BÁO CÁO CÔNG VIỆC

**Công việc số:** 60

**Mô tả công việc:** Tìm hiểu phương án tạo service Python

**Người thực hiện:** Hạ Quang Dũng

**Ngày bắt đầu:** 25/11/2024

**Ngày kết thúc:** 25/11/2024

**NỘI DUNG TÀI LIỆU**

[**Hướng Dẫn Cài Đặt và Quản Lý YOLOService 1**](#_bjubnywzp8uj)

[**1. Giới thiệu 2**](#_chs5fkqbrxqg)

[**2. Mã nguồn 2**](#_pqlapdu9sj28)

[2.1. Windows Service sử dụng Socket 2](#_nxume0f073kd)

[2.2. Windows Service sử dụng API 4](#_o0ia8eauliv)

[2.3. Windows Service sử dụng gRpc 7](#_9ungspbf561)

[**3. Hướng dẫn cài đặt 9**](#_ks4itgn5bcwb)

[3.1: Cài đặt các thư viện cần thiết 9](#_f54k2ix1pxeh)

[3.2: Cài đặt YOLOService 9](#_917gg62mw1xc)

[**4. Quản lý YOLOService 10**](#_v6xpxtvpvknj)

[**5. Tương tác với YOLOService từ C# 10**](#_sdwbssoitia)

[**6. Xử lý lỗi 16**](#_hfgz0z3x1gnx)

[**7. Gỡ bỏ YOLOService 17**](#_wr565yf6ijpo)

[**8. Kết quả 17**](#_gdfee9y3opg5)

[**HƯỚNG DẪN ĐÓNG GÓI VÀ SỬ DỤNG THƯ VIỆN PYTHON 19**](#_yjyw8dvvfaj)

[1. Cấu trúc thư viện 19](#_e2worw6eijx0)

[2. Chi tiết mã nguồn 19](#_w4go8q43aw7u)

[2.1. image\_service.proto 19](#_yrze5mg6b1ya)

[2.2. \_\_init\_\_.py 20](#_36ruiutkpk5b)

[2.3. server.py 20](#_xawzqlaomcip)

[2.4. setup.py 22](#_5m5bls2astn2)

[3. Hướng dẫn đóng gói và cài đặt thư viện 22](#_nczxd0cfr84c)

[4. Sử dụng trên dự án 23](#_pcsouo2m0eqk)

# **Hướng Dẫn Cài Đặt và Quản Lý YOLOService**

# 1. Giới thiệu

YOLOService là một Windows Service được viết bằng Python, sử dụng mô hình YOLO (You Only Look Once) để phát hiện đối tượng trong ảnh. Service này được thiết kế để giao tiếp với các ứng dụng khác (ví dụ: ứng dụng C#), thông qua giao thức socket, giúp xử lý và trả về thông tin bounding box của các đối tượng được phát hiện.

**Tính năng chính:**

* Phát hiện đối tượng trong ảnh sử dụng YOLO.
* Giao tiếp qua socket với các ứng dụng bên ngoài.
* Tương thích với môi trường Windows Service, chạy nền hiệu quả.

## 2. Mã nguồn

### 2.1. Windows Service sử dụng Socket

**File: yolo\_service.py**

| from ultralytics import YOLO import socket import cv2 import numpy as np import json import win32serviceutil import win32service import win32event import servicemanager  # Tải mô hình YOLO model = YOLO('./Model/yolov8n.pt')  # Hàm phát hiện đối tượng def detect\_objects(image):  results = model(image)  detections = []  for result in results:  for box in result.boxes:  x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()  confidence = float(box.conf[0].cpu().numpy())  class\_id = int(box.cls[0].cpu().numpy())  label = model.names[class\_id]  detections.append({  "x": int(x1),  "y": int(y1),  "w": int(x2 - x1),  "h": int(y2 - y1),  "label": label,  "confidence": round(confidence, 2)  })  return detections  # Service class class YOLOWindowsService(win32serviceutil.ServiceFramework):  \_svc\_name\_ = "YOLOService"  \_svc\_display\_name\_ = "YOLO Object Detection Service"  \_svc\_description\_ = "A service for object detection using YOLO."   def \_\_init\_\_(self, args):  win32serviceutil.ServiceFramework.\_\_init\_\_(self, args)  self.hWaitStop = win32event.CreateEvent(None, 0, 0, None)  self.running = True   def SvcStop(self):  self.ReportServiceStatus(win32service.SERVICE\_STOP\_PENDING)  self.running = False  win32event.SetEvent(self.hWaitStop)   def SvcDoRun(self):  servicemanager.LogMsg(  servicemanager.EVENTLOG\_INFORMATION\_TYPE,  servicemanager.PYS\_SERVICE\_STARTED,  (self.\_svc\_name\_, "")  )  self.main()   def main(self):  host, port = 'localhost', 5000  server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  server\_socket.bind((host, port))  server\_socket.listen(1)  while self.running:  try:  conn, addr = server\_socket.accept()  img\_size = int(conn.recv(16).decode())  img\_data = b''  while len(img\_data) < img\_size:  packet = conn.recv(4096)  if not packet:  break  img\_data += packet  np\_data = np.frombuffer(img\_data, np.uint8)  image = cv2.imdecode(np\_data, cv2.IMREAD\_COLOR)  if image is None:  raise ValueError("Invalid image data.")  bounding\_boxes = detect\_objects(image)  result = json.dumps(bounding\_boxes).encode()  conn.sendall(result)  except Exception as e:  print(f"Error: {e}")  finally:  conn.close()  if \_\_name\_\_ == '\_\_main\_\_':  win32serviceutil.HandleCommandLine(YOLOWindowsService) |
| --- |

### 2.2. Windows Service sử dụng API

| import sys import os import time import json import win32serviceutil import win32service import win32event import win32api from flask import Flask, request, jsonify import cv2 import numpy as np from ultralytics import YOLO from threading import Thread  # Tải mô hình YOLO model = YOLO('./Model/yolov8n.pt')  # Tạo ứng dụng Flask app = Flask(\_\_name\_\_)  # Hàm phát hiện đối tượng def detect\_objects(image\_path):  image = cv2.imread(image\_path)  results = model(image)  detections = []   for result in results:  for box in result.boxes:  x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()  confidence = float(box.conf[0].cpu().numpy())  class\_id = int(box.cls[0].cpu().numpy())  label = model.names[class\_id]   detections.append({  "x": int(x1),  "y": int(y1),  "w": int(x2 - x1),  "h": int(y2 - y1),  "label": label,  "confidence": round(confidence, 2)  })  return detections   # API endpoint for object detection @app.route('/detect', methods=['POST']) def detect():  try:  # Lấy thời gian nhận yêu cầu  receive\_time = time.time() # Thời gian nhận (tính bằng giây từ epoch)   # Lấy đường dẫn ảnh từ yêu cầu  image\_path = request.json.get('image\_path')  if not image\_path:  return jsonify({"error": "No image path provided"}), 400   # Kiểm tra ảnh tồn tại  if not os.path.exists(image\_path):  return jsonify({"error": "Image path does not exist"}), 400   # Bắt đầu đo thời gian xử lý  start\_time = time.time()   # Phát hiện đối tượng  detections = detect\_objects(image\_path)   # Kết thúc đo thời gian xử lý  end\_time = time.time()   # Tính toán các mốc thời gian  process\_time = end\_time - start\_time # Thời gian xử lý  receive\_timestamp = time.strftime('%Y-%m-%d %H:%M:%S', time.gmtime(receive\_time))  receive\_timestamp += f".{int((receive\_time - int(receive\_time)) \* 1000):03d}" # Thêm mili giây   # Trả về kết quả dưới dạng JSON  response = {  "detections": detections,  "receive\_time": receive\_timestamp,  "process\_time": process\_time # Không làm tròn  }  return jsonify(response)   except Exception as e:  return jsonify({"error": str(e)}), 500   # Service class để chạy Flask API dưới dạng Windows Service class YoloService(win32serviceutil.ServiceFramework):  \_svc\_name\_ = "YoloService"  \_svc\_display\_name\_ = "YOLO Object Detection Service"  \_svc\_description\_ = "A service to run YOLO object detection using Flask API"    def \_\_init\_\_(self, args):  win32serviceutil.ServiceFramework.\_\_init\_\_(self, args)  self.\_stop\_event = win32event.CreateEvent(None, 0, 0, None)    def SvcStop(self):  self.ReportServiceStatus(win32service.SERVICE\_STOP\_PENDING)  win32event.SetEvent(self.\_stop\_event)   def SvcDoRun(self):  # Khởi tạo Flask API trong một thread riêng để tránh blocking  api\_thread = Thread(target=self.start\_flask)  api\_thread.start()  win32event.WaitForSingleObject(self.\_stop\_event, win32event.INFINITE)   def start\_flask(self):  app.run(host='0.0.0.0', port=5000)  # Chạy service if \_\_name\_\_ == "\_\_main\_\_":  win32serviceutil.HandleCommandLine(YoloService) |
| --- |

### 2.3. Windows Service sử dụng gRpc

| import grpc  import os  import json  from concurrent import futures  import image\_service\_pb2  import image\_service\_pb2\_grpc  import cv2  from ultralytics import YOLO  import time  import win32serviceutil  import win32service  import win32event  import win32api  from threading import Thread  model = YOLO('./Model/yolov8n.pt')  # Hàm phát hiện đối tượng  def detect\_objects(image):  results = model(image)  detections = []  for result in results:  for box in result.boxes:  x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()  confidence = float(box.conf[0].cpu().numpy())  class\_id = int(box.cls[0].cpu().numpy())  label = model.names[class\_id]  detections.append({  "x": int(x1),  "y": int(y1),  "w": int(x2 - x1),  "h": int(y2 - y1),  "label": label,  "confidence": round(confidence, 2)  })  return detections  class ImageTransferServicer(image\_service\_pb2\_grpc.ImageTransferServicer):  def SendImage(self, request, context):  try:  image\_path = request.path    # Kiểm tra file tồn tại  if not os.path.exists(image\_path):  return image\_service\_pb2.ImageResponse(  success=False,  message=f"File not found: {image\_path}"  )    # Đọc ảnh  image = cv2.imread(image\_path)    if image is None:  return image\_service\_pb2.ImageResponse(  success=False,  message=f"Cannot read image: {image\_path}"  )    # Phát hiện đối tượng  detections = detect\_objects(image)    # Chuyển danh sách phát hiện đối tượng thành chuỗi JSON  detections\_json = json.dumps(detections)    return image\_service\_pb2.ImageResponse(  success=True,  message="Detection successful",  data=detections\_json # Trả về dữ liệu JSON  )    except Exception as e:  return image\_service\_pb2.ImageResponse(  success=False,  message=str(e)  )  class GRPCServerService(win32serviceutil.ServiceFramework):  \_svc\_name\_ = "Yolo service - gRPCImageDetectionService"  \_svc\_display\_name\_ = "Yolo service - gRPC Image Detection Service"  \_svc\_description\_ = "Service that runs a gRPC server for image detection using YOLO"  def \_\_init\_\_(self, args):  win32serviceutil.ServiceFramework.\_\_init\_\_(self, args)  self.\_stop\_event = win32event.CreateEvent(None, 0, 0, None)  def SvcStop(self):  self.ReportServiceStatus(win32service.SERVICE\_STOP\_PENDING)  win32event.SetEvent(self.\_stop\_event)  def SvcDoRun(self):  # Khởi tạo gRPC server trong một thread riêng để tránh blocking  grpc\_thread = Thread(target=self.\_run\_grpc\_server)  grpc\_thread.start()  win32event.WaitForSingleObject(self.\_stop\_event, win32event.INFINITE)  def \_run\_grpc\_server(self):  # Tạo server gRPC  server = grpc.server(futures.ThreadPoolExecutor(max\_workers=10))    # Đăng ký service  image\_service\_pb2\_grpc.add\_ImageTransferServicer\_to\_server(  ImageTransferServicer(), server)    # Lắng nghe trên port 50051  server.add\_insecure\_port('[::]:50051')  server.start()    print("gRPC server started, listening on port 50051")    while not self.\_stop\_event.is\_set():  time.sleep(1)  server.stop(0)  print("gRPC server stopped")  if \_\_name\_\_ == '\_\_main\_\_':  win32serviceutil.HandleCommandLine(GRPCServerService) |
| --- |

## 3. Hướng dẫn cài đặt

### 3.1: Cài đặt các thư viện cần thiết

Sử dụng lệnh sau để cài đặt các thư viện cần thiết:

| pip install ultralytics pywin32 opencv-python numpy |
| --- |

### 3.2: Cài đặt YOLOService

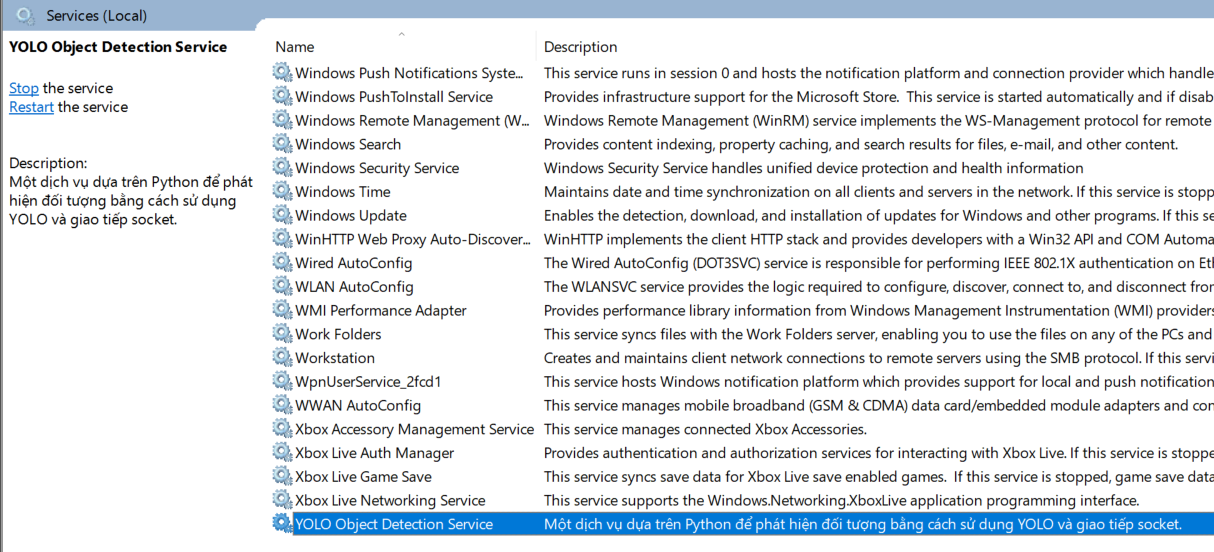
Mở Command Prompt hoặc PowerShell với quyền Administrator.

Điều hướng đến thư mục chứa file yolo\_service.py:

| cd D:\Documents\Work\NganGiang\HAQUANGDUNG\Job60 |
| --- |

Chạy lệnh cài đặt:

| python yolo\_service.py install |
| --- |



## 4. Quản lý YOLOService

| **Lệnh** | **Chức năng** |
| --- | --- |
| python yolo\_service.py start | Khởi động service. |
| python yolo\_service.py stop | Dừng service. |
| python yolo\_service.py restart | Khởi động lại service. |
| python yolo\_service.py remove | Gỡ cài đặt service. |
| python yolo\_service.py debug | Chạy service ở chế độ debug. |

## 5. Tương tác với YOLOService từ C#

Bạn có thể sử dụng socket trong ứng dụng C# để gửi ảnh và nhận thông tin bounding box từ service.

**Mã nguồn C#**

Gửi ảnh tới service:

| using Emgu.CV;  using VehicleDetection.src.CSharp.Models;  using VehicleDetection.src.CSharp.Services;  using System.Net.Sockets;  using System.Text;  using Newtonsoft.Json;  using System.Diagnostics;  using System.IO.Pipes;  namespace VehicleDetection\_8.\_0\_  {  public partial class MainForm : Form  {  #region Thuộc tính và Khởi tạo  private readonly ImageExtractor \_imageExtractor;  private readonly string \_rootDir;  private readonly string \_extractImageFolder;  private string \_videoPath;  private int \_frameSkipQuantity;  private Dictionary<string, DetectionResult> frameTimeExecute = new Dictionary<string, DetectionResult>();  private string host = "127.0.0.1";  private int port = 5000;  private Stopwatch stopwatch = new Stopwatch();  private string \_logFilePath;  public MainForm()  {  InitializeComponent();  \_rootDir = Path.GetFullPath(Path.Combine("..", "..", ".."));  \_extractImageFolder = Path.Combine(\_rootDir, "resources", "Image", "ExtractFromVideo");  \_logFilePath = Path.Combine(\_rootDir, "resources", "Logs", "Log.txt");  \_imageExtractor = new ImageExtractor(\_extractImageFolder);  \_frameSkipQuantity = (int)nmrframeSkip.Value;  }  #endregion  private async void Form1\_Load(object sender, EventArgs e)  {  string scriptPath = Path.Combine(\_rootDir, "src", "Python", "SocketServer.py");  string modelPath = Path.Combine(\_rootDir, "model", "yolov8n.pt");  string outputPath = Path.Combine(\_rootDir, "resources", "Image", "OutputDetection", "vehicle\_image\_detected.jpg");  //var pythonExecutor = new PythonExecutor("python", scriptPath);  //string arr = "";  //var result = pythonExecutor.Execute(arr);  }  #region Chọn File Video  private void btnSelectFile\_Click(object sender, EventArgs e)  {  using (var openFileDialog = new OpenFileDialog  {  Title = "Chọn File Video",  Filter = "Video Files|\*.mp4;\*.avi;\*.mov;\*.mkv|All Files|\*.\*"  })  {  if (openFileDialog.ShowDialog() == DialogResult.OK)  {  \_videoPath = openFileDialog.FileName;  wmpVideo.URL = \_videoPath;  wmpVideo.Ctlcontrols.play();  }  }  }  #endregion  #region Trích xuất Hình ảnh  private void btnExtractImages\_Click(object sender, EventArgs e)  {  if (\_videoPath == null)  {  MessageBox.Show("Vui lòng chọn Video trước khi trích xuất!");  return;  }  stopwatch = Stopwatch.StartNew();  Task.Factory.StartNew(() => ProcessImage());  }  private async Task ProcessImage()  {  ImageExtractor imageExtractor = new ImageExtractor(\_extractImageFolder);  imageExtractor.ExtractImages(\_videoPath, \_frameSkipQuantity);  // Các phần mở rộng của file ảnh mà bạn muốn lấy  string[] imageExtensions = { ".jpg", ".jpeg", ".png", ".bmp", ".gif", ".tiff" };  try  {  DetectionResult detectionResult = new DetectionResult();  // Lấy danh sách file từ thư mục  string[] allFiles = Directory.GetFiles(\_extractImageFolder);  // Lọc file ảnh dựa trên phần mở rộng  foreach (string file in allFiles)  {  stopwatch = Stopwatch.StartNew();  stopwatch.Start();  string apiUrl = "http://localhost:5000/detect";  var results = await SendImagePathToApi(apiUrl, file);  stopwatch.Stop();  // Cập nhật kết quả vào dictionary  foreach (var box in results.boundingBoxes)  {  string label = box.label.ToString();  if (detectionResult.VehicleCounts.ContainsKey(label))  {  detectionResult.VehicleCounts[label]++;  }  else  {  detectionResult.VehicleCounts[label] = 1;  }  }  Invoke(new Action(() => DisplayImageWithBoundingBoxes(file, results.boundingBoxes)));  Invoke(new Action(() => pictureBox1.Refresh()));  detectionResult.TotalTime = (double)stopwatch.Elapsed.TotalSeconds;  detectionResult.TotalVehicles = (int)detectionResult.VehicleCounts.Values.Sum();  UpdateDataGridView(detectionResult.VehicleCounts);  File.Delete(file);  Invoke(new Action(() => DisplayDetectionResult(detectionResult)));  detectionResult.VehicleCounts.Clear();  }  }  catch (Exception ex)  {  Console.WriteLine($"Có lỗi xảy ra: {ex.Message}");  }  }  // Gửi yêu cầu POST đến Flask API  private async Task<(List<dynamic> boundingBoxes, string receiveTime, double processTime)> SendImagePathToApi(string apiUrl, string imagePath)  {  Stopwatch test = new Stopwatch();  test = Stopwatch.StartNew();  using (var client = new HttpClient())  {  test.Start();  var requestData = new { image\_path = imagePath };  string json = JsonConvert.SerializeObject(requestData);  test.Stop();  double time1 = (double)test.Elapsed.TotalSeconds;  DateTime sendTime = DateTime.UtcNow; // Thời điểm gửi yêu cầu (UTC)  Console.WriteLine(sendTime.ToString());  test = Stopwatch.StartNew();  \_logFilePath = Path.Combine(\_rootDir, "resources", "Logs", "Log.txt");  WriteLog(\_logFilePath, $"[CSharp] Send Time (UTC): {sendTime:o}");  test.Start();  var content = new StringContent(json, Encoding.UTF8, "application/json");  HttpResponseMessage response = await client.PostAsync(apiUrl, content);  test.Stop();  double time2 = (double)test.Elapsed.TotalSeconds;  if (response.IsSuccessStatusCode)  {  string responseJson = await response.Content.ReadAsStringAsync();  // Trả về danh sách dynamic để xử lý trực tiếp các trường  //return JsonConvert.DeserializeObject<List<dynamic>>(responseJson);    dynamic result = JsonConvert.DeserializeObject<dynamic>(responseJson);  // Lấy boundingBoxes, receive\_time và process\_time từ phản hồi  var boundingBoxes = JsonConvert.DeserializeObject<List<dynamic>>(result["detections"].ToString());  string receiveTime = result["receive\_time"];  double processTime = result["process\_time"];  WriteLog(\_logFilePath, $"[Python] Receive Time (UTC): {receiveTime}");  WriteLog(\_logFilePath, $"[Python] Process Time (UTC): {processTime}");  WriteLog(\_logFilePath, $"[CSharp] Response received successfully from {apiUrl}. Status: {response.StatusCode}");  return (boundingBoxes, receiveTime, processTime);  }  else  {  throw new Exception($"API Error: {response.StatusCode}");  }  }  }  private void DisplayImageWithBoundingBoxes(string filePath, dynamic boundingBoxes)  {  using (Image img = Image.FromFile(filePath))  {  Bitmap bitmap = new Bitmap(img);  using (Graphics g = Graphics.FromImage(bitmap))  {  Color customColor = ColorTranslator.FromHtml("#33FF66");  using (Pen pen = new Pen(customColor, 2)) // Độ dày là 2  {  foreach (var box in boundingBoxes)  {  int x = box.x, y = box.y, w = box.w, h = box.h;  g.DrawRectangle(pen, x, y, w, h);  g.DrawString(box.label.ToString(), new Font("Arial", 12), new SolidBrush(ColorTranslator.FromHtml("#33FF66")), x, y - 24);  }  }  }  pictureBox1.Image = bitmap;  pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;  }  }  private void nmrframeSkip\_ValueChanged(object sender, EventArgs e)  {  \_frameSkipQuantity = (int)nmrframeSkip.Value;  }  #endregion  #region Cập nhật Giao diện Người dùng (UI)  private void DisplayDetectionResult(DetectionResult detectionResult)  {  if (detectionResult.VehicleCounts == null) return;  lbTotalTime.Text = $"Tổng thời gian thực hiện: {detectionResult.TotalTime} giây";  lbTotalVehicles.Text = $"Tổng số phương tiện: {detectionResult.TotalVehicles}";  dataGridView1.Columns.Clear();  dataGridView1.Columns.Add("VehicleType", "Loại Phương Tiện");  dataGridView1.Columns.Add("Count", "Số Lượng");  dataGridView1.Rows.Clear();  foreach (var result in detectionResult.VehicleCounts)  {  dataGridView1.Rows.Add(result.Key, result.Value);  }  }  private async void UpdateDataGridView(Dictionary<string, int> results)  {  Invoke((Action)(() =>  {  dataGridView1.Columns.Clear();  dataGridView1.Columns.Add("VehicleType", "Loại Phương Tiện");  dataGridView1.Columns.Add("Count", "Số Lượng");  dataGridView1.Rows.Clear();  foreach (var result in results)  {  dataGridView1.Rows.Add(result.Key, result.Value);  }  }));  }  #endregion  #region Ghi Log  private void WriteLog(string filePath, string message)  {  try  {  // Kiểm tra và tạo file nếu chưa tồn tại  if (!File.Exists(filePath))  {  using (var fileStream = File.Create(filePath))  {  // Đảm bảo file được tạo trước khi đóng  }  }  // Ghi log với timestamp  using (StreamWriter writer = new StreamWriter(filePath, true))  {  writer.WriteLine($"[{DateTime.UtcNow:yyyy-MM-dd HH:mm:ss.fff}] {message}");  }  }  catch (Exception ex)  {  Console.WriteLine($"Lỗi khi ghi log: {ex.Message}");  }  }  #endregion  }  } |
| --- |

## 6. Xử lý lỗi

**Access Denied (5)**: Đảm bảo chạy Command Prompt với quyền Administrator.

**Không tìm thấy pythonservice.exe**: Cài đặt lại pywin32:

| python -m pywin32\_postinstall |
| --- |

## 7. Gỡ bỏ YOLOService

Để gỡ bỏ service, chạy lệnh:

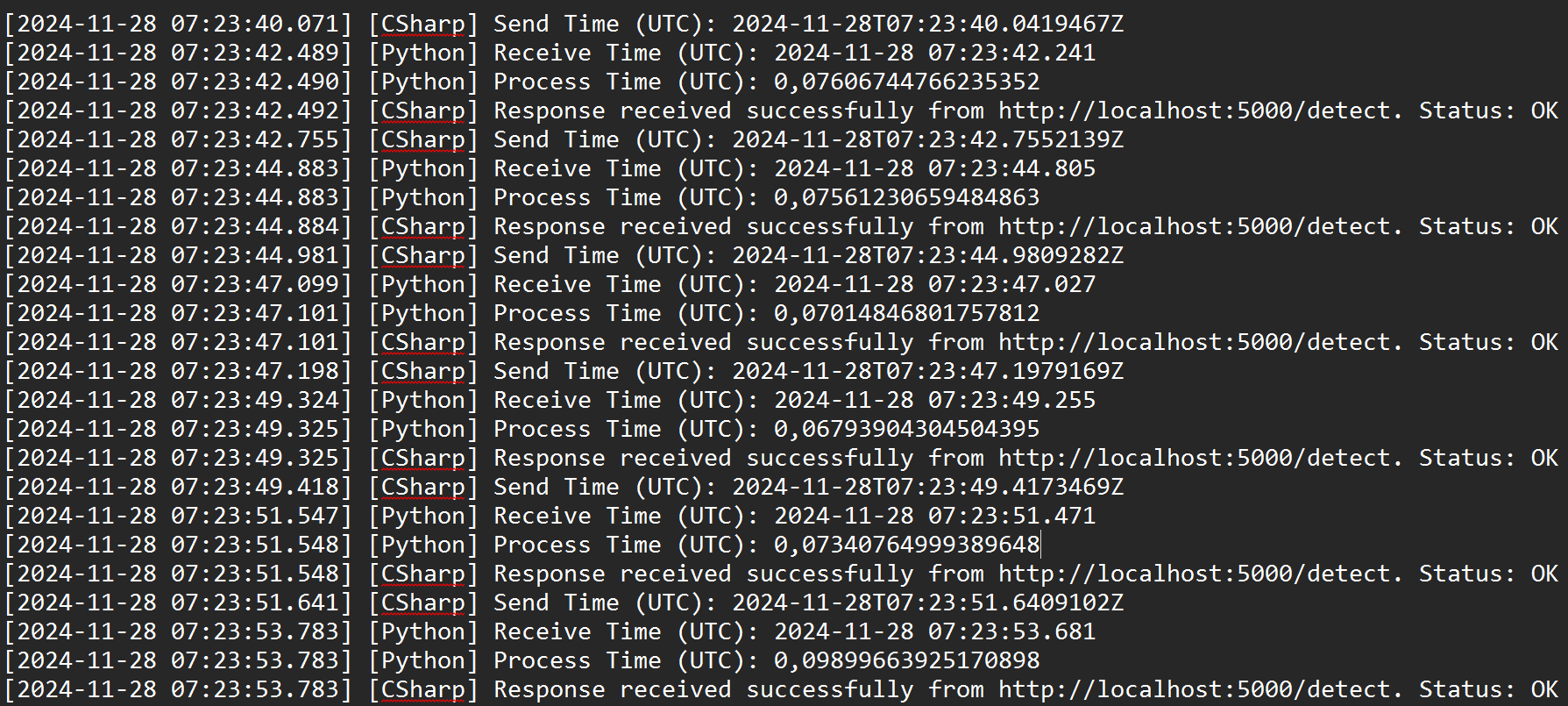
| python yolo\_service.py remove |
| --- |

Nếu không thành công, sử dụng:

| sc delete YOLOService |
| --- |

## 8. Kết quả

Thời gian thực thi nhận diện sử dụng Windows service (Socket) nhanh hơn so với Windows Service (API). Dưới đây là Log của Windows Service API



Thời gian sử dụng với grpc khi chuyển đổi thành Windows service sẽ khiến quá trình gửi dữ liệu bị lâu hơn. Để cải thiện, ta sẽ chuyển đổi cách sử dụng server gRpc như bên dưới đây.

Nhận xét:

| **Tiêu Chí** | **WebSocket** | **HTTP/REST** | **gRPC** |
| --- | --- | --- | --- |
| **Kết Nối** | Kết nối liên tục, hai chiều | Kết nối ngắt quãng, một chiều | Kết nối RPC, nhiều kênh |
| **Tốc Độ** | Rất nhanh, độ trễ thấp | Chậm hơn, độ trễ cao | Rất nhanh |
| **Tính Realtime** | Cao, giao tiếp ngay lập tức | Thấp, yêu cầu polling | Trung bình |
| **Chiều Truyền** | Hai chiều (full-duplex) | Một chiều | Hai chiều |
| **Giao Thức** | Giao thức riêng (ws://, wss://) | HTTP/HTTPS | HTTP/2 |
| **Sử Dụng Tốt Nhất** | Ứng dụng realtime, chat, game | API thông thường | Microservices, hệ thống phân tán |
| **Tài Nguyên** | Hiệu quả, ít tốn băng thông | Nhiều tài nguyên hơn | Hiệu quả |
| **Độ Phức Tạp** | Trung bình | Thấp | Cao |

# HƯỚNG DẪN ĐÓNG GÓI VÀ SỬ DỤNG THƯ VIỆN PYTHON

## 1. Cấu trúc thư viện

| GRPC\_LIBRARY/  ├── yolo\_grpc/ # Thư mục chính chứa mã nguồn  │ ├── \_\_init\_\_.py # File để đánh dấu đây là một gói Python  │ ├── image\_service.proto # Các file mã nguồn của thư viện  │ └── serve.py  ├── setup.py # File cấu hình đóng gói  ├── README.md # Mô tả thư viện |
| --- |

Vì có tệp .proto nằm trong thư mục yolo\_grpc, hãy chạy lệnh protoc từ thư mục gốc của dự án (GRPC\_LIBRARY):

| python -m grpc\_tools.protoc -I./yolo\_grpc --python\_out=./yolo\_grpc --grpc\_python\_out=./yolo\_grpc ./yolo\_grpc/image\_service.proto |
| --- |

**Giải thích lệnh:**

* -I./yolo\_grpc: Đường dẫn đến thư mục chứa tệp .proto (trong trường hợp này là yolo\_grpc).
* --python\_out=./yolo\_grpc: Chỉ định thư mục nơi các file Python sẽ được tạo ra. Đây là thư mục yolo\_grpc.
* --grpc\_python\_out=./yolo\_grpc: Chỉ định thư mục nơi các file Python gRPC sẽ được tạo ra. Đây cũng là thư mục yolo\_grpc.
* ./yolo\_grpc/image\_service.proto: Đường dẫn đến tệp .proto.

## 2. Chi tiết mã nguồn

### 2.1. image\_service.proto

| syntax = "proto3";  service ImageTransfer {  rpc SendImage (ImageRequest) returns (ImageResponse);  }  message ImageRequest {  string path = 1;  }  message ImageResponse {  bool success = 1;  string message = 2;  string data = 3;  } |
| --- |

### 2.2. \_\_init\_\_.py

| from .server import (  ImageTransferServicer,  serve  )  \_\_version\_\_ = "0.1.0"  \_\_all\_\_ = [  'ImageTransferServicer',  'serve'  ] |
| --- |

### 2.3. server.py

| import grpc  from concurrent import futures  from yolo\_grpc import image\_service\_pb2  from yolo\_grpc import image\_service\_pb2\_grpc  import cv2  import os  import numpy as np  from ultralytics import YOLO  import json  model = YOLO('./Model/yolov8n.pt')  # Hàm phát hiện đối tượng  def detect\_objects(image):  results = model(image)  detections = []  for result in results:  for box in result.boxes:  x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()  confidence = float(box.conf[0].cpu().numpy())  class\_id = int(box.cls[0].cpu().numpy())  label = model.names[class\_id]  detections.append({  "x": int(x1),  "y": int(y1),  "w": int(x2 - x1),  "h": int(y2 - y1),  "label": label,  "confidence": round(confidence, 2)  })  return detections  class ImageTransferServicer(image\_service\_pb2\_grpc.ImageTransferServicer):  def SendImage(self, request, context):  try:  image\_path = request.path    # Kiểm tra file tồn tại  if not os.path.exists(image\_path):  return image\_service\_pb2.ImageResponse(  success=False,  message=f"File not found: {image\_path}"  )    # Đọc ảnh  image = cv2.imread(image\_path)    if image is None:  return image\_service\_pb2.ImageResponse(  success=False,  message=f"Cannot read image: {image\_path}"  )    # Phát hiện đối tượng  detections = detect\_objects(image)    # Chuyển danh sách phát hiện đối tượng thành chuỗi JSON  detections\_json = json.dumps(detections)    return image\_service\_pb2.ImageResponse(  success=True,  message="Detection successful",  data=detections\_json # Trả về dữ liệu JSON  )    except Exception as e:  return image\_service\_pb2.ImageResponse(  success=False,  message=str(e)  )  def serve():  server = grpc.server(futures.ThreadPoolExecutor(max\_workers=10))    image\_service\_pb2\_grpc.add\_ImageTransferServicer\_to\_server(  ImageTransferServicer(), server)    # Lắng nghe trên port 50051  server.add\_insecure\_port('[::]:50051')  server.start()    print("gRPC server started, listening on port 50051")  server.wait\_for\_termination() |
| --- |

### 2.4. setup.py

| from setuptools import setup, find\_packages  setup(  name='yolo\_grpc',  version='0.1.0',  packages=find\_packages(include=['yolo\_grpc', 'yolo\_grpc.\*']),  install\_requires=[  'grpcio',  'opencv-python',  'ultralytics',  'numpy',  'torch',  'PyYAML',  ],  author='Ha Quang Dung',  author\_email='dungha.4work@gmail.com',  description='A gRPC service for object detection using YOLO',  long\_description=open('README.md', encoding='utf-8').read(),  long\_description\_content\_type='text/markdown',  url='https://github.com/<your-github-username>/yolo\_grpc',  classifiers=[  'Programming Language :: Python :: 3',  'License :: OSI Approved :: MIT License',  'Operating System :: OS Independent',  ],  python\_requires='>=3.7',  ) |
| --- |

## 3. Hướng dẫn đóng gói và cài đặt thư viện

Chạy lệnh sau tại thư mục có file setup.py

| python -m build |
| --- |

Sau khi chạy lệnh này, sẽ có thư mục **dist** và thư mục **yolo\_grpc.egg-info** được sinh ra. Tiếp tục cài đặt thư viện bằng lệnh sau:

| pip install dist/yolo\_grpc-0.1.0-py3-none-any.whl |
| --- |

Như vậy thư viện đã được cài đặt thành công

## 4. Sử dụng trên dự án

Tạo file GRPCServer.py

| from yolo\_grpc import serve  serve() |
| --- |

Khi mã nguồn sau đây vào sự kiện loadForm của chương trình Winform bên C#

| string scriptPath = Path.Combine(\_rootDir, "src", "Python", "GRPCServer.py");  string modelPath = Path.Combine(\_rootDir, "model", "yolov8n.pt");  string outputPath = Path.Combine(\_rootDir, "resources", "Image", "OutputDetection", "vehicle\_image\_detected.jpg");  pythonExecutor = new PythonExecutor("python", scriptPath);  pythonExecutor.Execute(""); |
| --- |