Automatic number-plate recognition

Peter Han, Samuel Lee, Manyang Piyin

Introduction

Background

- ANPR adopts optical character recognition on images to read vehicle registration plates to create vehicle location data.
- Application ranges from allowing law enforcement to check if a vehicle is registered to enabling drivers to park conveniently in the parking lot with the ANPR system

Introduction

Why are we interested?

- ANPR is an area of study that is both relevant and practical for those interested in machine learning applications. It has numerous applications, making it a valuable field to explore. It’s also deeply intertwined with our daily routines.

Introduction

What we want to achieve?

- Enhancing ANPR accuracy and efficiency through algorithm development and modeling

Method

Step 1

Collect Data

- Collect a large and diverse dataset of annotated images that contain vehicle registration plates.

- Used to train and test the machine learning algorithm.

Step 2

Write Algorithm

- Write an algorithm that can detect and segment the number plate from an image.

- Involves technique such as YOLO and Object Character Recognition

Step 3

Training

- Train the program using the annotated dataset.

- Feed the images into the algorithm and adjusting the weights and parameters to optimize its accuracy.

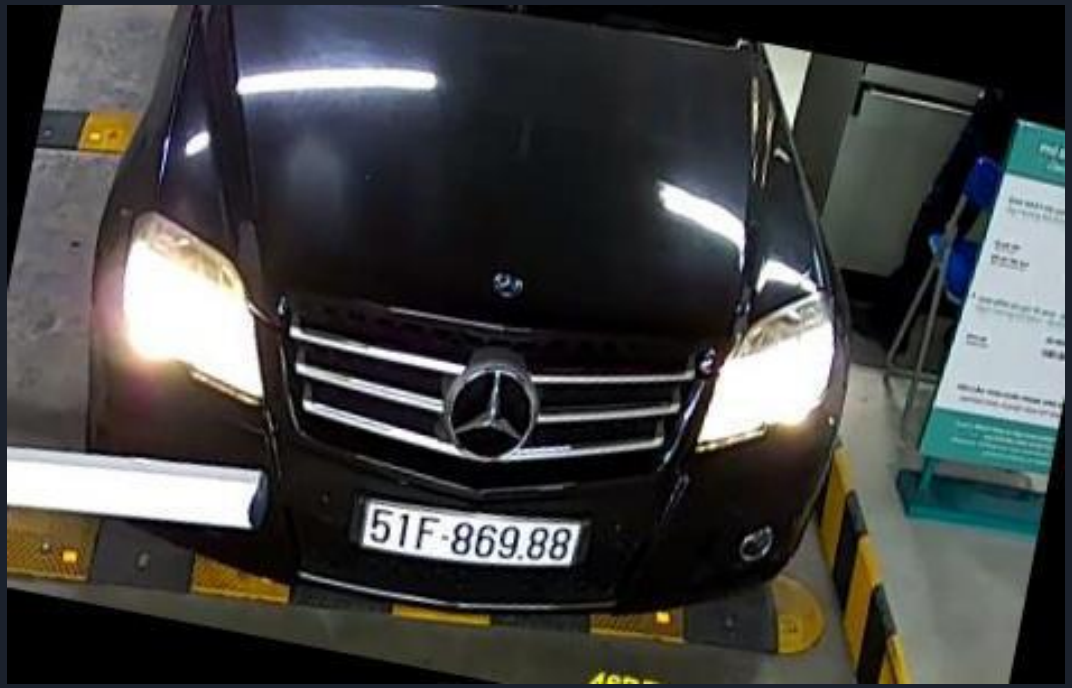
Step 4

Post-training

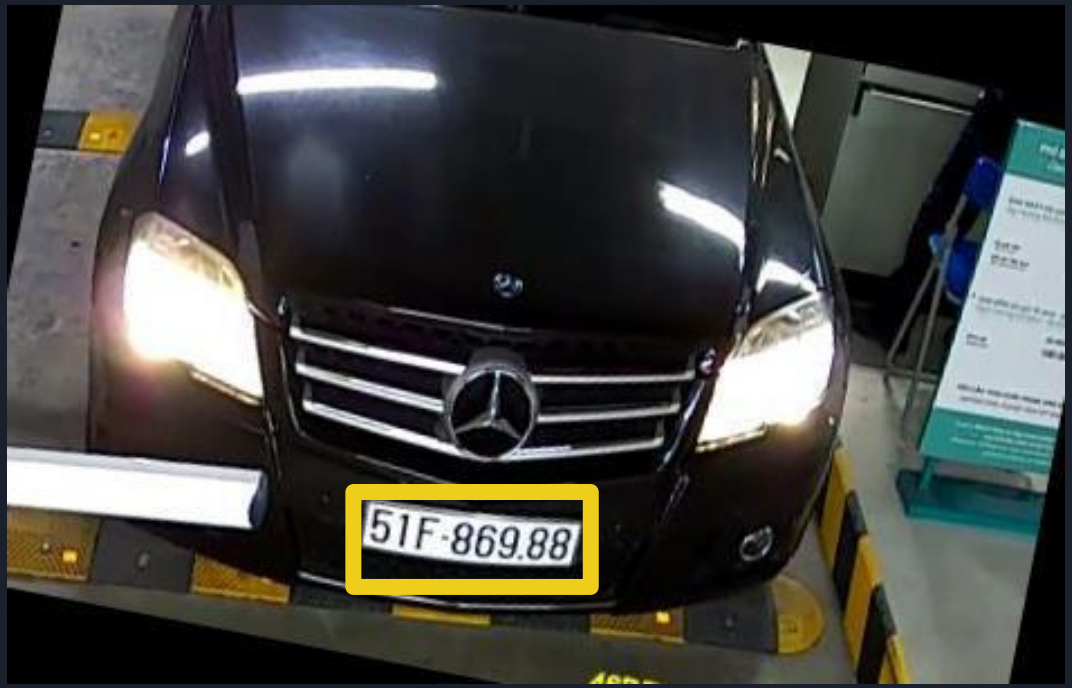
- After training, the algorithm is fine-tuned and optimized to improve its efficiency and accuracy.

- Test the algorithm on a validation dataset and adjusting the parameters to improve performance.

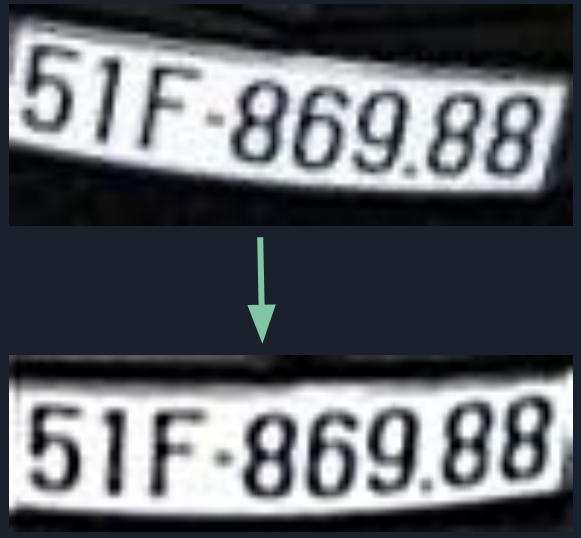
Algorithm



Algorithm



Algorithm



Algorithm

- The ANPR is broken down into 2 sections:
  1. License plate detection
  2. Reading the license plate for the numbers

License plate detection

- Our job: implement the code that will process each step
    1. License plate detection
        a. Haar Cascades
            i. Fast, low computational power requirements
            ii. Limited accuracy, struggles with varying conditions
        b. Faster R-CNN
            i. Accurate and excels in varying scenes
            ii. High computational requirements, relatively slow
        c. YOLO
            i. “Goldilocks” Model

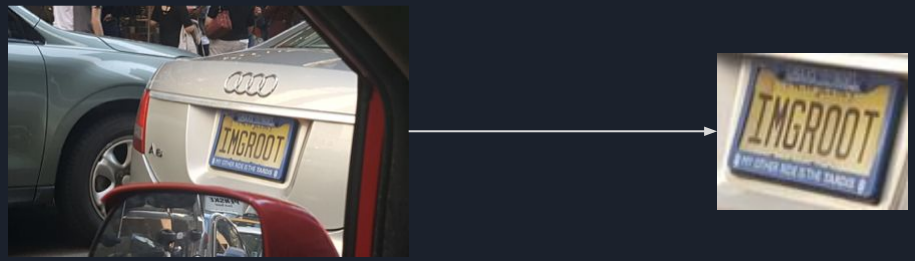
YOLO v5 and Implementation

- What is YOLO?
  - an object detection algorithm created by Ultralytics, which is a deep learning model used for detecting and recognizing objects within images and videos
- How do we use it?
  - Create a file to highlight the objects that we are looking to detect
    - In this case vehicles and license plates
    - Also include path to image files

```
1   train: ../CarPhotos/train/images
2   val: ../CarPhotos/valid/images
3
4   nc: 2
5   names: ['license-plate', 'vehicle']
```

YOLO v5 and Implementation

- How do we use it? (Continued)
  - Training
    - Utilize the included train.py file in the YOLOv5 model alongside the file created previously
  - Detecting
    - Utilize the included detect.py file in the YOLOv5 model, alongside the file created by the training part, as well as the input picture
- The result!`



Reading the license plate for the numbers

2. Image pre-processing
    a. OpenCV (Open Source Computer Vision)
3. Reading the plate number (Optical Character Recognition)
    a. EasyOCR

EasyOCR

- python based pyTorch library that falls upon good GPU to show accurate results
- What are components of easyOCR?

  - Features extraction

  - Sequence labeling

  - Decoding.

Review of the Architecture of Our Model

Collecting Dataset (YOLO)

- We looked for online plates dataset to train our YOLOv5 model
- The dataset consists of:
  1. Test set: 36 images
  2. Training set: 246 images
  3. Validation set: 71 images

Labeling Dataset (YOLO)

- The label consists of the following:
  1. Class: In our case “license-plate or vehicle”
  2. X: The x position of the object within the image
  3. Y: The y position of the object within the image
  4. Width: the width of the object
  5. Height: the height of the object

```
0 0.46634615384615385 0.5396634615384616 0.17427884615384615 0.09254807692307693
1 0.09735576923076923 0.3641826923076923 0.19471153846153846 0.45913461538461536
1 0.47836538461538464 0.4483173076923077 0.65625 0.7956730769230769
1 0.9122596153846154 0.39903846153846156 0.17427884615384615 0.4074519230769231
```

Collecting Dataset (Entire Model)

- We looked for online plates dataset for our project
- The dataset consists of 433 images in total

Labeling Dataset (Entire Model)

- We adopted Object Detection to assist us labeling the plates
- The label consists of the following:
  - License plate number

Results

How efficient was our Model?

- 95 % accuracy rate
- Reading a text using easyOCR took 3 seconds to return the results

```
#cropped_image = cropping(gray,location)
show_cropped(img)
```

✓ 0.1s Python


```
# using Easy OCR to read text
result = readText(img)
print(result)
```

✓ 2.2s Python

```
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
[[[[29, 7], [177, 7], [177, 49], [29, 49]], 'ALR 486', 0.6714471980204154)]
```