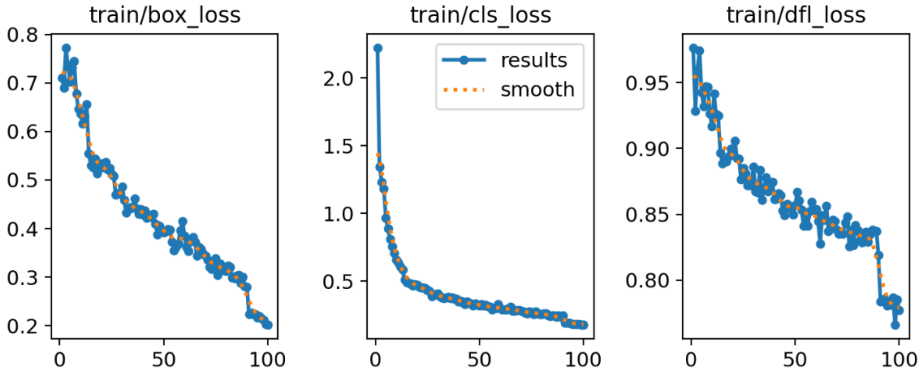



Figure 9: Overall training loss

- Box loss: model ability to minimizing location error between bounding boxes.
- Class loss: model ability to classifying the object right.
- Dlf loss: model ability to differentiate between similar objects or hard sample.

We evaluated the model using the trained parameters for the dataset and received the following result.

| Evaluation metric | Result |
|---|---|
| Precision | 0.99 |
| Recall | 1 |
| mAP50 | 0.995 |
| mAP50-95 | 0.978 |

### 3.1.2 Testing

We testing the model with an unseen test dataset. The performance of model is illustrated below. In overall, the model performance on the test set is similar to the result of valid set.

| Evaluation metric | Result |
|---|---|
| Precision | 0.99 |
| Recall | 1 |
| mAP50 | 0.995 |
| mAP50-95 | 0.983 |

Below is a test result image after using the trained YOLOv8 model to detect. With a rotate angle, the model can detect correctly the license plate with ... confidence.
**Comment on the result:** The model can balance between precision and recall, with high value. The model also acheive high mAP result with different threshold, which means it can detect correctly under many conditions.

## 3.2 Image Processing

In our project, we used image processing for enhanced the image quality before doing OCR, since the image can be unclear after being crop out from the previous detection model.

- Improving contrast: by converting RGB image to Gray scale image to avoid the impact of RGB color in OCR and using histogram equalization techniques.

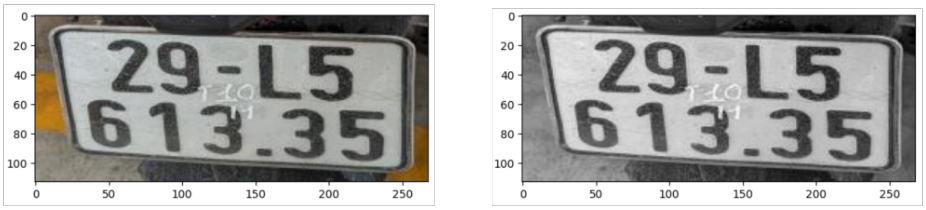



Figure 10: Converting to gray scale image

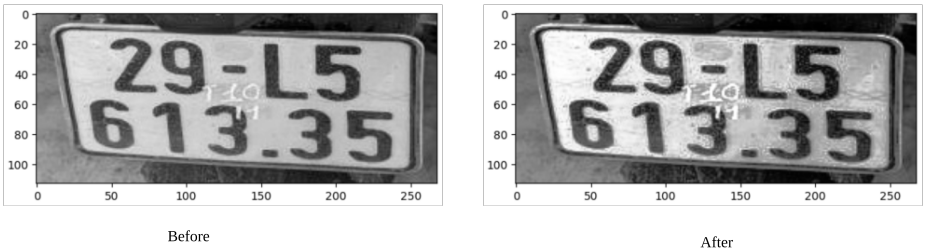



Figure 11: Histogram equalization image

- Removing Noise: by using Gaussian filter, we can smoothing the image and remove noise.

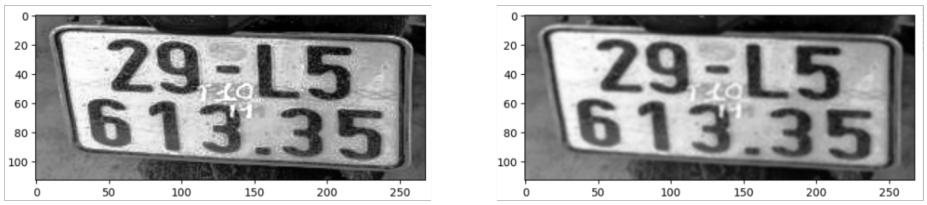



Figure 12: Remove noise

- Enhancing edges: we denoises the image using the Non-Local Means algorithm, this step will also remove noise while preserving edges of image.

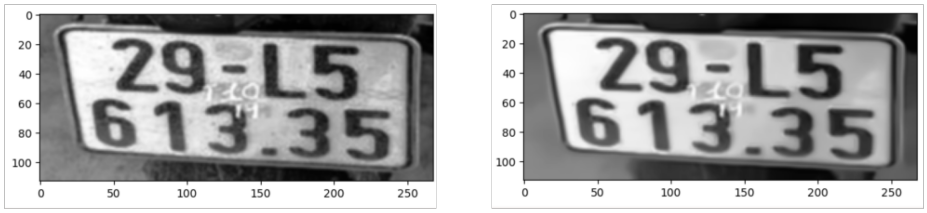



Figure 13: Edge enhancement

Below is an image before and after processing. We can see a different between these 2 images.


Figure 14: Image processing result

## 3.3 Information Extraction

Optical character recognition (OCR) is a process to extracting text from image, scanned documents, pdf files,etc. In our project, we consider OCR as recognizing text from a license plate. We first analyze the characteristic of a Vietnamese license plate, based on these features, we can improve the OCR process.

- Language: English, containing 20 letters of English alphabet (from A to Z except I,O,J,Q,R,W) and 10 numbers (from 0 to 9).
- Type: 1 line and 2 line license plate. The first type usually appears on car and the second type is used for many vehicles such as buses, cars, motorbikes,etc.
- Color: Can be black and white, blue and white or red and white, base on each type of vehicles.
- Other: On the license plate, sometimes there will be special mark or yellow blur due to long time usage. This can affect the OCR result.

### 3.3.1 EasyOCR

This is an OCR Python package for detecting and recognizing text, which can be used to detect English characters.

- Implementation: we using *readtext()* function from easyocr package in python to extract the plate information. After that, we store them into a .csv file.
- Testing: this is an image of license plate after running the model. The model will draw a bounding box around the text area and give the corresponding text of the box.

**Evaluation**: After testing with the test set, we have the following evaluation metrics. In overall, the model perform well on the dataset, it can recognize character from both type of licenses.

| **Metrics** | **Result** |
|---|---|
| Accuracy | 0.84 |
| Overall CER | 0.16 |

This is an example of using EasyOCR to extract license plate information. The result for this image is reported below:

| **Evaluation metric** | **Result** |
|---|---|
| CER | 0 |
| Accuracy | 1 |
| License information | 30A 098.83 |

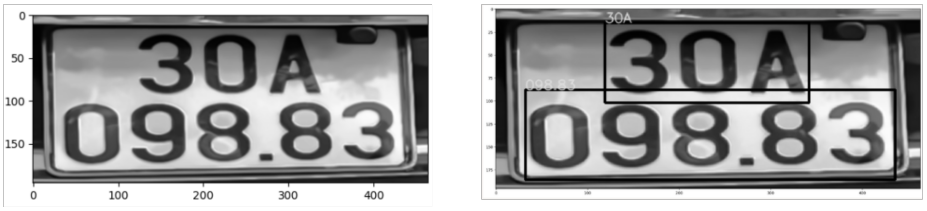



Figure 15: Example of EasyOCR

Another example of using EasyOCR, in this case the algorithm failed to detect the exact information of a license plate.

| **Evaluation metric** | **Result** |
|---|---|
| CER | 0.1 |
| Accuracy | 0.9 |
| License information | 30g-665.22 |

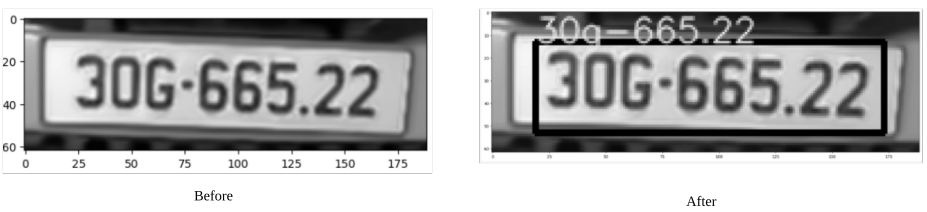



Figure 16: Example of EasyOCR

### 3.3.2 PyTesseract

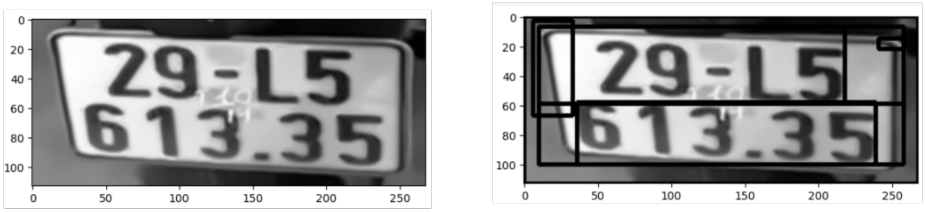
- Implementation: we using image to string function from pytesseract package in python to extract the plate information.
- Testing: this is an image of license plate after running the model. The model will also draw a bounding box around the text area and give the corresponding text of the box as EasyOCR

**Evaluation**: After testing with the test set, we have the following evaluation metrics. In overall, the model perform bad on the dataset, the accuracy is low and the CER is very high.

| **Metrics** | **Result** |
|---|---|
| Accuracy | 0.4 |
| Overall CER | 0.6 |

This is an example of using PyTesseract to extract license plate information. The result for this image is reported below. In overall, the algorithm failed to extract the license plate information correctly.

| **Metrics** | **Result** |
|---|---|
| CER | 0.42 |
| Accuracy | 0.58 |
| License information | — 29-15 — 613.35 |




Before After

Figure 17: Example of PyTesseract

The overall comparison of the two methods can be found below. In genral, EasyOCR tends to outperform the PyTesseract algorithm in both metrics and output a better result for license plate information extraction.

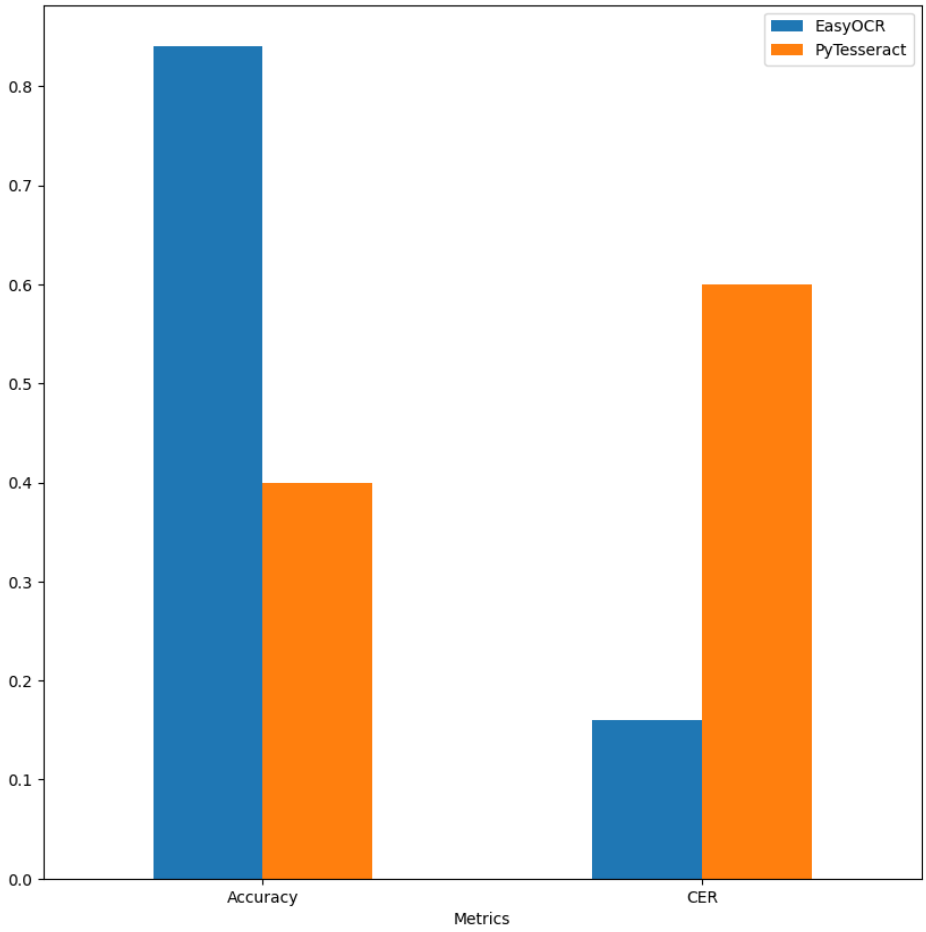



Figure 18: Overall Comparison