University of Science and Technology of Ha Noi

Information and Communication Technology Department

MACHINE LEARNING II

FINAL PROJECT REPORT

Topic: Vietnamese License Plate Detection

Major: Data Science

Student Name	Student ID
1. Chu Hoang Viet	22BI13462
2. Nguyen Thi Van	22BI13459
3. Dao Hoang Dung	22BI13101
4. Bui Dinh Lam	22BI13234
5. Nguyen Ngoc Nhi	22BI13351

Submission Date : 6/6/2024

Contents

1 Introduction **2**
1.1 Motivation 2
1.2 Goal 2

2 Project pipeline **2**

3 Experiments **2**
3.1 Data collection 2
3.2 Data pre-processing 3
3.3 Object Detection 4
3.3.1 Training model 5
3.3.2 Evaluating model 5
3.3.3 Testing model 5
3.4 Image Processing 6
3.5 Information Extraction 7
3.5.1 EasyOCR 8
3.5.2 PyTesseract 9
3.5.3 Evaluation 9

4 Improvement **10**

1 Introduction

1.1 Motivation

License plate detection serves multiples purpose in the daily life, including traffic management, public safety and in business operations. Due to its importance and multiple function, our team have built a system for Vietnamese license plate detection.

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.

2 Project pipeline

We have **2 main phases - Object Detection and OCR**. The workflow is illustrated below.

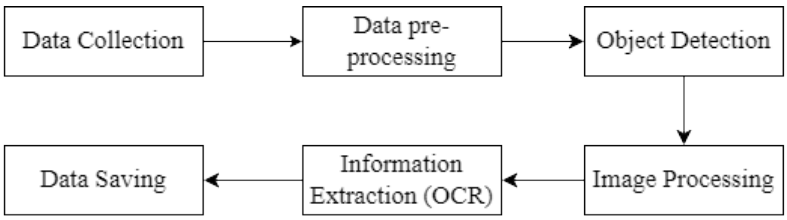

Figure 1: Project Workflow

3 Experiments

3.1 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 2: Image example

3.2 Data pre-processing

Below is the pipeline for this data pre-processing part.

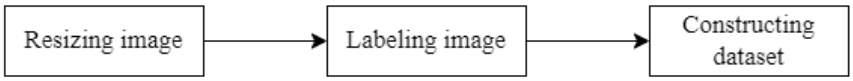

Figure 3: Data Pre-processing Pipeline

To prepare for the training part, we need to pre-processing the data. The model we used for the first part is **YoloV8**, so we need to modify the data to fit with the requirements of the model, in which:

- **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.
- **Labeling image**: 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.

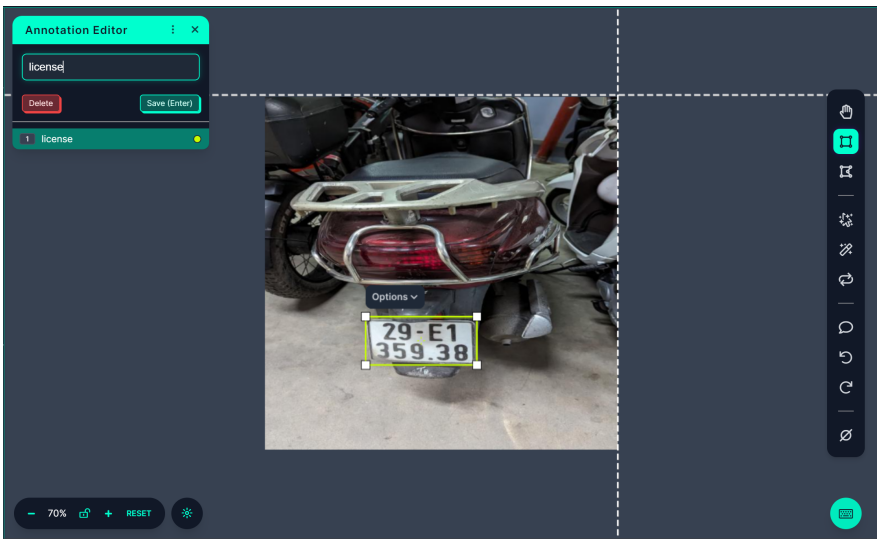

Figure 4: Labeling image using Roboflow

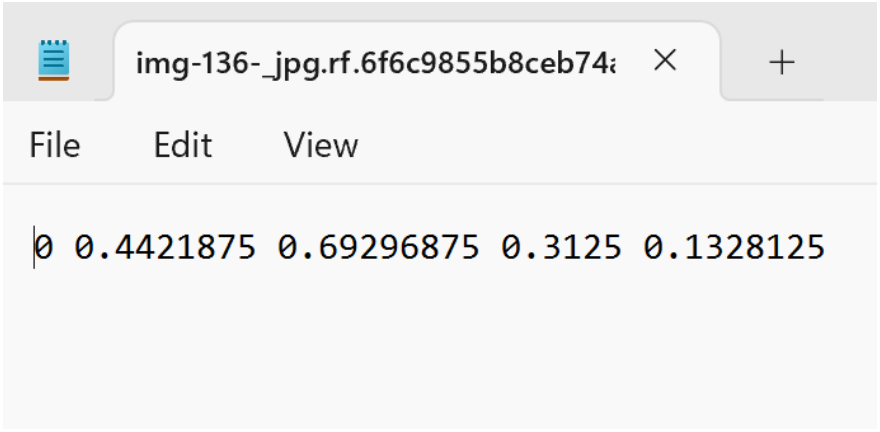

Figure 5: Label file format for the image

- **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.

3.3 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.
Pipeline:

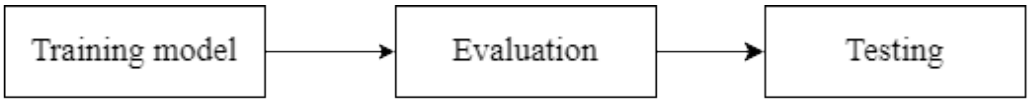

Figure 6: Object Detection Pipeline

- Step 1: Training model: We training the detection model using **YOLOv8** and the pre-trained parameters from **yolov8n.pt** file.

- Step 2: Evaluating model: Then we evaluate the model by using **val** method and trained parameters.
- Step 3: Testing model: Using the **predict** method from the model to testing with the test set.

3.3.1 Training model

After **6 hours** training the data with **100** epochs, we received a **best.pt** file which contains trained model parameters.
Evaluation: the overall loss of model is decrease over time and reach the minimum at the last epochs.

- 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.

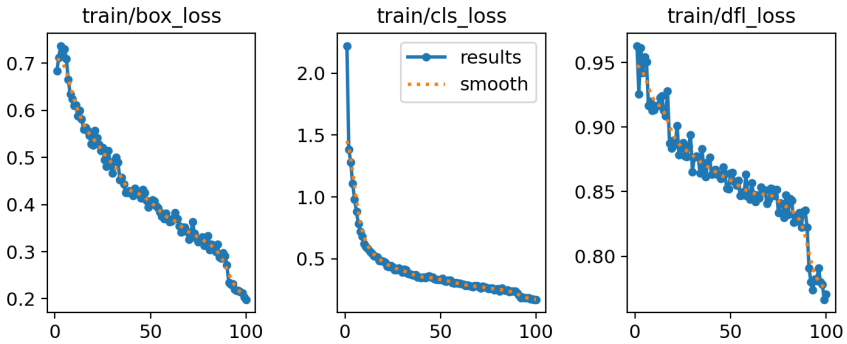

Figure 7: Overall training loss

3.3.2 Evaluating model

We evaluated the model using the trained parameters for the dataset and received the following result. These results will be an estimation for the test result.

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

Table 1: Model evaluation

3.3.3 Testing model

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

Table 2: Model evaluation

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 0.96 confidence.

Figure 8: Result after testing

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.

3.4 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.
- Removing Noise: by using Gaussian filter, we can smoothing the image and 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.

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

Figure 9: Image processing result

3.5 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.

We using 2 OCR model: EasyOCR and PyTesseract for this part. The pipeline for both 2 models is illustrated in below.

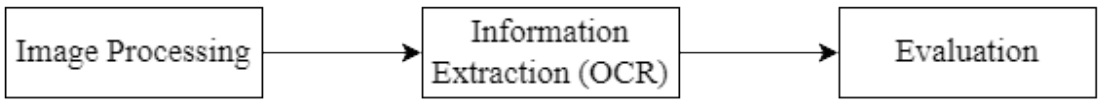

Figure 10: OCR pipeline

Evaluation: we using 2 metrics for evaluating an OCR model.

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

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

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

$$\text{Accuracy} = 1 - \text{CER}$$

3.5.1 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.

License plate number	30A 098.83
CER	0
Accuracy	1

Table 3: Test result and evaluation for example 1

Figure 11: EasyOCR result 1

We have another image result, which is a 1 line license plate.

License plate number	30g-665.22
CER	0.1
Accuracy	0.9

Table 4: Test result and evaluation for example 2

Figure 12: EasyOCR result 2

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
Overall CER	0.16
Accuracy	0.84

Table 5: Overall EasyOCR performance

3.5.2 PyTesseract

This is a Python library that provides an interface to the Tesseract optical character recognition (OCR) engine.

- **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

License plate number	$\|29 - 15\|613.35$
CER	0.42
Accuracy	0.58

Table 6: Test result and evaluation for example

Figure 13: Pytesseract result

The model detect the character poorly, since it is confused the edges with the character. To solve this problem, we propose some method in the improvement part.

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 too high.

Metrics	Result
Overall CER	0.6
Accuracy	0.4

Table 7: Overall PyTesseract performance

3.5.3 Evaluation

After testing 2 model with the test dataset, we have a comparison between 2 model performances. In overall, EasyOCR performance better on the dataset than PyTesseract on both accuracy and CER values.

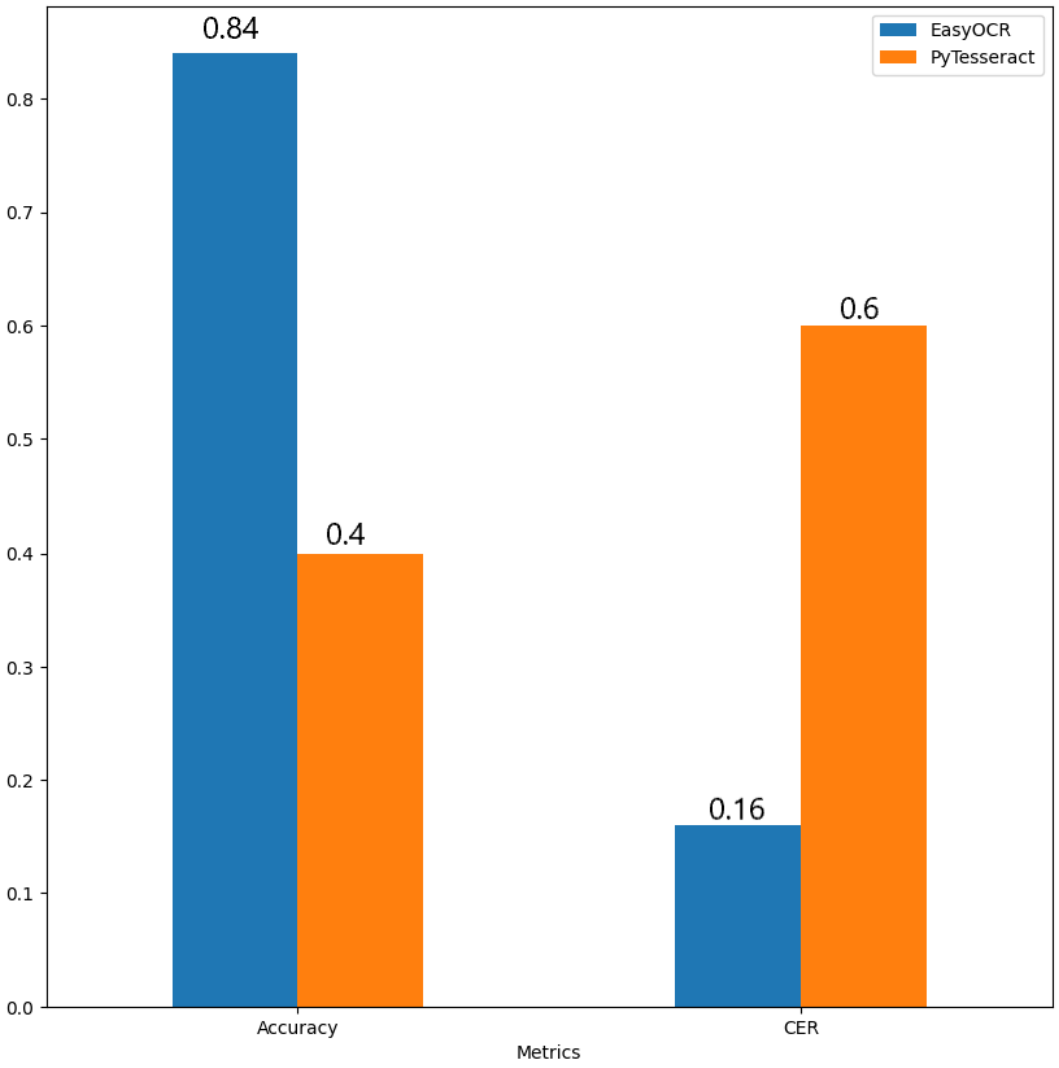

Figure 14: Enter Caption

4 Improvement

Due to the performance of the 2 OCR models on the dataset, we propose some methods for further improvement.

- EasyOCR: retrained the model with a customize dataset for Vietnamese license plate to increase the performance.
- PyTesseract: preprocess the image more effectively by remove the border of the license plate to avoid wrong detection from the model.
- Overall OCR: We can extract each character from the image and train them separately by using CNN model since this model is powerful in terms of character recognition.