

**MINISTRY OF EDUCATION AND TRAINING
CAN THO UNIVERSITY
SCHOOL OF INFORMATION & COMMUNICATION
TECHNOLOGY**



**INDUSTRY YEARBOOK REPORT
COMPUTER NETWORKS AND DATA COMMUNICATIONS**

LICENSE PLATE RECOGNITION

Instructor: **TRAN DUY QUANG**

Student: **LE HOANG PHI**

Student ID: **B2013551**

SEMESTER II, 2023-2024

CAN THO, 4/2024

[illegible]

Tran Duy Quang

ACKNOWLEDGMENTS

In order for this topic to achieve good results, I have received the facilitating support, help and encouragement of teachers, friends and family. With deep and sincere affection, I allow me to express his gratitude above all to all the individuals who have accompanied him throughout the process. In particular, I would like to thank Mr. Tran Duy Quang for wholeheartedly guiding, helping and guiding me to successfully complete this year.

At the same time, I would also like to thank the lecturers of the School of Information and Communication Technology in particular and Can Tho University in general for imparting knowledge, helping me accumulate valuable experience during the study period to be able and experience to complete the topic: "License plate recognition".

In the process of working on the project, I tried a lot but still could not avoid certain shortcomings as well as limitations. I look forward to receiving the dedicated comments and guidance of teachers and friends to make my topic more complete and practical.

Thank you very much!

Can Tho, April 25, 2024

Student implementation

Le Hoang Phi B2013551

TABLE OF CONTENTS

EVALUATION OF THE RESULTS OF THE IMPLEMENTATION OF THE INDUSTRY YEARBOOK	2
ACKNOWLEDGMENTS	3
TABLE OF CONTENTS	4
ABBREVIATE	5
CHAPTER 1: INTRODUCTION	6
1.1. INTRODUCTION	6
1.2. Project objectives	6
1.3. Topic limitation	6
1.4. Research methodology	6
1.5. Subjects and scope of research	6
CHAPTER 2: THEORETICAL FOUNDATIONS	7
2.1. ABOUT YOLO:	7
2.1.1. History	7
2.1.2. The YOLO Model Machine Learning Method	10
2.1.3. Advantages and Limitations of YOLOv8	11
2.2. ABOUT CNN	12
2.2.1. What is CNN?	12
2.2.2. What is the structure of the CNN network?	12
2.3. INTRODUCTION TO THE OS FUNCTION	14
2.3.1. What is os function?	14
2.3.2. Create output folder if it does not exist using os	14
CHAPTER 3: PROGRAM DEVELOPMENT	15
3.1. CREATE DATA: RECOGNIZE LICENSE PLATE AREA	15
3.2. TRAINING DATA INCLUSION	16
CHAPTER 4: EXPERIMENTAL RESULTS	19
4.1. RESULTS AFTER COACHING	19
4.2. RESULTING IMAGES	19
CONCLUDE	23
ADDENDUM	24
BIBLIOGRAPHY	26

ABBREVIATE

Today, with the development of science and technology and advanced and modern techniques, the application of such techniques to people's lives is really necessary, because these things not only help people's lives improve but also make society develop rapidly, contributing to promoting the progress of science and technology as well as catching up with the trends of the times.

One of the modern fields leading the current trend in the development of the 4.0 era is the study of artificial intelligence. Artificial intelligence is the most developing field in the world, using machine learning or deep learning methods,... and some other methods have helped the field of research on artificial intelligence is leading the trend not only in Vietnam but also outside the world. Therefore, in the field of research on artificial intelligence, up to now, there have been many applications for human life such as face recognition, lane detection, virtual assistant, handwriting recognition, license plate recognition,...

In particular, along with the growth of the world's population, the need to move or use of vehicles is also increased, due to too many means of transportation,

Participating in traffic, the management is also very difficult. In this report, will build Build an application developed on many platforms used to recognize license plates.

CHAPTER 1: INTRODUCE

1.1. INTRODUCE

YOLO (You Only Look Once) is one of the fastest and most accurate object recognition algorithms available today. YOLOv8 is the latest version of the YOLO series, which refers to the ability to predict all objects present in an image in a single data transmission. YOLO models are pre-trained on large datasets such as COCO and ImageNet. This allows them to both be able to provide extremely accurate predictions with trained classes, and to learn new classes with relative ease.

YOLOv8 is an algorithm that recognizes objects in images. The main goal of YOLOv8 is to detect and locate objects in images accurately and quickly. YOLOv8 is capable of detecting objects and drawing bounding boxes around them to indicate the position and size of that object in the image.

With the use of YOLOv8, users can apply this algorithm to various applications such as security monitoring, self-driving cars, healthcare, and many other fields to recognize and classify objects in images automatically and efficiently.

Thanks to the development of YOLOv8, we have the basis to implement the topic "License plate recognition".

1.2. Topic objectives

The topic "Face recognition" can be applied in the following systems:

- Traffic monitoring system
- Automatic parking lot
- Security protection

1.3. Topic limitations

The topic mainly uses libraries available in python programming, and at the same time applies mathematical models in artificial neural networks. The input condition of the model is that the image files are limited, the output of analyzing the license plate position is included with relative accuracy. As for functionality, there are limitations due to the understanding of programming and how the model is applied to large systems.

1.4. Research methodology

In this section, I designed a facial recognition model based on YOLOv8 using a special network architecture, usually a CNN (Convolutional Neural Network) model with multiple convolutional layers and full link layers. Based on the main research methods such as analysis methods, reference materials, methods of synthesizing theoretical documents...

1.5. Subjects and scope of research

The subjects to be studied and the scope of research can solve the topic:

- Object of study: Machine learning technology, python programming.
- Scope of research: YOLOv8, modules in python: YOLO, numpy, os, image.

YOLOv7 was released on April 30, 2022. YOLOv7 was developed by Alexey Bochkovskiy and a team from Taiwan National Academy of Science and Technology (NCTU) led by Chien-Yao Wang and HongYuan Mark Liao.

YOLOv8 was released in late 2022. YOLOv8 was developed by Alexey Bochkovskiy, Chien-Yao Wang and HongYuan Mark Liao. This model continues to evolve from YOLOv7 and YOLOv8, keeping the same collaboration with previous authors during the development and improvement of the model.

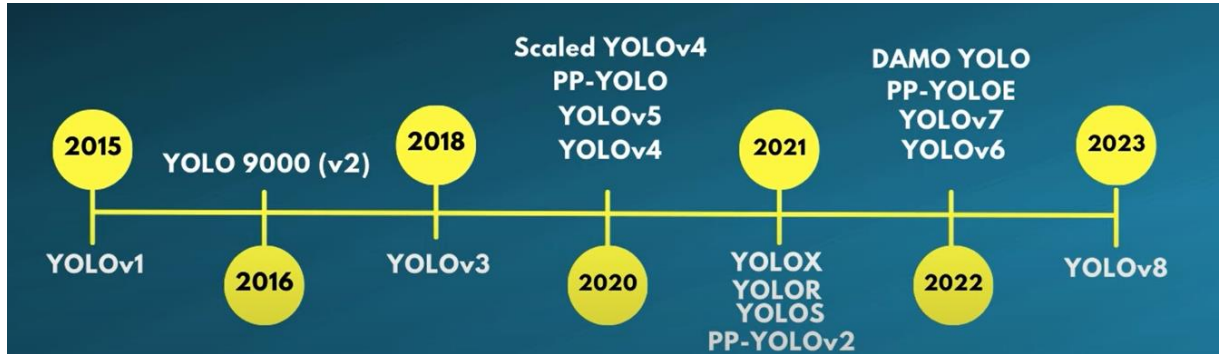


Image 2: YOLO Development Timeline

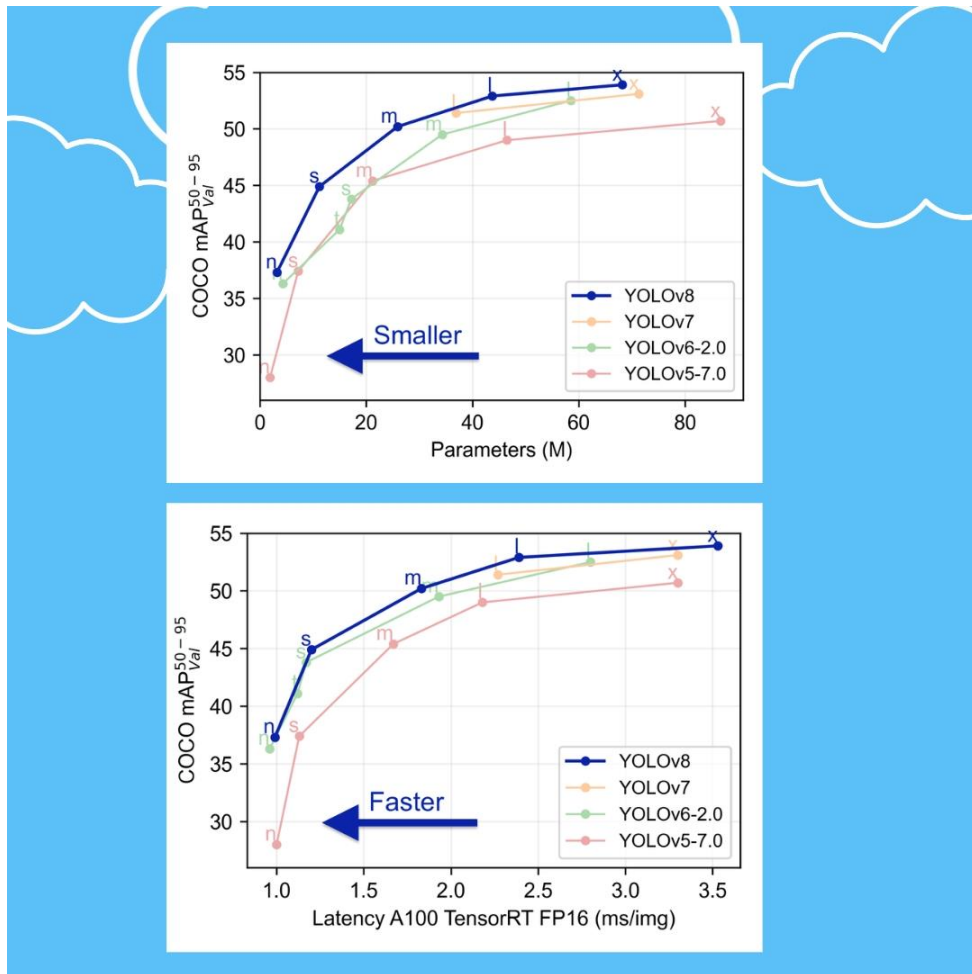


Image 3: Image from Ultralytics YOLOv8 archive

YOLOv8 is a convolutional neural network (CNN)-based object recognition model developed by Joseph Redmon and his research team at the University of Washington.

YOLOv8's architecture is based on previous versions of the YOLO algorithm. YOLOv8 uses a convolutional neural network that can be divided into two main parts: the backbone and the head.

The backbone of YOLOv8 is formed from a modified version of the CSPDarknet53 architecture. The architecture consists of 53 convolutional layers and uses cross-stage partial connections to improve the flow of information between different layers.

The first part of YOLOv8 consists of multiple convolution layers, followed by a sequence of full connection layers. These layers are responsible for predicting bounding boxes, objectness scores and layer probabilities for objects detected in an image.

One of the key features of YOLOv8 is the use of a self-attention mechanism in the early part of the network. This mechanism allows the model to focus on different parts of the image and adjust the importance of different features based on their relevance to the task.

Another important feature of YOLOv8 is its ability to perform multi-scale object recognition. The model uses a feature pyramid network to identify objects of different sizes and scales in an image.

This pyramid-specific network consists of multiple layers to recognize objects at different scales, allowing the model to recognize large and small objects in an image.



Image 4: Potential Use Cases of YoloV8

We can also visualize the YOLOv8 architecture ourselves by converting it to the ONNX format. ONNX stands for Open Neural Network Exchange. It is an open format used to represent machine learning models. It can be seen as a generic representation of ML models, because of its universality for any model, whether the model code is written in PyTorch, TensorFlow, or any other framework.

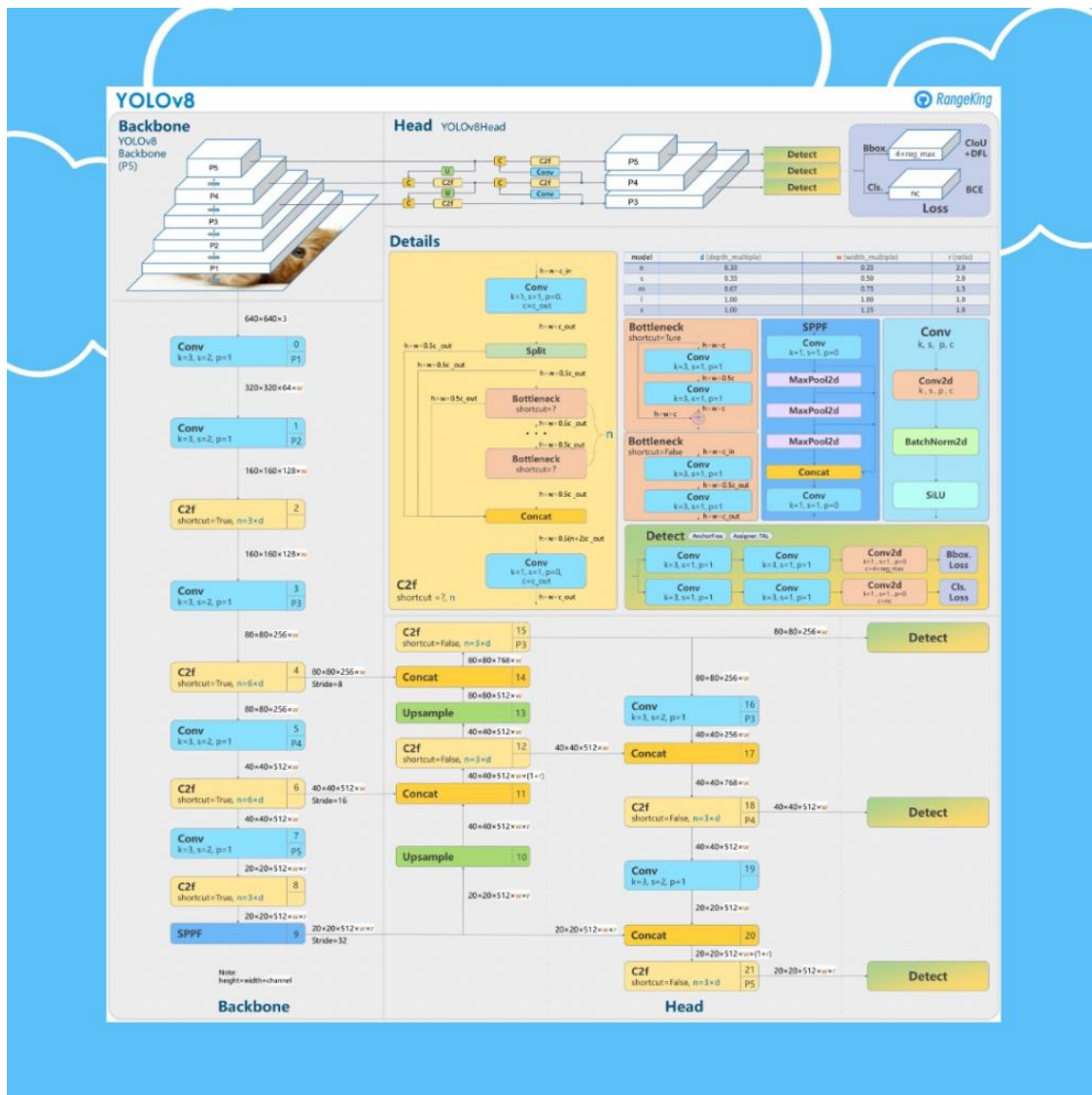


Image 5: YOLO model architecture. Image by GitHub user RangeKing

Converting any YOLO model to the ONNX format is also quite easy. Just copy and run the code below:

```
!pip install ultralytics

from ultralytics import YOLO
model = YOLO('yolov8m.pt')
model.export(format = "onnx")
```

Image 6: Format switching code

2.1.2. How machine learning works for the YOLO model?

The input image will be divided by the YOLO algorithm into $S \times S$, usually 3×3 , 7×7 , 9×9 ,...

With an input data of 1 image, the model output will be a 3-dimensional matrix of size

$S \times S \times (5 \times B + C)$ with the number of parameters per cell is $(5 \times B + C)$ where B is the number of bounding

Box, C is the class that each cell needs to predict.

We have a bounding box consisting of 5 predicted components (x,y,w,h, confidence) with (x,y) as coordinates

The center of the bounding box relative to the boundary of the grid cell, (w,h) is the width, the height is

prediction vs. the whole picture, and finally predict confidence representing IOUs (Intersection over Union) between predicted and fixed location

predicted box, ground truth box.

Each grid cell predicts class probabilities: $\Pr(\text{Class}_i | \text{Object})$

Confidence score: if there is no object in the cell, then the score will be 0, otherwise by $\Pr(\text{Object}) * \text{IOU}$

$$\Pr(\text{Class}_i | \text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

Image 7: The formula calculates the accuracy of object detection.

According to the above formula, we will give the probability that the final detection will predict and include the arguments

The statue in that image is as shown below:

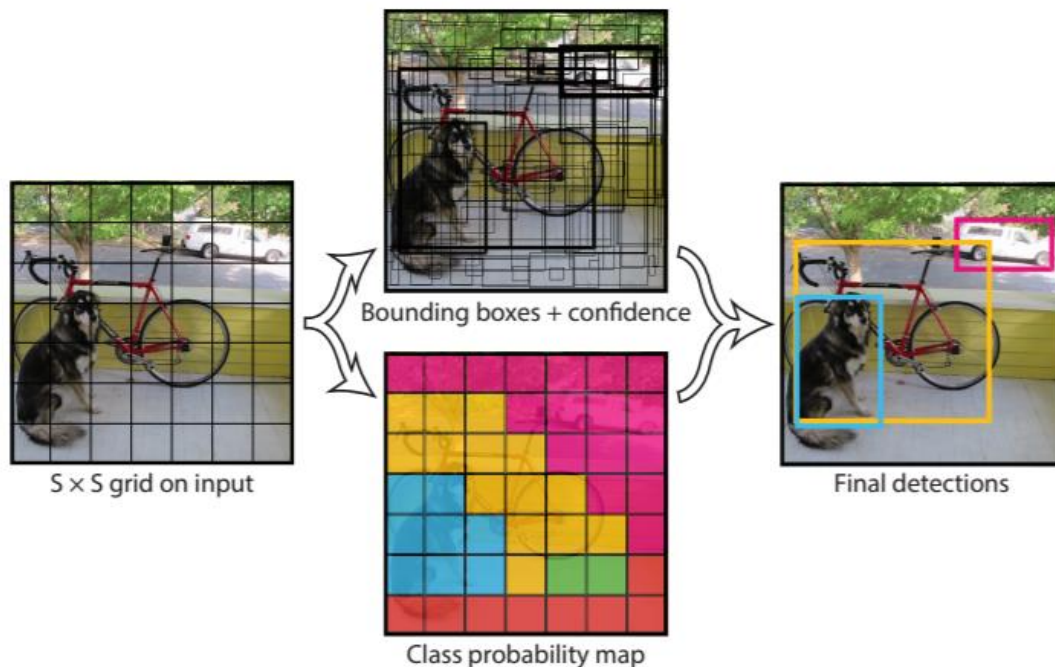


Image 8: Detects object recognition by formula.

2.1.3. Advantages and drawbacks of YOLOv8

Advantage

Speed: YOLOv8 is considered fast and low response time, helping to handle object recognition and image segmentation tasks in real time.

Accuracy: YOLOv8 builds on advances in deep learning and computer vision, ensuring high accuracy in object recognition.

Flexibility: YOLOv8 supports object recognition and segmentation on both GPUs and CPUs, leveraging technologies such as Nvidia's TensorRT and Intel's OpenVino.

Restrict

To use YOLOv8 effectively, it is necessary to:

In-depth knowledge of Machine Learning, Deep Learning and related algorithms.

It needs to be trained on a dataset large enough and diverse to be most effective.

Requires high computational resources to achieve fast and accurate processing speed.

The YOLOv8 algorithm is not open source and is only available through license agreements with its creator, Joseph Redmon.

YOLOv8 may not perform well in all environments and may need additional tuning or optimization to achieve optimal performance.

In a nutshell, YOLOv8 is a computer vision algorithm used for object detection based on a convolutional neural network (CNN). The input of the model is a photo, the model will identify whether the photo has any objects or not, then will determine the coordinates of the object in the photo. YOLOv8 has outstanding advantages such as high speed and accuracy, flexibility when it comes to object recognition and segmentation on both GPU and CPU. But YOLOv8 may not perform well in all environments and may need additional tuning or optimization to achieve optimal performance.

2.2. ABOUT CNN

2.2.1. What does CNN mean?

CNN is the ideal architecture when solving the problem of image data, one of the advanced deep learning models. It helps us build intelligent systems with high accuracy today. CNN is used a lot in problems to identify objects in images.

2.2.2. What is the structure of the CNN network?

The CNN network is one of the collections of the Superimposed Convolution class. The CNN network also uses nonlinear activation functions (such as ReLU and fishy) to trigger weights in the node. Once the function has been passed, this layer gains weight in the nodes and generates more abstract information for neighboring layers.

The key characteristics of CNNs are immutability and coherence. This ensures that the network can recognize objects in the image whether they are shifted, stretched or rotated. To achieve this, we use pooling classes to create immutability and combine local characteristics.

The pooling layer makes the network immutable to transformations such as shifting, scaling, and spinning. In addition, local matching between layers allows the network to learn characteristics from low to high levels through the application of convolution filters. This helps the network automatically learn important characteristics from training data.

During training, CNN will learn to recognize features by fine-tuning the parameters of the convolution classes. This process is similar to how humans recognize objects in the real world.

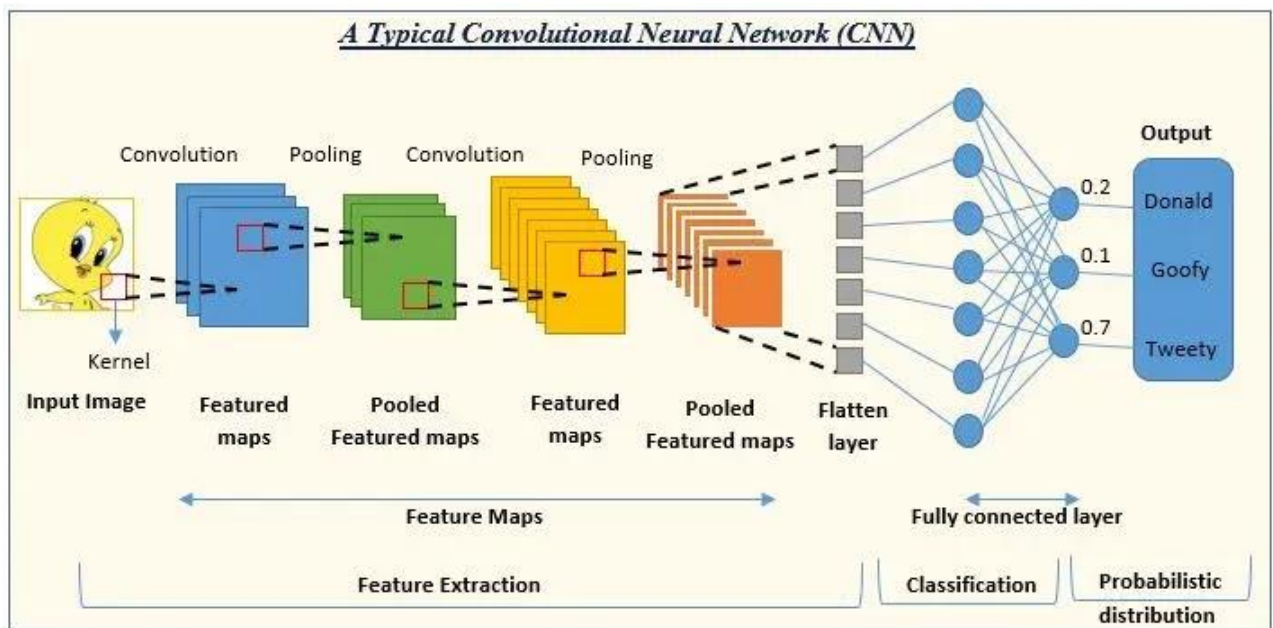


Image 9: The structure of the CNN network

The most basic structure of CNN will consist of 3 main parts, which are:

Local receptive field: The task of the local field is to separate and filter image data and information, and then select the areas of the image with the highest use value.

Shared weights and bias: In the CNN network, this component has the effect of minimizing the number of parameters that have a large effect. Each convolution will contain many different feature maps, each feature has the ability to help identify some features in the image.

Pooling layer: Pooling layer is the final layer, with the ability to simplify output information. Once the calculation and scanning through the layers are completed, the pooling layer will be created for the purpose of removing unnecessary information and optimizing the output. This helps users get satisfactory and satisfactory results as required or desired.

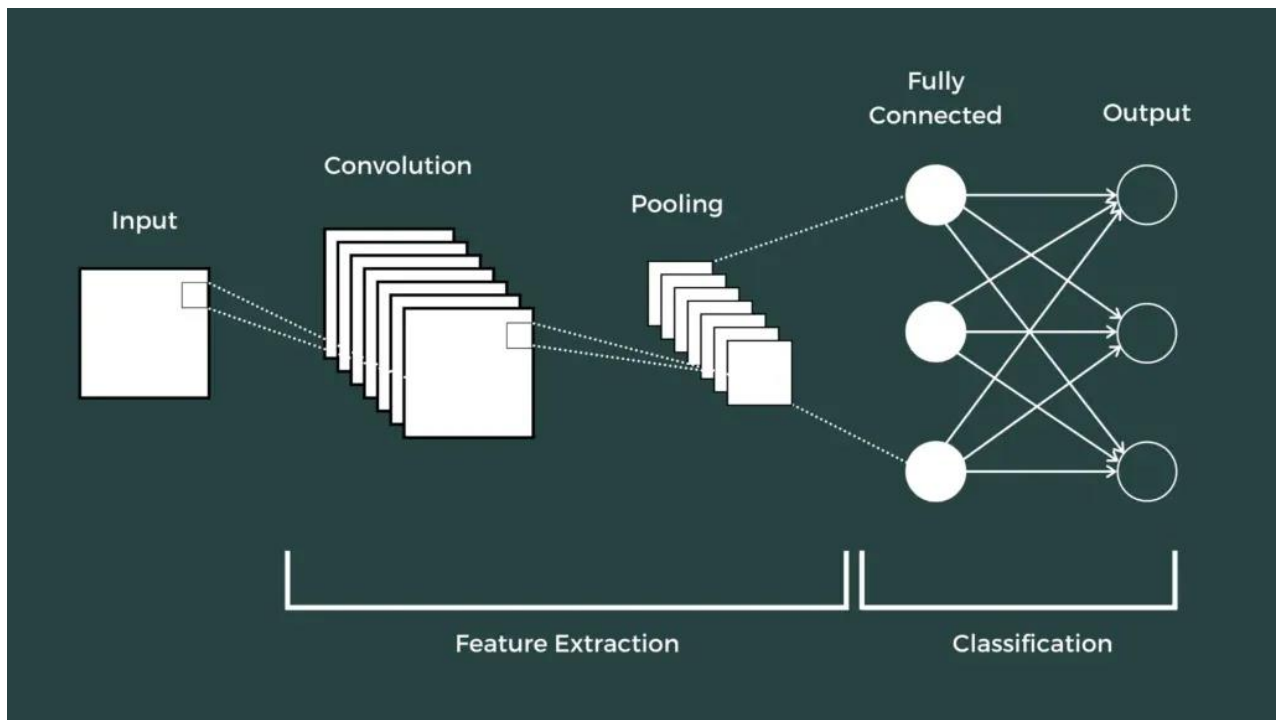


Image 10: CNN is widely used

2.3. ABOUT THE OS FUNCTION

2.3.1. What is the os function?

"os" can be thought of as a module that encapsulates POSIX calls or simulates them on some platform without POSIX. You'll see almost exactly the same named functions as in C and other languages, and how to do the same. It is easy to recognize constants such as `os.R_OK`, `os.WNOHANG` ... correctly named C/POSIX.

2.3.2. Create an output folder if it does not exist using os

`os.makedirs(output_directory, exist_ok=True)` is used to create an output directory if it doesn't already exist. The `makedirs` function from the `os` library receives input as the path `output_directory` and parameter `exist_ok=True` to ensure that if the directory already exists, the function will not create a new one, but will remain the same and continue to execute without causing errors. This keeps the code running smoothly and ensures the directory creation process is done efficiently.

```
# Create the output directory if it doesn't exist
os.makedirs(output_directory, exist_ok=True)
```

Image 11: Create the output directory if it doesn't exist

CHAPTER 3: PROGRAM DEVELOPMENT

To be able to conduct number plate recognition, we can divide into 3 main steps of number plate identification, which are:

Step 1: Create a dataset.

Step 2: Train the data to be included.

Step 3: Solve the problem of cutting license plates and give results.

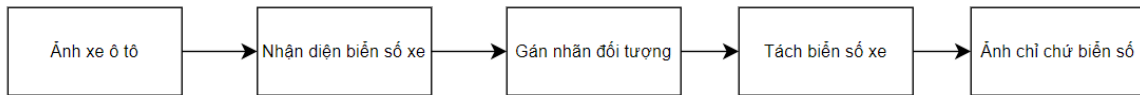


Image 12: The general scheme of the block determines the area containing the license plate

The above steps are based on the YOLOv8 model, which is a higher version

YOLO models were introduced earlier, as they are easier to install and optimized models result in higher recognition speed and accuracy. Based on the results after training, the dataset will produce number plate recognition results through high-precision stages.

3.1. GENERATE DATA: IDENTIFY LICENSE PLATE AREAS

In this report, we took data of 5 images containing license plates on the Label studio website, of which the image was used for machine learning, 3 images for value and 2 images for testing. Each image that has been marked with license plates will create a file that .txt is the label of the corresponding number plate. This txt file contains information to identify the license plate number of that image including class, x, y coordinates, length and height.



Image 13: Assign labels to objects

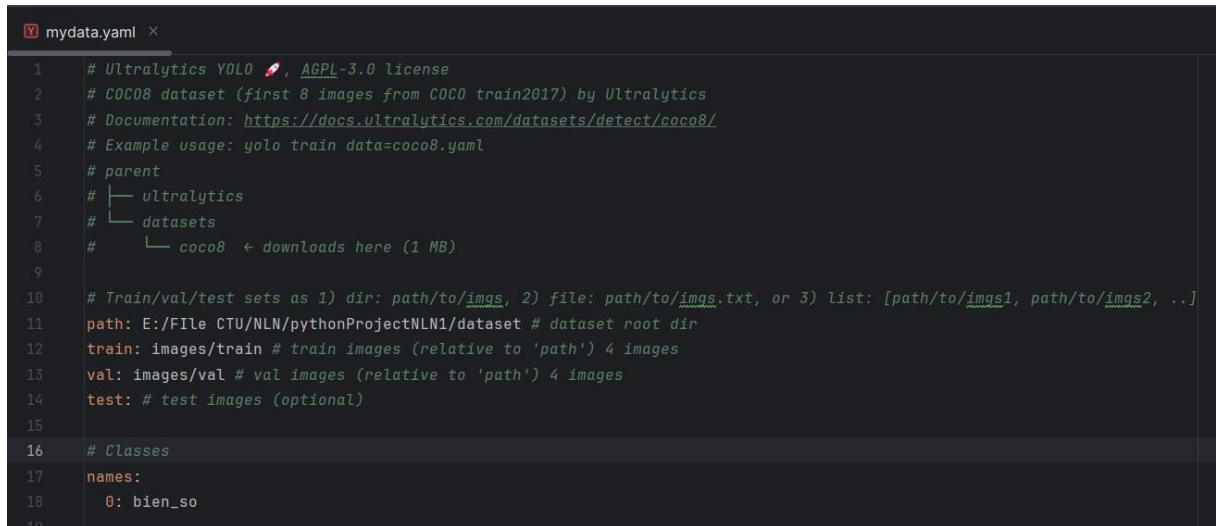
Once enough label data has been collected for the images. Finally, data training for machine learning will be performed to ensure that for each input any image containing motorcycle license plates is identifiable. Specifically, the data training is carried out as follows:

All machine learning is done on Pycharm. First, proceed to download the source code of YOLOv8 on github. This is open source, so it can be used for data training for

machine learning with pre-collected datasets along with some modifications to suit the research process as well as in identifying the area with license plates.

3.2. TRAINING DATA INCLUSION

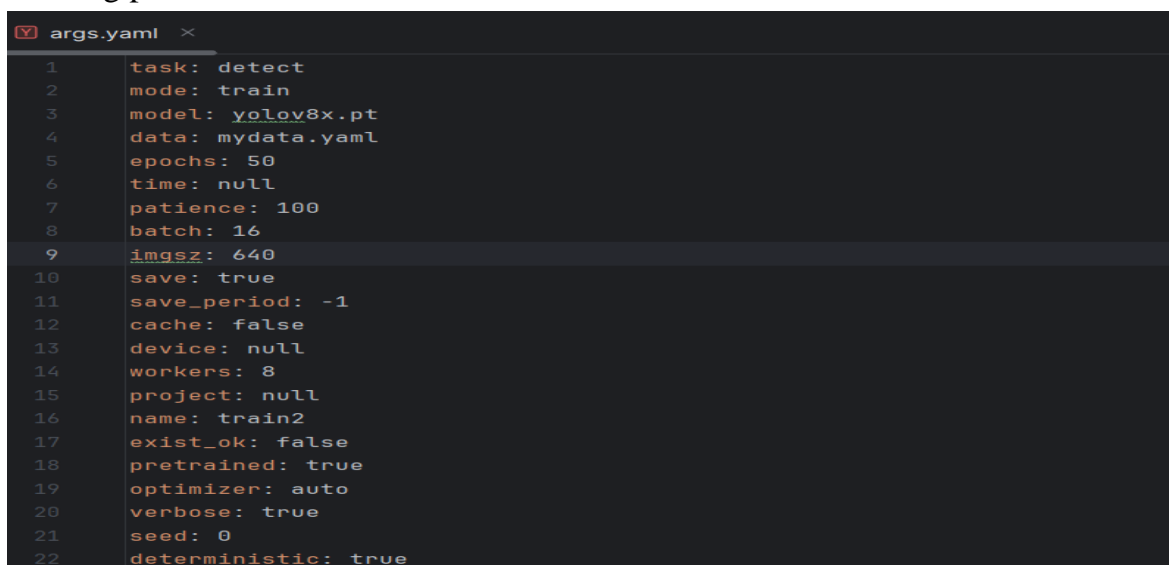
Create the file "mydata.yaml" with the declared class number of 0, because here to recognize the license plate, there is only one object, the license plate, in the declared array, configure the path to the folder containing the data file to start training. It is necessary to be able to train the data against the previously collected dataset.



```
1 # Ultralytics YOLO 🚀, AGPL-3.0 license
2 # COCO8 dataset (first 8 images from COCO train2017) by Ultralytics
3 # Documentation: https://docs.ultralytics.com/datasets/detect/coco8/
4 # Example usage: yolo train data=coco8.yaml
5 # parent
6 # └─ ultralytics
7 #   └─ datasets
8 #     └─ coco8 ← downloads here (1 MB)
9
10 # Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
11 path: E:/File CTU/NLN/pythonProjectNLN1/dataset # dataset root dir
12 train: images/train # train images (relative to 'path') 4 images
13 val: images/val # val images (relative to 'path') 4 images
14 test: # test images (optional)
15
16 # Classes
17 names:
18   0: bien_so
19
```

Image 14: Mydata.yaml file contents

After editing the necessary contents in the file, start the machine learning process for number plate recognition to take place, so that the machine learning process is performed to produce results with high accuracy, setting some parameters for the machine learning process as follows: Since this is a machine learning process that relies on a dataset consisting of images and labels of each corresponding image, set the parameter of "images", to 640 (since all images in the dataset are no larger than 640px), "batch" to 16 (during training, The number of training images at the same time is 16), "epochs" is 50 (the number of training repetitions is 50), with the parameters set above ensures that the data training process takes place with the highest accuracy, the machine learning process is shown as shown:



```
1 task: detect
2 mode: train
3 model: yolov8x.pt
4 data: mydata.yaml
5 epochs: 50
6 time: null
7 patience: 100
8 batch: 16
9 imgsz: 640
10 save: true
11 save_period: -1
12 cache: false
13 device: null
14 workers: 8
15 project: null
16 name: train2
17 exist_ok: false
18 pretrained: true
19 optimizer: auto
20 verbose: true
21 seed: 0
22 deterministic: true
```

Image 15: Data training process

After machine learning training with license plate determination, the results are shown as follows:

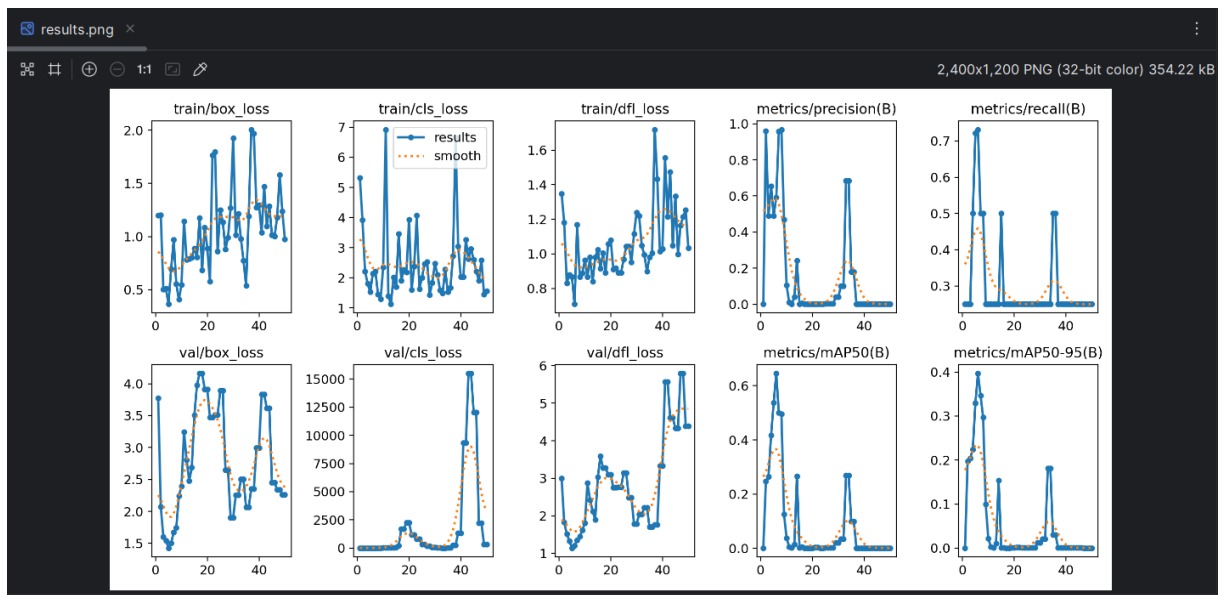


Image 16: Data training results evaluation chart

According to the figure above, it is noticed that the level of data loss gradually decreases asymptotic to zero, that is, during the data training iteration, the loss value decreases gradually because during the training process the sea has gradually been identified through iterations during data training, similar to the value of the object in the data loss chart. Since only one layer is a license plate during training, the number of layers lost is zero.

3.3 Solve the problem of cutting license plates and give results.

After the license plate number in an image has been identified, continue to deal with the problem of separating the license plate from the photograph or otherwise cropping the image containing the identified number plate into an image containing only the image of that number plate using the available cropped coordinates

```
# Create image filename based on detection index
image_filename = f'{i + 1}.png'
# Open original image
image_path = os.path.join(input_image_path, image_filename)
# Check if image file exists
if os.path.exists(image_path):
    original_image = Image.open(image_path)
    # Crop the license plate from the original image
    cropped_image = original_image.crop(box)
```

Image 17: Variable image cropping script

Shaped steel, coordinates consist of x_min.y_min, x_max, y_max, accordingly, proceed to crop the image along the vertical axis (y) from [y_min.y_max], along the horizontal

axis (x) from [x_min.x_max] so there will be a number plate image that has been cropped from the source image.



Image 18: The result of cutting license plates has been identified

With the use of the above number plate cut, it is almost possible to separate the license plate from a large image and focus only on that motorcycle number plate. This is the most essential and necessary stage for the problem of license plate recognition.

Because if there is no above stage, it may lead to discrepancies in the identification of characters in the license plate due to too large viewing frame, as well as the recognized characters may also be inaccurate, affecting the results of recognizing the characters of the license plate. If a character is misidentified or it is not in the area of the number plate, it will result in incorrect identification. This should not happen, in order to avoid this problem and as well as solve the recognition of license plate characters only in the area of the number plate, this is an extremely important stage.

CHAPTER 4: EXPERIMENTAL RESULTS

4.1. RESULTS AFTER TRAINING

```
C:\Users\ADMIN\.conda\envs\pythonProjectNLN1\python.exe "E:\File CTU\NLN\pythonProjectNLN1\predict.py"

image 1/5 E:\File CTU\NLN\pythonProjectNLN1\test\1.png: 544x640 1 bien_so, 1313.8ms
image 2/5 E:\File CTU\NLN\pythonProjectNLN1\test\2.png: 448x640 1 bien_so, 1001.4ms
image 3/5 E:\File CTU\NLN\pythonProjectNLN1\test\3.png: 448x640 1 bien_so, 992.2ms
image 4/5 E:\File CTU\NLN\pythonProjectNLN1\test\4.png: 480x640 1 bien_so, 1041.2ms
image 5/5 E:\File CTU\NLN\pythonProjectNLN1\test\5.png: 384x640 3 bien_sos, 869.5ms
Speed: 3.8ms preprocess, 1043.6ms inference, 3.7ms postprocess per image at shape (1, 3, 384, 640)
Cropped license plate saved to: E:/File CTU/NLN/pythonProjectNLN1/results\kq_1_bien_so.png
Cropped license plate saved to: E:/File CTU/NLN/pythonProjectNLN1/results\kq_2_bien_so.png
Cropped license plate saved to: E:/File CTU/NLN/pythonProjectNLN1/results\kq_3_bien_so.png
Cropped license plate saved to: E:/File CTU/NLN/pythonProjectNLN1/results\kq_4_bien_so.png
Cropped license plate saved to: E:/File CTU/NLN/pythonProjectNLN1/results\kq_5_bien_so.png

Process finished with exit code 0
```

Image 19: Result information

4.2. RESULTING IMAGE

- Compare results before and after license plate recognition:



Image 20: Photos before recognition



Image 21: Images after identification

- Compare results before and after recognizing multiple license plates:



Image 22: Photos before recognizing 5 license plates

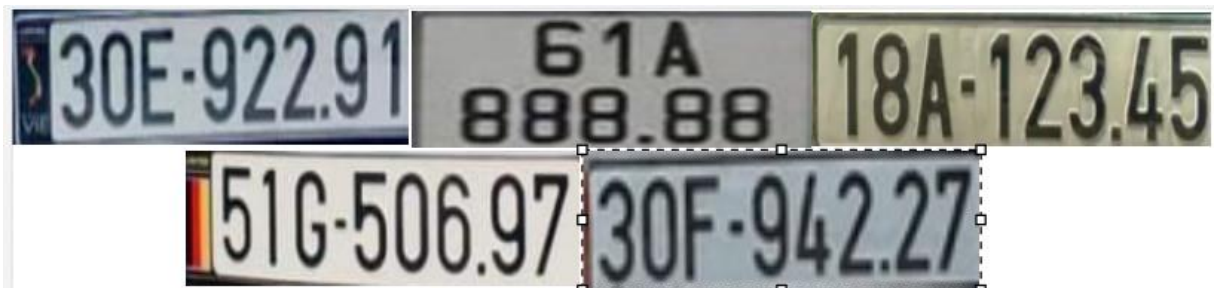


Image 23: Image after recognizing 5 license plates


```
predict x
:
image 2/100 E:\File CTU\NLN\pythonProjectNLN1\test\10.png: 512x640 5 bien_sos, 1118.4ms
image 3/100 E:\File CTU\NLN\pythonProjectNLN1\test\100.png: 480x640 2 bien_sos, 1051.2ms
image 4/100 E:\File CTU\NLN\pythonProjectNLN1\test\11.png: 416x640 2 bien_sos, 927.7ms
image 5/100 E:\File CTU\NLN\pythonProjectNLN1\test\12.png: 448x640 1 bien_so, 969.0ms
image 6/100 E:\File CTU\NLN\pythonProjectNLN1\test\13.png: 384x640 3 bien_sos, 821.2ms
image 7/100 E:\File CTU\NLN\pythonProjectNLN1\test\14.png: 384x640 1 bien_so, 814.7ms
image 8/100 E:\File CTU\NLN\pythonProjectNLN1\test\15.png: 448x640 (no detections), 957.6ms
image 9/100 E:\File CTU\NLN\pythonProjectNLN1\test\16.png: 448x640 11 bien_sos, 938.1ms
image 10/100 E:\File CTU\NLN\pythonProjectNLN1\test\17.png: 448x640 2 bien_sos, 941.9ms
image 11/100 E:\File CTU\NLN\pythonProjectNLN1\test\18.png: 352x640 4 bien_sos, 802.4ms
image 12/100 E:\File CTU\NLN\pythonProjectNLN1\test\19.png: 512x640 1 bien_so, 1057.7ms
image 13/100 E:\File CTU\NLN\pythonProjectNLN1\test\2.png: 416x640 3 bien_sos, 872.0ms
image 14/100 E:\File CTU\NLN\pythonProjectNLN1\test\20.png: 384x640 1 bien_so, 867.1ms
image 15/100 E:\File CTU\NLN\pythonProjectNLN1\test\21.png: 384x640 3 bien_sos, 849.7ms
image 16/100 E:\File CTU\NLN\pythonProjectNLN1\test\22.png: 416x640 2 bien_sos, 903.0ms
image 17/100 E:\File CTU\NLN\pythonProjectNLN1\test\23.png: 480x640 3 bien_sos, 1004.9ms
image 18/100 E:\File CTU\NLN\pythonProjectNLN1\test\24.png: 448x640 5 bien_sos, 960.3ms
image 19/100 E:\File CTU\NLN\pythonProjectNLN1\test\25.png: 416x640 2 bien_sos, 860.6ms
image 20/100 E:\File CTU\NLN\pythonProjectNLN1\test\26.png: 352x640 4 bien_sos, 816.7ms
image 21/100 E:\File CTU\NLN\pythonProjectNLN1\test\27.png: 448x640 1 bien_so, 948.2ms
image 22/100 E:\File CTU\NLN\pythonProjectNLN1\test\28.png: 448x640 7 bien_sos, 982.1ms
image 23/100 E:\File CTU\NLN\pythonProjectNLN1\test\29.png: 448x640 8 bien_sos, 997.4ms
image 24/100 E:\File CTU\NLN\pythonProjectNLN1\test\3.png: 448x640 1 bien_so, 950.0ms
image 25/100 E:\File CTU\NLN\pythonProjectNLN1\test\30.png: 480x640 1 bien_so, 1028.3ms
image 26/100 E:\File CTU\NLN\pythonProjectNLN1\test\31.png: 544x640 4 bien_sos, 1164.6ms
image 27/100 E:\File CTU\NLN\pythonProjectNLN1\test\32.png: 448x640 1 bien_so, 894.2ms
image 74/100 E:\File CTU\NLN\pythonProjectNLN1\test\75.png: 384x640 3 bien_sos, 861.9ms
image 75/100 E:\File CTU\NLN\pythonProjectNLN1\test\76.png: 448x640 2 bien_sos, 989.9ms
image 76/100 E:\File CTU\NLN\pythonProjectNLN1\test\77.png: 448x640 2 bien_sos, 958.5ms
image 77/100 E:\File CTU\NLN\pythonProjectNLN1\test\78.png: 640x640 5 bien_sos, 1386.2ms
image 78/100 E:\File CTU\NLN\pythonProjectNLN1\test\79.png: 640x480 3 bien_sos, 1056.8ms
image 79/100 E:\File CTU\NLN\pythonProjectNLN1\test\8.png: 480x640 2 bien_sos, 1102.6ms
image 80/100 E:\File CTU\NLN\pythonProjectNLN1\test\80.png: 480x640 1 bien_so, 1047.8ms
image 81/100 E:\File CTU\NLN\pythonProjectNLN1\test\81.png: 480x640 1 bien_so, 1005.7ms
image 82/100 E:\File CTU\NLN\pythonProjectNLN1\test\82.png: 384x640 4 bien_sos, 845.7ms
image 83/100 E:\File CTU\NLN\pythonProjectNLN1\test\83.png: 480x640 1 bien_so, 1058.2ms
image 84/100 E:\File CTU\NLN\pythonProjectNLN1\test\84.png: 448x640 1 bien_so, 967.9ms
image 85/100 E:\File CTU\NLN\pythonProjectNLN1\test\85.png: 480x640 3 bien_sos, 983.2ms
image 86/100 E:\File CTU\NLN\pythonProjectNLN1\test\86.png: 416x640 3 bien_sos, 938.4ms
image 87/100 E:\File CTU\NLN\pythonProjectNLN1\test\87.png: 448x640 5 bien_sos, 972.9ms
image 88/100 E:\File CTU\NLN\pythonProjectNLN1\test\88.png: 448x640 (no detections), 1009.5ms
image 89/100 E:\File CTU\NLN\pythonProjectNLN1\test\89.png: 448x640 1 bien_so, 954.6ms
image 90/100 E:\File CTU\NLN\pythonProjectNLN1\test\9.png: 448x640 2 bien_sos, 973.6ms
image 91/100 E:\File CTU\NLN\pythonProjectNLN1\test\90.png: 448x640 1 bien_so, 943.8ms
image 92/100 E:\File CTU\NLN\pythonProjectNLN1\test\91.png: 480x640 1 bien_so, 1026.7ms
image 93/100 E:\File CTU\NLN\pythonProjectNLN1\test\92.png: 480x640 3 bien_sos, 1038.4ms
image 94/100 E:\File CTU\NLN\pythonProjectNLN1\test\93.png: 448x640 1 bien_so, 972.0ms
image 95/100 E:\File CTU\NLN\pythonProjectNLN1\test\94.png: 480x640 2 bien_sos, 1052.3ms
image 96/100 E:\File CTU\NLN\pythonProjectNLN1\test\95.png: 384x640 3 bien_sos, 827.6ms
image 97/100 E:\File CTU\NLN\pythonProjectNLN1\test\96.png: 384x640 2 bien_sos, 890.2ms
image 98/100 E:\File CTU\NLN\pythonProjectNLN1\test\97.png: 384x640 8 bien_sos, 875.5ms
image 99/100 E:\File CTU\NLN\pythonProjectNLN1\test\98.png: 480x640 (no detections), 1043.5ms
```

Image 24: Run a license plate recognition program

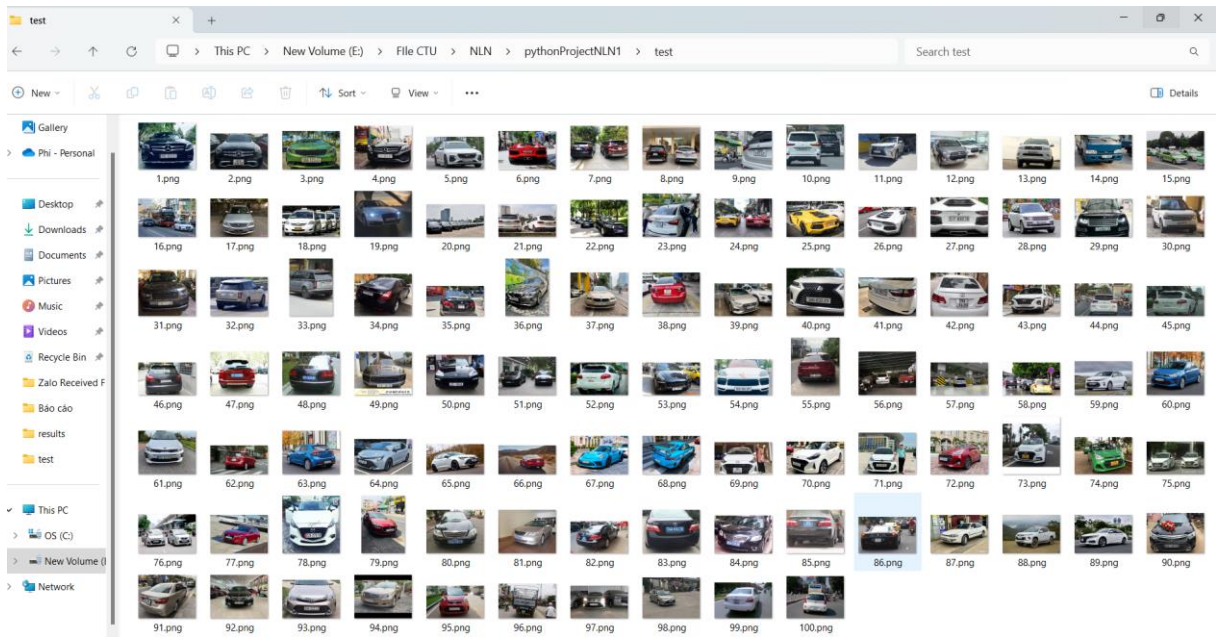


Image 25: Image before recognizing multiple license plates

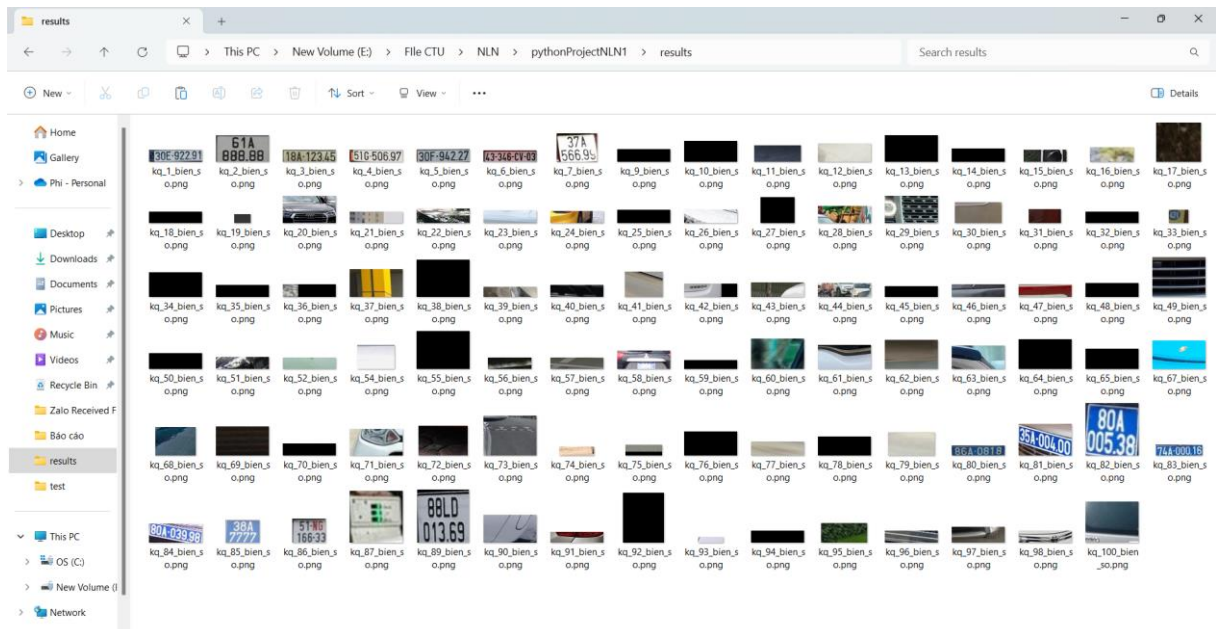


Image 26: Image after license plate recognition

*Observe:

- For license plate recognition, the results are accurate and at the same time detect the license plate number to be identified.
- For recognizing 5 license plates, the results are similar to recognizing 1 license plate and have high accuracy.
- For the identification of multiple license plates, the result is that the rate of identifying multiple license plates at the same time is not highly accurate, only 17%.

CONCLUDE

In this report, I have explored and studied the theory of image processing models, number plate and number plate recognition. This paper focuses on technology and its practical application to vehicle management.

Thus, it can be confirmed that my model is moderately complex, suitable for medium processing systems and offers the potential for feasibility in practical application.

Although the complexity of the model is low compared to other models, the test results show that the effectiveness of layering is quite high. Currently, due to calculation conditions, only a low number of training sessions is applied, if trained at a deeper level, it is expected to bring higher results.

To further develop the model, I will learn and design an image data collection system to create a diverse training data set for the model, thereby building an application for practical problems such as traffic monitoring systems, automatic parking, security area monitoring systems, mobile and digital applications.

ADDENDUM

The program is implemented on the Anaconda environment with 3 programs as follows:

+ Create training data files:

```
mydata.yaml x
1 # Ultralytics YOLO 🚀, AGPL-3.0 license
2 # COCO8 dataset (first 8 images from COCO train2017) by Ultralytics
3 # Documentation: https://docs.ultralytics.com/datasets/detect/coco8/
4 # Example usage: yolo train data=coco8.yaml
5 # parent
6 # └─ ultralytics
7 #   └─ datasets
8 #     └─ coco8 ← downloads here (1 MB)
9
10 # Train/val/test sets as 1) dir: path/to/images, 2) file: path/to/images.txt, or 3) list: [path/to/images1, path/to/images2, ..]
11 path: E:/File CTU/NLN/pythonProjectNLN1/dataset # dataset root dir
12 train: images/train # train images (relative to 'path') 4 images
13 val: images/val # val images (relative to 'path') 4 images
14 test: # test images (optional)
15
16 # Classes
17 names:
18   0: bien_so
```

To train data from an existing image folder, we include the path from the dataset folder.

+ Data set training:

```
main.py x
1 from ultralytics import YOLO
2 # Load a pretrained YOLO model (recommended for training)
3 model = YOLO('yolov8x.pt')
4 # Train the model using the 'mydata.yaml' dataset for 3 epochs
5 results = model.train(data='mydata.yaml', epochs=50, device='0')
6
```


+ License plate identification:

```
import os
import numpy as np
from PIL import Image

# Load a pretrained YOLO model
model = YOLO('E:/File
CTU/NLN/pythonProjectNLN1/runs/detect/train2/weights/best.pt')

# Input image path
input_image_path = 'E:/File CTU/NLN/pythonProjectNLN1/test'

# Run inference
results = model(input_image_path)

# Specify the directory to save results
output_directory = 'E:/File CTU/NLN/pythonProjectNLN1/results'

# Create the output directory if it doesn't exist
os.makedirs(output_directory, exist_ok=True)

# Label index for license plate (you need to check the label index in your
dataset)
license_plate_label_index = 0 # Replace with the correct label index

# Save images with license plate
for i, r in enumerate(results):
    for label, box in zip(r.names, np.array(r.boxes.xyxy).tolist()):
        # Check if the detected object is a license plate
        if label == license_plate_label_index:
            # Convert box coordinates to integers
            box = [int(coord) for coord in box]

            # Create image filename based on detection index
            image_filename = f'{i + 1}.png'
            # Open original image
            image_path = os.path.join(input_image_path, image_filename)
            # Check if image file exists
            if os.path.exists(image_path):
                original_image = Image.open(image_path)
                # Crop the license plate from the original image
                cropped_image = original_image.crop(box)
                # Save cropped image
                save_path = os.path.join(output_directory, f'kq_{i +
1}_bien_so.png')
                cropped_image.save(save_path)
                print(f'Cropped license plate saved to: {save_path}')
            else:
                print(f'Image file {image_filename} not found!')
        break # Stop checking other objects in this image
```

BIBLIOGRAPHY

- [1] ultralytics. [Online]. Available: <https://docs.ultralytics.com/vi#yolo-licenses-how-is-ultralytics-yolo-licensed>.
- [2] A. Mehra, 16 8 2023. [Online]. Available: <https://www.labellerr.com/blog/understanding-yolov8-architecture-applications-features/>.
- [3] G. J. T. C. D. |. U. Y. Docs, 2023. [Online]. Available: https://docs.ultralytics.com/yolov5/tutorials/train_custom_data/.
- [4] p. S. S. f. O. D. |. R-CNN. [Online]. Available: <https://www.geeksforgeeks.org/selective-search-for-object-detection-r-cnn/>.
- [5] P. S. H. t. T. Y. O. D. o. a. C. Dataset, 2023. [Online]. Available: <https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/>.
- [6] J. S. a. Francesco, 2023. [Online]. Available: <https://blog.roboflow.com/whats-new-in-yolov8/>.
- [7] VBD, 2023. [Online]. Available: https://vinbigdata.com/cong-nghe-hinh-anh/yolov8-co-gi-nang-cap-so-voi-cac-phien-ban-truoc.html#YOLOv8_la_gi.
- [8] S. Son Nguyen| Train custom object recognition AI models with yolov8, 2024. [Online]. Available: <https://www.youtube.com/watch?v=VdkwO4VQMR0&list=LL&index=17>.
- [9] H. Hung Nguyen | What is the CNN algorithm? Learn about Convolutional Neural Network, 2023. [Online]. Available: <https://vietnix.vn/cnn-la-gi/#:~:text=M%E1%BA%A1ng%20CNN%20l%C3%A0%20m%E1%BB%99t%20trong,cho%20c%C3%A1c%20l%E1%BB%9Bp%20k%E1%BA%BF%20c%E1%BA%ADn..>