Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Програмний засіб розпізнавання автомобільних номерів для комп'ютерної системи відеоспостереження»

08-54.БКР.017.00.000 ПЗ

Виконав студент 2 курсу, групи 1КІ-23мс
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма — «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)
Солодюк В.П.
(прізвище та ініціали)
Керівник д.т.н., проф. каф. ОТ
Мартинюк Т.Б.
(прізвище та ініціали)
Рецензент д.т.н., проф. каф. ПЗ
Романюк О. Н.
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.

«11» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Галузь знань — 12 Інформаційні технології
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф._____ Азаров О. Д.
« 21 » березня 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Солодюку Віталію Петровичу

1 Тема роботи: «Програмний засіб розпізнавання автомобільних номерів для комп'ютерної системи відеоспостереження», керівник роботи Мартинюк Тетяна Борисівна, д.т.н., професор кафедри ОТ, затверджено наказом вищого навчального закладу від «20» березня 2025 року № 97.

2 Строк подання студентом роботи 13.05.2025 р.

3 Вихідні дані до роботи: документація OpenCV, документація YOLO, набір даних для навчання моделі, технології розробки: мова програмування Python, бібліотека для роботи з базами даних SQLAlchemy.

4 Зміст пояснювальної записки: вступ; аналіз методів та засобів розпізнавання номерних знаків автомобілів; розробка способу та послідовності розпізнавання номерних знаків автомобілів; розробка програмного засобу; тестування програмного засобу; висновки.

5 Перелік графічного матеріалу: алгоритм роботи програмного засобу.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завда прий
1-4	д.т.н., проф. каф. ОТ Мартинюк Т.Б.	21.03.2025	13.05.
Нормоконтроль	асистент каф. ОТ Швець С. І.	14.05.2025	30.05.

7 Дата видачі завдання 21.03.2025 р.

8 Календарний план приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання	Прим
1.	Аналіз завдання	21.03.2025 — 23.03.2025	вик.
2.	Аналіз предметної області	24.03.2025 — 29.03.2025	вик.
3.	Аналіз та вибір методів локалізації та сегментації номерних знаків	30.03.2025 — 5.04.2025	вик.
4.	Вибір технологій та інструментів для розробки	6.04.2025 — 13.04.2025	вик.
5.	Розробка програмного засобу	14.04.2025 — 26.04.2025	вик.
6.	Тестування програмного засобу	27.04.2025 — 3.05.2025	вик.
7.	Оформлення пояснювальної записки та ілюстративного матеріалу	4.05.2025 — 12.05.2025	вик.
8.	Попередній захист БКР та усунення недоліків	13.05.2025	вик.

Студент Віталій СОЛОДЮК

Керівник роботи д.т.н., проф. Тетяна МАРТИНЮК

АНОТАЦІЯ

УДК 004.932

Солодюк В.П. Програмний засіб розпізнавання автомобільних номерів для комп’ютерної системи відеоспостереження. Бакалаврська кваліфікаційна робота зі спеціальності 123 «Комп’ютерна інженерія»; освітня програма «Комп’ютерна інженерія». Вінниця: ВНТУ, 2025. 98 с.

На укр. мові. Бібліогр.: 23 назв; рис.:25; табл. 3.

Бакалаврська робота присвячена розробці програмного засобу для автоматичного розпізнавання автомобільних номерних знаків.

У роботі було виконано огляд та аналіз сучасних методів і алгоритмів локалізації та сегментації об’єктів на зображенні. Було використано гібридний підхід, що об’єднує можливості спеціально навченої нейронної мережі, архітектура якої є різновидом згорткових нейронних мережах (CNN), і класичних методів комп’ютерного зору та обробки зображення. Результатом роботи є повнофункціональний програмний засіб, здатний ефективно виконувати завдання локалізації та розпізнавання автомобільних номерів у реальному часі з високою точністю.

Ключові слова: комп’ютерний зір, сегментація, нейронна мережа, глибоке машинне навчання, розпізнавання символів, розпізнавання номерних знаків.

ABSTRACT

UDC 004.932

Solodyuk V.P. Software Tool for License Plate Recognition in a Computer-Based Video Surveillance System. Bachelor's Qualification Thesis in Specialty 123 «Computer Engineering»; Educational Program "Computer Engineering". Vinnytsia: VNTU, 2025. 98 p.

In Ukrainian. Bibliography: 23 sources; Illustrations: 25; Tables: 3.

This bachelor's thesis is dedicated to the development of a software tool for automatic license plate recognition.

The work involved a comprehensive review and analysis of modern methods and algorithms for object localization and segmentation in images. A hybrid approach was employed, combining the capabilities of a specially trained neural network—based on a Convolutional Neural Network (CNN) architecture—with classical computer vision and image processing techniques. The outcome of the project is a fully functional software tool capable of effectively performing real-time vehicle license plate localization and recognition with high accuracy.

Keywords: computer vision, segmentation, neural network, deep learning, character recognition, license plate recognition.

ЗМІСТ

ВСТУП

Невід’ємним елементом сучасного життя є транспортні засоби, які оточують нас усюди, і тенденція по збільшенню їх чисельності лише набирає темпів, що робить ручне відстеження та контроль за ними надзвичайно складною задачею. До переліку задач, які потребують вирішення в цьому контексті, належать організація ефективного керування дорожнім рухом, керуванням системи доступу на в’їздах до охоронюваних об’єктів та закритих територій, відстеження викрадених транспортних засобів, ідентифікації автомобілів які були помічені при скоєнні злочинів та контроль за дотриманням правил дорожнього руху і виявлення правопорушень. Саме тому існує постійна та зростаюча потреба в залученні до виконання цих функцій спеціалізованих автоматизованих систем, здатних виконувати розпізнавання номерних знаків та надавати доступ до зібраних даних для ефективного виконання вищезазначених задач. Використання цих систем дозволяє автоматизувати процеси та дії, які раніше вимагали безпосередньої участі людини, такі як постійний візуальний контроль чи ручне введення даних.

Слід зазначити, що системи розпізнавання номерних знаків існують досить давно [1] і пройшли складний багатоетапний шлях від використання традиційних методів обробки зображення до більш спеціалізованих інструментів комп'ютерного зору [2], далі до машинного навчання, а тепер — до глибокого навчання. І все ж незважаючи на значний прогрес в ефективності їх роботи, існуючі рішення все ще стикаються з низкою проблем та обмежень, а алгоритми та методи, що лежать в їх основі не завжди демонструють потрібну ефективність. Слід врахувати і досить високу вартість комерційних варіантів таких систем, що не сприяє достатньому впровадженню їх усюди де існує в них потреба. З огляду на вищезазначене можна чітко стверджувати, що даний клас систем має ще досить значний потенціал до покращення як в плані вдосконалення вже існуючих алгоритмів та методів так і в створенні більш фінансово доступних аналогів, здатних конкурувати з дорожчими варіантами.

Мета даної дипломної роботи є розробка та вдосконалення методів виділення

та розпізнавання автомобільних номерних знаків із застосуванням сучасних підходів, зокрема згорткових нейронних мереж (CNN), їх модифікацій та інструментів оптичного розпізнавання символів (OCR).

Задачі дослідження бакалаврської роботи:

— зробити аналіз сучасних підходів та технологій до розпізнавання автомобільних номерних знаків з зображень.

— дослідити існуючі на даний час алгоритми та методи виділення та розпізнавання автомобільних номерів, обрати найбільш ефективні з них.

— побудувати на їх основі або реалізувати нові, які б стали вдосконаленою основою для побудови ефективної системи розпізнавання автомобільних номерних знаків.

— реалізація теоретичних та практичних напрацювань у програмному засобі, що становить цілісну систему, придатну для практичного застосування.

Об’єкт дослідження — процес отримання даних реєстраційного номеру автомобілів шляхом знаходження номерного знаку на зображенні та розпізнавання символів у виділеній частині зображення з ним.

Предмет дослідження — методи оброблення цифрового зображення для виокремлення та розпізнавання символів у реєстраційному номеру автомобілів.

Практичне значення одержаних результатів — створено програмний засіб для розпізнавання номерних знаків, який здійснює виокремлення та ідентифікацію символів на зображеннях. Результати роботи зберігаються у форматі, що легко інтегрується з іншими системами.

Особистий внесок здобувача — усі результати, викладені у бакалаврській дипломній роботі, отримані автором особисто.

Апробація результатів бакалаврської роботи — було опубліковано в матеріалах LIV Всеукраїнської науково-технічній конференції підрозділів Вінницького національного технічного університету (2025) [3].

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ АВТОМОБІЛІВ

1.1 Історія появи систем розпізнавання автомобільних номерів

Розпізнавання автомобільних номерів – це автоматизована технологія, в основі якої лежить пошук та оптичне розпізнавання символів на зображенні, а також збереження отриманих фото та текстових даних у форматі, який дозволяє їх практичне використання. Перші такі системи з’явилися ще в 1976 році у Великобританії, а її творцем вважається Дуглас Слінгер. Система виявилася достатньо ефективною щоб дозволити налагодити її промислове виробництво. Спочатку випуском займалася EMI Electronics, а згодом права на випуск перейшли до Computer Recognition System. Сучасні системи розпізнавання автомобільних номерів набули широкого використання у правоохоронних структур у всьому світі. Також їх використовують для виконання адміністративних функцій, таких як верифікація реєстрації транспортних засобів та наявність відповідних ліцензій, попередження заторів, забезпечення безпеки, систем паркування тощо.

У своїй роботі системи розпізнавання автомобільних номерів використовують низку специфічних алгоритмів та методів для обробки зображення, щоб виявити, нормалізувати та покращити зображення номерного знаку. Процес розпізнавання ускладнює наявність великого різноманіття шрифтів, розмірів та колірних схем, кількість символів, а також розташування тексту, яке може бути як однорядковим, так і дворядковим. Тому для досягнення високої точності розпізнавання АНР часто підлягають індивідуальній адаптації під конкретні умови експлуатації чи країну.

Прикладом того, як дизайн номерних знаків впливає на системи розпізнавання, може слугувати зміна стилю голландських номерних знаків у 2002 році. Тоді було змінено шрифт у якому додали пропуски в деяких літерах, таких як P та R, що покращило чіткість та читабельність символів. В Україні теж відбулися реформи у структурі номерних знаків. Так, у 2015 році набули чинності нові стандарти державних реєстраційних номерних знаків: введено новий шрифт, що

мав більш чіткі геометричні контури, збільшено міжсимвольні інтервали та мінімізовано візуальну спорідненість між деякими символами. Іншою зміною було вдосконалення світловідбивного покриття, що помітно покращило видимість знаку для камери при поганому освітленні та вночі. Все ці зміни позитивним чином вплинула на ефективність розпізнавання номерних знаків.

1.2 Аналіз методів локалізації та сегментації номерних знаків

Для ефективного вирішення задач, пов’язаних з локалізацією та сегментацією номерних знаків найбільш доцільно застосувати методи машинного зору. Ці методи забезпечують ефективне та точне розпізнавання різних об’єктів на зображенні, тому добре підходять для вирішення нашої задачі.

Залежно від принципів обробки зображень та підходів до аналізу візуальної інформації, методи машинного зору можна поділити на кілька категорій:

Класичні методи машинного зору — виникли досить давно, ще у другій половині минулого століття. Це набір математичних алгоритмів, який дозволяв виконувати такі операції як класифікація, регресія та кластеризація. Основу цих підходів складають такі алгоритми як порогова обробка, гістограмна рівномірність, методи Канні, Собеля.

Переваги:

— прості в реалізації;

— невелика обчислювальна складність.

Недоліки:

— низька адаптивність до шумів, нерівномірного освітлення, перекриття;

— ручне налаштування параметрів;

— погано справляються розпізнаванні складних об’єктів.

Більш прогресивний спосіб вирішення задач пов’язаних з машинним зором передбачає використання машинного або глибокого навчання.

Машинне навчання [4] передбачає процес навчання певного алгоритму чи моделі на наборі даних великих обсягів. В процесі відбувається виявлення закономірностей та взаємозв’язків. Результатом такого навчання є модель здатна

виконувати класифікацію об’єктів, прогнозування результатів та інше.

В свою чергу навчання буває двох типів:

— кероване, у якому вихідні данні є «сумішшю» даних кількох типів кожному з яких присвоєно унікальний клас. Модель при аналізі вхідних даних враховує інформацію про клас, що дозволяє досить якісно знаходити відмінності між об’єктами різних класів.

— некероване, у якому дані не мають додаткових класифікаторів на зразок міток чи класів, що спонукає алгоритм машинного навчання більш шукати характерні ознаки та закономірності у об’єктів самостійно.

Переваги методу:

— обробка великих обсягів даних;

— автоматизація рутинних завдань;

— підвищення точності прогнозування.

Недоліки методу:

— залежність якості від кількості даних;

— витрати та труднощі при підготовці даних;

— висока вартість та ресурсоємність впровадження.

Глибоке навчання — є підмножиною машинного навчання [5], але з деякими суттєвими відмінностями головна з яких це використання штучних нейронних мереж з великою кількістю шарів. Такий підхід направлений на спробу імітації будови мозку живих істот. Нейронні мережі здатні набагато якісніше проводити автоматичне виокремлення більш тонких ознак та складніших закономірностей що проявляється у вищій продуктивності та точності відносно алгоритмів машинного навчання.

Переваги методу:

— автоматичне вилучення ознак;

— висока ефективність на складних даних;

— виявлення більш тонких закономірностей.

Недоліки методу:

— потреба у великому навчальному наборі даних;

— висока обчислювальна складність.

1.3 Аналіз існуючих систем розпізнавання номерних знаків

Розвиток сучасної мікроелектроніки, подальше зменшення технологічних норм виробництва, а також широке поширення ARM-архітектури та накопичувачів на основі флеш-пам’яті відкривають можливості для створення компактних, автономних та енергоефективних систем. Такі пристрої володіють достатньою обчислювальною потужністю для виконання складних завдань, зокрема в галузях обробки зображень, машинного навчання, Інтернету речей (IoT), робототехніки та вбудованих систем. Завдяки низькому енергоспоживанню, високій продуктивності та можливості інтеграції різноманітних сенсорів і бездротових модулів, ці системи можуть досить ефективно застосовуватися у системах відеоспостереження та розпізнаванні об’єктів.

Застосування такі пристрої знаходять в задачах організації безпеки на різних об’єктах, таких як торгівельні центри, парковки, будівельні майданчики, у громадських місцях для моніторингу порядку, тощо.

В Україні системи відеоспостереження широко використовуються органами правопорядку, юридичними та фізичними особами, а також органами місцевого самоврядування. Масштаб таких систем останніми роками стрімко зростає, значною мірою завдяки глобальному поширенню інтернету, який виконує роль комунікаційного каналу. Це дозволяє об’єднати велику кількість камер у єдину систему з централізованим керуванням і можливістю тривалого зберігання відеофайлів, яке зазвичай становить щонайменше місяць.

Сучасні комерційні системи розпізнавання номерних знаків, можна поділити на кілька різновидів: окремі аналогові системи, спеціалізовані LPR камери

Апаратні рішення, що функціонують як окремі аналогові системи на основі розумних IP-камер, які можуть бути додатково інтегровані у кіберфізичні системи, до яких належать шлагбауми, світлофори, інформаційні табло. Такі системи умовно можна поділити на дів категорії:

— спеціалізовані LPR камери (рис. 1.1);

— програмне забезпечення для встановлення на відповідні IP-камери.

Рисунок 1.1 — LPR-камера ZIP-5241DLM-X12CP

Це LPR (License Plate Recognition) IP-камера, з реалізованою на апаратному рівні функцією розпізнавання номерних знаків [6]. В її функції входить не лише пошук та виділення сегменту з номерним знаком на зображенні, а й оптичне розпізнавання символів. Камера надає по веб-інтерфейсу доступ до своїх налаштувань де можна задати такі опції як регіон номерного знаку, регіон на кадрі в межах якого буде здійснюватися пошук, ступінь впевненості що отримане зображення є номерним знаком (рис. 1.2).

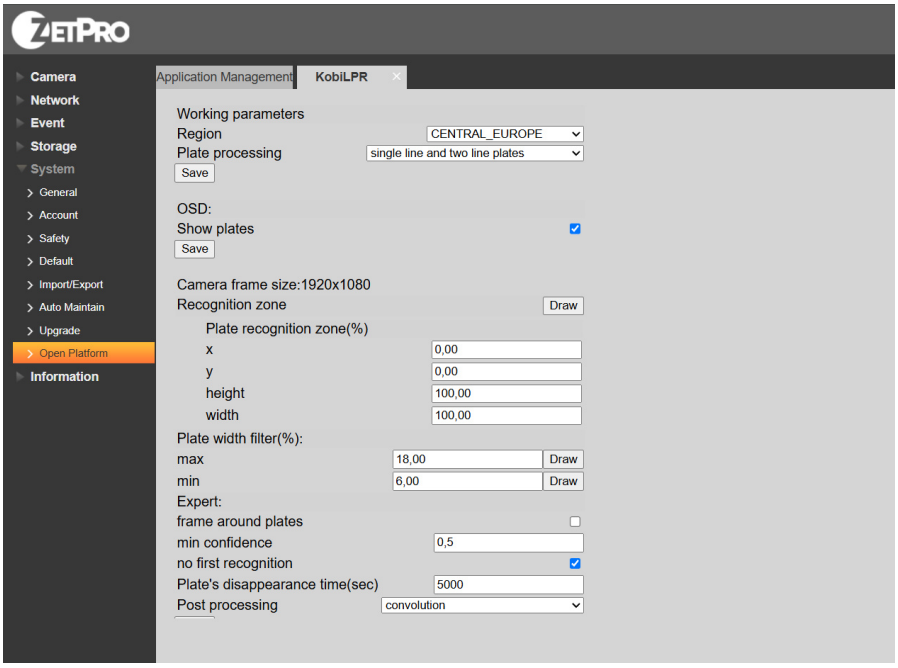


Рисунок 1.2 — Меню налаштування розпізнавання LPR камери

Також камера обладнана аналітичним модулем, здатний аналізувати додаткові атрибути автомобіля, такі як марка, модель чи колір. Має функцію

адаптивного ІЧ-підсвічування, що допомагає ефективно виявляти номерні знаки в вечірній та нічний час на відстані до 50 метрів. Вагомим недоліком є ціна, яка станом на 2025 рік становить 74 тисячі гривень. Це помітно обмежує доступність для широкого кола потенційних користувачів.

Після обробки інформація пересилаються на сервер, де спеціалізоване програмне забезпечення ZetPro VMS (рис. 1.3) зберігає отримані дані.

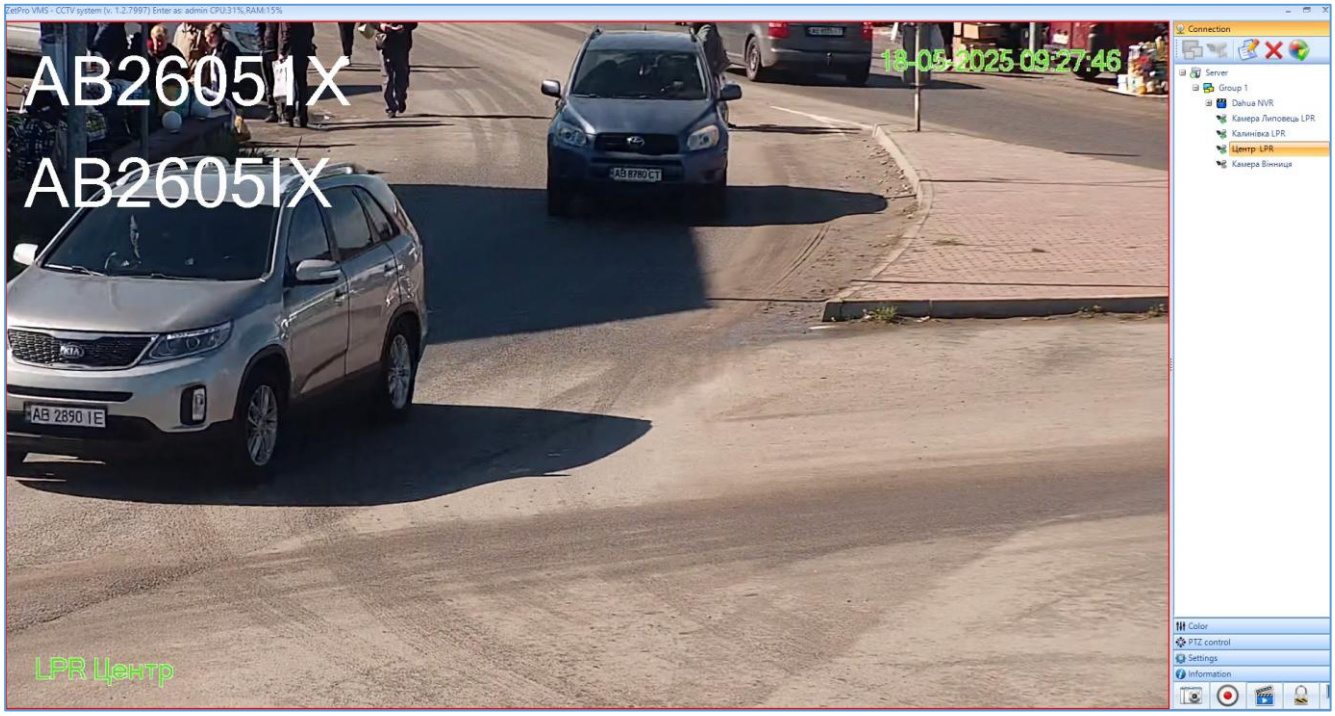


Рисунок 1.3 — Інтерфейс програми ZetPro VMS

Програмні рішення, які реалізуються у вигляді програмного забезпечення для встановлення на відповідні IP-камери та використовують такі сервери баз даних, як SQLite, Firebird тощо.

В таких системах можуть використовуватися як спеціалізовані, так і звичайні камери, що надсилають відеодані на сервер і всі потрібні функції виконує спеціалізоване програмне забезпечення. Перевагами таких систем є можливість використання звичайних камер, що дозволяє їх інтегрувати у вже існуючу інфраструктуру відеоспостереження.

Прикладом такого програмного рішення є Milestone XProtect LPR (рис. 1.4).

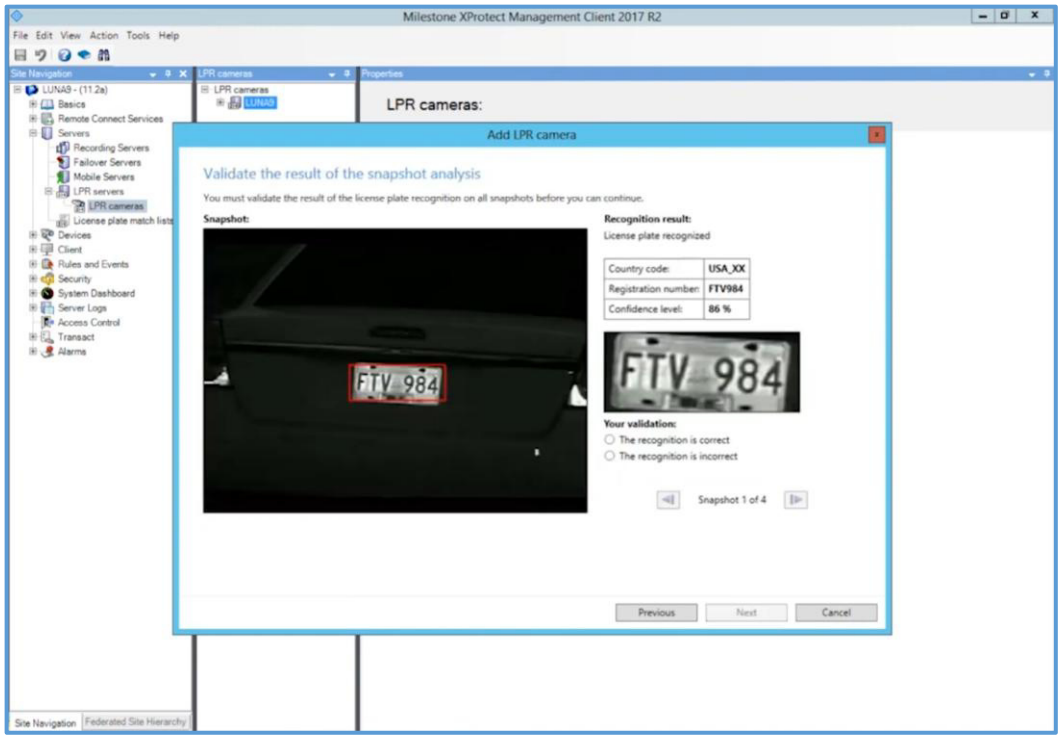


Рисунок 1.4 — Вікно програмного додатку Milestone XProtect LPR

Часто до складу такого програмного забезпечення включено спектр функцій, які можуть бути незатребуваними, але все одно потребують оплати. Оскільки всі обчислення та обробка даних виконуються на виділеному сервері це висуває значні вимоги до його продуктивності.

1.4 Постановка задачі розробки програмного засобу

Проаналізувавши переваги та недоліки існуючих програмних та апаратних рішень для розпізнавання номерних знаків було виявлено ряд важливих аспектів, які потребують удосконалення. На основі цього аналізу було сформовано набір задач у вигляді вимог до функцій, які мають бути вирішені в процесі розробки програмного засобу. Ці вимоги можна розділити на кілька категорії:

Технічні вимоги:

— Висока стабільність та надійність роботи на протязі тривалого періоду експлуатації;

— коректна обробка виняткових та нестандартних ситуацій без переривання

роботи програмного засобу;

— підтримка гнучкого конфігурування ключових параметрів системи;

— забезпечення одночасного доступу до інструментів системи кільком користувачам;

— оптимізація продуктивності системи на різних етапах обробки даних;

— реалізація механізму автоматичного створення резервних копій даних.

Інтерфейс користувача:

— розробка зручного та інтуїтивно зрозумілого дизайну інтерфейсу;

— можливість гнучкого задання критеріїв пошуку номерних знаків;

— можливість перегляду результатів пошуку та розпізнавання у зручній для користувача формі;

— вивантаження результатів.

Підтримка та обслуговування:

— технічна простота встановлення програмного засобу на цільовій робочій системі;

— надання високоінформативних службових повідомлень для моніторингу.

Реалізація перелічених задач дозволить створити програмний засіб високої якості, що не тільки задовольнить існуючі потреби в системах розпізнавання номерних знаків автомобілів, але й допоможе усунути ключові недоліки існуючих рішень що зрештою робить його ефективним інструментом для широкого кола застосувань.

У першому розділі було розглянуто історію виникнення та розвитку систем автоматичного розпізнавання номерних знаків і проаналізовано методи локалізації та сегментації об’єктів. Проведений аналіз дозволив встановити переваги та недоліки кожного з методів і надати перевагу підходу, що базується на використанні глибокого навчання. Додатково було проаналізовано реальні сучасні програмні та апаратні рішення, що дозволило виділити їх сильні та слабкі сторони. Отримані результати дають достатньо матеріалу для формування базових вимог та принципів роботи власної системи розпізнавання номерних знаків.

2 РОЗРОБКА ПОСЛІДОВНОСТІ ТА ЗАСОБІВ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ АВТОМОБІЛІВ

2.1 Розробка методу локалізації та виділення автомобільних номерів

Вирішення задачі по локалізації та виділенню автомобільних номерних знаків є досить нетривіальним завданням у зв’язку з присутністю великої кількості зовнішніх факторів, які тим чи іншим чином ускладнюють її розв’язання. Насамперед, це різні умови освітлення: у вечірню пору погане освітлення призводить до появи шумів на зображенні та зменшує його загальну контрастність. Удень сонячні промені можуть створювати на номерному знаку тіні та пересвічені області, що також ускладнюють його аналіз. Несприятливі погодні умови, такі як дощ, сніг, туман та пил, різною мірою перешкоджають видимості номерного знаку та викликають його спотворення. Сюди ще можна додати забруднення поверхні знаку, його механічні пошкодження чи наявність сторонніх предметів, що перекривають частину символів. Усе це вимагає застосування складних та адаптивних алгоритмів, і навіть у цьому випадку не можна бути повністю впевненим щодо якості отриманого результату розпізнавання.

У процесі розробки програмного засобу було розглянуто набір методів, які є найбільш перспективними та придатними для розв’язання задачі локалізації та розпізнавання номерного знаку та відібрано серед них найбільш підходящі для реалізації проекту.

Незважаючи на велике різноманіття програмних та апаратних засобів, які виконують процедуру розпізнавання номерних знаків, в основі їхньої роботи лежать дуже подібні алгоритми. Відмінності полягають лише в індивідуальних реалізаціях їх конкретних підпунктів. Сам алгоритм обов’язково передбачає наявність таких етапів:

— попередня обробка зображення (препроцесинг);
— та виділення номерного знаку (сегментація);
— вирівнювання сегменту з номерним знаком (нормалізація).

Розглянемо призначення кожного з цих пунктів а також набір методів, які

підходять для його реалізації:

Попередня обробка зображення — це процес, що може включати кілька етапів проміжної обробки зображення кожен з яких призначений для усунення різного типу артефактів, коригування гами, контрасту.

Одна частина методів потребує людського втручання для налаштування параметрів, це інтерактивні методи покращення зображення. Як правило, вони включають налаштування різноманітних фільтрів постпроцесингу та доступні безпосередньо на відеокамерах, де оператор може регулювати такі параметри, як масштаб, фокусна відстань, витримка затвора, яскравість, контрастність та баланс білого.

Інші методи зосереджені на роботі в автоматичному режимі і використовують алгоритми адаптивної обробки що коригують характеристики зображення без людського втручання але все ж дають можливість задання певних ключових робочих параметрів. До таких методів належить адаптивне посилення контрасту (CHALE) та корекція експозиції. Також активно застосовується зниження шуму на основі згладжування (розмиття по Гауссу, медіанна та білатеральна фільтрація), підсилення контурів та алгоритм Лапласа.

Компресія зображення хоч і не є методом у прямому розумінні цього терміну, але все ж впливає на якість зображення, тому доцільно включити її до цього списку. Вона призначена для зменшення обсягу даних, необхідного для представлення зображення, що дозволяє оптимізувати його зберігання та передачу. Використання цього механізму продиктоване технічними обмеженнями каналів комунікації між джерелом зображення (камерою) та пристроєм фіксації або збереження відеоданих. Для порівняння, відео у роздільній здатності 1920×1080 пікселів (Full HD) без застосування компресії вимагає приблизно 3 Гбіт/с пропускної здатності при частоті 30 кадрів за секунду та з 8-бітною глибиною кольору. Це робить некомпресоване відео непридатним для передачі і тим більше для зберігання. Також подальша обробка таких відеоданих буде займати суттєво більше часу, тому на практиці завжди застосовують компресію. Методи стиснення без втрат зберігають усі деталі зображення, але забезпечують відносно низький рівень

стиснення. Натомість методи стиснення з втратами значно зменшують розмір даних, використовуючи алгоритми усунення надлишкової інформації, що призводить до зниження якості зображення, і при низькому бітрейту може додавати до відеоданих досить помітні артефакти, особливо у нічний час.

Сегментація — метою цього процесу є розділення зображення на набір незалежних областей які мають між собою візуальні відмінності, відрізняються деякими характеристиками чи властивостями, такими як колір або текстура, рівень яскравості. Розглянемо доступні методи сегментації:

Метод порогового сегментування — це найпростіший метод сегментації зображень, коли з зображення у градаціях сірого перетворюють на бінарне зображення. Суть його полягає в заміні кожного пікселя зображення чорним, якщо його значення менше певного, заданого наперед, порогового значення. Або навпаки — якщо значення пікселя більше за порогове значення, піксель стає білим. Даний метод має кілька варіацій, які відрізняються алгоритмом вибору порогового значення. До переваг цього методу належить його простота реалізації та невисока обчислювальна складність, що знижує вимоги до обчислювальних ресурсів. Недоліком є низька адаптивність до зображень з нерівномірним освітленням або складною структурою. Також вибір порогового значення — як вручну, так і за допомогою алгоритмів, далеко не завжди забезпечує потрібний результат. Приклад роботи даного методу зображено на рис. 2.1.

Рисунок 2.1 — Приклад застосування методу порогового сегментування

Метод кластеризації — суть полягає в поділі зображення на кластери (неперетинні групи) пікселів, що мають схожі характеристики. Одним з найбільш уживаних алгоритмів кластеризації є K-means, який є простим алгоритмом машинного навчання без нагляду. Суть цього методу полягає в розподілі простору зображення на k груп пікселів, які представляють k-центроїдів. Далі кожен піксель відноситься до певної групи на основі відстані між ним та відповідним центроїдом. Після завершення початкового розподілу відбувається оновлення центрів кластерів. Нові центроїди обчислюються як середнє значення всіх пікселів, що належать кожному кластеру. Потім знову відбувається призначення пікселів до найближчого центроїда, що викликає його оновлення з повторним обчисленням середнього значення пікселів у кожному кластері. Цей процес триває доти, доки центроїди перестануть змінювати своє положення або поки кількість ітерацій не досягне заданого критичного значення. На виході маємо сегментоване зображення, де кожен кластер представляє окремий колір або текстуру.

Перевагами методу є простота реалізації та ефективна робота у випадках, коли об’єкти на зображенні мають добре виражені відмінності в кольорі.

Недоліком є залежність результату від початкового вибору координат центрів кластерів що ускладнює автоматизацію використання даного методу.

Метод графового розрізу — даний метод базується на представлені зображення у вигляді графа, де кожен піксель відповідає вершині, а зв’язки між пікселями є ребрами, що мають певну вагу, і її значення означає ступінь подібності між пікселями. Наприклад у якості подібності пікселів може виступати колір чи яскравість, і чим більша схожість між двома групою пікселів, тим більше значення ваги мають їхні зв’язки. Додатково до графа вводяться два спеціальні вузли – джерело та стік, перший асоціюється з самим об’єктом а другий з фоном. Кожен піксель також з’єднується або з джерелом, або з стоком залежно від ймовірності того до якого типу він належить. Алгоритм працює за принципом пошуку такого розрізу в графі, який дозволяє мінімізувати сумарну вагу зв’язків, що розриваються між пікселями різних кластерів. Таким чином досягається оптимальний поділ зображення на сегменти — об’єкт та фон. У більшості випадків метод реалізується

в інтерактивному режимі: користувач вручну задає приклади пікселів, які належать до фону та об’єкта, а далі система автоматично виконує поділ.

До переваг методу можна віднести його високу точність, особливо при роботі зі складними зображеннями, де об’єкти мають нечіткі межі або складну текстуру. Метод також є гнучким, оскільки дозволяє враховувати різноманітні характеристики зображення, зокрема яскравість, колір, контрастність або текстуру.

До його недоліків належать висока обчислювальна складність, яка нелінійно зростає зі збільшенням розміру зображення, а також потреба вручну задавати зразки фону та об’єкта. Без цього ефективність роботи алгоритму помітно знижується.

Метод виявлення країв Кенні [7] є одним з популярніших алгоритмом для виявлення країв на зображенні. Він був створений у 1986 році Джоном Кенні, як помітно точніша заміна детектору Собеля. В процесі роботи алгоритм застосовує фільтр Гауса для розмиття, і таким чином зменшення шуму. Потім він виявляє місця найбільших перепадів яскравості, а точніше — обчислює, наскільки і в якому напрямку) змінюється яскравість у кожній точці зображення. Приклад використання цього методу наведено на рисунку 2.2.

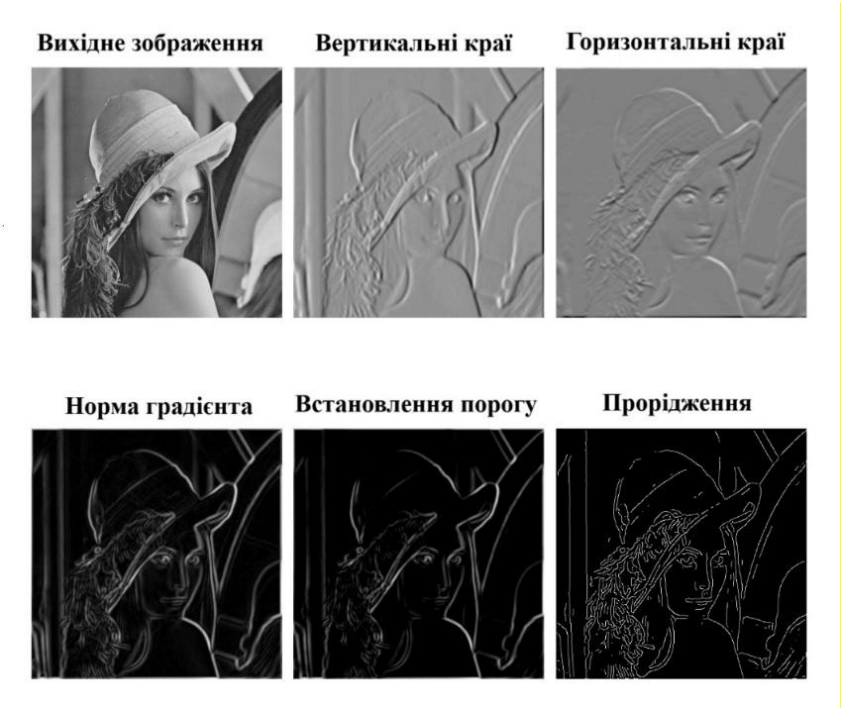


Рисунок 2.2 — Приклад застосування методу Кенні

Таким чином знаходяться лінії країв на межі максимальної зміни яскравості, враховуючи напрямок градієнту.

Сегментація зображення з використанням нейронних мереж є найсучасніший та найефективніший на сьогодні підхід для цього класу завдань. Серед усього різноманіття доступних архітектур, найкраще для цього підходять згорткові нейронної мережі (CNN), які найбільш ефективно виділяють на зображенні значущі області на основі характерних ознак, отриманих у процесі навчання [8].

До переваг цього методу належать його висока точність та стійкість до різноманітних спотворень, таких як шуми, неоднорідне освітлення, забруднення та різні ракурси зйомки. Крім того, метод здатний виділяти досить складні або багатовимірні ознаки, які ігноруються традиційними математичними алгоритмами машинного зору.

До недоліків методу належать високі вимоги до обчислювальних потужностей та складність підготовки необхідного обсягу якісних навчальних даних.

Вирівнювання сегменту з номерним знаком — етап обробки зображення, що виконується після процесу локалізації номерного знаку, але перед оптичним розпізнаванням (OCR). Служить для нормалізації положення номерного знаку і передбачає компенсацію перекосу відносно горизонту чи неправильну перспективу (коли номерний знак розміщено не перпендикулярно камері). Даний процес зазвичай здійснюється за допомогою одного з наступних методів [9] або, за потреби, їхньої комбінації:

Перспективне перетворення — цей метод призначений для усунення спотворень зображення, спричинених неправильною перспективою, тобто коли номерний знак потрапляє у кадр під кутом, а не прямо. В основі роботи методу лежить використання просторових трансформацій з використанням інформації про координати чотирьох кутів номерного знаку. Це перетворення видозмінює зображення номерного знаку таким чином, щоб він виглядав як рівний прямокутник.

Корекція нахилу — для цього методу найкраще підходить перетворення

Хафа, що дозволяє знайти на зображенні прямі лінії, що представляють контури знаку та розрахувати кут відносно горизонту і вже знаючи його виконати потрібний поворот для усунення нахилу.

Афінні перетворення — цей метод має обмеження, а саме застосовуються лише при незначних нахилах або поворотах об’єкта. Його перевагою є мінімальні спотворення чи викривлення форми: афінні перетворення зберігають паралельність ліній і співвідношення відстаней між точками, що не гарантується при використанні перспективних трансформацій. Для застосування афінного перетворення необхідно визначити три контрольні точки на вихідному зображенні. Такими точками можуть бути наприклад кути номерного знаку, якщо вони чітко видимі, або характерні ознаки символів номерного знаку, які легко ідентифікувати. Далі вказується, куди ці точки мають бути зміщені на вирівняному зображенні. Після цього обчислюється матриця перетворення, яку потім застосовують до сегментованої області зображення.

Використання спеціалізованої архітектури нейронних мереж, зокрема Spatial Transformer Network.

Завдяки автоматичній адаптації до особливостей вхідного зображення, і при цьому не потребуючи додаткового ручного налаштування параметрів, здатна усунути майже усі види недоліків, пов’язаних з нахилом, перспективними спотвореннями чи поворотом об’єкта, притому одночасно.

До недоліків можна віднести складність створення якісного навчального датасету. З одного боку, він має бути досить об’ємним, а з іншого — включати для кожного об’єкта як правильні, так і спотворені варіанти (з різними ракурсами, кутами, освітленням тощо).

Розглянувши переваги та недоліки кожного з методів, а також врахувавши специфіку задачі, було вирішено для виконання локалізації та сегментації номерного знаку задіяти нейронну мережу, побудовану на архітектурі CNN. Хоча її використання накладає жорсткі вимоги до апаратного забезпечення, цей підхід має ряд вагомих переваг. Це і значно вища точність розпізнавання в несприятливих умовах експлуатації, таких як недостатнє освітлення, забруднення номерного

знаку. Також її використання дозволяє об'єднати етапи локалізації та сегментації в єдиний процес, що зменшує кількість проміжних перетворень, які можуть бути додатковим джерелом помилок. В подальшому нейронну мережу можна донавчити на нових типах даних, можливості розпізнавання, притому без внесення помітних змін в архітектуру програмного засобу.

2.2 Розробка методу розпізнавання символів номерних знаків

Розпізнавання символів на зображенні номерного знаку буде виконуватися з використанням Tesseract, що є системою розпізнавання тексту (OCR). Цей програмний засіб доступний за ліцензією Apache 2.0, тобто є безкоштовним для використання, вивчення, зміни та розповсюдження. Він був розроблений компанією Hewlett-Packard у 1980-х роках. Тривалий час проєкту не приділялося особливої уваги, аж до 2006 року, коли ним зацікавилася компанія Google і почала його спонсорувати. На сьогодні це потужний інструмент для оптичного розпізнавання символів, який став певним стандартом у розв’язанні завдань, пов’язаних з цією сферою.

Вибір саме Tesseract OCR, а не якогось іншого інструменту для оптичного розпізнавання тексту зумовлений його високою швидкодією, що для нашої задачі є однією з ключових характеристик, оскільки дозволяє розпізнавати символи в реальному часі. Крім того, починаючи з версії 4.0, у Tesseract було інтегровано новий рушій на основі нейронної мережі LSTM [10], для якого доступні офіційні мовні моделі, навчені для понад 100 мов та більше ніж 35 сценаріїв розпізнавання.

Оскільки Tesseract офіційно не підтримує графічного інтерфейсу, його налаштування виконуються через передачу у командному рядку потрібних параметрів. Першим параметром, який потрібно задати, є вибір режиму сегментації зображення. Серед усіх чотирнадцяти які доступні найбільший практичний інтерес представляють дві із них:

Режим --psm 7. Призначений для розпізнавання тексту, представленого як один рядок. Він ігнорує аналіз розмітки та не робить припущень щодо взаємозв’язоку між потенційними блоками тексту. Оскільки номерний знак сам по

собі є типовим прикладом одного рядка тексту, вибір було зупинено саме на цьому режимі.

Режим --psm 10 призначений для розпізнавання одного символу, тому до зображення номерного знаку також потрібно застосувати процедуру сегментації зображення щоб отримати кожен символ у вигляді окремого зображення.

Оскільки автомобільний номерний знак є не стандартним словом, а малозмістовною послідовністю символів, використання словників може спотворити результат або призвести до помилок, намагаючись "зібрати" номер у відоме слово, тому доцільно вимкнути їх підтримку. Це робиться шляхом встановлення змінних load_system_dawg та load_freq_dawg в значення false.

Так як номерний знак складається з комбінації цифр (використовуються усі від 0 до 9) та обмеженого набору літер (не всі літери алфавіту), тому для підвищення точності розпізнавання Tesseract'ом доцільно обмежити набір символів, які він розпізнає, лише допустимим набором. Це можна зробити задавши через tessedit_char_whitelist набір символів, доступний для розпізнавання.

Доречно також інтегрувати алгоритм, який буде аналізувати коректність символьного представлення розпізнаного номерного знаку на відповідність шаблону дві літери, чотири цифри, дві літери, і у випадку невідповідності проводити спробу коригування якщо на місці літери виявиться цифра, і навпаки. Наприклад якщо в позиціях 1,2,5,6 замість літер стоять цифри, то це ознака помилкового розпізнавання і потрібно здійснити заміну на найбільш імовірні відповідні літери, наприклад, цифру «0» — на «O», «1» — на «I» або «L», «3» — на «З», "5" — на "S", "6" — на «G», «8» — на «B».

Точність роботи OCR можна суттєво покращити, застосувавши інструменти бібліотеки комп’ютерного зору OpenCV для попередньої обробки зображень номерних знаків. Ці інструменти спрямовані на зменшення шуму, покращення контрасту та підвищення читабельності окремих символів. Розглянемо детальніше послідовність кроків, які потрібно виконати для досягнення цієї мети:

Приведення зображень номерних знаків до однакових розмірів – цей крок має на меті стандартизувати розміри вхідного зображення. Це важливо, оскільки деякі

методи обробки, такі як застосування фільтрів чи ядер, мають наперед задані параметри, оптимізовані під конкретні розміри зображень. Використання зображень інших розмірів може призвести до некоректних результатів їх роботи. Крім того, стандартизація розмірів допомагає зменшити обчислювальне навантаження, уникаючи необхідності обробляти надмірно великі зображення. Виконується з застосуванням методу cv2.Resize, якому у якості параметрів передається висота та ширина зображення в пікселях та спосіб інтерполяції. Оптимальним по швидкості буде бікубічна інтерполяція, яка задається константою cv2.INTER_CUBIC

Перетворення кольорового зображення у відтінки сірого – це виконується з кількох причин. З одного боку, це необхідно для застосування деяких методів обробки зображень, які працюють виключно з монохромними даними, наприклад бінарізація. З іншого боку, обробка зображення у відтінках сірого вимагає менше обчислювальних потужностей. Це перетворення не призводить до втрати важливої для розпізнавання інформації, оскільки для OCR критичною є відмінність в інтенсивності пікселів тексту та фону, яка зберігається при перетворенні у відтінки сірого. Для цієї операції використовується cv2.cvtColor, який приймає в якості аргументів зображення та константу cv2.COLOR_BGR2GRAY.

Бінаризація — дуже важливий етап подальшої обробки зображення. Як випливає з самої назви, він перетворює зображення, представлене у відтінках сірого, на бінарне, тобто таке, що має лише два кольори: чорний (0) та білий (255). Це дозволяє відділити символи номерного знаку від фону. Суть роботи методу полягає в тому, що він порівнює значення яскравості кожного пікселя з деяким наперед заданим порогом. Якщо яскравість пікселя рівна або перевищує поріг, то він стає білим (255), інакше — чорним (0). OpenCV має кілька реалізацій алгоритму бінаризації, проте в нашому випадку найкраще використати метод, що базується на алгоритмі Оцу (Otsu's method). Цей підхід дозволяє автоматично визначити оптимальне порогове значення для бінаризації, аналізуючи гістограму зображення, без необхідності ручного підбору порогу. Для реалізації використовується функція cv2.threshold, яка приймає вхідне зображення та параметр cv2.THRESH_OTSU, що

активує автоматичний розрахунок порога.

Дилатація — необов’язковий, але бажаний етап. Це морфологічна операція, яка розширює межі об’єкта на зображенні [11]. Це виконується шляхом згортки зображення зі структурним елементом, який визначає розмір і форму дилатації. У якості структурного елемента виступає невелика матриця, яка використовується для дослідження вхідного зображення, а останнє повинно бути представлене у бінарному вигляді, тобто утворене після процесу бінарізації. Результатом роботи цього методу є розгортання елементів переднього плану (білі області збільшуються назовні), заповнення невеликих пропусків та прогалин в об’єктах. Приклад застосування дилатації зображено на рисунку 2.3.

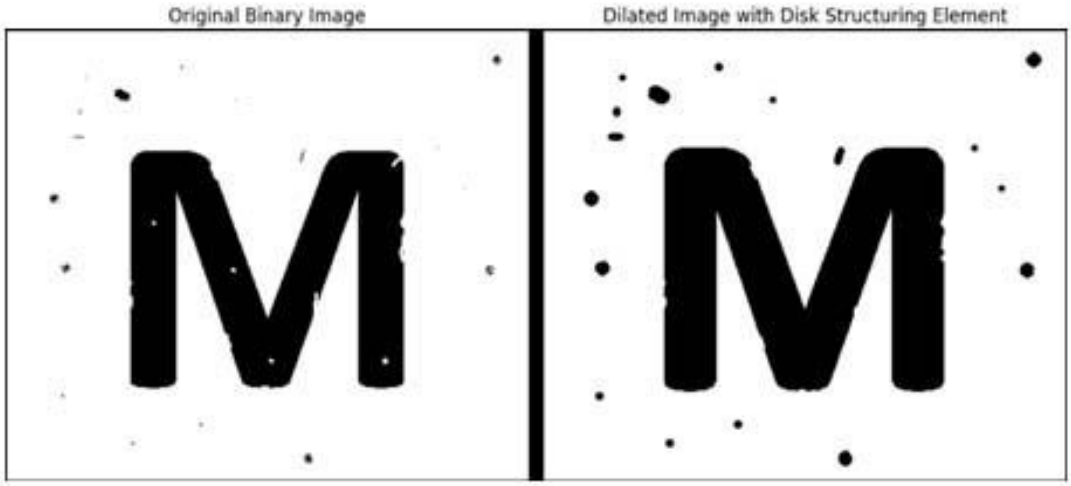


Рисунок 2.3 — Приклад застосування дилатації до бінарного зображення

Ерозія — це морфологічна операція, яка стискає або зменшує межі об’єктів переднього плану на певну величину на основі структурного елемента (ядра) [11]. Вона виконується за допомогою невеликого структурного елемента, зазвичай розміром 2x2, 3x3. Її застосування призводить до видалення дрібних деталей та шуму на передньому плані, які з’являються, наприклад, після бінаризації зображення. Також вона спричиняє з’їдання країв об’єктів всередину. У підсумку ми отримуємо зображення, очищене від шуму та дуже дрібних елементів, таких як тонкі лінії. Приклад застосування ерозії наведено на рисунку 2.4.

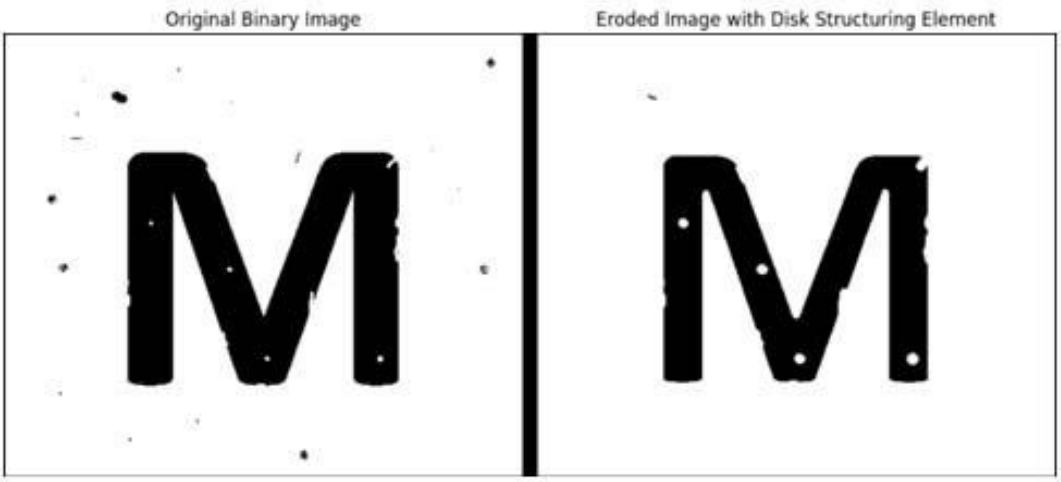


Рисунок 2.4 — Приклад застосування ерозії до бінарного зображення

Для сегментації окремих символів номерного знаку на зображенні найкраще використати метод cv2.findContours з бібліотеки OpenCV. Цей метод призначений для пошуку та виділення контурів об'єктів на бінаризованому зображенні. Знайшовши контури, можна за допомогою функції cv2.boundingRect отримати обмежувальний прямокутник для кожного з них. Аналізуючи властивості цих прямокутників, передусім їх площу та співвідношення сторін — можна надійно відрізнити контури, що належать символам від інших елементів.

2.3 Розробка структурної схеми програмного засобу

Слід наперед уточнити, що програмний засіб складається з двох окремих, але взаємопов’язаних підсистем. Серверна підсистема, яка виконує обробку даних та збереження результатів та клієнтська підсистема, яка відповідає за взаємодію з користувачем через графічний інтерфейс. Кожна з них, у свою чергу, розбита на декілька окремих модулів, кожен з яких виконує певну, вузькоспеціалізовану функцію. Такий підхід підвищує гнучкість системи, спрощує її розробку і подальшу підтримку та додавання нових функцій.

Для наглядного представлення взаємодії модулів між собою було розроблено структурну схему (рис.2.5).

Алгоритм роботи, який відповідає даній схемі наведено на рисунку В.1.

Рисунок 2.5 — Структурна схема програмного засобу

Модуль пошуку автомобільних номерних знаків виконує процес сегментації зображення на основі попередньо навченої моделі нейронної мережі, тобто здійснює пошук та виділення значущих об’єктів на зображенні (Areas of Interest - AOI), що будуть представляти собою області, що можуть містити автомобільні номерні знаки;

Модуль попередньої обробки зображення виконує попередню обробку сегмента зображення, виділеного на попередньому етапі. Його основна функція полягає у покращенні візуальної якості виокремленого зображення номерного знаку. Це досягається шляхом застосування до нього заданого набору

спеціалізованих методів та фільтрів, представлених в модулі OpenCV, спрямованих на зменшення шуму, підвищення контрастності та різкості, а також виділення важливих деталей, що формують символи номера. Крім того, модуль здійснює масштабування зображення до заданого розміру. Усі ці кроки виконуються з єдиною метою – зробити зображення номерного знаку більш придатним для подальшого розпізнавання символів допомагаючи системі OCR краще розрізняти межі між символами та фоном, що підвищує точність розпізнавання.

Модуль розпізнавання символів приймає адаптоване та оптимізоване зображення номерного знак від попереднього модуля. Використовуючи OCR Tesseract, налаштований на використання білого списку, що містить обмежений, дозволений набір символів, а також налаштований на розпізнавання одного рядка тексту (оскільки номерний знак являє собою саме один рядок символів), модуль виконує аналіз зображення. Цей аналіз включає сегментацію символів, що здійснюється шляхом аналізу різних візуальних характеристик зображення, таких як контраст, текстура, зміни яскравості. Отримавши таким чином набір окремих сегментів, кожен з яких містить один символ, застосовується процедура витягування ознак (Feature Extraction). Метою цієї процедури є визначення візуальних характеристик символу, таких як наявність петель, прямих ліній чи кутів. Після отримання певного набору ознак стає можливим ідентифікувати символ шляхом порівняння цих ознак з ознаками відомих символів із навчальної моделі Tesseract та вибору найкращого відповідника серед символів із дозволеного білого списку. На виході цього модуля ми маємо розпізнаний номерний знак у вигляді текстового рядка.

Модуль реконструкції символів застосовується у випадках, коли розпізнаний номерний знак не відповідає шаблону або структурі номерного знаку. Основною метою модуля є автоматичне коригування результату з урахуванням типових помилок оптичного розпізнавання символів. Слід зазначити що це не гарантує правильного результату, лише підвищує ймовірність отримання коректного номерного знаку.

Модуль збереження результатів призначений для забезпечення можливості

практичного використання даних, зібраних попереднім модулем, завдяки організації їх довгострокового зберігання. Сама інформація має зберігатися як текстові дані в оптимізованій для ефективного та швидкого доступу формі, для чого найкраще підходять реляційні бази даних, та включає не лише розпізнані номерні знаки, але й відповідні кадри, з яких вони були отримані (при цьому зберігається не сам кадр, а шлях до нього на носії). Крім того, додатково автоматично додається інша службова інформація, така як точний час фіксації номера. Сам модуль надає зручний програмний інтерфейс (API), що дозволяє іншим програмним засобам легко отримувати доступ до всієї збереженої інформації та повноцінно її використовувати.

Модуль отримання даних є тим інструментом, завдяки якому користувач має можливість взаємодіяти з системою через через веб-інтерфейс формуючи запити та отримуючи відповіді.

Модуль взаємодії користувача з програмним засобом забезпечує зручний спосіб взаємодії користувача із системою. Він реалізований у вигляді веб-інтерфейсу, що значно підвищує його універсальність та зручність використання. Завдяки веб-інтерфейсу користувач не прив’язаний до конкретного робочого місця і може підключатися з різноманітних пристроїв, таких як персональний комп'ютер, ноутбук чи мобільний телефон, або навіть віддалено, через інтернет, якщо дотримані відповідні вимоги до безпеки, наприклад підключення здійснюється через VPN.

У веб-інтерфейсі передбачено поле для введення номерного знаку для пошуку його в базі даних. При цьому знак може бути введений не повністю, і тоді система відобразить список усіх номерів, що відповідають заданій масці пошуку. Реалізовані додаткові критерії пошуку через можливість задати часовий інтервал (рис.Д.2). Результатом пошуку є список, що містить точний час фіксації автомобіля. Також кожен запис списку є гіперпосиланням на відповідний файл зображення що є дуже корисним та зручним і дозволяє швидко переглянути зафіксований транспортний засіб (рис. Д.3).

2.4 Використання згорткової нейронної мережі

Свою назву та основні функції цей тип нейронних мереж бере від класичної математичної операції згортання. Згортання вже багато десятиліть застосовується в різноманітних інженерних дисциплінах, в тому числі і обробку сигналів та комп'ютерну графіку. Зазвичай при виконанні такої процедури користувач задає набір параметрів, що відповідають певному функціональному задуму, наприклад підсвітка країв або пригнічення високочастотних звуків. Навіть така загальновідома програма як Photoshop в основі роботи багатьох своїх фільтрів містить цю процедуру. Окрім використання в задачах розпізнавання цей тип мереж досить добре себе проявив і в задачах обробки природньої мови та семантичного парсисингу, а також моделюванні та класифікації речень.

Згорткові нейронні мережі (CNN) здобули широку популярність у задачах розпізнавання зображень, оскільки вони значно ефективніше, ніж традиційні повністю зв'язані нейронні мережі, долають виклики, пов'язані з обробкою візуальних даних. Для ефективного розпізнавання об'єктів, особливо при роботі з великими обсягами даних, модель має володіти високою здатністю до навчання та точно виділяти й інтерпретувати ознаки зображення [12].

Ключові проблеми традиційних повнозв'язаних мереж при обробці високорозмірних зображень полягають у надмірній кількості параметрів, що виникає через повне зв'язування кожного нейрона з усіма вхідними елементами. Це вимагає надзвичайно великих обсягів навчальних даних для коректного налаштування всіх зв'язків, інакше швидко виникає перенавчання, коли модель навчається на нехарактерних другорядних ознаках. Крім того, такі мережі не враховують природну просторову локальну структуру зображення (високу кореляцію сусідніх пікселів) і є нестійкими до зміщень об'єктів чи змін їх ракурсу. CNN вирішують ці проблеми завдяки своїй унікальній архітектурі, яка дозволяє істотно зменшити кількість навчуваних параметрів та зв'язків, одночасно ефективно працюючи з просторовою структурою даних .

Процес навчання нейронної мережі починається з приведення формату

вхідних даних до єдиного стандарту. У випадку використання зображення це зміна розміру, перетворення трьохканального кольорового на одноканальне. Додатково виконується нормалізація пікселів коли їх значення кольору, яке зазвичай знаходиться в діапазоні [0-255] переводиться у діапазон [0-1].

Слід зазначити, що CNN це не одна конкретна мережа а архітектура, на основі якої будується ціле сімейство вже конкретних моделей нейронних мереж. Історично першою була створена R-CNN [13], яка була певним симбіозом нейронної мережі та класичних методів комп'ютерного зору. Останній виконував сегментацію зображення і таким чином отримували набір регіонів. Кожен такий регіон потім незалежно оброблялася через AlexNet для вилучення ознак. Кожен запропонований регіон змінювався за розміром і подавався до попередньо навченої CNN для отримання вектора ознак фіксованої довжини. Вилучені ознаки використовувалися для класифікації об'єкта в межах регіону, а лінійна регресія застосовувалася для уточнення координат обмежувальної рамки. Незалежна обробка тисяч регіонів робила R-CNN обчислювально дуже дорогим. Крім того, через те, що багато запропонованих регіонів перекривалися, CNN виконувала одне й те саме вилучення ознак кілька разів, що призводило до надлишкових обчислень. Повільне генерування пропозицій регіонів за допомогою вибіркового пошуку також обмежувало загальну швидкість роботи.

Еволюційним розвитком R-CNN став Fast R-CNN який запропонував значні покращення архітектури [14]. Для підвищення ефективності, замість обробки кожної області окремо, усе зображення проходило через CNN лише один раз, формуючи карту згорткових ознак. З цієї спільної карти ознак за допомогою RoI pooling вилучалися ознаки фіксованого розміру для кожної запропонованої області. Перевага RoI pooling полягає у стандартизації розміру вихідних ознак, незалежно від розмірів вхідної області. Отримані ознаки RoI потім використовувалися повністю зв'язними шарами для класифікації об'єктів та уточнення їхніх обмежувальних рамок.

Значним проривом у галузі виявлення об'єктів стала розробка архітектури Faster R-CNN. Ключовим нововведенням є інтеграція мережі пропозицій регіонів,

відомої як RPN (Region Proposal Network). RPN призначена для автоматичного створення пропозицій потенційних областей з об'єктами безпосередньо всередині основної мережі. Вона реалізована як повністю згорткова мережа, що на основі вхідних даних прогнозує межі й оцінює ступінь об'єктності для кожної можливої позиції. Вирішальним фактором ефективності є спільне використання згорткових ознак між мережею RPN та подальшою мережею виявлення Fast R-CNN. Таке спільне використання робить етап генерування пропозицій регіонів обчислювально надзвичайно швидким і практично не вимагає додаткових ресурсів. У результаті це дозволило досягти швидкості виявлення об'єктів, близької до реального часу. Наприклад, час обробки одного зображення міг становити лише 0,2 секунди, а продуктивність сягала 5 кадрів на секунду при використанні мережі VGG-16 на графічному процесорі GPU.

YOLO (You Only Look Once) — без перебільшення стала революційною open-source архітектурою, представленою у 2015 році Джозефом Редмоном, Сантошем Діваллою та Россом Гіршиком.

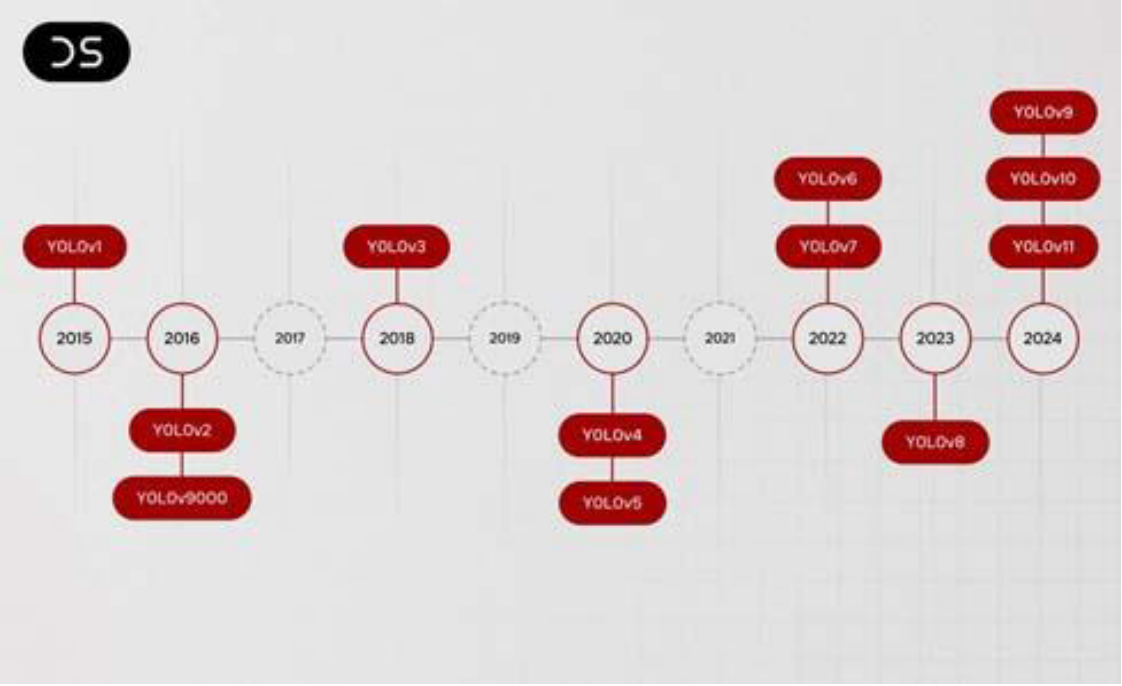


Рисунок 2.6 — Хронологія виходу версій нейронної мережі YOLO

Перша версія вийшла в 2015 році і могла виконувати лише детекцію об’єктів на

зображенні. Оригінальна реалізація була написана на C та CUDA, що додавало апаратного прискорення на графічних процесорах Nvidia. Розглянемо коротко особливості деяких версій:

— YOLO V2: прив'язка з доданими K-середніми, двоетапне навчання, повна згорточна мережа;

— YOLO V3: багатомасштабне виявлення за допомогою FPN;

— YOLO V4: SPP, функція активації MISH, покращення даних Mosaic/Mixup, функція втрати GIOU (узагальнене перетину через об'єднання);

— YOLO V5: Гнучке керування розміром моделі, застосування функції активації Hardswish і розширення даних.

Остання на сьогоднішній день версія, вже здатна виконувати класифікацію, трекінг об'єктів на відео, задачі pose estimation та інші, і що важливо все це виконується в режимі реального часу, це одна з найшвидших моделей. З виходом YOLO комп'ютерний зір отримав інструмент, який не лише працював швидше за всіх попередників, але й мав досить хороші метрики (mAP — mean Average Precision), інакше кажучи точність. Розглянемо більш детально основні можливості YOLO:

— класифікація (classification) — це визначення наявності або відсутності об'єкта певного типу на зображенні;

— локалізація (localization) дозволяє не лише класифікувати об'єкт а й визначити його розташування на зображенні та відобразити відповідні координати за допомогою обмежувальної рамки (bounding box);

— семантична сегментація (Semantic Segmentation) визначає для кожного пікселя зображення, до якої категорії він належить, наприклад розділяє зображення на такі області як «дорога», «дерево», «небо»;

— детекція об'єктів (Object Detection), яка дозволяє знайти, локалізувати та класифікувати на зображенні одночасно кілька об'єктів по наявній базі класів, і саме цей механізм буде використовуватися в подальшому для детекції.

Нейронна мережа YOLO вирізняється кількома ключовими архітектурними особливостями, які роблять її особливо придатною для вирішення завдань детекції.

Оскільки автори YOLO розглядають детекцію виключно як задачу регресії, будова моделі є простішою, якщо порівнювати з тими підходами, що реалізовують інші архітектури нейронних мереж, наприклад двоетапні методи. Це суттєво підвищує швидкість не лише під час детекції, а й під час навчання [15]. На відміну від тієї ж Faster R-CNN, YOLO під час навчання та інференсу виконує аналіз усього зображення, а не його ділянок, як «region proposals», завдяки чому YOLO здатна кодувати контекстну інформацію про об’єкти та їх оточення, що позитивно відображається на точності передбачень і зменшує кількість помилкових виявлень. Також модель при навчанні враховує загальні характеристики об’єктів, що покращує її ефективність роботи на нових наборах даних. Це підвищує адаптивність без використання додаткового навчання.

2.5 Використання бібліотек та модулів із відкритим кодом

При розробці програмного засобу для реалізації заявлених функцій було використано декілька бібліотек та модулів з відкритим вихідним кодом (рис. Г.1).

OpenCV (Open Source Computer Vision Library) — бібліотека з відкритим кодом для роботи з алгоритмами комп’юторного зору, обробки зображень. Переважна більшість її коду написана на мові C++, також може бути задіяна в проектах написаних на Python, JavaScript, Ruby. Здатна працювати на широкому спектрі операційних системах, таких як Windows, Linux, MacOS, IOS та Android. Вона є певним стандартом для проектів, які використовують машинний зір. Її бібліотека нараховує близько 2500 інструментів та алгоритмів комп’ютерного зору та машинного навчання, хоча останнє не користується широким попитом у зв’язку з деякими обмеженнями даних інструментів. Основними з них є: відсутність підтримки сучасних моделей глибокого навчання, таких як рекурентні та згорткові нейронні мережі чи трансформери, присутній лише алгоритми k-NN, SVM (Support Vector Machines) та дерева рішень. Вони є досить повільними та малоефективними для складних задач. Іншим суттєвим мінусом є відсутність можливості завантажити попередньо навчені моделі, створені у Tensorflow чи PyTorch. З огляду на перелічені обмеження роботи OpenCV з нейромережами, ці можливості

залишиться незадіяним. Завдяки тому, що переважна більшість коду самого OpenCV реалізована на C++ маємо високу швидкодію, достатню для обробки зображень в режимі реального часу що є надзвичайно важливим, оскільки чим більше кадрів здатна обробити система, тим з більшою ймовірністю буде зафіксований автомобільний номерний знак.

SQLAlchemia — це бібліотека, написана на мові програмування Python, створена Майклом Баєром. Вона є майже універсальним інструментом для роботи з базами даних. SQLAlchemy дозволяє розробникам ефективно взаємодіяти з різними типами баз даних, надаючи потужні та гнучкі засоби для створення запитів, управління даними та організації взаємодії між об'єктами Python та таблицями бази даних. Ця бібліотека підтримує такі популярні бази даних як MySQL, SQLite, Postgres, Oracle та багатьох інших, наприклад навіть таку, досить екзотичну, як Amazon Redshift, яка використовує власний діалект PostgreSQL.

Одна з головних переваг цієї бібліотеки є можливість писати універсальний код який буде ідентично працювати з різними базами даних незалежно від її типу чи специфічних особливостей. Також присутнє правильне екранування та очищення вхідних даних перед їх надсиланням до бази, що допомагає боротися з такою розповсюдженою вразливістю як SQL-ін’єкція.

Має два основні режими роботи: Core та ORM [16], і між ними є досить значні відмінності. Core режим має схемоцентричний підхід, що орієнтований на ключі, таблиці та індексні структури. Даний режим добре оптимізований для роботи зі звітами, аналізом чи іншими випадками коли потрібно забезпечити надійний контроль над запитами або працювати з немодельованими даними. ORM працює іншим чинам, він здійснює певну інкапсуляцію базової структури що робить роботу з базою даних схожою на роботу зі звичайним кодом на Python. Інакше кажучи неважливо яка база даних використовується, це може бути Oracle Data Warehousing Amazon Redshift чи звичний MySQL взаємодія з ними відбувається ідентично. Це підвищує швидкість розробки так як дозволяє зосередитись на логіці роботи програмного продукту а не на особливостях взаємодії з конкретною базою даних. Також ORM автоматизує такі процеси як створення запитів, управляння

транзакціями та відображення об’єктів в базі даних і сам код є більш читабельним та легшим в підтримці чим ті ж сирі запити SQL.

PyTorch — це фреймворк для машинного навчання [17], розроблений компанією Meta AI в 2016 році. З часом став найпопулярнішим фреймворком витіснивши з першого місця TensorFlow.

FastAPI — завдяки асинхронності є одним з найшвидших веб-фреймворків. написаних на Python. На відміну від інших підтримує обов’язкову анотацію та валідацію типів даних, що зменшує ймовірність допущення помилок, пов’язаних з неправильним використанням змінних. Значним плюсом є наявність вичерпної офіційної документації з великою кількістю прикладів що значно спрощує його вивчення та використання.

В цьому розділі було розроблено послідовність та засоби розпізнавання номерних знаків автомобілів, адаптовані для ефективного виконання задачі пошуку та розпізнавання навіть при наявності значної кількості зовнішніх несприятливих факторів. Обґрунтовано вибір згорткової нейронної мережі YOLO для пошуку номерних знаків та системи OCR Tesseract для оптичного розпізнавання символів. Додатково послідовність було розширено модулем, який виконують попередню обробку зображення для підвищення коректності розпізнавання та реалізовано алгоритм корекції помилок, який дозволяє виправляти неправильно розпізнані символи. Розроблена модульна структура програмного засобу дозволила підвищити гнучкість та розширила можливості для подальшого вдосконалення розробленого програмного засобу.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ АВТОМОБІЛІВ

3.1 Вибір інструментальних засобів програмування

Мова програмування Python була обрана як основний інструмент розробки програмного засобу. Цей вибір обумовлений низкою переваг та особливостей ключові з яких буде розглянуто нижче.

Python — це потужна мова програмування, що підтримує багато парадигм, яка оптимізована для забезпечення високої продуктивності програмістів, читабельності коду та якості програмного забезпечення і однаково придатний як для написання автономних програм так і для сценарних додатків в широкому різноманітті предметних областей. Він має лаконічний синтаксис, а відсутність дужок для виділення ділянки коду, відсутність строгої типізації і відповідно необхідності вказувати тип для змінних робить код візуально чистим, неперенавантаженим елементами синтаксису що підвищує продуктивність праці програміста, інколи дуже суттєво. Так, для прикладу, якщо реалізувати якусь задачу на мові програмування C++ чи Java і на Python, то код на Python в 3-4 рази меншим по об’єму.

Іншою перевагою Python є переносимість коду. Це означає можливість його запуску на різних операційних системах (Windows, Linux, MacOS) чи процесорних архітектурах (x86-64, arm64, risc-v) з мінімальними змінами. Це досягається завдяки інтерпретованій природі мови та кросплатформеній сумісності. Це дає змогу створювати додатки, скрипти чи веб – сервіси, які легко адаптувати до різних середовищ.

Хоча Python і підтримує ООП, його реалізація є простою у порівнянні з деякими іншими мовами, такими як C# або Java. У ньому відсутні різні модифікатори доступу, такі як public, private, protected у C#, а всі члени класу є публічними, що частково порушує принцип інкапсуляції і для покращення ситуації з приватністю членів класу серед розробників було прийнято певні домовленості щодо іменування, які передбачають обмеження доступу до таких членів.

Серед усіх середовищ розробки які підтримують мову програмування Python вибір було зупинено на двох, це VS Code від компанії Microsoft [18] та Jupiter Notebook. Розглянемо особливості, переваги та недоліки кожної з них.

Jupiter Notebook — це безкоштовний веб-застосунок, який надає інтерактивне середовище для створення та обміну документами, що містять код, рівняння, візуалізації та текстові пояснення. Присутня підтримка різних мов програмування, зокрема Python.

До його переваг можна віднести високу наочність та зручність. Завдяки тому, що весь код, пояснення, результати та візуалізації зібрані в одному місці ми маємо всі необхідні матеріали під рукою і можемо оформлювати їх у зручному та зрозумілому форматі. Ще однією перевагою є можливість організації спільної роботи завдяки наявності спільного сервера, за яким може працювати кілька користувачів одночасно. Крім того Jupyter Notebook дозволяє виводити результати виконання коду безпосередньо в робочому середовищі а не в новому вікні, як у інших IDE. Для створення форматованого тексту, що включає заголовки, списки, жирний та курсивний шрифт, гіперпосилання, зображення та математичні формули використовуються клітинки Markdown що розширює можливості документування та представлення інформації.

VS Code — це сучасний, потужний та популярний редактор коду, створений компанією Microsoft, що надає розробнику всі основні інструменти для швидкого циклу створення та налагодження та управління кодом чи керування версіями. Це крос-платформний редактор що підтримує Windows, Linux та MacOS, також він є легшим та швидшим у порівнянні з іншими популярними IDE, наприклад такими як PyCharm від компанії JetBrains чи Atom.

Завдяки великій кількості додаткових модулів, базові можливості VS Code можуть бути значно розширені та доповнені. Це стосується підтримки нових мов програмування, таких як C++, Java, JavaScrypt, C#, Python, Rust, Go та багато інших. Інсталяція та керування розширеннями доступно через платформу Marketplace, яка інтегрована безпосередньо в графічний інтерфейс VS Code. Це своєрідний онлайн-магазин, де розробники розміщують свої розширення як на безкоштовній, так і на

комерційній основі. Кожен такий плагін містить інформацію про свого розробника, опис можливостей та функцій, історію версій, кількість завантажень та рейтинг від користувачів. Це дозволяє сформувати уявлення про плагін, його особливості, доцільність використання, а також переваги та недоліки перед аналогами і доцільність його використання.

IntelliSense — це загальний термін для цілого набору функцій для редагування коду. Він працює на основі мовної служби, яка забезпечує інтелектуальне завершення коду на основі семантики мови та аналізу вихідного коду і має вигляд набору пропозицій, що змінює списки членів, змінних та методів по мірі того як користувач продовжує вводити символи. Значним нововведенням останніх версій VS Code є Copilot, який розширює пропозиції варіантів автодоповнення. Він здатен не просто доповнювати окремі слова чи фрагменти, а пропонувати цілі блоки коду, засновані на заданому контексті. Це досягається завдяки аналізу не лише того рядка, в якому в даний момент відбувається написання коду, а й сусідніх що дозволяє з високою ймовірністю видати актуальний варіант коду. До інших корисних можливостей Copilit можна віднести допомогу у виявленні потенційних помилок та відповідні пропозиції змін для їх усунення. Ще він може пояснити доступними термінами як працює деякий фрагмент коду чи уся програма, логіку роботи функцій чи принципи використання бібліотек.

Засоби налагодження коду є невід’ємною частиною функцій будь-якого середовища розробки і VS Code не є винятком. Окрім вбудованої підтримки для таких мов як JavaScript, TypeScript і Node.js через Visual Studio Marketplace доступно широкий вибір розширень налагодження інших мов програмування, таких як C++, Python, Java, PHP.

Важливим механізмом налагодження коду є використання точок зупину, що призупиняють виконання коду у певних місцях та дають змогу проаналізувати стан певних змінних. Додатково можна задати умови спрацювання точок зупину на основі виразів, кількості звернень або комбінації обох. Існують також ініційовані точки робота яких полягає в вмиканні тоді, коли спрацьовує інша точка. Це буває

корисно під час діагностики випадків збою в коді, які трапляються лише після певної передумови.

Якщо розробник одночасно працює над кількома проектами, то слід очікувати, що деякі з них можуть потребувати різні версії одних і тих самих бібліотек. Розробник – початківець за потреби інсталювати якусь бібліотеку потрібну для його проекту інсталює її глобально, що може призвести до конфлікту залежностей, якщо буде інший проект що потребує іншої версії цієї бібліотеки. Наприклад можливий випадок, коли є проект який вже працює, потім створюється інший проект що потребує бібліотеки, спільної для обох цих проектів, але новішої версії. Це може призвести до порушення роботи або взагалі зламає роботу першого проекту, який залежить від старої версії цієї бібліотеки. Вирішенням цієї проблеми є використання віртуального середовища суть якого полягає в створенні ізольованого простору, певного контейнера, в якому містяться усі файли проекту включно з залежностями та інтерпретатором Python що робить таке середовище самодостатнім, кожне таке середовище буде містити та використовувати ті версії бібліотек, які потрібні саме цьому проекту що вирішує проблему конфлікту. Також ще одним плюсом використання віртуального середовища є можливість легко перенести проект на інше робоче місце просто скопіювавши його робочу папку та надавши список необхідних залежностей, які потрібно інсталювати, зазвичай для цього створюється файл requirements.txt з відповідним списком бібліотек.

3.2 Розробка структури та функцій графічного інтерфейсу

Наразі програмне забезпечення залежно від способу реалізації графічного інтерфейсу можна поділити на дві категорії:

Віконні додатки — представляють собою тип програмного забезпечення, що виконується безпосередньо в операційній системі користувача і взаємодіє з ним через графічний інтерфейс, організований у вигляді одного або декількох вікон. Прикладом є майже всі програмні засоби, що присутні на комп’ютері пересічного користувача.

Веб-інтерфейси є програмними інтерфейсами користувача, що функціонують

у середовищі веб-браузера та слугують засобом для доступу й взаємодії користувача з функціональністю веб-додатку або онлайн-сервісу. Порівняно з віконними додатками, перевагами веб-інтерфейсів є їх кросплатформність, відсутність потреби встановлювати клієнтське програмне забезпечення (окрім веб-браузера, який зазвичай є на всіх пристроях), а також можливість організації одночасної роботи кількох користувачів.

Проаналізувавши переваги та недоліки кожного з підходів було вирішено при розробці графічного інтерфейсу надати перевагу варіанту на основі веб-технологій.

Особливу увагу було приділено логічній послідовності розміщення елементів інтерфейсу.

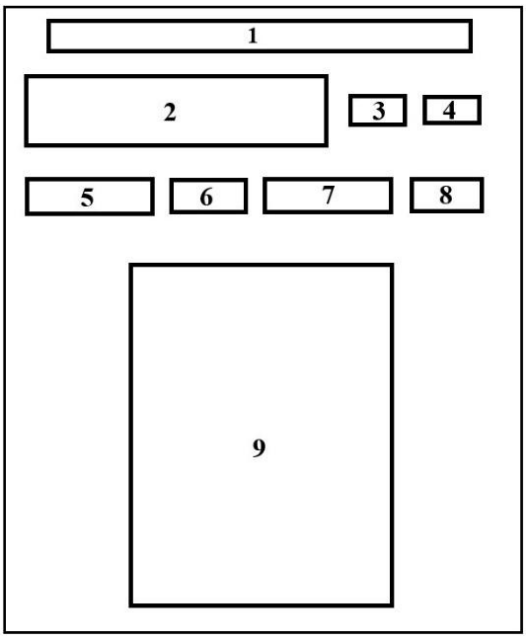


Рисунок 3.1 — Структурна схема веб-інтерфейсу програмного застосунку

Позиції елементів інтерфейсу, наведеного на рис. 3.1:

1) назва сторінки;
2) поле для введення номерного знаку;
3) кнопку пошуку;
4) кнопка очистки некоректного введення;
5) поля для задання дати початку пошуку;

6) поле для задання години та хвилини початку пошуку;

7) поле для ведення дати закінчення пошуку;

8) поле для задання години та хвилини закінчення пошуку;

9) список з результатами пошуку.

Усі поля вводу номерного знаку є опційними, тобто немає вимоги щодо обов'язкового заповнення їх вмісту. Наприклад можливі такі варіанти.

У випадку якщо користувач не ввів жодного значення, то виведуться усі номерні знаки за увесь період.

Якщо не введено лише кінцеву дату, то відобразяться результати від вказаної дати початку і дотепер.

Якщо не введено лише початкову дату, то відобразяться результати за період до вказаної кінцевої дати.

Якщо користувач ввів обидві дати (початок і кінець), то відобразяться номерні знаки в межах зазначеного періоду. По аналогічному принципу працює введення часу (рис. Г.2).

Також реалізовано деякі інші функції, спрямовані на те, щоб зробити процес введення даних, виконання пошуку та перегляду результатів максимально простим, швидким та інтуїтивним для користувача:

— введення номерного знаку розбито на окремі поля вводу, візуально імітуючи номерний знак за допомогою фонового зображення (рис. 3.2);

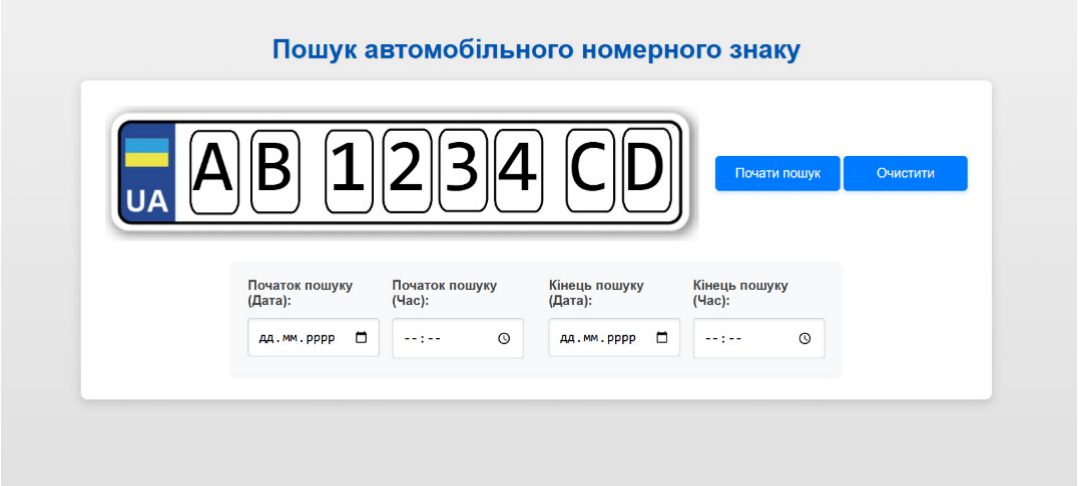


Рисунок 3.2 — Загальний вигляд веб-інтерфейсу

— допускається пропустити один або кілька символів при введенні в поле пошуку номерного знаку і тоді відповідність шуканому варіанту буде перевірятися тільки за введеними позиціями, дозволяючи пропущеним позиціям містити будь - який інший символ з переліку дозволених;

— при клацанні на область номерного знаку курсор автоматично переходить в перше поле, і після введення кожного символу автоматично переходить до наступної позиції;

— адаптація положення елементів інтерфейсу під тип пристрою та розміри дисплею (рис. 3.3);

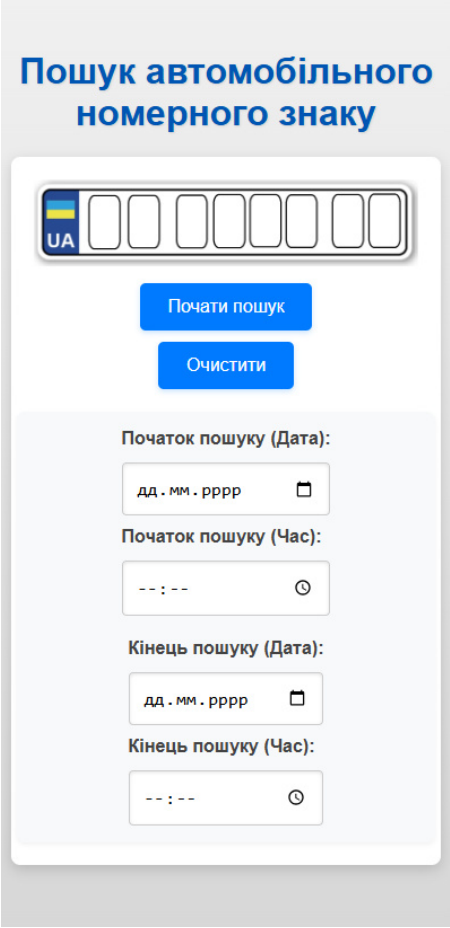


Рисунок 3.3 — Адаптація інтерфейсу під розмір екрану

— результати пошуку відображаються у вигляді таблиці. В першому стовпчику міститься номерний знак, а в другому точний час фіксації який є гіперпосиланням ні відповідний кадр зображення, з якого цей номерний знак був отриманий (рис. 3.4).

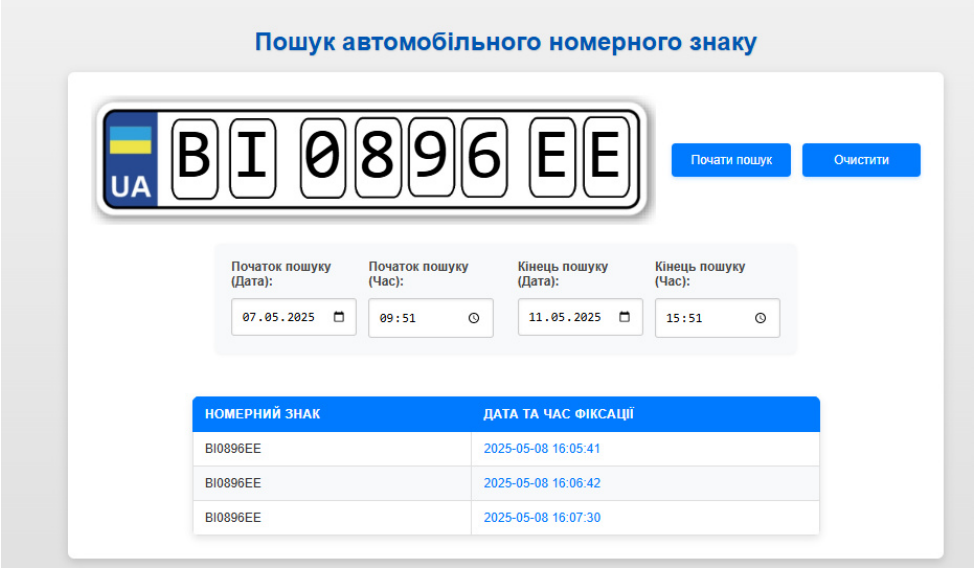


Рисунок 3.4 — Виконання пошуку та відображення результатів

В підсумку можна стверджувати що розроблений графічний інтерфейс є зручним та інтуїтивно зрозумілим для користувача, побудований на сучасних технологіях. Він пропонує гнучкі критерії для пошуку та містить ряд покращень, які значно покращують процес введення даних, а результати пошуку відображаються в зручному та інформативному вигляді (рис. Г.3).

3.3 Навчання та валідація нейронної мережі YOLO

Процес підготовки до використання нейромережі YOLO складається з двох основних етапів: формування навчального набору даних і безпосередньо навчання моделі. Розглянемо докладно кожен із цих етапів.

Для створення якісного навчального датасету необхідно підготувати достатню кількість зображень з автомобілями різних типів та моделей, бажано не менше 400, де вони представлені у різноманітних ракурсах, рівню освітленності та на різній відстані від камери. Сам номерний знак повинен мати максимально можливу кількість варіацій масштабів та поворотів. Корисним буде включити зображення, зроблені при різних погодних умовах та різний час доби та фони (міські вулиці, парковки, автостради, сільська місцевість). Це підвищить універсальність та адаптованість моделі.

Наступним кроком потрібно виконати процес розмітки зібраних зображень,

для чого буде використано спеціалізований інтернет-сервіс CVAT [19] (Computer Vision Annotation Tool). У ньому потрібно створити новий проект та завантажити за один раз усі підготовлені зображення. Процес розмітки полягає в ручному виділенні на зображенні ділянки, що відповідає номерному знаку (рис. 3.5).

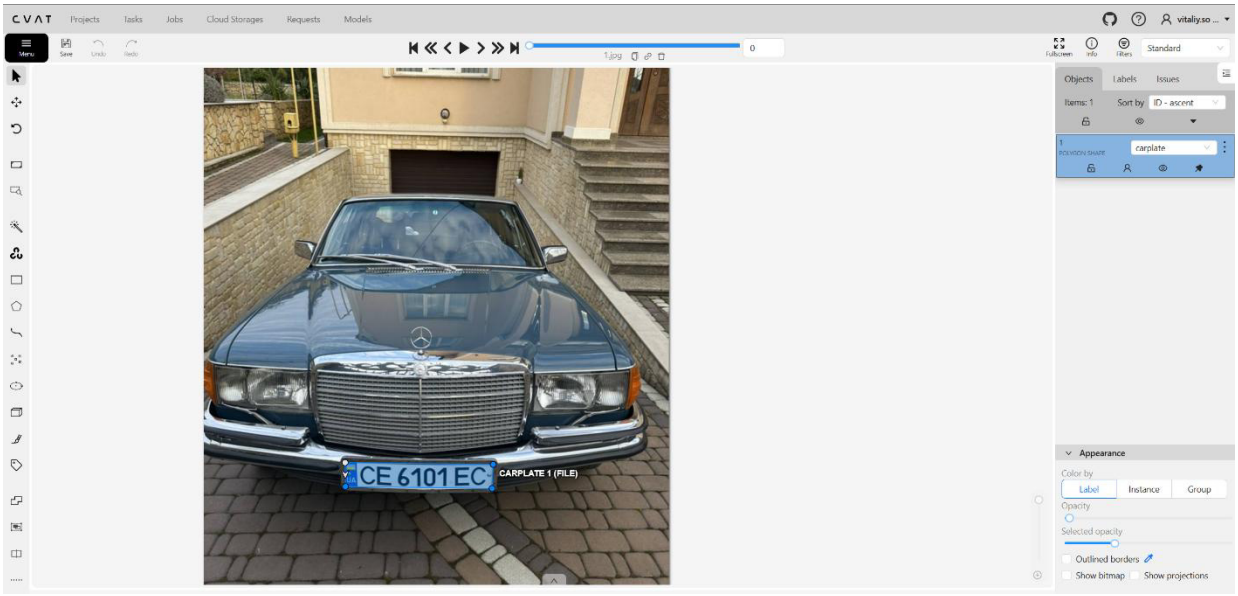


Рисунок 3.5 — Процес розмітки зображень через сервіс CVAT

Таким чином створюється свого роду система «питання — відповідь». Виконуючи розмітку ми показуємо моделі, де саме на зображенні знаходиться шуканий об’єкт, формуючи таким чином правильні відповіді для усіх навчальних зображень. Це дозволяє виокремити характерні візуальні ознаки номерних знаків. Чим більший та різноманітніший вихідний датасет, тим краще модель зможе узагальнити дані та в подальшому точніше виконувати детекцію номерних знаків.

По закінченню розмітки потрібно зберегти результати у форматі YAML [20]. В ньому буде міститися координати обмежувального прямокутника до кожного з зображень. Аналогічним чином створюється ще один, валідаційний набір, який буде використовуватися для оцінки ефективності проходження навчання.

Після підготовки датасету він ще не готовий до використання оскільки формат COCO не підтримується інтерфейсом YOLO. Тому потрібно застосувати сторонні інструменти для конвертації [20].

Надалі необхідно визначитися з версією моделі. Для використання доступно

кілька моделей YOLO [21]. Вибір було здійснено на основі двох ключових критеріїв: точності моделі, яка оцінюється за показником mAP50 (табл. 3.1), та продуктивності.

Таблиця 3.1 — Інформація про точність різних версій моделі

Модель	mAP50
YOLO11n-obb	78.4
YOLO11s-obb	79.5
YOLO11m-obb	80.9
YOLO11l-obb	81.0
YOLO11x-obb	81.3

Як видно, різниця в точності mAP50 виявлення об’єктів між наймолодшою модулю та найстаршою становить менше трьох відсотків, що досить незначна різниця, тому орієнтуватися при виборі моделі будемо на продуктивність, яку ми виміряємо для кожної моделі окремо за допомогою коду, наведеного в лістингу 3.1.

Лістинг 3.1 — Скрипт тестування продуктивності моделей YOLO

```
import cv2
import time
from ultralytics import YOLO
VIDEO_PATH = 'others/Test.mp4'
YOLO_MODEL_PATH = 'yolo11x-obb.pt'
FRAME_COUNT = 2000
DEVICE = 'cuda'
# DEVICE = 'cpu'
cap = cv2.VideoCapture(VIDEO_PATH)
model = YOLO(YOLO_MODEL_PATH)
model.to(DEVICE)
processed_frame_count = 0
start_time = time.time()
while processed_frame_count < FRAME_COUNT:
    ret, frame = cap.read()
    if not ret:
        break
    results = model(frame, verbose=False)
    processed_frame_count += 1
```

```
    end_time = time.time()
    total_time = end_time - start_time
    dev = ('відеокарті' if DEVICE == 'cuda' else 'процесорі')
    print("\n--- Результати тесту продуктивності ---")
    print(f"Час обробки на {dev} становить {processed_frame_count} кадрів:
{total_time:.4f} секунд")
    cap.release()
```

Після завершення тестування всі отримані результати були занесені в таблицю 3.2. Проаналізувавши отримані значення робимо висновок, що найбільш оптимальну продуктивність мають моделі «m» та «s», але враховуючи помітно більше середнє навантаження на GPU моделі «m» свій вибір зупиняємо на «s».

Таблиця 3.2 — Порівняння продуктивності різних моделей YOLO

Назва моделі YOLO	Час обробки 2000 кадрів, с	Середнє завантаження GPU, %	Середня кількість кадрів за секунду
YOLO11n-obb	38	40	52
YOLO11s-obb	43.9	65	45
YOLO11m-obb	46.9	80	43
YOLO11l-obb	57.4	92	35
YOLO11x-obb	63	93	32

Далі переходимо безпосередньо до процесу навчання моделі для чого потрібно виконати скрипт наведений в лістингу 3.2.

Лістинг 3.2 — Скрипт виконання навчання моделі на створеному датасеті.

```
from ultralytics import YOLO
model = YOLO('yolo11s-obb.pt')
result = model.train(
    data='YOLO_dataset/data.yaml',
    epochs=96,
    imgsz=1024,
    batch=8,
    patience=50,
    augment=True,
    degrees=10,
```

```
    translate=0.1,
    scale=0.5,
    shear=2,
    perspective=0.001,
    flipud=0.5,
    fliplr=0.5,
    mosaic=1.0,
    mixup=0.2,
    lr0=0.001,
    lrf=0.01,
    optimizer='AdamW',
    cos_lr=True,
)
```

Тут параметр data задає шлях до конфігураційного файлу, що містить перелік класів даних та шляхи до папок train і val з даними для тренування та валідації. Через epoch задається кількість епох, або інакше кажучи циклів проходу по датасету, imgsz визначає розмір, до якого масштабується вхідне зображення перед подачею на вхід нейронної мережі.

Параметр batch визначає розмір пакета даних, який обробляється за одну ітерацію навчання для обчислення градієнтів та оновлення параметрів моделі (рис. 3.6). Використання пакетів такого розміру сприяє оптимізації процесу навчання, забезпечуючи більш стабільну оцінку градієнта порівняно з використанням одного прикладу та є значно ефективнішим за метод, ніж обчислення градієнту на всьому наборі даних одночасно.

Інший важливий параметр augment дозволяє використання технології аугментації даних, яка полягає у модифікацією вихідних зображень за допомогою масштабування, повороту, зміни кольору та освітлення, зміщення, перспективні перетворення, віддзеркалення та змішування кількох зображень в одне. Ціллю всіх цих маніпуляцій є збільшення варіативності та різноманіття вхідних даних, що з одного боку допомагає моделі краще узагальнювати ознаки та покращити якість детекції об’єктів для нових, раніше незадіяних даних та допомагає уникнути перенавчання моделі. По завершенню навчання модель надає результати у вигляді графіків та звітів, які допомагають оцінити її ефективність.

Рисунок 3.6 — Згенерований з використанням аугментації батч

Найбільший інтерес для нас представляє статистика прогресу тренування моделі з інформацією про хід тренування по кожній із епох (рис. 3.7)

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      61/64      5.34G     0.4578     0.3321      1.224          3       1024: 100%|██████████| 22/22 [00:03<00:00,  6.41it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00, 10.45it/s]
                   all         21         21      0.997          1      0.995      0.926

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      62/64      5.34G     0.4371     0.3179       1.18          4       1024: 100%|██████████| 22/22 [00:03<00:00,  6.31it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00, 10.75it/s]
                   all         21         21      0.997          1      0.995      0.927

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      63/64      5.34G     0.4415     0.3173      1.244          4       1024: 100%|██████████| 22/22 [00:03<00:00,  6.29it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00, 10.27it/s]
                   all         21         21      0.997          1      0.995      0.931

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      64/64      5.34G     0.4205     0.3135      1.154          4       1024: 100%|██████████| 22/22 [00:03<00:00,  6.30it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00, 10.91it/s]
                   all         21         21      0.997          1      0.995      0.936

64 epochs completed in 0.086 hours.
Optimizer stripped from runs\obb\train\weights\last.pt, 20.0MB
Optimizer stripped from runs\obb\train\weights\best.pt, 20.0MB
```

Рисунок 3.7 — Статистика прогресу тренування моделі по епохам

Розглянемо отримані ключові метрики та зробимо висновок про ефективність тренування моделі.

Перша метрика box_loss характеризує втрати (неточності) при спробі передбачити координати обмежувальних рамок об’єктів і чим значення менше — тим краще. Упродовж навчання можна спостерігати значне поступове зниження цього параметра: від 1,454 в першій епосі до 0,4205 в останній. Це свідчить, що модель успішно навчається та покращує свою здатність локалізувати об’єкти.

Наступна метрика mAP50 вказує наскільки добре модель вміє знаходити об’єкти на зображенні. Суть її в тому, що при виявленні об’єкта нейронна мережа намагається визначити його границі і створює обмежувальну рамку чи маску, яка візуально його покриває. Таким чином об’єкт вважається правильно знайденим якщо область цієї маски більше чи дорівнює 50% від реальних меж об’єкта. Показник менше 0,6 вважається поганим, а задовільним є значення у 0,8. У нашому випадку це значення становить 0,995 що є дуже гарним показником.

Метрика mAP50-95 аналогічна попередній по змісту з тією лиш різницею, що це є середнє арифметичне на основі розрахунку декількох порогів (0,50; 0,55; 0,60; 0,65; …, 0,90; 0,95), а не одного (0,50). Це найбільш важлива метрика оскільки показує наскільки точно модель визначає межі об’єкта в залежності від значення мінімально допустимого перекриття. Наприкінці навчання цей параметр знаходився в діапазоні від 0,926 до 0,936, що є досить високим показником, а отже модель буде знаходити межі номерних знаків на зображенні з точністю приблизно 93%.

Також на виході ми отримали графік, шо показує, як змінюється якість детекції в залежності від заданого порогу впевненості (рис. 3.8). На ньому по осі X міститься значення впевненості моделі у достовірності своїх результатів, а по осі Y знаходиться F1 оцінка, що дає загальне уявлення про те, наскільки добре модель виявляє об’єкти. Інакше кажучи це показник означає відсоток скільки правильних об’єктів змогла знайти модель відносно усіх виявлених нею об’єктів. Цей параметр критично важливий для розробки програмного забезпечення, адже він визначає рівень впевненості нейронної мережі при розпізнаванні об'єктів.

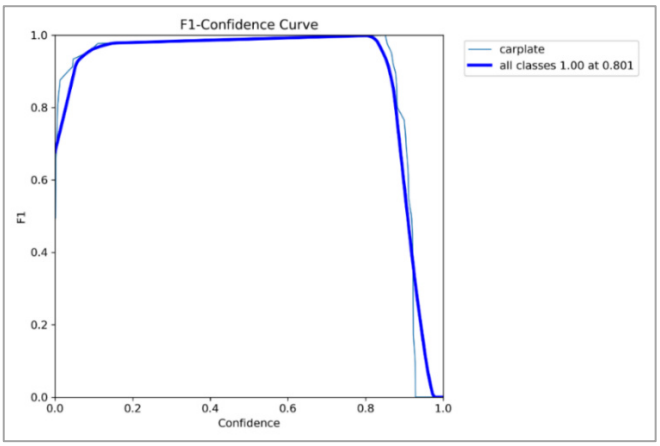


Рисунок 3.8 — Графік залежності F1 оцінки від порогу впевненості

Як видно з графіку, оптимальний поріг впевненості становить 0,8. При вищих порогах модель починає відкидати навіть правильно визначені об’єкти та обмежувальні рамки.

Для оцінки ефективності проведеного навчання проводиться тестування на новому наборі зображень, який не приймав участі в навчанні. По результату цієї процедури можна судити наскільки ефективно модель навчилася розпізнавати номерні знаки. Результат проходження самотестування зображено на рисунку 3.9.

Рисунок 3.9 — Тестування детекції номерних знаків на новому наборі даних

В процесі навчання може виникнути перенавчення моделі, коли вона показує

дуже високі результати на наборі даних, на якому проходила тестування, але погані при обробці тестових зображень. Це трапляється внаслідок того, що модель дуже добре запам'ятовує тренувальний набір даних, фактично запам'ятовує його, а не робить вивчення та узагальнення закономірностей та характерних ознак. Причиною цього може бути замалий датасет, його низька різноманітність або ж завелика кількість епох навчання. Свідченням того, що модель почала перенавчатися є те, що тренувальні втрати box_loss знижуються, тоді як валідаційні втрати перестають падати. Провірити це можна на сумарному графіку, який зображений на рисунку 3.10.

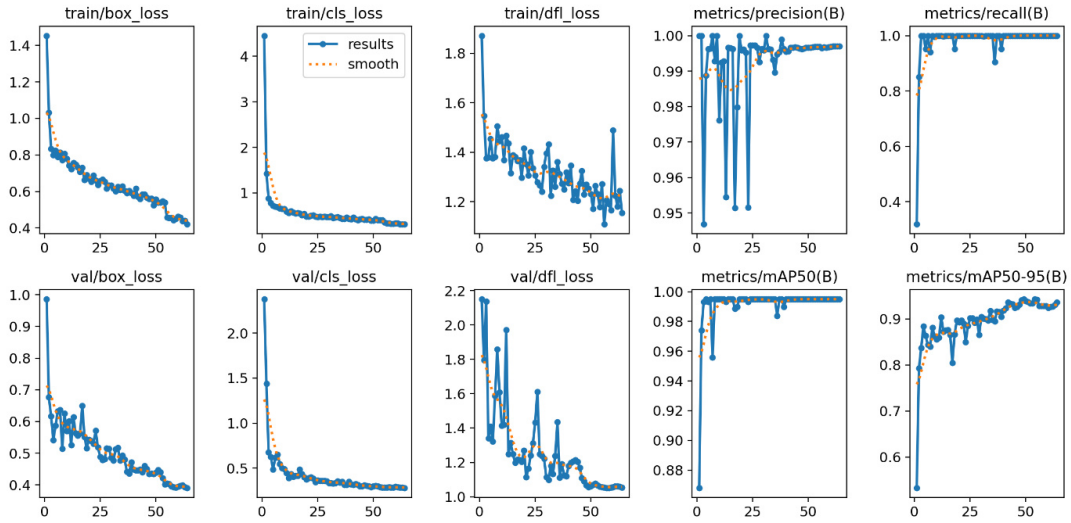


Рисунок 3.10 — Загальні графіки результатів тренування та валідації моделі

Як бачимо, показники train/box_loss та val_box_loss продовжують знижуватися майже за ідентичною кривою, що свідчить про те, що модель продовжує навчатися, і стадія перенавчання ще не досягнута, і в разі потреби кількість епох можна навіть збільшити.

3.4 Розробка програмного засобу розпізнавання номерних знаків

Робота програмного засобу починається із запуску на виконання файлу process.py, який містить опис майже усіх класів програми, за виключенням функцій для роботи з базою даних SQLite. В процесі запуску відбувається імпорт зовнішніх бібліотек у вигляді модулів та зчитування конфігураційного файлу config.py, в

якому містяться глобальні налаштування програмного засобу. Далі починається виконання коду, розташованого після точки входу if __name__ == "__main__". Першим кроком ініціалізується рушій SQLAlchemy Engine, який виконує підключення до бази даних SQLite. Потім за допомогою функції sessionmaker в SQLAlchemy створюється фабрика сесій, яка в подальшому спрощує створення та керування окремими екземплярами сесій. Далі відбувається очищення від старих даних. За це відповідає імпортований метод delete_old_data, який видаляє застарілі записи з бази SQLite та файли зображень вік яких більше кількості днів заданої через змінну DAYS_TO_KEEP. Наступним кроком створюється екземпляр класу LicensePlateSystem і виклик у нього методу process_video, який може обробляти як відеофайл, так і потік з відеокамери. Дуже важливим моментом є те, що виклик методу process_video відбувається через ThreadPoolExecutor, що створює пул потоків, кількість яких визначається автоматично через виклик методу os.cpu_count, який верне кількість процесорних ядер. Це дозволяє ефективно розпаралелити обробку на усі доступні процесорні ядра таким чином максимально задіяти доступні обчислювальні потужності. Це дозволяє не чекати поки буде оброблено поточний кадр а одразу приступити до обробки наступного, поки попередній обробляється в іншому потоці. Також якщо в одному кадрі буде виявлено більше одного номерного знаку, усі вони будуть оброблені не послідовно а паралельно.

LicensePlateSystem є головним класом програмного засобу. В ньому міститься загальна логіка роботи програми. Через свій конструктор приймає шлях до навченої моделі YOLO, потім створює об’єкти класів CarDetector, LicensePlateDetector, ImageProcessor, OCR_Reader, LicensePlateValidator, Utils, Debug через які отримує доступ до їх функцій. Містить метод process_video, який приймає відеодані (відеофайл або RSTP – потік з камери) та об’єкт фабрики сесій. Робота цього методу полягає зчитуванні один за одним кадрів за допомогою методу cv2.VideoCapture, захоплює кадри один за однім і перетворює їх на масив numpy. Цей масив передається в метод process_frame, який застосовує нейронну мережу YOLO для пошуку номерних знаків. Метод повертає координати повернутих

обмежувальних прямокутників для кожного з виявлених номерних знаків. Додатково відбувається фільтрація розмірів обмежувальних прямокутників щоб відкинути завеликі чи занадто малі. Якщо YOLO знайде номерні знаки та поверне результат, то буде визвано внутрішній метод _process_lp, який приймає через параметри поточний кадр зображення та координати обмежувальних рамок та виконає ланцюжок обробки, в якому кожен наступний метод приймає результат роботи попереднього.

Метод crop_skewed_rectangle(frame, lp_points) — статичний внутрішній метод, який приймає кадр зображення, а також список координат вершин обмежувальних рамок та ітерується по ньому. Він вирізає ділянку зображення, яка задається чотирма вершинами прямокутника. Всередині методу реалізовано алгоритм, що сортує точки у правильному порядку (верхній лівий, верхній правий, нижній правий, нижній лівий) якщо раптом цей порядок порушено.

Додатково задача ускладнюється тим, що прямокутник може бути похилим, тобто його сторони не паралельні горизонтальній та вертикальній осі. Це не дозволяє стандартними методами вирізати задану ним ділянку зображення, і для вирішення проблеми потрібно додатково застосувати два методи з бібліотеки OpenCV — getPerspectiveTransform та warpPerspective. Перший з них дозволяє розрахувати матрицю перспективного перетворення, яка визначає, як саме потрібно «повернути площину» вихідного зображення, щоб перетворити похилий прямокутник на звичайний. Інший метод застосовує цю матрицю до зображення та поверне вирізаний з нього фрагмент.

Метод preprocess об’єкта класу Image_Processor — складається із семи методів бібліотеки OpenCV, що застосовуються послідовно до вхідного зображення. Спочатку зліва відрізається приблизно вісім відсотків від ширини зображення номерного знаку, оскільки ця область містить символіку України і не може містити корисної інформації. Потім відбувається приведення розміру зображення до одного стандарту – висоти у 300 пікселів. Для масштабування та згладжування зображення застосовується бікубічна інтерполяція. Оскільки деякі методи бібліотеки OpenCV працюють лише з одноканальним зображенням, то

перетворюємо його у відтінки сірого за допомогою методу cv2.cvtColor, задавши константу cv2.COLOR_BGR2GRAY. Далі відбувається відокремлення об’єктів від фону шляхом виклику методу threshold, який виконує бінаризацію зображення, використовуючи метод Оцу. Наступні два методи – dilate та erode – працюють у парі та виконують розширення і стиснення білих областей, що робить зображення більш чистим, усуваючи шум та інші зайві дрібні елементи.

Метод read_text об’єкта класу OCR_Reader приймає виокремлене та оброблене методом preprocess, зображення номерного знаку та виконує оптичне розпізнавання символів. Метод повертає рядок, що містить розпізнаний текст. Розпізнавання можливе у двох режимах. Один із них запускає Tesseract у режимі --psm 7, в якому номерний знак представляється як один рядок тексту, і сегментація символів виконується безпосередньо самим Tesseract. Інший використовує режим --psm 10, але для його використання потрібно виконати сегментацію власним способом. Саме це робить внутрішній метод _segment_lp, у якому за допомогою cv2.findContours виконується пошук зовнішніх контурів. Оскільки зображення вже є бінарним, то це досить легко виконати. Таким чином отримуємо список контурів, по якому здійснюємо ітерацію, вираховуючи площу та співвідношення сторін обмежувального прямокутника для цього контуру (його отримуємо за допомогою методу cv2.boindingRect). Також відсіюються контури, які не підпадають під розміри та пропорції символів (співвідношення сторін має бути в діапазоні від 0,1 до 1,5 та площа більше 100). Далі сортуємо за координатою осі X та, знаючи завдяки cv2.boundingRect, координати обмежувального прямокутника, вирізаємо кожен символ окремо, додаючи по 30 пікселів білого фону з усіх сторін, це спрощує розпізнавання. Логіка роботи методу наступна: спочатку виконуємо розпізнавання номерного знаку як однієї стрічки тексту. Потім виконується спроба сегментації символів за допомогою методу _segment_lp. Якщо вдалося отримати вісім сегментів, то повертаємо ці сегменти у вигляді списку, інакше — результат від розпізнавання несегментованого зображення. Це менш бажаний варіант, оскільки він більш схильний до помилок, таких як хибні або зайві символи. Якщо жоден результат не задовольняє умовам, то повертається порожній рядок.

Статичний метод find_pattern об’єкта класу LicensePlateValidator — приймає як параметр «сирий» текстовий рядок з номерним знаком, отриманий від OCR Tesseract. Він призначений для коригування типових помилок розпізнавання тексту. Спочатку видаляються з розпізнаного рядка усі зайві та неприпустимі символи, це пробіли, коми, тощо. Потім виконується пошук за шаблоном ([A-Z]{2}\d{4}[A-Z]{2}). Цим перевіряється, чи міститься в отриманому рядку потрібний підрядок, що відповідає шаблону: дві великі літери на початку, за ними чотири цифри та знову дві літери. Якщо знайдено точний збіг, він повертається як результат виклику методу, інакше проводиться додаткова фільтрація.

Якщо текст з номером містить менше восьми символів, він відкидається як некоректно розпізнаний. Якщо більше восьми, то він містить зайві символи, які потрібно видалити. Як правило, цифри в центрі номера досить добре розпізнаються, а проблеми виникають на кінцях. Відповідно робиться спроба знайти чотири цифри однією послідовністю, і якщо їх знайдено, переходимо до наступного кроку, інакше номер відкидається як некоректно розпізнаний. Якщо зліва, чи справа, від цих чотирьох цифр кількість букв більше двох, то крайні букви відсікаються.

Ще один тип реконструкції застосовується якщо виникає помилка оптичного розпізнавання викликана візуальною подібністю символів, коли на місці літери опиняється число, і навпаки. Нам відомо, що на позиціях 0,1,6,7 мають бути літери, а на позиціях 2,3,4,5 — числа. Якщо це не так, то виконується спроба замінити символ на підходящий, подібний за формою. Це не гарантує правильності отриманого результату але робить з завідомо некоректного варіанту більш правдоподібний, який може відповідати реальному номерному знаку.

Набір функцій для роботи з базою даних SQLite було винесено в окремий файл db_funk.py. Це зроблено для уникнення дублювання коду оскільки ці функції використовуються також модулем отримання даних при запитах пошуку номерного знаку.

Метод add_license_plate приймає номерний знак у вигляді текстового рядка та шлях до зображення з автомобілем, з якого цей номерний знак був отриманий та

додає їх в базу. Також додається інформація про поточний час.

Метод search_license_plate виконує пошук в базі даних номера, отриманого через параметр license_plate. У відповідь повертає список кортежів, кожен елемент якого містить в собі час фіксації автомобіля, та шлях до відповідного зображення на диску.

Метод check _record перед додаванням нового запису про до бази даних здійснюється перевірка, чи не був вже він зафіксований нещодавно (протягом визначеного проміжку часу, значення якого задається через константу config.THRESHOLD_DETECT). Це робиться для уникнення дублювання записів, якщо номерний знак був розпізнаний кілька разів поспіль, доки автомобіль знаходився в кадрі.

Метод get_lp_records здійснює пошук записів в базі даних конкретного номерного знаку. У випадку виявлення підходящих записів повертає їх у вигляді списку. Якщо користувач додатково задав інтервал, то пошук буде проводитися тільки в заданому часовому діапазоні.

Метод search_by_pattern за допомогою регулярного виразу проводить пошук в базі даних номерного знаку, який може бути неповним, тобто у ньому можуть бути відсутні один, чи кілька символів. Повертає список усіх записів що відповідають даному шаблону.

Також додатково реалізовано кілька класів з методами, які виконують службову функцію або використовуються в цілях відладки програмного коду.

Utils є службовим класом і містить метод get_timestamp, який використовується для отримання поточного часу. Цей час потрібен, щоб знати, коли саме був зафіксований номерний знак при додаванні запису до бази даних. Другий метод називається gen_image_name. Він генерує з кількох частин ім'я файлу для зображення. Це ім'я включає сам текст номерного знаку і поточну дату та час.

Клас Debug містить метод draw_lp_on_image призначений для оцінки правильності локалізації та розпізнавання номерних знаків. Він додає обмежувальні рамки та текстовий рядок з розпізнаним номерним знаком до зображення з автомобілем та зберігає його в заданому через

config.LP_IMG_FOLDER каталозі.

3.6 Визначення системних вимог

Визначення системних вимог є критично важливим етапом процесу розробки. Вони задають вимоги до продуктивності ключових компонентів обладнання, на яких має працювати програмний засіб. Це необхідно для забезпечення повноцінної роботи системи без шкоди її ефективності чи стабільності. Відповідні мінімальні вимоги на наведено в таблиці 3.1.

Таблиця 3.1 — Мінімальні системні вимоги до апаратного забезпечення

Центральний процесор (CPU)	Intel Core I5 12100, AMD Ryzen 3600
Графічний процесор (GPU)	GeForce RTX 2060
Оперативна пам’ять	8 Гб DDR4 SDRAM
Накопичувач	SSD NVMe 256 Гб

Дуже важливою умовою є наявність високопродуктивного графічного процесора з підтримкою технології CUDA, здатного виконувати паралельні обчислення значно ефективніше, ніж центральні процесори. При роботі з подібними навантаженнями різниця в продуктивності між GPU та CPU може сягати 5-10 разів, що робить останній значно менш придатним для подібних задач.

Оперативна пам’ять використовується як кеш для накопичення кадрів зображення тому її використання залишається завжди на високому рівні і відсутність потрібного об’єму може причиною уповільнення роботи.

Вимоги до програмного забезпечення:

Операційна система: Microsoft Windows 10 або новіша, macOS 10.14 (Mojave) або новіша, дистрибутив на базі Linux (рекомендується: Ubuntu 24.04.2 LTS, Debian 12.10).

Програмне забезпечення: веб-браузер Google Chrome, Mozilla Firefox, Opera, Microsoft Edge. Python 3.9 або новіший.

Інші вимоги: Мережеве підключення зі швидкістю від 100 Мбіт/с є опційною вимогою. Необхідність у ньому виникає у випадках, якщо передбачається віддалений доступ до веб-інтерфейсу або застосовуються камери, що передають дані по протоколу RTSP.

У третьому розділі було реалізовано усі етапи розробки програмного засобу для розпізнавання номерних знаків автомобілів. Вибрано мову програмування Python, як основний інструмент розробки, а середовищем розробки — VS Code, завдяки його гнучкості та розвиненій платформі додаткових модулів. Було розроблено веб-інтерфейс з урахуванням принципів інтуїтивності користування та адаптивності, що значно покращує зручність використання програмного засобу. Проведено навчання моделі YOLO на якісно підготовленому навчальному датасеті та отримано високі показники mAP50 та mAP50-96 при тестуванні, шо підтверджує ефективність обраного підходу.

4. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ АВТОМОБІЛІВ

4.1 Аналіз методів тестування програмного засобу

Етап тестування є критично важливим етапом розробки [23] і повнота та комплексність його проведення буде вирішальним фактором, що забезпечує високу надійність, продуктивність та відповідність заявленим функціям. В даному розділі буде розглянуто деякі методи тестування, використання яких забезпечить реалізацію поставлених задач до якості програмного засобу.

Модульне тестування передбачає перевірку роботи окремих частин програми. Найчастіше це різного роду функції, які приймають на вхід аргументи та повертають результат. У нашому випадку це попередня обробка зображення, локалізація номерного знаку, оптичне розпізнавання символів, збереження результатів в базу. Зазвичай таке тестування проводиться на ранній стадії розробки конкретного модуля програмного засобу і слугує для перевірка коректної роботи кожної із функцій на наборі тестових даних, які часто штучно генеруються розробником та охоплюють специфічні випадки.

Інтеграційне тестування — наступний етап тестування, що проводиться після модульного тестування і зосереджений на перевірці взаємодії та обміну даними між різними компонентами або модулями. Оскільки програмний засіб представляє собою набір взаємопов’язаних функцій модулів, які тим чи іншим чином взаємодіють або залежать один від одного. Для підвищення достовірності результатів бажано відтворити в достатній мірі тестове середовище, що імітує реальні умови експлуатації, це забезпечить більш точний результат . Це дозволить переконатися, що всі компоненти працюють разом коректно.

Системне тестування — це тестування вже відлагодженого програмного застосунку на різних типах та версіях операційних систем (наприклад Windows, Linux чи MacOS) та апаратному забезпеченні (різні типи центральних та графічних процесорів, відеокамер, пристроїв зберігання даних). Метою такого тестування є перевірка стабільності, оцінка продуктивності та сумісності з максимально

широким спектром програмного-апаратних комбінацій.

Повне та якісне виконання вищенаведених методик дасть змогу отримати програмний продукт, що повністю відповідає всім заявленим можливостям, стабільності та продуктивності.

4.2 Тестування якості виявлення та розпізнавання номерних знаків

Для перевірки якості роботи системи розпізнавання номерних знаків буде використано набір тестових зображень автомобілів.

Результати роботи буде представлено у вигляді видозміненого оригінального зображення з нанесеним обмежувальним прямокутником що задає область виявленого номерного знаку, та результат його оптичного розпізнавання.

Спочатку протестуємо роботу на зображенні, на якому автомобіль розташований недалеко від камери, а номерний знак є досить чітким та майже перпендикулярним до спостерігача (рис. 4.1).

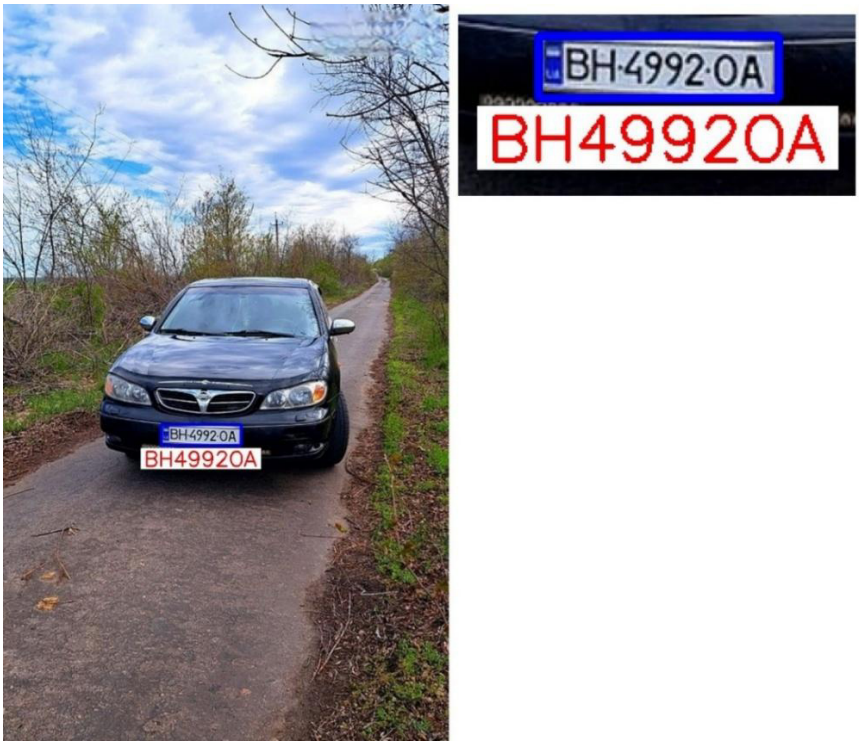


Рисунок 4.1 — Перший етап тестування

На другому етапі тестування використовуються зображення з типом транспортного засобу, зображень якого у навчальному наборі даних для мережі

було досить мало, що теоретично могло ускладнити виявлення номерного знак (рис.4.2).

Рисунок 4.2 — Другий етап тестування

На третьому етапі тестування використовується зображення з номерним знаком, повернутим відносно камери на значний кут, що призводить до помітного спотворення його проекції і ускладнює оптичне розпізнавання символів (рис. 4.3).

Рисунок 4.3 — Третій етап тестування

В четвертому етапі використаємо повернуте на значний кут відносно горизонту зображення (рис 4.4).

Рисунок 4.4 — Четвертий етап тестування

Завдяки використанню повернутих обмежувальних рамок [21] стає можливим чітке виділення номерного знаку без захоплення сусідніх ділянок.

На останньому етапі тестування використаємо зображення, що містить кілька автомобілів одночасно (рис. 4.5).

Рисунок 4.5 — П’ятий етап тестування

Як бачимо, в усіх випадках було правильно визначено положення номерних знаків, а система OCR Tesseract правильно виконала розпізнавання символів. Однак, як видно на деяких зразках, через те, що розмір номерного знаку на зображенні дуже малий, спостерігається значна пікселізація, яка спричиняє спотворення символів. Це, у свою чергу, може бути причиною збільшення кількості помилок у процесі розпізнавання. Тому дуже важливо, щоб відеокамера, яка здійснює відеофіксацію, відповідала мінімальним вимогам до яких належить роздільна здатність не менше HD Ready (1280×720 пікселів), швидка та якісна матриця, що буде створювати мінімальне розмивання зображення автомобілів, що швидко рухаються. Саме за таких умов якість зображення буде достатньою для правильного визначення форми та границь символів.

У цьому розділі було визначено вимоги до проведення тестування, яке є важливим етапом розробки та забезпечує відповідність розробленої системи вимогам до якості та стабільності. Тестування передбачає комплексний підхід, що охоплює різні види перевірки — модульний, інтеграційний та системний, кожен з яких виконуються на різних етапах розробки програмного засобу. Завершальним етапом розробки було тестування програмного засобу на різноплановому наборі зображень, в результаті чого було отримано результати, що свідчили про коректну роботу усіх модулів і високі показники точності локалізації та розпізнавання автомобільних номерних знаків.

ВИСНОВКИ

В процесі виконання бакалаврської дипломної роботи було розроблено програмний засіб розпізнавання автомобільних номерів, що здатний виявляти, розпізнавати та зберігати номерні знаки. Додатково реалізовано зручний та інтуїтивно зрозумілий веб – інтерфейсом із доступом до широких можливостей пошуку та перегляду.

У першому розділі наведено історію появи систем автоматичного розпізнавання номерних знаків, розглянуто різні методи машинного зору, проаналізовано їх переваги та недоліки. Також було проведено аналіз та порівняння існуючих комерційних рішень, що дало змогу виявити їх недоліки та сформувати список вимог та функцій, до програмного засобу, що розробляється.

Другий розділ містить обґрунтування вибору конкретних методів та технологій для процесів сегментації зображення та розпізнавання символів номерного знаку. Далі на їх базі розроблено модель та алгоритм функціонування програмного засобу. Це дало міцну основу для наступних етапів розробки.

У третьому розділі описано призначення класів та роботу методів програмного засобу. Обґрунтовано вибір середовища розробки та додаткових модулів, які підвищили зручність та продуктивність при написанні коду. Для навчання моделі нейронної мережі YOLO було створено якісний та різноманітний датасет. Також було розроблено веб – інтерфейс для реалізації запланованих функцій по практичному використанню отриманих даних.

У четвертому розділі було розроблено комплексний підхід до тестування якості програмного засобу. Його виконання дозволило отримати програмний засіб, що відповідає всім заявленим вимогам та функціям. Відповідність підтверджено тестуванням на різноплановому наборі зображень автомобілів, яке продемонструвало високу точність та ефективність розробленого програмного засобу.

ПЕРЕЛІК ДЖЕРЕД ПОСИЛАННЯ

1. Системи машинного зору роботів. Історія розвитку: вебсайт. :URL https://robotics.ua/systemy-mashynnoho-zrenyia.-ystoryia-prymery-plany/.

2. Що таке комп'ний зір: вебсайт. URL: https://robotics.ua/systemy-mashynnoho-zrenyia.-ystoryia-prymery-plany/.

3. Солодюк В. П., Мартинюк Т. Б., Очкуров М. А.. Програмний засіб для ком'ютерної системи розпізнавання номерних знаків автомобілів. // LIV Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025). Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24058/19938.

4. Що таке машинне навчання: вебсайт. URL: https://qudata.com/uk/blog/what-is-machine-learning/.

5. У чому різниця між машинним та глибоким навчанням. URL : https://qudata.com/uk/blog/machine-learning-vs-deep-learning-know-the-differences/

6. LPR-камера ZIP-5241DLM-X12CP: вебсайт. URL: https://kobi.ua/product/smart-ip-kamera-2mp-zip-5241dlm-x12cp-lpr-onboard/.

7. OpenCV documentation: вебсайт. URL:https://docs.opencv.org/4.11.0/index.html

8. Суботтін С.О. Нейронні мережі: теорія та практика: навч. посіб. / С.О. Субботін. – Житомир: Вид. О. О. Євенюк, 2020. – 184 с.

9. Morphological Operations In image Processing: вебсайт. URL: https://blog.roboflow.com/morphological-operations/.

10. Tesseract. User Manual: вебсайт. URL: https://tesseract-ocr.github.io/tessdoc/

11. Erosion and Dilation in Image Processing: вебсайт. URL: https://www.scaler.com/topics/erosion-and-dilation-in-image-processing/.

12. Introduction to Convolution Neural Network: вебсайт. URL: https://www.geeksforgeeks.org/introduction-convolution-neural-network/.

13. Kaiming He, Ross Girshick, Jian SunFaster. R-CNN: Towards Real-Time

Object Detection with Region Proposal Networks: вебсайт. URL: https://arxiv.org/pdf/1506.01497v3.pdf.

14. Object detection algorithms: R-CNN, Fast R-CNN, Faster R-CNN, and YOLO: вебсайт. URL: https://www.analyticsvidhya.com/blog/2024/07/object-detection-algorithms/.

15. Performance comparasion of YOLO object Detection Models: вебсайт. URL: https://learnopencv.com/performance-comparison-of-yolo-models/.

16. Jason Myers, Rick Copeland. Essential SQLAlchemy. Mapping python to databases. 2 edition 2016 p.

17. Eli Stevensen, Luca Antiga, Thomas Viehmann. Deep Learning with PyTorch. 2020 p.

18. VS Code: вебсайт. URL: https://code.visualstudio.com/docs

19. CVAT official documentation: сайт. URL: https://docs.cvat.ai/docs/manual/advanced/

20. COCO to YOLO format converter: вебсайт. URL: https://github.com/Koldim2001/COCO_to_YOLOv8

21. YOLO OBB model comparasion: вебсайт. URL: https://docs.ultralytics.com/ru/tasks/obb/#models.

22. Mohamed Elgendy. Grookking Deep Learning for Computer Vision. 2019 p. 112 – 150.

23. Азаров О.Д., Богомолов С.В., Швець С.І. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 123 «Комп'ютерна інженерія» : методичні вказівники Вінниця: ВНТУ, 2023.

ДОДАТОК А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

д.т.н., проф. Азаров О. Д.

« 21 » березня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

«Програмний засіб розпізнавання автомобільних номерів для комп’ютерної системи відеоспостереження»

08-54.БКР.017.00.000.ТЗ

Науковий керівник: професор, д.т.н.

_____ Мартинюк Т. Б.

Студент групи 1КІ-23мс

_____ Солодюк В. П.

Вінниця ВНТУ – 2025 рік

1 Підставою для виконання бакалаврської кваліфікаційної роботи (БКР)

1.1 Актуальність розробки програмного засобу для розпізнавання автомобільних номерів у комп’ютерних системах відеоспостереження зумовлена зростаючою потребою в спеціалізованих автоматизованих системах контролю та відстеженню транспортних засобів. Такі системи широко застосовуються на пунктах контролю доступу до закритих територій, для фіксації порушень правил дорожнього руху та правоохоронними органами.

1.2 Наказ про затвердження теми БКР

2 Мета БКР і призначення розробки

2.1 Мета роботи — створення програмного засобу, здатного ефективно виконувати задачу по пошуку та розпізнавання номерного знаку автомобіля на зображенні, отриманому з відеокамери. Зберігати отримані результати і надавати зручний та інтуїтивно зрозумілий графічний інтерфейс для взаємодії з користувачем.

2.2 Практичне застосування — розроблений програмний засіб має значний потенціал для практичного запровадження в системи відеоспостереження, де існує потреба в автоматизації контролю доступу, а також для виявлення підозрілих або викрадених транспортних засобів. За можливості інтеграції з іншими спеціалізованими системами може бути застосований для фіксації порушень правил дорожнього руху чи моніторингу дорожнього трафіку.

3 Вихідні дані для виконання БКР

3.1 Аналіз сучасних технологій та методів виділення об’єктів на зображенні;

3.2 Визначення модулів програмного засобу та їх функцій;

3.3 Навчання нейронної мережі YOLO на підготовленому датасеті;

3.4 Реалізація модулів програмного засобу.

4 Вимоги до виконання БКР

Головна вимога — розроблений програмний засіб повинен відповідати усім

заявленим функціям, мати високі показники точності локалізації та розпізнавання символів з номерного знаку. Працювати стабільно на різних типах операційних систем та апаратному забезпеченні. Надавати зручний та інтуїтивно зрозумілий інтерфейс для взаємодії.

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 – Етапи БКР

№ етапу	Назва етапу	Термін виконання	Очікувані результати
1	Аналіз завдання. Вступ	21.03.2025 — 23.03.2025	Формулювання завдань та цілей
2	Аналіз технологій та методів пошуку об’єктів на зображенні, пошук літературних джерел	24.03.2025 — 29.03.2025	Розділ 1
3	Аналіз та вибір методів локалізації та сегментації номерних знаків. Огляд існуючих програмних рішень	30.03.2025— 5.04.2025	Розділ 1
4	Розробка послідовності та методів сегментації об’єктів, способу розпізнавання символів	6.04.2025— 13.04.2025	Розділ 2
5	Розробка програмного засобу, інтерфейсу користувача, проведення навчання нейронної мережі.	14.04.2025— 26.04.2025	Розділ 3
6	Тестування готового програмного засобу на відповідність заявленим функціям та показникам ефективності	27.04.2025— 3.05.2025	Розділ 4
7	Оформлення пояснювальної записки та ілюстративної частини	4.05.2025— 12.05.2025	ПЗ, графічний матеріал, презентація
8	Попередній захист БКР та усунення недоліків	13.05.2025	Усунення зауважень

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, ілюстративні матеріали,

протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, анотації до БКР українською та іноземною мовами, довідка про відповідність оформлення БКР діючим вимогам.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлювання БКР використовуються

— ДСТУ 3008: 2015 "Звіти в сфері науки і техніки. Структура та правила оформлювання";

— ДСТУ 8302: 2015 "Бібліографічні посилання. Загальні положення та правила складання";

— міждержавний ГОСТ 2.104-2006 “Єдина система конструкторської документації. Основні написи”;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 – "Комп’ютерна інженерія" (освітня програма

“Комп’ютерна інженерія”). Кафедра обчислювальної техніки ВНТУ 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в “Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21”.

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний засіб розпізнавання автомобільних номерів для комп'ютерної системи відеоспостереження.

Тип роботи: _____ бакалаврська кваліфікаційна робота _____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism __1__%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

____Крупельницький Л.В., доц. каф ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ___Захарченко С.М.____
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

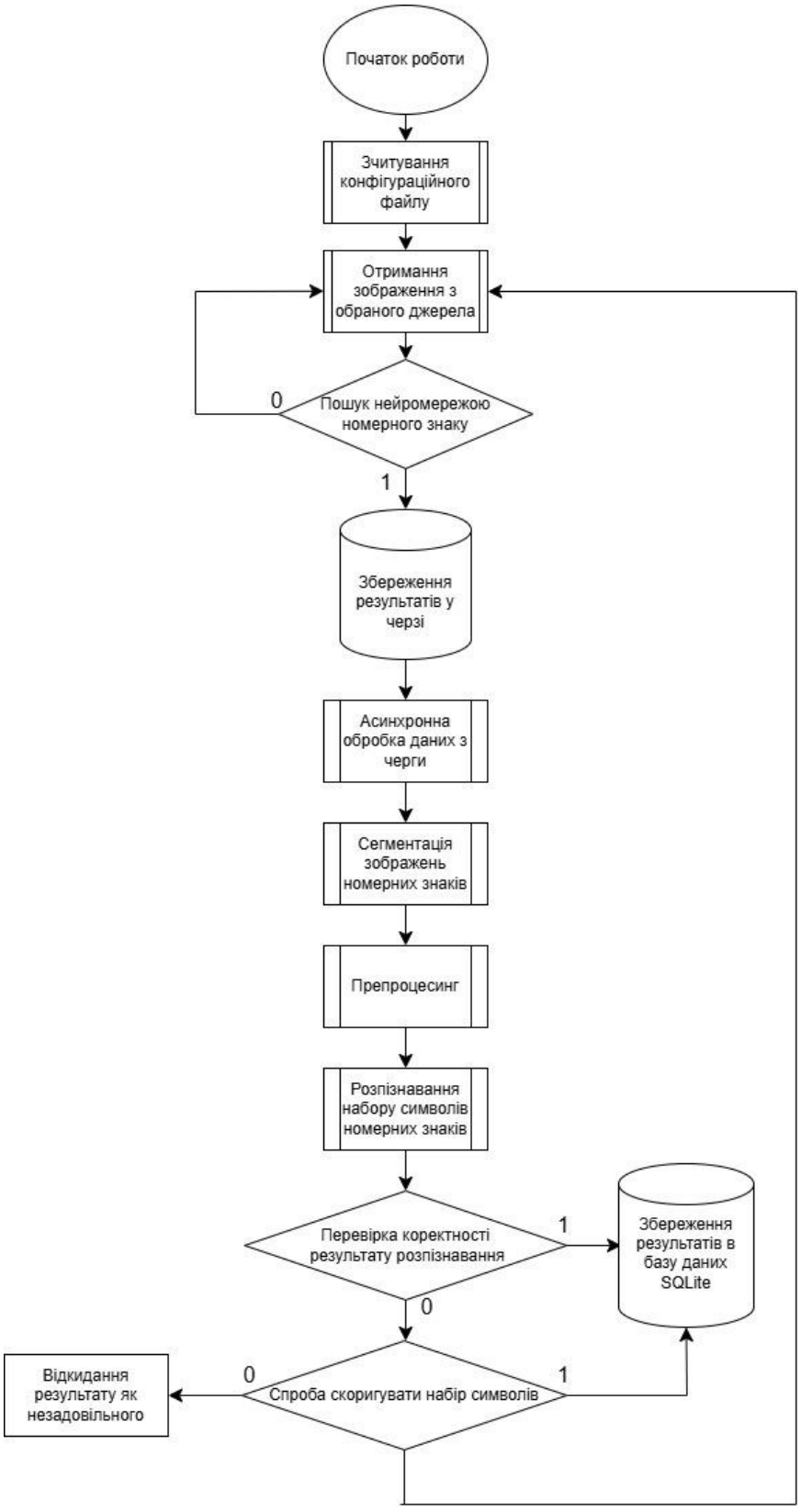
Керівник _____ _____Мартинюк Т.Б., проф. кафедри ОТ_____
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____Солодюк В.П._____
(підпис) (прізвище, ініціали)

ДОДАТОК В

Графічна частина

ПРОГРАМНИЙ ЗАСІБ РОЗПІЗНАВАННЯ АВТОМОБІЛЬНИХ НОМЕРІВ ДЛЯ КОМП’ЮТЕРНОЇ СИСТЕМИ ВІДЕОСПОСТЕРЕЖЕННЯ



Рисунку В.1 — Алгоритм роботи програмного засобу

ДОДАТОК Г

Ілюстративна частина

ПРОГРАМНИЙ ЗАСІБ РОЗПІЗНАВАННЯ АВТОМОБІЛЬНИХ НОМЕРІВ ДЛЯ КОМП’ЮТЕРНОЇ СИСТЕМИ ВІДЕОСПОСТЕРЕЖЕННЯ

Рисунок Г.1 — Середовище розробки та бібліотеки

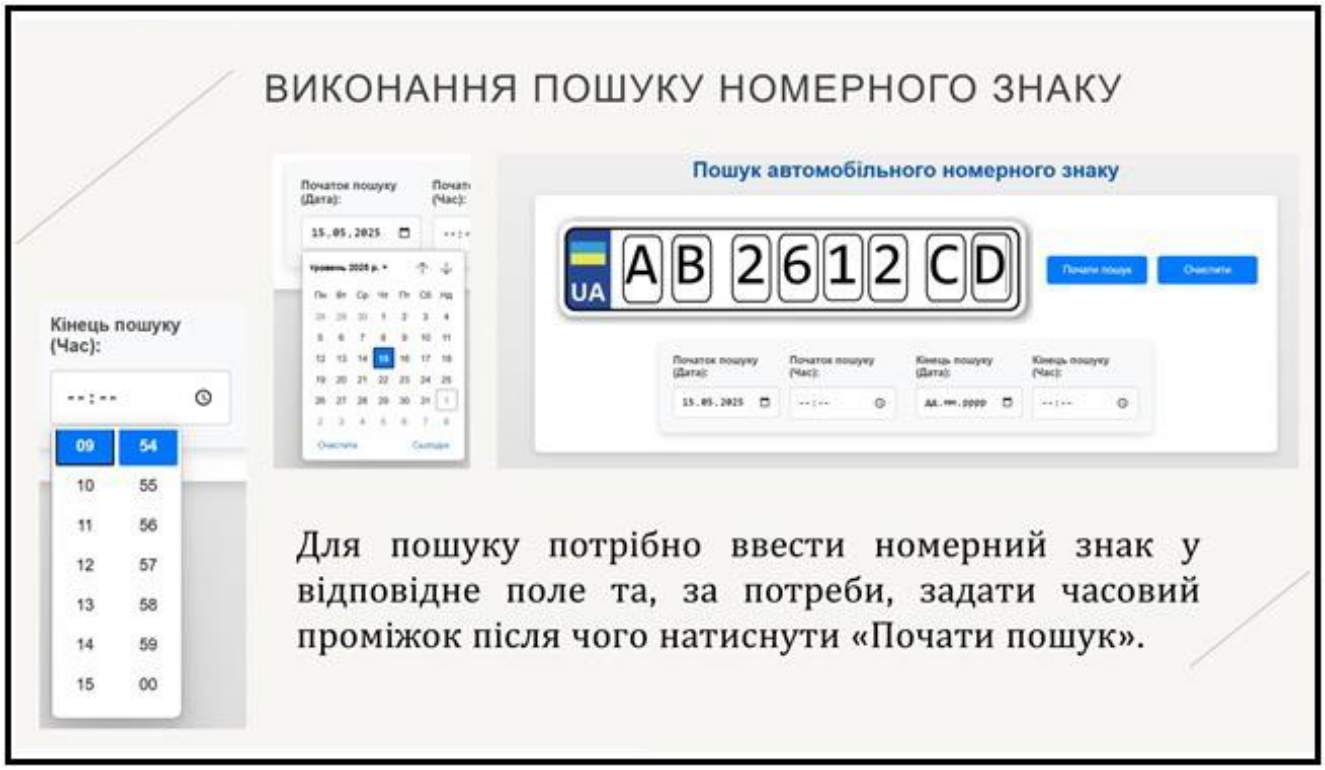


Рисунок Г.2 —Виконання пошуку номерного знаку

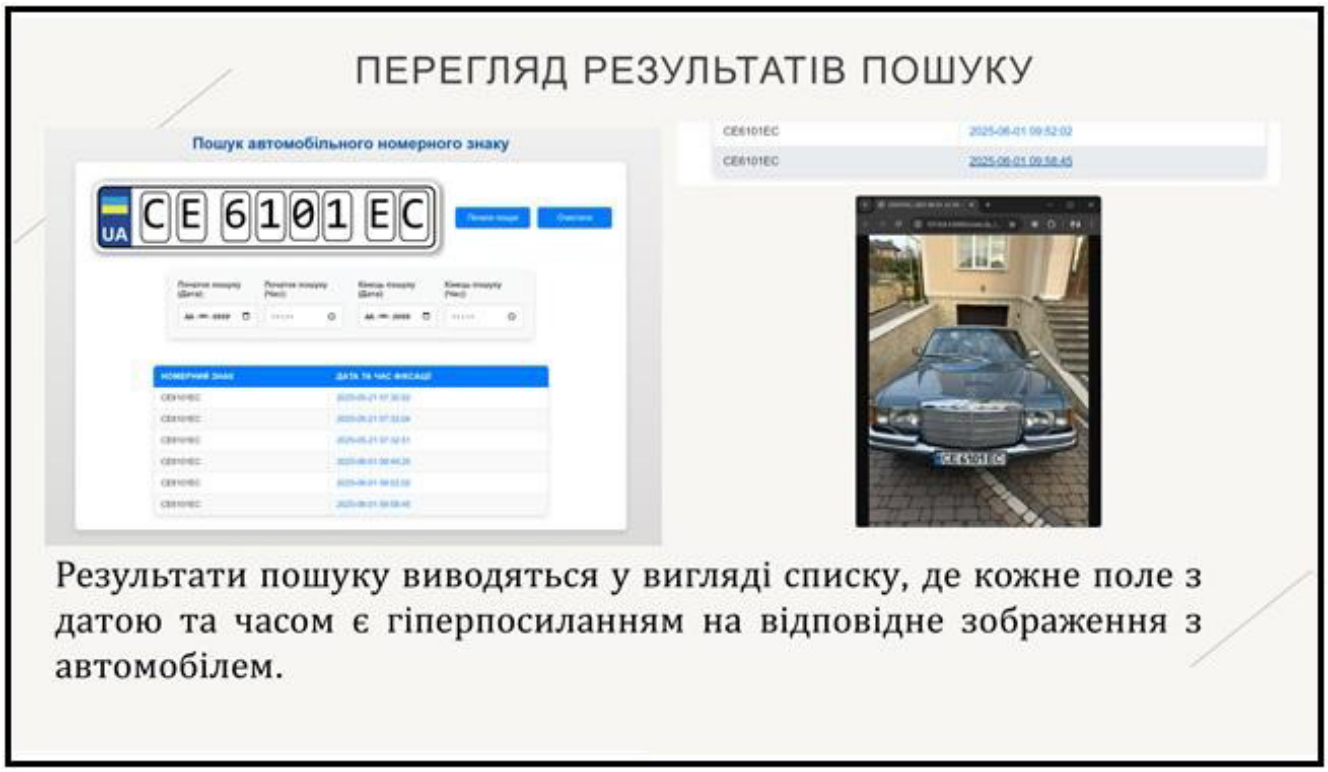


Рисунок Г.3 — Перегляд результатів пошуку

ДОДАТОК Д

Лістинг програмного застосунку

```
import cv2
import concurrent.futures
from typing import List, Tuple, Optional
from ultralytics import YOLO
import numpy as np
import config
import random
import os
import re
import pytesseract
from abc import ABC, abstractmethod
from db_func import create_database_tables, add_license_plate, check_record, delete_old_data
from datetime import datetime
import torch
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

pytesseract.pytesseract.tesseract_cmd = 'C:\\Program Files\\Tesseract-OCR\\tesseract.exe'

class ObjectDetector(ABC):
    def __init__(self, model_path: str):
        self.model = YOLO(model_path).to('cuda' if torch.cuda.is_available() else 'cpu')
    @abstractmethod
    def detect(self, frame):
        pass

class CarDetector(ObjectDetector):
    def detect(self, frame) -> List[List[float]]:
        cars_list = []
        cars_result = self.model.predict(frame, verbose=False)
        find_cars = cars_result[0].boxes.data.cpu().tolist()
        for car in find_cars:
            x1, y1, x2, y2, score, class_id = car
            if int(class_id) in config.VEHICLES and score >= config.THRESHOLD:
                cars_list.append([x1, y1, x2, y2])
        return cars_list

class LicensePlateDetector(ObjectDetector):
    def detect(self, frame, cars_list: List[List[float]]) -> Tuple[List[np.ndarray], List[np.ndarray]]:
        lp_obb_list_filtered = []
        lp_bb_list_filtered = []
        lp_results = self.model.predict(frame, verbose=False)
        if not hasattr(lp_results[0], 'obb') or lp_results[0].obb is None or len(lp_results[0].obb.xyxyxyxy)
== 0:
            return lp_obb_list_filtered, lp_bb_list_filtered
        find_obb = lp_results[0].obb.xyxyxyxy.cpu().numpy()
        find_bb = lp_results[0].obb.xyxy.cpu().numpy()
        scores = lp_results[0].obb.conf.cpu().numpy()
```

```
        for i in range(len(find_obb)):
            lp_points = find_obb[i]
            lp_box_bb = find_bb[i]
            lp_score = scores[i]
            if lp_score < config.THRESHOLD:
                continue
            x1, y1, x2, y2 = lp_box_bb
            lp_width = x2 - x1
            lp_height = y2 - y1
            if (config.MIN_LP_WIDTH is not None and lp_width < config.MIN_LP_WIDTH) or \
               (config.MAX_LP_WIDTH is not None and lp_width > config.MAX_LP_WIDTH) or \
               (config.MIN_LP_HEIGHT is not None and lp_height < config.MIN_LP_HEIGHT) or \
               (config.MAX_LP_HEIGHT is not None and lp_height > config.MAX_LP_HEIGHT):
                continue
            lp_center_x = np.mean(lp_points[::2])
            lp_center_y = np.mean(lp_points[1::2])
            is_inside_car = False
            if config.CHECK_CAR and cars_list:
                for car_box in cars_list:
                    car_x1, car_y1, car_x2, car_y2 = car_box
                    if car_x1 <= lp_center_x <= car_x2 and car_y1 <= lp_center_y <= car_y2:
                        is_inside_car = True
                        break
            if not config.CHECK_CAR or is_inside_car:
                lp_obb_list_filtered.append(lp_points)
                lp_bb_list_filtered.append(lp_box_bb)
        return lp_obb_list_filtered, lp_bb_list_filtered

class ImageProcessor:
    @staticmethod
    def crop_skewed_rectangle(frame: np.ndarray, lp_points) -> np.ndarray:
        if frame is None or frame.size == 0:
            return np.array([])
        points = np.array(lp_points, dtype=np.float32).reshape(-1, 2)
        rect = np.zeros((4, 2), dtype=np.float32)
        s = points.sum(axis=1)
        diff = np.diff(points, axis=1)
        rect[0] = points[np.argmin(s)]
        rect[2] = points[np.argmax(s)]
        rect[1] = points[np.argmin(diff)]
        rect[3] = points[np.argmax(diff)]
        width_a = np.linalg.norm(rect[0] - rect[1])
        width_b = np.linalg.norm(rect[2] - rect[3])
        max_width = int(max(width_a, width_b))
        height_a = np.linalg.norm(rect[0] - rect[3])
        height_b = np.linalg.norm(rect[1] - rect[2])
        max_height = int(max(height_a, height_b))
        if max_width <= 0 or max_height <= 0:
            return np.array([])
        dst_points = np.array([
            [0, 0],
            [max_width - 1, 0],
```

```
            [max_width - 1, max_height - 1],
            [0, max_height - 1]
        ], dtype=np.float32)
        M = cv2.getPerspectiveTransform(rect, dst_points)
        cropped_image = cv2.warpPerspective(frame, M, (max_width, max_height))
        return cropped_image

    @staticmethod
    def preprocess(lp_image: np.ndarray) -> np.ndarray:
        if lp_image is None or lp_image.size == 0:
            return np.array([])
        gray = cv2.cvtColor(lp_image, cv2.COLOR_BGR2GRAY) if len(lp_image.shape) == 3 else
lp_image
        crop_percentage = 0.08
        _, original_width = gray.shape[:2]
        pixels_to_crop = int(original_width * crop_percentage)
        if pixels_to_crop >= original_width:
            pixels_to_crop = original_width - 1
            if pixels_to_crop < 0: pixels_to_crop = 0
        cropped_image = gray[:, pixels_to_crop:]
        if cropped_image is None or cropped_image.size == 0:
            return np.array([])
        desired_height = 300
        cropped_height, cropped_width = cropped_image.shape[:2]
        if cropped_height == 0:
            return np.array([])
        aspect_ratio = cropped_width / cropped_height
        desired_width = int(desired_height * aspect_ratio)
        if desired_width <= 0 or desired_height <= 0:
            return np.array([])
        resized = cv2.resize(cropped_image, (desired_width, desired_height),
interpolation=cv2.INTER_CUBIC)
        if np.max(resized) == np.min(resized):
            thresh = np.zeros_like(resized) if np.max(resized) < 128 else np.full_like(resized, 255)
        else:
            _, thresh = cv2.threshold(resized, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
        kernel = np.ones((2, 2), np.uint8)
        dilated = cv2.dilate(thresh, kernel, iterations=1)
        erode = cv2.erode(dilated, kernel, iterations=1)
        if config.DEBUG:
            timestamp = datetime.now().strftime("%Y%m%d%H%M%S%f")
            debug_dir = 'debug_images'
            os.makedirs(debug_dir, exist_ok=True)
            cv2.imwrite(os.path.join(debug_dir, f'original_lp_{timestamp}.png'), lp_image)
            cv2.imwrite(os.path.join(debug_dir, f'preprocessed_lp_{timestamp}.png'), erode)
        return erode

class OCR_Reader:
    def read_text(self, preproc_img: np.ndarray) -> str:
        if preproc_img is None or preproc_img.size == 0:
            return ""
        ocr_text = pytesseract.image_to_string(
```

```
            preproc_img,
            config=config.CUSTOM_TESSERACT_CONFIG_A
        ).strip()
        segmented_lp = ''
        segmented_img_list = OCR_Reader._segment_lp(preproc_img)
        if segmented_img_list:
            for img_char in segmented_img_list:
                if img_char is not None and img_char.size > 0:
                    char = pytesseract.image_to_string(
                        img_char,
                        config=config.CUSTOM_TESSERACT_CONFIG_B
                    ).strip()
                    segmented_lp += char
        if segmented_lp and len(segmented_lp) == 8:
            return segmented_lp
        elif ocr_text:
            return ocr_text
        else:
            return ""

    @staticmethod
    def _segment_lp(image: np.ndarray) -> List[np.ndarray]:
        if image is None or image.size == 0:
            return []
        binary = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
if len(image.shape) == 2 else image
        kernel = np.ones((3, 3), np.uint8)
        binary = cv2.morphologyEx(binary, cv2.MORPH_CLOSE, kernel)
        binary = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)
        contours, _ = cv2.findContours(binary.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        potential_contours = []
        for contour in contours:
            x, y, w, h = cv2.boundingRect(contour)
            aspect_ratio = w / float(h) if h > 0 else 0
            area = cv2.contourArea(contour)
            if area > 50 and 0.1 < aspect_ratio < 2.0:
                potential_contours.append((x, y, w, h, area))
        if not potential_contours:
            return []
        heights = [h for _, _, _, h, _ in potential_contours]
        areas = [area for _, _, _, _, area in potential_contours]
        median_height = np.median(heights) if heights else 0
        median_area = np.median(areas) if areas else 0
        character_contours = []
        for x, y, w, h, area in potential_contours:
            if median_height > 0 and median_area > 0:
                if (0.5 * median_height <= h <= 1.5 * median_height) and area >= 0.3 * median_area:
                    character_contours.append((x, y, w, h))
            else:
                character_contours.append((x, y, w, h))
        character_contours.sort(key=lambda x: x[0])
```

```
    segmented_chars = []
    for (x, y, w, h) in character_contours:
        char_img = image[y:y+h, x:x+w]
        if char_img is not None and char_img.size > 0:
            padding = 30
            char_img_padded = cv2.copyMakeBorder(char_img, padding, padding, padding, padding,
cv2.BORDER_CONSTANT, value=255)
            segmented_chars.append(char_img_padded)
    return segmented_chars

class LicensePlateValidator:
    @staticmethod
    def find_pattern(raw_lp_number: str) -> Optional[str]:
        if not raw_lp_number:
            return None
        raw_lp_number = raw_lp_number.replace(" ", "").upper()
        whitelist = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        filtered_lp = ''.join(c for c in raw_lp_number if c in whitelist)
        if not filtered_lp:
            return None
        potential_match = None
        pattern_strict = r"([A-Z]{2}\d{4}[A-Z]{2})"
        regex_strict = re.compile(pattern_strict)
        match_strict = regex_strict.search(filtered_lp)
        if match_strict:
            potential_match = match_strict.group()
        else:
            digits_pattern = r"\d{4}"
            digits_match = re.search(digits_pattern, filtered_lp)
            if digits_match:
                digits_start = digits_match.start()
                digits_end = digits_match.end()
                digits_part = filtered_lp[digits_start:digits_end]
                prefix_raw = filtered_lp[:digits_start]
                prefix = ''.join(c for c in prefix_raw if c.isalpha())
                if len(prefix) > 2:
                    prefix = prefix[-2:]
                suffix_raw = filtered_lp[digits_end:]
                suffix = ''.join(c for c in suffix_raw if c.isalpha())
                if len(suffix) > 2:
                    suffix = suffix[:2]
                reconstructed_lp = prefix + digits_part + suffix
                if len(reconstructed_lp) == 8 and prefix.isalpha() and len(prefix) == 2 and \
                    digits_part.isdigit() and len(digits_part) == 4 and \
                    suffix.isalpha() and len(suffix) == 2:
                    potential_match = reconstructed_lp
                else:
                    if len(filtered_lp) == 8:
                        potential_match = filtered_lp
        if potential_match is None or len(potential_match) != 8:
            return None
        correction_letter = {
```

```
            "0": "O", "1": "I", "8": "B", "4": "A"
        }
        correction_digit = {
            "O": "0", "I": "1", "T": "7", "L": "1", "S": "5", "B": "8"
        }
        corrected_lp_chars = list(potential_match)
        for i in range(8):
            char = corrected_lp_chars[i]
            if i in [0, 1, 6, 7]:
                if char.isdigit():
                    corrected_lp_chars[i] = correction_letter.get(char, char)
            elif i in [2, 3, 4, 5]:
                if char.isalpha():
                    corrected_lp_chars[i] = correction_digit.get(char, char)
        correct_lp = ''.join(corrected_lp_chars)
        if len(correct_lp) == 8 and correct_lp[0:2].isalpha() and \
            correct_lp[2:6].isdigit() and correct_lp[6:8].isalpha():
            return correct_lp
        else:
            return None

class Utils:
    @staticmethod
    def get_timestamp() -> str:
        timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
        return timestamp

    @staticmethod
    def gen_image_name(lp_number: str) -> str:
        timestamp = Utils.get_timestamp()
        rnd_number = random.randint(100, 999)
        safe_lp_number = re.sub(r'[^\w\s.-]', '_', lp_number)
        safe_lp_number = safe_lp_number.replace(' ', '_')
        filename = f"{safe_lp_number}_{timestamp}_{rnd_number}.jpg"
        return filename

class Debug:
    @staticmethod
    def draw_lp_on_image(frame: np.ndarray, bb, lp_string: str, filename: str) -> None:
        if frame is None or frame.size == 0:
            return
        output_img = frame.copy() if not frame.flags['WRITEABLE'] else frame
        if bb is None or len(bb) != 4:
            if config.DEBUG:
                debug_dir = 'debug_images'
                os.makedirs(debug_dir, exist_ok=True)
                timestamp = datetime.now().strftime("%Y%m%d%H%M%S%f")
                safe_filename = re.sub(r'[^\w\s.-]', '_', filename)
                save_path = os.path.join(debug_dir, f'frame_no_draw_{timestamp}_{safe_filename}.png')
                cv2.imwrite(save_path, output_img)
            return
        x_min, y_min, x_max, y_max = map(int, bb)
```

```
        h, w = output_img.shape[:2]
        x_min = max(0, x_min)
        y_min = max(0, y_min)
        x_max = min(w - 1, x_max)
        y_max = min(h - 1, y_max)
        if x_max <= x_min or y_max <= y_min:
            if config.DEBUG:
                debug_dir = 'debug_images'
                os.makedirs(debug_dir, exist_ok=True)
                timestamp = datetime.now().strftime("%Y%m%d%H%M%S%f")
                safe_filename = re.sub(r'[^\w\s.-]', '_', filename)
                save_path = os.path.join(debug_dir,
f'frame_no_draw_invalid_coords_{timestamp}_{safe_filename}.png')
                cv2.imwrite(save_path, output_img)
            return
        cv2.rectangle(output_img, (x_min, y_min), (x_max, y_max), (255, 0, 0), 3)
        if lp_string:
            text_size, _ = cv2.getTextSize(lp_string, cv2.FONT_HERSHEY_SIMPLEX, 1.2, 2)
            text_x = (x_min + x_max - text_size[0]) // 2
            text_y = y_max + text_size[1] + 10
            text_x = max(0, text_x)
            text_y = max(text_size[1], text_y)
            text_y = min(h - 1, text_y)
            text_background_x1 = text_x - 5
            text_background_y1 = text_y - text_size[1] - 5
            text_background_x2 = text_x + text_size[0] + 5
            text_background_y2 = text_y + 5
            text_background_x1 = max(0, text_background_x1)
            text_background_y1 = max(0, text_background_y1)
            text_background_x2 = min(w, text_background_x2)
            text_background_y2 = min(h, text_background_y2)
            cv2.rectangle(output_img, (text_background_x1, text_background_y1), (text_background_x2,
text_background_y2), (255, 255, 255), -1)
            cv2.putText(output_img, lp_string, (text_x, text_y), cv2.FONT_HERSHEY_SIMPLEX, 1.2,
(0, 0, 255), 2)
        if config.DEBUG:
            debug_dir = 'debug_images'
            os.makedirs(debug_dir, exist_ok=True)
            timestamp = datetime.now().strftime("%Y%m%d%H%M%S%f")
            safe_filename = re.sub(r'[^\w\s.-]', '_', filename)
            save_path = os.path.join(debug_dir,
f'detected_{lp_string}_{timestamp}_{safe_filename}.png')
            cv2.imwrite(save_path, output_img)

class LicensePlateSystem:
    def __init__(self, car_model_path: str, license_plate_model_path: str):
        self.car_detector = CarDetector(car_model_path)
        self.license_plate_detector = LicensePlateDetector(license_plate_model_path)
        self.image_processor = ImageProcessor()
        self.ocr_reader = OCR_Reader()
        self.validator = LicensePlateValidator()
        self.utils = Utils()
```

```
        self.debug = Debug()

    def _process_lp(self, frame: np.ndarray, lp_obb, lp_bb, session_maker):
        db = session_maker()
        try:
            lp_img = self.image_processor.crop_skewed_rectangle(frame, lp_obb)
            if lp_img is None or lp_img.size == 0:
                return
            processed_img = self.image_processor.preprocess(lp_img)
            if processed_img is None or processed_img.size == 0:
                return
            raw_lp_string = self.ocr_reader.read_text(processed_img)
            if not raw_lp_string:
                return
            lp_string = self.validator.find_pattern(raw_lp_string)
            if not lp_string:
                return
            if not check_record(db, lp_string):
                path_lp_img_filename = self.utils.gen_image_name(lp_string)
                path_lp_img_full = os.path.join(config.LP_IMG_FOLDER, path_lp_img_filename)
                if config.DEBUG and lp_bb is not None:
                    self.debug.draw_lp_on_image(frame, lp_bb, lp_string, path_lp_img_filename)
                elif not config.DEBUG:
                    os.makedirs(config.LP_IMG_FOLDER, exist_ok=True)
                    cv2.imwrite(path_lp_img_full, frame)
                add_license_plate(db, lp_string, path_lp_img_full)
                db.commit()
        finally:
            db.close()

    def process_frame(self, frame: np.ndarray) -> Tuple[List[np.ndarray], List[np.ndarray]]:
        if frame is None or frame.size == 0:
            return [], []
        cars_list = []
        if config.CHECK_CAR:
            cars_list = self.car_detector.detect(frame)
        lp_obb_list, lp_bb_list = self.license_plate_detector.detect(frame, cars_list)
        return lp_obb_list, lp_bb_list

    def process_video(self, video_path: str, executor: concurrent.futures.ThreadPoolExecutor,
session_maker) -> None:
        if not os.path.exists(video_path):
            return
        cap = cv2.VideoCapture(video_path)
        if not cap.isOpened():
            return
        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break
            lp_obb_list, lp_bb_list = self.process_frame(frame)
            if lp_obb_list:
```

```
            for i, lp_obb in enumerate(lp_obb_list):
                lp_bb = lp_bb_list[i] if i < len(lp_bb_list) else None
                executor.submit(self._process_lp, frame.copy(), lp_obb, lp_bb, session_maker)
        cap.release()

if __name__ == "__main__":
    engine = create_engine(config.DB_PATH)
    create_database_tables(engine)
    SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
    db_cleanup_session = SessionLocal()
    deleted_count = delete_old_data(db_cleanup_session)
    db_cleanup_session.close()
    lp_system = LicensePlateSystem(
        car_model_path=config.CAR_MODEL_PATH,
        license_plate_model_path=config.LICENSE_PLATE_MODEL_PATH
    )
    process_mode = 'image'
    num_workers = os.cpu_count()
    if num_workers is None:
        num_workers = 4
    if process_mode == 'video':
        video_to_process = 'others/Test.mp4'
        with concurrent.futures.ThreadPoolExecutor(max_workers=num_workers) as executor:
            lp_system.process_video(video_to_process, executor, SessionLocal)
    elif process_mode == 'image':
        image_to_process = 'others/1.jpg'
        if not os.path.exists(image_to_process):
            exit()
        frame = cv2.imread(image_to_process)
        if frame is None or frame.size == 0:
            exit()
        lp_obb_list, lp_bb_list = lp_system.process_frame(frame)
        if lp_obb_list:
            with concurrent.futures.ThreadPoolExecutor(max_workers=num_workers) as executor:
                for i, lp_obb in enumerate(lp_obb_list):
                    lp_bb = lp_bb_list[i] if i < len(lp_bb_list) else None
                    executor.submit(lp_system._process_lp, frame.copy(), lp_obb, lp_bb, SessionLocal)
            executor.shutdown(wait=True)
```

Лістинг файлу db_func.py

```
from sqlalchemy import String, DateTime, delete
from sqlalchemy.orm import declarative_base, Mapped, mapped_column, Session
from datetime import datetime, timedelta, timezone
from typing import List, Optional
import config
import os
import logging

logger = logging.getLogger(__name__)
Base = declarative_base()
```

```
class Data(Base):
    __tablename__ = "license_plate_base"
    id: Mapped[int] = mapped_column(primary_key=True)
    license_plate: Mapped[str] = mapped_column(String(8), unique=False, nullable=True, index =
True)
    timestamp: Mapped[datetime] = mapped_column(DateTime(timezone=True), default=lambda:
datetime.now(timezone.utc), nullable=False, index = True)
    image_path: Mapped[str] = mapped_column(String(255), nullable=False)

def create_database_tables(engine):
    Base.metadata.create_all(engine)

def add_license_plate(db: Session, license_plate: str, image_path: str) -> None:
    if license_plate is None:
        return
    record = Data(license_plate=license_plate, image_path=image_path)
    db.add(record)

def check_record(db: Session, lp_string: str) -> bool:
    if lp_string is None:
        return False
    time_threshold = datetime.now(timezone.utc) - timedelta(seconds=config.THRESHOLD_DETECT)
    recent_entry_exists = (
        db.query(Data.id)
        .filter(Data.license_plate == lp_string)
        .filter(Data.timestamp >= time_threshold)
        .first()
    )
    is_recent = recent_entry_exists is not None
    return is_recent

def get_lp_records(db: Session, lp_string: str, start_date: Optional[datetime] = None, end_date:
Optional[datetime] = None) -> List[Data]:
    if lp_string is None:
        return []
    query = db.query(Data).filter(Data.license_plate == lp_string)
    if start_date:
        query = query.filter(Data.timestamp >= start_date)
    if end_date:
        query = query.filter(Data.timestamp <= end_date)
    records = query.all()
    return records

def search_by_pattern(db: Session, pattern: str, start_date: Optional[datetime] = None, end_date:
Optional[datetime] = None) -> List[Data]:
    if pattern is None:
        return []
    regex_pattern = '^' + pattern.replace('?', '.') + '$'
    query = db.query(Data).filter(Data.license_plate.op('REGEXP')(regex_pattern))
    if start_date:
        query = query.filter(Data.timestamp >= start_date)
```

```
    if end_date:
        query = query.filter(Data.timestamp <= end_date)
    records = query.all()
    return records

def delete_old_records_and_images(db: Session) -> int:
    try:
        cutoff_date = datetime.now(timezone.utc) - timedelta(days=config.DAYS_TO_KEEP)
        image_paths_to_delete = [
            path for path, in db.query(Data.image_path).filter(Data.timestamp < cutoff_date).all()
        ]
        delete_statement = delete(Data).where(Data.timestamp < cutoff_date)
        result = db.execute(delete_statement)
        deleted_db_count = result.rowcount
        db.commit()
        deleted_image_count = 0
        for image_path in image_paths_to_delete:
            if image_path and os.path.exists(image_path) and os.path.isfile(image_path):
                os.remove(image_path)
                deleted_image_count += 1
        return deleted_db_count
    except Exception as e:
        db.rollback()
        return 0
```

Лістинг файлу web.py

```
from fastapi import FastAPI, Form, Request, Depends
from fastapi.templating import Jinja2Templates
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, Session
import config
from db_func import get_lp_records, search_by_pattern, create_database_tables
from fastapi.staticfiles import StaticFiles
import uvicorn
from datetime import datetime, timedelta

app = FastAPI()
engine = create_engine(config.DB_PATH, connect_args={"check_same_thread": False})
create_database_tables(engine)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
templates = Jinja2Templates(directory="templates")
ap.mount("/static", StaticFiles(directory="static"), name="static")
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
@app.get("/")
async def search_page(request: Request):
```

```
    return templates.TemplateResponse("search.html", {
        "request": request,
        "records": [],
        "lp_string": "",
        "start_date": "",
        "start_time": "",
        "end_date": "",
        "end_time": ""
    })

@app.post("/")
async def search_records(request: Request, license_plate: str = Form(...), db: Session =
Depends(get_db)):
    records = get_lp_records(db, license_plate)
    return templates.TemplateResponse("search.html", {
        "request": request,
        "records": records,
        "lp_string": license_plate,
        "start_date": "",
        "start_time": "",
        "end_date": "",
        "end_time": ""
    })

@app.post("/search_by_pattern")
async def search_by_pattern_page(
    request: Request,
    db: Session = Depends(get_db),
    lp_char0: str = Form(""), lp_char1: str = Form(""),
    lp_char2: str = Form(""), lp_char3: str = Form(""),
    lp_char4: str = Form(""), lp_char5: str = Form(""),
    lp_char6: str = Form(""), lp_char7: str = Form(""),
    start_date: str = Form(None),
    start_time: str = Form(None),
    end_date: str = Form(None),
    end_time: str = Form(None)
):
    lp_chars = [lp_char0 or '?', lp_char1 or '?', lp_char2 or '?', lp_char3 or '?',
                lp_char4 or '?', lp_char5 or '?', lp_char6 or '?', lp_char7 or '?']
    lp_pattern = ''.join(lp_chars).upper()

    start_datetime_obj = None
    end_datetime_obj = None

    if start_date:
        try:
            if start_time:
                start_datetime_str = f"{start_date} {start_time}"
                start_datetime_obj = datetime.strptime(start_datetime_str, "%Y-%m-%d %H:%M")
            else:
                start_datetime_obj = datetime.strptime(start_date, "%Y-%m-%d")
        except ValueError:
```

```
            print(f"Помилка парсингу дати/часу початку: {start_date} {start_time}")
            pass

    if end_date:
        try:
            if end_time:
                end_datetime_str = f"{end_date} {end_time}"
                end_datetime_obj = datetime.strptime(end_datetime_str, "%Y-%m-%d %H:%M")
                end_datetime_obj = end_datetime_obj + timedelta(minutes=1)
            else:
                end_datetime_obj = datetime.strptime(end_date, "%Y-%m-%d") + timedelta(days=1)
        except ValueError:
            print(f"Помилка парсингу дати/часу кінця: {end_date} {end_time}")
            pass

    records = search_by_pattern(db, lp_pattern, start_date=start_datetime_obj,
end_date=end_datetime_obj)

    formatted_records = [(record.license_plate, record.timestamp.strftime("%Y-%m-%d %H:%M:%S")
if record.timestamp else '', record.image_path) for record in records]

    return templates.TemplateResponse("search.html", {
        "request": request,
        "records": formatted_records,
        "lp_string": lp_pattern,
        "search_pattern": True,
        "start_date": start_date,
        "start_time": start_time,
        "end_date": end_date,
        "end_time": end_time
    })

if __name__ == "__main__":
    uvicorn.run("web:app", host="127.0.0.1", port=8000, reload=True)
```

Лістинг файлу search.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>License Plate Search</title>
    <meta name="theme-color" content="#007bff">
    <style>
        body {
            display: flex;
            flex-direction: column;
            align-items: center;
            font-family: sans-serif;
            background: linear-gradient(to bottom, #f0f0f0, #d6d6d6);
```

```
    min-height: 100vh;
    margin: 0;
    padding: 20px 10px;
    color: #333;
}

form, .date-fields {
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
}

form {
    padding: 30px;
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 25px;
    width: auto;
    max-width: 95%;
}

h2 {
    margin-bottom: 20px;
    color: #0056b3;
    font-size: 2em;
    font-weight: bold;
    text-align: center;
    text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.1);
}

.top-section {
    display: flex;
    align-items: center;
    gap: 20px;
    flex-wrap: wrap;
    justify-content: center;
    width: 100%;
}

.plate-wrapper {
    position: relative;
    display: inline-block;
    flex-shrink: 0;
}

.plate-image {
    width: 720px;
    height: auto;
    display: block;
    max-width: 100%;
}
```

```
.input-box {
    position: absolute;
    width: 80px;
    height: 90px;
    font-size: 90px;
    text-align: center;
    border: none;
    background: transparent;
    outline: none;
    color: black;
    top: 28px;
    font-family: monospace;
    padding: 0;
    box-sizing: border-box;
    display: flex;
    justify-content: center;
    align-items: center;
}

#lp_char0 { left: 90px; }
#lp_char1 { left: 164px; }
#lp_char2 { left: 254px; }
#lp_char3 { left: 324px; }
#lp_char4 { left: 392px; }
#lp_char5 { left: 460px; }
#lp_char6 { left: 546px; }
#lp_char7 { left: 614px; }

.button-group {
    display: flex;
    gap: 15px;
    flex-wrap: wrap;
    justify-content: center;
    align-items: center;
    flex-grow: 1;
    min-width: 200px;
}

button {
    padding: 12px 24px;
    font-size: 16px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease, transform 0.1s ease;
    min-width: 150px;
    box-shadow: 0 2px 5px rgba(0, 123, 255, 0.2);
}
```

```
button:hover {
    background-color: #0056b3;
    box-shadow: 0 4px 8px rgba(0, 123, 255, 0.3);
}

button:active {
    transform: scale(0.98);
}

.date-fields {
    display: flex;
    justify-content: center;
    gap: 30px;
    width: 100%;
    max-width: 700px;
    background-color: #f8f9fa;
    padding: 20px;
    flex-wrap: wrap;
}

.date-time-group {
    display: flex;
    gap: 15px;
    flex: 1;
    min-width: 280px;
    flex-wrap: wrap;
}

.date-field, .time-field {
    display: flex;
    flex-direction: column;
    gap: 8px;
    flex-grow: 1;
    flex-basis: 0;
    min-width: 120px;
}

.time-field {
    min-width: 80px;
}

.date-fields label {
    font-weight: bold;
    color: #444;
    font-size: 16px;
    margin-bottom: 4px;
}

.date-fields input[type="date"],
.date-fields input[type="time"] {
    padding: 12px;
    border: 1px solid #ccc;
```

```
    border-radius: 4px;
    font-size: 16px;
    width: 100%;
    box-sizing: border-box;
    box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.1);
    transition: border-color 0.3s;
}

.date-fields input:focus {
    border-color: #007bff;
    outline: none;
    box-shadow: 0 0 5px rgba(0, 123, 255, 0.5);
}

.results-table {
    margin-top: 30px;
    border-collapse: collapse;
    width: 100%;
    max-width: 800px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    background-color: #fff;
    border-radius: 8px;
    overflow: hidden;
}

.results-table th, .results-table td {
    border: 1px solid #e0e0e0;
    padding: 12px 15px;
    text-align: left;
}

.results-table th {
    background-color: #007bff;
    color: white;
    font-weight: bold;
    text-transform: uppercase;
    letter-spacing: 0.5px;
    border-color: #007bff;
}

.results-table tbody tr:nth-child(even) {
    background-color: #f8f9fa;
}

.results-table tbody tr:hover {
    background-color: #e9ecef;
    cursor: pointer;
}

.results-table td a {
    color: #007bff;
    text-decoration: none;
```

```
}

.results-table td a:hover {
    color: #0056b3;
    text-decoration: underline;
}

@media (max-width: 768px) {
    form {
        padding: 20px;
        gap: 20px;
    }

    .top-section, .button-group, .date-fields, .date-time-group {
        flex-direction: column;
        gap: 15px;
    }

    .plate-wrapper {
        width: 100%;
        display: block;
    }

    .input-box {
        width: 10%;
        height: auto;
        font-size: 10vw;
        top: 4%;
    }

    #lp_char0 { left: 10%; }
    #lp_char1 { left: 20%; }
    #lp_char2 { left: 32%; }
    #lp_char3 { left: 42%; }
    #lp_char4 { left: 52%; }
    #lp_char5 { left: 62%; }
    #lp_char6 { left: 74%; }
    #lp_char7 { left: 84%; }

    .button-group {
        align-items: center;
    }

    button {
        min-width: 120px;
    }

    .date-fields {
        padding: 15px;
        align-items: center;
    }
```

```
        .date-time-group {
            min-width: auto;
            max-width: 300px;
            margin: 0 auto;
        }

        .date-field, .time-field {
            width: 100%;
            min-width: auto;
        }

        .results-table th, .results-table td {
            padding: 10px;
            font-size: 14px;
        }
    }
  </style>
</head>
<body>

  <h2>Пошук автомобільного номерного знаку</h2>
  <form method="POST" action="/search_by_pattern">
    <div class="top-section">
      <div class="plate-wrapper">
        <img src="/static/images/8d8c5701-b1ab-4ea0-b88b-b97e8554b6a8.png" alt="Plate Image"
class="plate-image">
          {% for i in range(8) %}
            <input type="text" name="lp_char{{ i }}" id="lp_char{{ i }}" class="input-box"
maxlength="1" pattern="[A-Za-z0-9]" aria-label="Символ номерного знаку {{ i + 1 }}" {% if
lp_string|length > i %}{% if lp_string[i] != '?' %}value="{{ lp_string[i] }}"{% endif %}{% endif %}>
          {% endfor %}
        </div>
        <div class="button-group">
          <button type="submit">Почати пошук</button>
          <button type="button" onclick="clearInputs()">Очистити</button>
        </div>
      </div>

      <div class="date-fields">
        <div class="date-time-group">
          <div class="date-field">
            <label for="start_date">Початок пошуку (Дата):</label>
            <input type="date" name="start_date" id="start_date" value="{{ start_date or '' }}">
          </div>
          <div class="time-field">
            <label for="start_time">Початок пошуку (Час):</label>
            <input type="time" name="start_time" id="start_time" value="{{ start_time or '' }}">
          </div>
        </div>
        <div class="date-time-group">
          <div class="date-field">
            <label for="end_date">Кінець пошуку (Дата):</label>
```

```
            <input type="date" name="end_date" id="end_date" value="{{ end_date or '' }}">
        </div>
        <div class="time-field">
            <label for="end_time">Кінець пошуку (Час):</label>
            <input type="time" name="end_time" id="end_time" value="{{ end_time or '' }}">
        </div>
    </div>
</div>

{% if records %}
    <table class="results-table">
        <thead>
            <tr>
                <th>Номерний знак</th>
                <th>Дата та час фіксації</th>
            </tr>
        </thead>
        <tbody>
            {% for record in records %}
                <tr>
                    <td>{{ record[0] }}</td>
                    <td><a href="{{ record[2] }}" target="_blank">{{ record[1] }}</a></td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% endif %}
</form>

<script>
    document.addEventListener("DOMContentLoaded", () => {
        const inputs = document.querySelectorAll(".input-box");
        inputs.forEach((input, i) => {
            input.addEventListener("input", () => {
                input.value = input.value.toUpperCase().replace(/[^A-Z0-9]/g, '');
                if (input.value.length === 1 && i < inputs.length - 1) {
                    inputs[i + 1].focus();
                }
            });
            input.addEventListener('click', () => {
                input.focus();
            });
            const lpString = "{{ lp_string }}";
            if (lpString.length > i && lpString[i] !== '?') {
                input.value = lpString[i];
            }
        });

        document.querySelector('.plate-wrapper').addEventListener('click', (event) => {
            if (event.target.classList.contains('plate-image') || event.target.classList.contains('plate-
wrapper')) {
                const firstEmptyInput = document.querySelector('.input-box:not([value])') ||
```

```
document.querySelector('.input-box');
                if(firstEmptyInput) {
                    firstEmptyInput.focus();
                }
            }
        });

        const startDate = "{{ start_date }}";
        const startTime = "{{ start_time }}";
        const endDate = "{{ end_date }}";
        const endTime = "{{ end_time }}";

        if(startDate && startDate !== 'None') document.getElementById('start_date').value = startDate;
        if(startTime && startTime !== 'None') document.getElementById('start_time').value =
startTime;
        if(endDate && endDate !== 'None') document.getElementById('end_date').value = endDate;
        if(endTime && endTime !== 'None') document.getElementById('end_time').value = endTime;
    });

    function clearInputs() {
        document.querySelectorAll('.input-box').forEach(el => el.value = '');
        document.getElementById('start_date').value = '';
        document.getElementById('start_time').value = '';
        document.getElementById('end_date').value = '';
        document.getElementById('end_time').value = '';
        document.querySelector('.input-box').focus();
    }
</script>

</body>
</html>
```

Лістинг файлу config.py

```
// Шлях до бази даних;
DB_PATH = 'sqlite:///database.db'
//Каталог, куди будуть зберігатися зображення;
LP_IMG_FOLDER = 'static/lp_images/'
//Шлях до зображення. Використовується в цілях діагностики;
IMG_PATH = 'others/1.jpg'
//  конфігурація OCR для розпізнавання зображення як один рядок символів;
CUSTOM_TESSERACT_CONFIG_A = (
    r'--oem 3 --psm 7 -c load_system_dawg=false '
    r'-c load_freq_dawg=false '
    r'-c tessdir_char_whitelist=0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
)
CUSTOM_TESSERACT_CONFIG_B = (    //  конфігурація OCR для
    r' --oem 3 --psm 10 -c load_system_dawg=false '
    r'-c load_freq_dawg=false '
    r'-c tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 '
)
VEHICLES = [2, 3, 5, 7]
```

```
THRESHOLD = 0.7
THRESHOLD_DETECT = 30
CAR_MODEL_PATH = 'yolo11n.pt'
LICENSE_PLATE_MODEL_PATH = r'runs\obb\train\weights\last.pt'
DEBUG = True
CHECK_CAR = False
DAYS_TO_KEEP = 20
MIN_LP_WIDTH = 120
MAX_LP_WIDTH = 500
MIN_LP_HEIGHT = 25
MAX_LP_HEIGHT = 250
```