**BÁO CÁO CÔNG VIỆC**

**Công việc số:** 74

**Mô tả công việc:** Tìm hiểu thuật toán nhận diện biển số xe dùng deeplearning.

**Người thực hiện:** Hạ Quang Dũng

**Ngày bắt đầu:** 06/03/2025

**Ngày kết thúc:** 04/03/2025

***Ghi chú:***

*Tài liệu tham khảo:* [Nhận diện biển số xe Việt Nam](https://viblo.asia/p/nhan-dien-bien-so-xe-viet-nam-Do754P9L5M6)

*Dữ liệu biển số xe (Dataset):*

[DataThiGiacMayTinh/GreenParking.zip at main · thigiacmaytinh/DataThiGiacMayTinh](https://github.com/thigiacmaytinh/DataThiGiacMayTinh/blob/main/GreenParking.zip)

*Ảnh đầu vào (trường hợp ảnh mờ lỗi, còn lại các trường hợp khác đều đúng):*



**NỘI DUNG TÀI LIỆU**

[**1. Gán nhãn và huấn luyện mô hình phát hiện biển số xe 3**](#_9n35kgkxqj21)

[1.1. Gán nhãn dữ liệu 3](#_ppm6q3on884t)

[1.1.1. Tạo project mới trên Roboflow 3](#_gyj94t9jdpga)

[1.1.2. Upload ảnh vào project 3](#_vlbd7ag50g80)

[1.1.3. Gán nhãn (Labeling) 3](#_emiqt88l3t9f)

[1.1.4. Quản lý, kiểm tra và xử lý ảnh 4](#_r1hykkz8r2pi)

[1.1.5. Chuẩn bị dữ liệu để huấn luyện 4](#_h20pbjcdf597)

[1.2. Huấn luyện mô hình 4](#_ere4rg5oih6m)

[**2. Phát hiện và chuyển đổi ký tự trên biển số xe bằng Google Vision API 6**](#_wbm4dwpb845e)

[2.1. Giới thiệu 6](#_jfuvdiudsz43)

[2.2. Luồng hoạt động 7](#_mh9d70oswfie)

[2.3. Mã nguồn chương trình 7](#_9sxuj32d765s)

[2.4. Kết quả 9](#_vb1h4n2vv8c7)

[**3. Phát hiện và chuyển đổi ký tự trên biển số xe bằng thư viện easyocr 10**](#_xij0n5wx407n)

[3.1. Giới thiệu 10](#_1l07q78n62z6)

[3.2. Cài đặt thư viện 10](#_q6uhj6iqq59p)

[3.3. Mô tả thuật toán 11](#_zckso97uv1g8)

[3.4. Mã nguồn 11](#_nf00s41iogo1)

[3.5. Kết quả 12](#_d6z1g1es6df0)

[3.6. Đánh giá 13](#_zfbetfn8iulh)

[**4. Phát hiện và chuyển đổi ký tự trên biển số xe bằng Deep Learning 13**](#_bw40s5vpkrj9)

[4.1. Gán nhãn dữ liệu 13](#_csx6jx8rmix2)

[4.1.1. Phát hiện và trích xuất các ký tự từ biển số xe 13](#_hadp4y3z52pe)

[4.1.2. Gán nhãn ký tự bằng tay 19](#_eq4dn4x2db4d)

[4.2. Huấn luyện mô hình CNN 22](#_67bnbnf4u7vq)

[4.2.1. Dữ liệu đầu vào 22](#_qo7p9vitx2fh)

[4.2.2. Tiền xử lý 22](#_ttz1snmqtr7c)

[4.2.3. Kiến trúc mô hình CNN 22](#_fls2oavpukex)

[4.2.4. Huấn luyện 22](#_3aae713hljz9)

[4.2.5. Đánh giá & trực quan hóa 22](#_5okjwp8cxc55)

[4.2.6. Mã nguồn chi tiết 23](#_v4blkyt1p1nt)

[4.3. Phát hiện và chuyển đổi ký tự từ ảnh biển số xe 28](#_a7lfsrz9vbru)

# 

# 1. Gán nhãn và huấn luyện mô hình phát hiện biển số xe

## 1.1. Gán nhãn dữ liệu

Tiến hành gán nhãn trên Roboflow

### 1.1.1. Tạo project mới trên Roboflow

* Truy cập: <https://app.roboflow.com>
* Đăng nhập hoặc đăng ký tài khoản.
* Nhấn "Create New Project":
* Project Type: Object Detection (vì ta cần detect vị trí biển số, không phải phân loại ảnh).
* Project Name: ví dụ LicensePlateDetection
* Project License: bạn chọn Private hoặc Public tùy nhu cầu.

### 1.1.2. Upload ảnh vào project

* Chọn Upload Images.
* Tải lên ảnh chứa xe và biển số.
* Chọn định dạng ảnh .jpg, .png, hoặc .jpeg.

### 1.1.3. Gán nhãn (Labeling)

Sau khi upload ảnh, Roboflow sẽ mở trình Image Annotator:

* Cách gán nhãn thủ công
  + Bấm vào ảnh bạn muốn annotate.
  + Kéo vùng chọn (bounding box) quanh biển số xe.
  + Gõ nhãn cho vùng chọn, ví dụ: "license\_plate" → Enter.
  + Nếu ảnh có nhiều biển số, bạn có thể vẽ nhiều vùng chọn cho cùng ảnh.
  + Sau khi gán nhãn xong, nhấn Save.
* Tip: bạn có thể dùng phím tắt B để bật chế độ vẽ box.

### 1.1.4. Quản lý, kiểm tra và xử lý ảnh

Sau khi annotate nhiều ảnh, bạn sẽ thấy:

* Số lượng ảnh đã labeled (annotated).
* Số ảnh chưa annotated.
* Bạn có thể sửa nhãn hoặc chỉnh bounding box bất kỳ lúc nào.

### 1.1.5. Chuẩn bị dữ liệu để huấn luyện

* Chuyển sang tab "Generate".
* Chọn định dạng xuất, ví dụ:
  + YOLOv8 (Nên dùng)
  + COCO JSON
  + Pascal VOC
* Chọn kích thước ảnh, resize (tuỳ mô hình bạn dùng).
* Nhấn "Generate Dataset" và Download.

## 1.2. Huấn luyện mô hình

* **Nền tảng gán nhãn: Roboflow** → Gán bounding box cho biển số xe, chia tập Train/Valid/Test, xuất định dạng YOLOv8.
* **Tải dữ liệu:** → Dùng API key Roboflow để tải version 2 của project "license-plate-detection-opksu" dưới định dạng YOLOv8.
* **Mô hình:** → Sử dụng YOLOv8n.pt (phiên bản nhẹ – tốc độ cao, ít tài nguyên).
* Huấn luyện:
  + 100 epochs
  + Kích thước ảnh: 640x640
  + Batch size: 16
  + Sử dụng file data.yaml từ Roboflow để định nghĩa dữ liệu.
* Kết quả:
  + Mô hình được lưu về Google Drive.
  + Có thể dự đoán ảnh mới và hiển thị kết quả phát hiện (bounding box và nhãn).

Mã nguồn chi tiết:

| # -\*- coding: utf-8 -\*- """Train.ipynb  Automatically generated by Colab.  Original file is located at  https://colab.research.google.com/drive/1XaW9fh8xsbRZklmArDO2joG7Y5wABFJk """  !pip install git+https://github.com/ultralytics/ultralytics.git !pip install roboflow  from roboflow import Roboflow rf = Roboflow(api\_key="EaizOCPRYmGzn34wUlz8") project = rf.workspace("dunghaquang").project("license-plate-detection-opksu") version = project.version(2) dataset = version.download("yolov8")  from ultralytics import YOLO model = YOLO('yolov8n.pt')  # Huấn luyện mô hình model.train(  data='/content/License-plate-detection-2/data.yaml', # Đường dẫn đến file data.yaml  epochs=100, # Số lượng epochs  imgsz=640, # Kích thước ảnh  batch=16, # Kích thước batch  workers=2 # Số lượng workers )  # Lưu mô hình vào thư mục Google Drive model.save("/content/drive/MyDrive/NGANGIANGINTERNSHIP/Job74/License-plate-detection.pt")  # Dự đoán với mô hình đã huấn luyện results = model("/content/drive/MyDrive/NGANGIANGINTERNSHIP/Job74/GreenParking/0000\_06886\_b.jpg") print(results[0]) # Hiển thị kết quả results[0].plot() # Hiển thị ảnh với các bounding box và nhãn |
| --- |

# 2. Phát hiện và chuyển đổi ký tự trên biển số xe bằng Google Vision API



## 2.1. Giới thiệu

Chương trình sử dụng mô hình YOLO để phát hiện biển số xe trên ảnh đầu vào, sau đó sử dụng Google Vision API để nhận diện ký tự trên biển số và hiển thị kết quả lên ảnh gốc.

**Công nghệ sử dụng:**

* YOLOv8: Phát hiện biển số xe trong ảnh.
* Google Vision API: Nhận diện ký tự trên biển số.
* OpenCV: Xử lý ảnh, cắt ảnh, vẽ bounding box và hiển thị kết quả.
* Python (re, os): Quản lý file và xử lý chuỗi ký tự.

**Kết quả mong đợi**

* Chương trình có thể tự động phát hiện và nhận diện biển số từ ảnh.
* Biển số xe nhận diện được hiển thị trên ảnh gốc cùng với bounding box.
* Kết quả trả về biển số sạch (chỉ bao gồm chữ và số, không có ký tự đặc biệt).

**Ghi chú**

* Đảm bảo ServiceAccountToken.json có quyền truy cập Google Vision API.
* Ảnh đầu vào cần có biển số rõ ràng để YOLO có thể phát hiện tốt.
* Nếu biển số bị che khuất hoặc bị mờ, kết quả nhận diện có thể không chính xác.

## 2.2. Luồng hoạt động

**Bước 1: Đọc ảnh đầu vào**

* Chương trình tải ảnh từ đường dẫn được cung cấp.
* Sử dụng OpenCV để đọc ảnh vào bộ nhớ.

**Bước 2: Phát hiện biển số xe bằng YOLO**

* Mô hình YOLO đã được huấn luyện trước để nhận diện biển số xe.
* Chương trình chạy YOLO trên ảnh đầu vào để phát hiện các vùng chứa biển số.
* Lấy danh sách các bounding box (tọa độ x1, y1, x2, y2) của biển số xe.

**Bước 3: Cắt biển số xe**

* Dựa vào tọa độ bounding box, cắt phần ảnh chứa biển số xe.
* Kiểm tra xem vùng cắt có hợp lệ hay không.
* Lưu ảnh biển số đã cắt vào thư mục Cropped.

**Bước 4: Nhận diện ký tự trên biển số bằng Google Vision API**

* Đọc ảnh biển số đã cắt và gửi lên Google Vision API để xử lý.
* API trả về kết quả nhận diện văn bản trên biển số.
* Lọc kết quả để chỉ giữ lại ký tự chữ và số, loại bỏ ký tự đặc biệt.

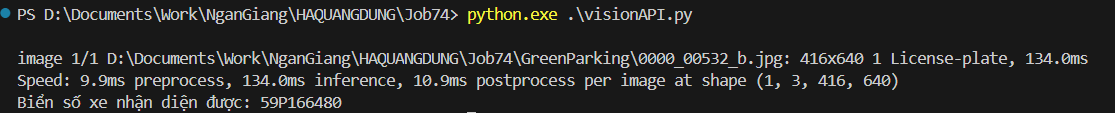
**Bước 5: Hiển thị kết quả lên ảnh gốc**

* Vẽ bounding box xung quanh biển số trên ảnh gốc.
* Hiển thị biển số nhận diện được bằng cv2.putText().
* Hiển thị ảnh đã xử lý.

## 2.3. Mã nguồn chương trình

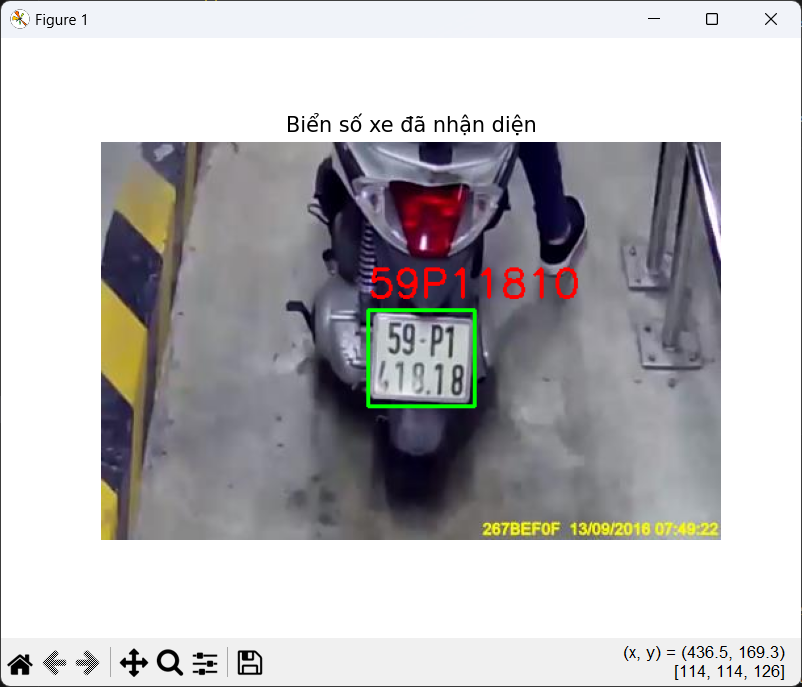
| import os import re import cv2 from google.cloud import vision from ultralytics import YOLO  # Cấu hình Google Cloud Vision API os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = r'./ServiceAccountToken.json'  # Load model YOLO đã train model = YOLO("./License-plate-detection.pt")  # Đọc ảnh image\_path = r"D:\Documents\Work\NganGiang\HAQUANGDUNG\Job74\GreenParking\0000\_00532\_b.jpg" img = cv2.imread(image\_path)  # Chạy mô hình để phát hiện biển số xe results = model(image\_path)  # Thư mục lưu ảnh biển số cắt output\_dir = r"D:\Documents\Work\NganGiang\HAQUANGDUNG\Job74\Cropped" os.makedirs(output\_dir, exist\_ok=True)  # Hàm nhận diện ký tự trên biển số xe def detect\_texts(path):  client = vision.ImageAnnotatorClient()  with open(path, "rb") as image\_file:  content = image\_file.read()   image = vision.Image(content=content)  response = client.text\_detection(image=image)  texts = response.text\_annotations   if not texts:  return "Không xác định"   # Chỉ lấy nội dung của biển số xe (dòng đầu tiên)  text = texts[0].description   # Giữ lại chỉ chữ và số  cleaned\_text = re.sub(r'[^A-Za-z0-9]', '', text)  return cleaned\_text  # Duyệt qua các kết quả phát hiện biển số xe for i, result in enumerate(results):  for j, box in enumerate(result.boxes):  # Lấy tọa độ của bounding box  x1, y1, x2, y2 = map(int, box.xyxy[0])   # Cắt vùng chứa biển số  license\_plate = img[y1:y2, x1:x2]   # Kiểm tra nếu ảnh cắt có kích thước hợp lệ  if license\_plate.size == 0:  print(f"Không thể cắt vùng biển số {j}, có thể bbox nằm ngoài ảnh!")  continue   # Lưu ảnh biển số gốc  cropped\_path = os.path.join(output\_dir, f"license\_plate\_{i}\_{j}.jpg")  cv2.imwrite(cropped\_path, license\_plate)   # Nhận diện biển số từ ảnh cắt  plate\_number = detect\_texts(cropped\_path)  print(f"Biển số xe nhận diện được: {plate\_number}")   # Vẽ bounding box và biển số lên ảnh gốc  cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)  cv2.putText(img, plate\_number, (x1, y1 - 5), cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (0, 255, 0), 2)  # Hiển thị ảnh gốc với biển số xe cv2.imshow("Detected License Plates", img) cv2.waitKey(0) cv2.destroyAllWindows() |
| --- |

## 2.4. Kết quả





# 3. Phát hiện và chuyển đổi ký tự trên biển số xe bằng thư viện easyocr



## 3.1. Giới thiệu

Phần này trình bày cách sử dụng thư viện **EasyOCR** để nhận diện ký tự trên biển số xe từ hình ảnh. Quy trình bao gồm:

* Phát hiện biển số xe bằng mô hình **YOLO**.
* Cắt và lưu biển số xe từ hình ảnh gốc.
* Nhận diện ký tự từ biển số xe bằng **EasyOCR**.

## 3.2. Cài đặt thư viện

Để chạy chương trình, cần cài đặt các thư viện sau:

| pip install opencv-python easyocr ultralytics |
| --- |

## 3.3. Mô tả thuật toán

1. **Phát hiện biển số xe**: Sử dụng mô hình YOLO đã được huấn luyện để xác định vùng chứa biển số xe trong ảnh.
2. **Cắt và lưu biển số xe**: Dựa vào tọa độ bounding box từ YOLO, cắt vùng chứa biển số và lưu lại.
3. **Nhận diện ký tự**: Dùng EasyOCR để trích xuất ký tự từ biển số xe đã cắt.
4. **Tiền xử lý kết quả**: Loại bỏ ký tự đặc biệt, chỉ giữ lại chữ cái và số để đảm bảo độ chính xác.

## 3.4. Mã nguồn

Dưới đây là mã nguồn Python thực hiện quy trình trên:

| import cv2 import re import easyocr import logging from pathlib import Path from ultralytics import YOLO  # Cấu hình logging logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")  # Load model YOLO đã train YOLO\_MODEL\_PATH = Path("./License-plate-detection.pt") model = YOLO(YOLO\_MODEL\_PATH)  # Thư mục ảnh IMAGE\_PATH = Path(r"D:\Documents\Work\NganGiang\HAQUANGDUNG\Job74\GreenParking\0002\_02183\_b.jpg") OUTPUT\_DIR = Path(r"D:\Documents\Work\NganGiang\HAQUANGDUNG\Job74\Cropped") OUTPUT\_DIR.mkdir(parents=True, exist\_ok=True)  # Khởi tạo EasyOCR reader reader = easyocr.Reader(['en'])  def detect\_license\_plate(image\_path):  """ Phát hiện biển số xe bằng YOLO và trả về danh sách bounding boxes """  results = model(str(image\_path))  return results  def crop\_and\_save(img, bbox, index):  """ Cắt và lưu ảnh biển số từ bounding box """  x1, y1, x2, y2 = map(int, bbox.xyxy[0])  license\_plate = img[y1:y2, x1:x2]    if license\_plate.size == 0:  logging.warning(f"Không thể cắt vùng biển số {index}, có thể bbox nằm ngoài ảnh!")  return None    cropped\_path = OUTPUT\_DIR / f"license\_plate\_{index}.jpg"  cv2.imwrite(str(cropped\_path), license\_plate)  return cropped\_path  def recognize\_text(image\_path):  """ Nhận diện ký tự từ ảnh biển số """  try:  result = reader.readtext(str(image\_path))  license\_plate = "".join(text for \_, text, \_ in result)  license\_plate = re.sub(r'[^A-Za-z0-9]', '', license\_plate)  return license\_plate  except Exception as e:  logging.error(f"Lỗi khi nhận diện biển số: {e}")  return None  def main():  """ Chạy toàn bộ quy trình nhận diện biển số xe """  img = cv2.imread(str(IMAGE\_PATH))  if img is None:  logging.error("Không thể đọc ảnh, vui lòng kiểm tra đường dẫn!")  return   results = detect\_license\_plate(IMAGE\_PATH)    for i, result in enumerate(results):  for j, box in enumerate(result.boxes):  cropped\_path = crop\_and\_save(img, box, f"{i}\_{j}")  if cropped\_path:  plate\_text = recognize\_text(cropped\_path)  logging.info(f"Biển số xe nhận diện: {plate\_text}")  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| --- |

## 3.5. Kết quả

* Ảnh biển số xe sau khi cắt được lưu tại thư mục **Cropped**.
* Ký tự biển số xe được hiển thị trong terminal/log file.

## 3.6. Đánh giá

* **Ưu điểm**:
  + Kết hợp YOLO và EasyOCR giúp nhận diện biển số chính xác hơn.
  + Chương trình có khả năng xử lý ảnh tự động, lưu trữ kết quả rõ ràng.
* **Nhược điểm**:
  + Một số ký tự có thể bị nhận diện sai nếu biển số bị mờ hoặc ánh sáng kém.
  + Cần cải thiện thuật toán xử lý lỗi khi biển số không được phát hiện.

# 4. Phát hiện và chuyển đổi ký tự trên biển số xe bằng Deep Learning



## 4.1. Gán nhãn dữ liệu

### 4.1.1. Phát hiện và trích xuất các ký tự từ biển số xe

**🧭 TỔNG QUAN CHỨC NĂNG**

* ✅ Đầu vào: ảnh biển số **đã cắt sẵn** (bỏ qua bước dùng YOLO).
* 🔍 Tiền xử lý ảnh để làm nổi bật các ký tự.
* ✂️ Trích xuất các ký tự bằng phương pháp phân tích thành phần liên thông (connected components).
* 🔠 Sắp xếp ký tự theo đúng thứ tự dòng và cột.
* 💾 Lưu ảnh các ký tự dưới dạng ảnh xám 28x28.

**⚙️ LUỒNG XỬ LÝ CHÍNH**

**1. main()**

* Duyệt toàn bộ ảnh trong thư mục ./Cropped.
* Gọi hàm process\_image() với từng ảnh.

**2. process\_image(image\_path)**

* Đọc ảnh biển số (đã cắt).
* Chuyển ảnh sang ảnh nhị phân (dùng adaptive threshold).
* Trích xuất và sắp xếp các ký tự.
* Lưu các ký tự thành ảnh nhỏ.

**3. preprocess\_license\_plate(plate\_img)**

* Chuyển ảnh sang HSV, lấy kênh sáng (V).
* Áp dụng threshold\_local (ngưỡng cục bộ kiểu Gaussian).
* Trả về ảnh nhị phân: chữ trắng, nền đen.

**4. extract\_and\_sort\_characters(thresh, plate\_img)**

a. Xử lý nhị phân:

* Đảo màu: nền trắng, chữ đen.
* Resize về chiều rộng 400 (giữ tỉ lệ), lọc nhiễu bằng medianBlur.

b. Trích xuất ký tự:

* Phân tích thành phần liên thông (skimage.measure.label).
* Với mỗi label (vùng), tính:
  + Tỷ lệ khung (aspect ratio)
  + Độ đặc (solidity)
  + Chiều cao so với ảnh gốc (height\_ratio)
* Lọc các vùng **có khả năng là ký tự** dựa trên tiêu chí trên.

c. Xử lý và lưu:

* Cắt vùng ký tự gốc từ ảnh màu.
* Đưa ảnh về dạng **vuông** (dùng convert\_to\_square()).
* Resize về 28x28.
* Thêm vào danh sách ứng viên.

d. Sắp xếp thứ tự ký tự:

* Dựa vào y để chia làm 2 dòng (nếu có).
* Trong mỗi dòng, sắp xếp từ trái sang phải theo x.

**5. save\_characters()**

* Lưu các ảnh ký tự với tên theo định dạng:

| <tên\_ảnh\_gốc>\_char\_00.png  <tên\_ảnh\_gốc>\_char\_01.png  ... |
| --- |

* Ảnh được lưu trong thư mục ./output\_chars.

**6. convert\_to\_square(image)**

* Làm ảnh thành vuông bằng cách **padding viền bằng màu trắng** (pad\_color=255).
* Hữu ích để resize ký tự mà không làm méo hình.

**📄 CẤU TRÚC THƯ MỤC**

| import os import cv2 import numpy as np from skimage.filters import threshold\_local import imutils from skimage import measure import matplotlib.pyplot as plt from ultralytics import YOLO  # Load YOLO model for license plate detection yolo\_model = YOLO("./License-plate-detection.pt")  # Directory paths input\_folder = "./output\_chars" output\_dir = r"D:\Documents\Work\NganGiang\HAQUANGDUNG\Job74\Cropped"  # Create output directories if they don't exist os.makedirs(output\_dir, exist\_ok=True)  def save\_characters(sorted\_candidates, original\_filename, save\_dir="output\_chars"):  """Lưu các ký tự đã trích xuất từ biển số vào thư mục"""  if not os.path.exists(save\_dir):  os.makedirs(save\_dir)    # Lấy tên file gốc làm tiền tố  file\_prefix = os.path.splitext(os.path.basename(original\_filename))[0]   # Lưu từng ký tự sau khi trích xuất  for index, (char, \_) in enumerate(sorted\_candidates):  filename = f"{save\_dir}/{file\_prefix}\_char\_{index:02d}.png"  cv2.imwrite(filename, char)  print(f"✅ Đã lưu ký tự: {filename}")  def convert\_to\_square(image):  """Chuyển đổi ảnh ký tự thành dạng vuông bằng cách thêm padding"""  img\_h, img\_w = image.shape[:2]   if img\_h > img\_w:  # Nếu chiều cao lớn hơn chiều rộng, thêm padding hai bên  diff = img\_h - img\_w  x1 = np.zeros((img\_h, diff // 2), dtype=image.dtype)  x2 = np.zeros((img\_h, diff // 2 + diff % 2), dtype=image.dtype)  return np.concatenate((x1, image, x2), axis=1)  elif img\_w > img\_h:  # Nếu chiều rộng lớn hơn chiều cao, thêm padding phía trên và dưới  diff = img\_w - img\_h  x1 = np.zeros((diff // 2, img\_w), dtype=image.dtype)  x2 = np.zeros((diff // 2 + diff % 2, img\_w), dtype=image.dtype)  return np.concatenate((x1, image, x2), axis=0)  return image  def preprocess\_license\_plate(license\_plate):  """Tiền xử lý ảnh biển số để chuẩn bị trích xuất ký tự"""  # Chuyển đổi ảnh sang không gian màu HSV  hsv = cv2.cvtColor(license\_plate, cv2.COLOR\_BGR2HSV)   # Lấy kênh V (độ sáng)  V = cv2.split(hsv)[2]   # Áp dụng Adaptive Thresholding để làm nổi bật ký tự  T = threshold\_local(V, 15, offset=10, method="gaussian")    # Trả về ảnh nhị phân (chỉ chứa ký tự)  return (V > T).astype("uint8") \* 255  def extract\_and\_sort\_characters(thresh, license\_plate):  """Trích xuất và sắp xếp các ký tự từ biển số dựa trên phân tích thành phần liên thông"""  labels = measure.label(thresh, connectivity=2, background=0)  candidates = []   for label in np.unique(labels):  if label == 0:  continue    # Tạo một mask chỉ chứa vùng có ký tự hiện tại  mask = np.zeros(thresh.shape, dtype="uint8")  mask[labels == label] = 255   # Tìm contour của ký tự  contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)    if contours:  contour = max(contours, key=cv2.contourArea)  x, y, w, h = cv2.boundingRect(contour)   # Tính toán một số thông số đặc trưng của ký tự  aspect\_ratio = w / float(h) # Tỉ lệ chiều rộng/chiều cao  solidity = cv2.contourArea(contour) / float(w \* h) # Độ đặc của ký tự  height\_ratio = h / float(license\_plate.shape[0]) # Tỉ lệ chiều cao so với biển số   # Lọc ra những ký tự có hình dạng hợp lý  if 0.1 < aspect\_ratio < 1.0 and solidity > 0.1 and 0.35 < height\_ratio < 2.0:  candidate = np.array(mask[y:y + h, x:x + w])   # Chuyển ký tự về dạng ảnh vuông  square\_candidate = convert\_to\_square(candidate)   # Resize về kích thước 28x28 (dùng cho nhận diện ký tự sau này)  square\_candidate = cv2.resize(square\_candidate, (28, 28), cv2.INTER\_AREA)   # Định dạng lại thành ảnh 3 kênh để dễ xử lý sau này  square\_candidate = square\_candidate.reshape((28, 28, 1))   # Thêm vào danh sách ký tự  candidates.append((square\_candidate, (y, x)))   # Sắp xếp ký tự theo tọa độ trên biển số  candidates = sorted(candidates, key=lambda x: x[1][0])   # Xác định vị trí trung bình để chia làm 2 hàng trên và dưới (nếu có)  mid\_y = np.median([pos[0] for \_, pos in candidates])  top\_row = [c for c in candidates if c[1][0] < mid\_y]  bottom\_row = [c for c in candidates if c[1][0] >= mid\_y]   # Sắp xếp từ trái sang phải  top\_row = sorted(top\_row, key=lambda x: x[1][1])  bottom\_row = sorted(bottom\_row, key=lambda x: x[1][1])   return top\_row + bottom\_row  def process\_image\_for\_license\_plate(image\_path):  """Xử lý ảnh để phát hiện biển số và trích xuất ký tự"""  img = cv2.imread(image\_path)    if img is None:  print(f"⚠️ Không thể đọc ảnh: {image\_path}")  return   # Dự đoán biển số bằng mô hình YOLO  results = yolo\_model(image\_path)   for i, result in enumerate(results):  for j, box in enumerate(result.boxes):  x1, y1, x2, y2 = map(int, box.xyxy[0])  license\_plate = img[y1:y2, x1:x2]    if license\_plate.size == 0:  print(f"⚠️ Không thể cắt vùng biển số {j}")  continue   # Lưu biển số đã cắt  cropped\_path = os.path.join(output\_dir, f"license\_plate\_{i}\_{j}.jpg")  cv2.imwrite(cropped\_path, license\_plate)   # Tiền xử lý biển số (chuyển sang ảnh nhị phân)  thresh = preprocess\_license\_plate(license\_plate)  thresh = cv2.bitwise\_not(thresh) # Đảo màu nền và chữ  thresh = imutils.resize(thresh, width=400) # Resize để dễ xử lý hơn  thresh = cv2.medianBlur(thresh, 5) # Giảm nhiễu bằng bộ lọc trung vị   # Trích xuất và sắp xếp các ký tự trên biển số  candidates = extract\_and\_sort\_characters(thresh, license\_plate)   # Lưu các ký tự vào thư mục  save\_characters(candidates, image\_path)  # Process the license plate image image\_path = r"D:\Documents\Work\NganGiang\HAQUANGDUNG\Job74\Dataset\0000\_06886\_b.jpg" process\_image\_for\_license\_plate(image\_path)   # # Danh sách các tệp ảnh trong thư mục # image\_folder = r"./Dataset" # image\_files = sorted([f for f in os.listdir(image\_folder) if f.endswith(('.png', '.jpg'))]) # for image\_file in image\_files: # image\_path = os.path.join(image\_folder, image\_file)  # process\_image\_for\_license\_plate(image\_path) |
| --- |

### 4.1.2. Gán nhãn ký tự bằng tay

**TÓM TẮT CHỨC NĂNG**

| **Thành phần** | **Mô tả** |
| --- | --- |
| 📂 ./output\_chars/ | Thư mục chứa ảnh ký tự đầu vào |
| 🏷️ annotations.txt | File lưu nhãn ở dạng: path label |
| 📁 ./labeled\_images/<label>/ | Ảnh sau khi gán nhãn được di chuyển vào thư mục tương ứng |
| 🖼️ Hiển thị ảnh | Resize về 100x100, hiển thị 1 ảnh/ký tự |
| ⌨️ Nhập nhãn | Người dùng nhập 1 ký tự, nhấn Enter để gán nhãn |
| ❌ Không nhập gì → ảnh bị xóa |  |

Mã nguồn chi tiết

| import os import shutil import tkinter as tk from PIL import Image, ImageTk  # Đường dẫn thư mục chứa ảnh image\_folder = r"./output\_chars" # Đổi lại tên thư mục phù hợp output\_file = "annotations.txt" labeled\_folder = r"./labeled\_images" # Thư mục chứa ảnh đã gán nhãn  # Danh sách các tệp ảnh trong thư mục image\_files = sorted([f for f in os.listdir(image\_folder) if f.endswith(('.png', '.jpg'))]) print(f"Tìm thấy {len(image\_files)} tệp ảnh trong thư mục {image\_folder}")  # Khởi tạo cửa sổ Tkinter root = tk.Tk() root.title("Image Annotation Tool")  # Đặt kích thước cửa sổ đủ lớn để chứa ảnh root.geometry("1000x1000") # Điều chỉnh kích thước cửa sổ  # Hiển thị ảnh và ô nhập ký tự image\_label = tk.Label(root) image\_label.pack()  # Tạo ô nhập ký tự và làm nó lớn hơn entry = tk.Entry(root, font=("Arial", 40)) # Kích thước font lớn hơn entry.pack(pady=20, anchor='center') # Căn giữa ô nhập liệu entry.focus()  # Chức năng để hiển thị ảnh và cho phép nhập ký tự def show\_next\_image():  if not image\_files:  print("Tất cả ảnh đã được xử lý!")  root.quit()  return    # Lấy tên file ảnh tiếp theo  image\_file = image\_files.pop(0)  image\_path = os.path.join(image\_folder, image\_file)    # Mở và hiển thị ảnh  img = Image.open(image\_path)    # Resize ảnh cho phù hợp với cửa sổ  img = img.resize((100, 100)) # Resize ảnh theo kích thước mong muốn, ví dụ 500x500  img\_tk = ImageTk.PhotoImage(img)  image\_label.config(image=img\_tk)  image\_label.image = img\_tk    # Tạo hàm lưu nhãn và di chuyển ảnh  def save\_label(event=None):  label = entry.get().strip()    if label:  # Tạo thư mục nhãn nếu chưa tồn tại  label\_folder\_path = os.path.join(labeled\_folder, label)  if not os.path.exists(label\_folder\_path):  os.makedirs(label\_folder\_path)    # Di chuyển ảnh vào thư mục nhãn  new\_image\_path = os.path.join(label\_folder\_path, image\_file)  shutil.move(image\_path, new\_image\_path)    # Lưu nhãn vào file txt với đường dẫn mới  with open(output\_file, "a") as f:  f.write(f"{new\_image\_path} {label}\n")  print(f"Lưu nhãn: {label} cho ảnh {new\_image\_path}")    else:  # Nếu không nhập ký tự, xóa ảnh  os.remove(image\_path)  print(f"Đã xóa ảnh {image\_path} do không có nhãn.")   # Xóa nội dung ô nhập ký tự sau khi lưu nhãn  entry.delete(0, tk.END)    # Sau khi xử lý xong, chuyển sang ảnh tiếp theo  show\_next\_image()   # Khi nhấn Enter, lưu nhãn hoặc xoá ảnh  entry.bind("<Return>", save\_label)  # Bắt đầu hiển thị ảnh đầu tiên show\_next\_image()  # Chạy vòng lặp Tkinter root.mainloop() |
| --- |

## 4.2. Huấn luyện mô hình CNN

**🧠 Tóm tắt mô hình nhận dạng ký tự (Train.ipynb)**

### 4.2.1. Dữ liệu đầu vào

* Dữ liệu từ thư mục ./labeled\_images, gồm các ảnh ký tự đã được gán nhãn thủ công.
* Kích thước ảnh: 64x64, ảnh **grayscale** (1 kênh màu).
* Nhãn được ánh xạ thành chỉ số (label encoding).
* Dữ liệu được chuẩn hóa về [0,1].

### 4.2.2. Tiền xử lý

* **Chia dữ liệu**: 70% huấn luyện, 15% validation, 15% test (qua train\_test\_split).
* **Class weights**: Tự động tính toán bằng class\_weight.compute\_class\_weight để xử lý mất cân bằng nhãn.
* **Augmentation** với ImageDataGenerator: xoay, dịch, shear, zoom...

### 4.2.3. Kiến trúc mô hình CNN

* **Input**: (64, 64, 1)
* 3 block Conv2D + BatchNorm + MaxPooling + Dropout:
  + 32 → 64 → 128 filters
* Flatten → Dense(256) + Dropout(0.5)
* Output: Dense(num\_classes, activation='softmax')

### 4.2.4. Huấn luyện

* Optimizer: Adam(0.001)
* Loss: categorical\_crossentropy
* Sử dụng các callback:
  + ModelCheckpoint lưu mô hình tốt nhất theo val\_accuracy
  + EarlyStopping với patience=15
  + ReduceLROnPlateau giảm LR nếu val\_loss không giảm

### 4.2.5. Đánh giá & trực quan hóa

* Accuracy trên tập test.
* Đồ thị: độ chính xác và loss theo epoch.
* Confusion matrix hiển thị dự đoán so với thực tế (dùng seaborn).

### 4.2.6. Mã nguồn chi tiết

| # -\*- coding: utf-8 -\*- """Train.ipynb  Automatically generated by Colab.  Original file is located at  https://colab.research.google.com/drive/160QMj4y0kxbbCpk7HEtLwMdK2QIJ0lUy """  !pip install tensorflow keras !pip install h5py !pip install numpy opencv-python matplotlib scipy  import os import numpy as np import tensorflow as tf from tensorflow.keras.preprocessing.image import ImageDataGenerator from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization from tensorflow.keras.optimizers import Adam from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau from sklearn.model\_selection import train\_test\_split from sklearn.utils import class\_weight import cv2 import matplotlib.pyplot as plt import seaborn as sns from pathlib import Path from sklearn.metrics import classification\_report, confusion\_matrix  # Đường dẫn đến thư mục chứa dữ liệu trên Google Drive data\_dir = "/content/drive/MyDrive/NGANGIANGINTERNSHIP/Job74/DL/labeled\_images" # Cập nhật đường dẫn nếu cần  # Tham số cho mô hình img\_width, img\_height = 64, 64 batch\_size = 32 epochs = 50 grayscale = True  # Load dữ liệu def load\_data(data\_dir):  images = []  labels = []  class\_dirs = sorted(os.listdir(data\_dir))  label\_to\_idx = {class\_name: idx for idx, class\_name in enumerate(class\_dirs)}   for class\_name in class\_dirs:  class\_path = os.path.join(data\_dir, class\_name)  if not os.path.isdir(class\_path):  continue   for img\_file in os.listdir(class\_path):  img\_path = os.path.join(class\_path, img\_file)  try:  if grayscale:  img = cv2.imread(img\_path, cv2.IMREAD\_GRAYSCALE)  img = cv2.resize(img, (img\_width, img\_height))  img = np.expand\_dims(img, axis=-1)  else:  img = cv2.imread(img\_path)  img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  img = cv2.resize(img, (img\_width, img\_height))   if img is not None:  images.append(img)  labels.append(label\_to\_idx[class\_name])  except Exception as e:  print(f"Lỗi khi đọc ảnh {img\_path}: {e}")   return np.array(images), np.array(labels), label\_to\_idx  # Tải dữ liệu images, labels, label\_to\_idx = load\_data(data\_dir) idx\_to\_label = {v: k for k, v in label\_to\_idx.items()} num\_classes = len(label\_to\_idx)  print(f"Số lượng ảnh: {len(images)}") print(f"Số lượng nhãn: {num\_classes}")  # Chuẩn hóa dữ liệu images = images.astype('float32') / 255.0 print(f"Kích thước tập dữ liệu: {images.shape}")  # Chia tập dữ liệu X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(images, labels, test\_size=0.3, stratify=labels, random\_state=42) X\_val, X\_test, y\_val, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=0.5, stratify=y\_temp, random\_state=42)  y\_train\_onehot = tf.keras.utils.to\_categorical(y\_train, num\_classes) y\_val\_onehot = tf.keras.utils.to\_categorical(y\_val, num\_classes) y\_test\_onehot = tf.keras.utils.to\_categorical(y\_test, num\_classes)  print(f"Kích thước tập huấn luyện: {X\_train.shape}") print(f"Kích thước tập kiểm định: {X\_val.shape}") print(f"Kích thước tập kiểm thử: {X\_test.shape}")  # # Tính class weights # class\_weights = class\_weight.compute\_class\_weight('balanced', np.unique(y\_train), y\_train) # class\_weights\_dict = {i: weight for i, weight in enumerate(class\_weights)}  # print(f"Class weights: {class\_weights\_dict}")  # Đảm bảo y\_train là một mảng numpy y\_train = np.array(y\_train)  # Tính toán class weights class\_weights =class\_weight.compute\_class\_weight(  class\_weight='balanced',  classes=np.unique(y\_train),  y=y\_train )  # Chuyển thành dictionary class\_weights\_dict = {i: class\_weights[i] for i in range(len(class\_weights))}  print(f"Class weights: {class\_weights\_dict}")  # Data Augmentation datagen = ImageDataGenerator(  rotation\_range=15,  width\_shift\_range=0.1,  height\_shift\_range=0.1,  shear\_range=0.1,  zoom\_range=0.1,  fill\_mode='nearest' )  print("Datagen", datagen)  # Xây dựng mô hình CNN input\_shape = (img\_width, img\_height, 1) if grayscale else (img\_width, img\_height, 3)  model = Sequential([  Conv2D(32, (3, 3), activation='relu', padding='same', input\_shape=input\_shape),  BatchNormalization(),  MaxPooling2D(2, 2),  Dropout(0.2),   Conv2D(64, (3, 3), activation='relu', padding='same'),  BatchNormalization(),  MaxPooling2D(2, 2),  Dropout(0.3),   Conv2D(128, (3, 3), activation='relu', padding='same'),  BatchNormalization(),  MaxPooling2D(2, 2),  Dropout(0.4),   Flatten(),  Dense(256, activation='relu'),  BatchNormalization(),  Dropout(0.5),  Dense(num\_classes, activation='softmax') ])  model.compile(optimizer=Adam(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])  # Callbacks checkpoint = ModelCheckpoint("/content/drive/MyDrive/NGANGIANGINTERNSHIP/Job74/DL/best\_model2.h5", monitor='val\_accuracy', save\_best\_only=True, mode='max') early\_stop = EarlyStopping(monitor='val\_loss', patience=15, restore\_best\_weights=True) reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.5, patience=5, min\_lr=1e-6)  callbacks = [checkpoint, early\_stop, reduce\_lr]  # Huấn luyện history = model.fit(  datagen.flow(X\_train, y\_train\_onehot, batch\_size=batch\_size),  epochs=epochs,  validation\_data=(X\_val, y\_val\_onehot),  class\_weight=class\_weights\_dict,  callbacks=callbacks )  # Đánh giá mô hình \_, test\_acc = model.evaluate(X\_test, y\_test\_onehot) print(f"Độ chính xác trên tập kiểm tra: {test\_acc\*100:.2f}%")  # Lưu mô hình model.save("/content/drive/MyDrive/NGANGIANGINTERNSHIP/Job74/DL/model\_nhan\_dang\_bien\_so\_final\_2.h5")  # Vẽ đồ thị kết quả plt.figure(figsize=(12, 4)) plt.subplot(1, 2, 1) plt.plot(history.history['accuracy'], label='Độ chính xác huấn luyện') plt.plot(history.history['val\_accuracy'], label='Độ chính xác kiểm định') plt.legend() plt.title('Độ chính xác qua các epoch')  plt.subplot(1, 2, 2) plt.plot(history.history['loss'], label='Mất mát huấn luyện') plt.plot(history.history['val\_loss'], label='Mất mát kiểm định') plt.legend() plt.title('Mất mát qua các epoch') plt.show()  # Confusion Matrix y\_pred = np.argmax(model.predict(X\_test), axis=1) y\_true = np.argmax(y\_test\_onehot, axis=1) cm = confusion\_matrix(y\_true, y\_pred) plt.figure(figsize=(10, 8)) sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=idx\_to\_label.values(), yticklabels=idx\_to\_label.values()) plt.xlabel('Dự đoán') plt.ylabel('Thực tế') plt.title('Confusion Matrix') plt.show() |
| --- |

## 4.3. Phát hiện và chuyển đổi ký tự từ ảnh biển số xe

Sau khi đã huấn luyện mô hình CNN nhận diện ký tự (mục 4.2), giai đoạn tiếp theo là xây dựng một pipeline đầy đủ để phát hiện biển số trong ảnh gốc, tách ký tự, và nhận diện từng ký tự bằng mô hình đã huấn luyện. Quy trình thực hiện gồm các bước sau:

**Bước 1: Phát hiện và cắt ảnh biển số**

* Sử dụng mô hình YOLOv8 (ultralytics) đã được huấn luyện trước để phát hiện vùng chứa biển số trong ảnh đầu vào.
* Các bounding box được trích xuất và lưu thành từng ảnh biển số riêng biệt trong thư mục tạm ./Cropped.

**Bước 2: Tiền xử lý ảnh biển số**

* Mỗi ảnh biển số được chuyển sang ảnh nhị phân (đen trắng) bằng phương pháp ngưỡng cục bộ Gaussian (adaptive thresholding) trên kênh V của không gian màu HSV.
* Ảnh nhị phân này giúp dễ dàng phát hiện các vùng ký tự trên biển số.

**Bước 3: Tách và chuẩn hoá các ký tự**

* Áp dụng phương pháp phân vùng connected components (skimage.measure.label) để phát hiện các cụm điểm ảnh có khả năng là ký tự.
* Các vùng được lọc theo tiêu chí hình dạng như: tỷ lệ khung hình, độ đặc (solidity), và tỷ lệ chiều cao so với biển số để loại bỏ nhiễu.
* Sau đó, từng ký tự được:
  + Cắt ra khỏi ảnh gốc,
  + Đưa về dạng ảnh vuông bằng padding,
  + Resize về kích thước chuẩn (28x28),
  + Lưu vào thư mục ./output\_chars.

**Bước 4: Nhận diện từng ký tự**

* Lần lượt đọc từng ảnh ký tự đã lưu.
* Dùng mô hình CNN đã huấn luyện (model\_nhan\_dang\_bien\_so\_final.h5) để dự đoán lớp tương ứng.
* Dự đoán được ánh xạ trở lại nhãn (số hoặc chữ) nhờ bảng idx\_to\_label.
* Cuối cùng, các ký tự được ghép nối theo thứ tự để tái dựng lại biển số xe hoàn chỉnh.

**Bước 5: Dọn dẹp**

* Sau khi nhận diện xong, tất cả ảnh tạm được tự động xoá khỏi hệ thống để giữ sạch thư mục và tránh lưu trữ dư thừa.
* Quy trình này đảm bảo:
  + Tự động phát hiện biển số từ ảnh bất kỳ.
  + Nhận diện chính xác từng ký tự với mô hình đã huấn luyện.
  + Dễ mở rộng và tích hợp vào các hệ thống giám sát giao thông hoặc nhận diện biển số thực tế.