

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Автоматизоване розпізнавання державних номерів
для врахування часу стоянки та оплати»**

Виконав студент групи ПЗМ-21-1
ВАСИЛЕНКО Максим Костянтинович

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
ЛОКТИКОВА Тамара Миколаївна

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«18» вересня 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ВАСИЛЕНКО Максим Костянтинович**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: **«Автоматизоване розпізнавання державних номерів для врахування часу стоянки та оплати»,**

затверджена Наказом закладу вищої освіти від **«18» вересня 2023 р. № 479/с**

Вихідні дані для роботи: **методи штучного інтелекту, які використовуються в алгоритмах виявлення та розпізнавання автомобільних номерних знаків**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В	22.09.2022р.	22.09.2022р.
2	Власенко О.В	22.09.2022р.	22.09.2022р.
3	Власенко О.В	22.09.2022р.	22.09.2022р.

РЕФЕРАТ

Індивідуальне завдання кваліфікаційної магістерської роботи полягає в розробці та впровадженні алгоритму виявлення та розпізнавання автомобільних номерних знаків на зображеннях реальних сцен за допомогою апарату штучних нейронних мереж.

У дослідженні представлено аналітичний огляд методів та існуючих технічних і програмних систем, що використовують методи штучного інтелекту для розпізнавання автомобільних номерів. Проведено аналіз існуючих алгоритмів та запропоновано власний алгоритм розпізнавання номерних знаків на основі використання штучних нейронних мереж та їх ансамблів.

В результаті застосування запропонованого алгоритму вдалося вирішити задачу розпізнавання номерних знаків і досягти прийнятної точності результатів, отриманих при невеликій кількості дескрипторів.

Пояснювальна записка до кваліфікаційної магістерської роботи складається зі вступу, чотирьох розділів, списку літератури з 35 найменувань, 30 рисунків, 6 таблиць, 2 додатків та 12 формул. Повний обсяг роботи становить 69 сторінок.

КЛЮЧОВІ СЛОВА: АВТОМОБІЛЬНИЙ НОМЕРНИЙ ЗНАК, АЛГОРИТМ РОЗПІЗНАВАННЯ, РОЗПІЗНАВАННЯ СИМВОЛІВ, ЦИФРОВА ОБРОБКА ЗОБРАЖЕНЬ, ШТУЧНА НЕЙРОННА МЕРЕЖА, АНСАМБЛЬ НЕЙРОННИХ МЕРЕЖ, МЕТОД КОМІТЕТНОЇ КЛАСИФІКАЦІЇ.

ABSTRACT

The individual task of the qualifying master's thesis consists in the development and implementation of an algorithm for the detection and recognition of car license plates on images of real scenes using the apparatus of artificial neural networks.

The study presents an analytical review of methods and existing technical and software systems using artificial intelligence methods for recognizing car license plates. An analysis of existing algorithms was carried out and a proprietary license plate recognition algorithm based on the use of artificial neural networks and their ensembles was proposed.

As a result of the application of the proposed algorithm, it was possible to solve the problem of license plate recognition and achieve acceptable accuracy of the results obtained with a small number of descriptors.

The explanatory note to the qualifying master's thesis consists of an introduction, four chapters, a bibliography of 35 names, 30 figures, 6 tables, 2 appendices and 12 formulas. The full volume of the work is 69 pages.

KEY WORDS: CAR NUMBER SIGN, RECOGNITION ALGORITHM, CHARACTER RECOGNITION, DIGITAL IMAGE PROCESSING, ARTIFICIAL NEURAL NETWORK, NEURAL NETWORK ENSEMBLE, COMMITTEE CLASSIFICATION METHOD.

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1 ОГЛЯД МЕТОДІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ НОМЕРІВ АВТОМОБІЛІВ.....	10
1.1 Методи розпізнавання номерних знаків	10
1.1.1 Метод Віюли-Джонса	11
1.1.2 Мащини опорних векторів	13
1.2.3 Використання штучних нейронних мереж.....	14
1.2 Огляд систем розпізнавання номерних знаків	19
1.2.1 «Centurion»	19
1.2.2 «Plate Recognizer».....	21
1.2.3 «SecurOS Auto».....	24
1.3 Перевірка систем розпізнавання номерних знаків	29
1.4 Вимоги до розробленого алгоритму	29
Висновки до розділу 1	30
РОЗДІЛ 2 АЛГОРИТМ РОЗПІЗНАВАННЯ АВТОМОБІЛЬНИХ НОМЕРІВ НЕЙРОМЕРЕЖАМИ.....	31
2.1 Загальне рішення задачі	31
2.2 Поетапна робота з зображенням.....	33
2.2.1 Сегментація номерних знаків	33
2.2.2 Вилучення фрагмента з номерним знаком	35
2.2.3 Попередня обробка вирізаного фрагмента.....	37
2.2.4 Класифікація сегментів номерних знаків	39
Висновки до розділу 2	41
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	42
3.1 Мова розробки	42
3.2 Середовище розробки	45
3.3 Розробка програми	47
3.4 Робота з базою даних	49
3.5 Результат роботи програми.....	51

3.6 Тестування роботи програми	57
3.6.1 Набір даних	57
3.6.2 Результати тестування	57
3.7 Висновки за результатами експерименту	60
Висновки до розділу 3	60
ВИСНОВКИ.....	61
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	62
ДОДАТКИ.....	65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MH – Машинне навчання

ШІ – Штучний інтелект

NN – (англ. neural network) – нейронна мережа

CNN – (англ. convolutional neural network) – згорткова нейронна мережа

ВСТУП

Актуальність теми. В епоху стрімкого розвитку технологій, суспільство стикається з ростом інформатизації, що зумовлює автоматизацію широкого спектру процесів за допомогою сучасних комп'ютерних технологій. В цьому контексті важливим стає застосування автоматизації для полегшення та покращення життя. Яскравим прикладом успішного впровадження автоматизації є ідентифікації автомобілів за їхніми номерними знаками. На сьогодні системи розпізнавання номерних знаків набувають великої популярності та широко використовуються в різних сферах та для певних цілей таких як безпека дорожнього руху, контроль на автотранспортних підприємствах або заправних станціях, а також й на автомобільних стоянках. Актуальність цієї технології виявляється в ряді важливих аспектів, які варто врахувати.

По-перше, розпізнавання номерів є ключовим елементом забезпечення безпеки та ведення правопорядку. Застосування цієї технології дозволяє автоматизовано визначати та реєструвати транспортні засоби, надаючи правоохоронним органам ефективний інструмент для розслідування злочинів та забезпечення громадської безпеки.

Другий важливий аспект полягає в автоматизації різних сфер, від транспортних сервісів до містобудування. Системи розпізнавання номерних знаків дозволяють ефективно керувати транспортним потоком, сприяючи оптимізації паркувань, управлінню дорожньою інфраструктурою та навіть покращенню екологічних показників.

Отже розпізнавання номерних знаків є затребуваним у різних системах керування транспортними засобами. Ці системи можуть використовуватися різними державними установами, пунктами контролю швидкості тощо.

Мета роботи. Метою роботи є розробка та впровадження алгоритму виявлення та розпізнавання автомобільних номерних знаків на зображеннях реальних сцен за допомогою апарату штучних нейронних мереж.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. розробка та впровадження алгоритму виявлення номерних знаків;
2. попередня обробка зображення для подальшого розпізнавання;
3. розробка та впровадження алгоритму розпізнавання номерних знаків;
4. проведення чисельних експериментів.

Предмет дослідження. Предметом дослідження є методи штучного інтелекту, які використовуються в алгоритмах виявлення та розпізнавання автомобільних номерних знаків. Застосування згорткових нейронних мереж для сегментації, а також класифікатори буквено-цифрових символів автомобільних номерних знаків.

Об'єкт дослідження. Методи штучного інтелекту, які використовуються в алгоритмах сегментації та розпізнавання номерних знаків.

Практичне значення. В результаті застосування запропонованого алгоритму вдалося вирішити задачу розпізнавання номерних знаків і досягти прийнятної точності результатів, отриманих при невеликій кількості дескрипторів.

Сфера застосування розробленого алгоритму – програмні комплекси, завдання яких пов'язані з пошуком та ідентифікацією транспортних засобів, обліком проїзду через пункти пропуску, обмеженням в'їзду на приватну територію.

Наукова новизна роботи полягає в розробці алгоритму на базі відомих методів розпізнавання автомобільних номерів та дослідженні можливостей використання штучних нейронних мереж, а також їх ансамблів як елементів класифікації.

Апробація. За темою кваліфікаційної магістерської роботи було опубліковано тези конференції [1].

РОЗДІЛ 1 ОГЛЯД МЕТОДІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ НОМЕРІВ АВТОМОБІЛІВ

1.1 Методи розпізнавання номерних знаків

Комп'ютерний зір досі залишається актуальною науковою галуззю, яка вирішує задачі отримання високого рівня розуміння цифрових зображень [2]. Завдання включають методи збору, обробки, аналізу та розуміння цифрових зображень, їх можна розділити на чотири типи:

Класифікація - класифікація зображення за типом об'єкта, який воно містить;

Семантична сегментація - знаходження пікселів об'єктів одного класу на зображенні. Якщо кілька об'єктів одного класу перекриваються, їх пікселі жодним чином не відокремлюються один від одного;

Виявлення об'єктів - виявлення всіх об'єктів необхідних класів і побудова огорожувальної рамки для кожного з них;

Сегментація екземплярів - знаходження пікселів, які містить об'єкт кожного класу окремо;

Ілюстрація сегментації на прикладі повітряних куль наведена на рис. 1.1.

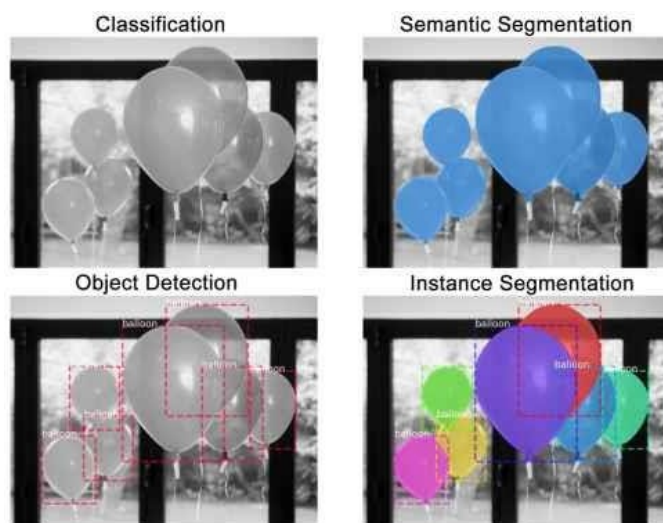


Рис. 1.1. Ілюстрація типів комп'ютерного зору

Для виявлення та розпізнавання номерних знаків використовується багато різних методів. Розглянемо деякі з них: метод Віоли-Джонса і машину опорних векторів.

Метод Віоли-Джонса, його недоліком є те, що при куті нахилу більше 30 градусів точність і ефективність методу значно знижується [3];

Опорна векторна машина, основним недоліком якої є те, що вона підходить лише для розв'язування задач з 2 класами.

У свою чергу, згорткові нейронні мережі показують найкращі результати для розпізнавання об'єктів на зображеннях. У зв'язку з цим в даній роботі було вирішено використовувати згорткові нейронні мережі для виявлення та розпізнавання номерних знаків автомобілів під різними кутами нахилу та шуму.

Розглянемо вищевказані методи більш детально.

1.1.1 Метод Віоли-Джонса

Метод Віоли-Джонса — це алгоритм виявлення об'єктів у реальному часі, запропонований у 2001 році П. Віолою та М. Джонсом. Його основне завдання — виявлення облич, але воно також використовується для розпізнавання різних класів зображень [3].

Метод можна розділити на дві частини: алгоритм навчання і алгоритм розпізнавання.

Принципи, на яких заснований метод:

- цілісне представлення зображень, що значно прискорює розрахунок необхідних об'єктів;
- використовуються Хаарівські ознаки, за допомогою яких здійснюється пошук потрібного об'єкта (в даному контексті кутових точок);
- побудова класифікатора на основі алгоритму boosting (від англ. boost — покращення, посилення) для вибору найбільш відповідних ознак для потрібного об'єкта в заданій частині зображення;

- всі ознаки вводяться в класифікатор, який дає результат «істина» або «хибна»;
- каскади функцій використовуються для швидкого відкидання незначних вікон без об'єкта.

Описані вище принципи дозволяють шукати об'єкт в режимі реального часу.

Довгий час навчання, а також використання великої кількості необхідних даних для навчання класифікатора є основним недоліком методу. Якщо кут нахилу об'єкта на зображенні перевищує 30 градусів, відсоток виявлення значно зменшується. Крім того, через підвищену освітленість або сильне затемнення об'єкта алгоритм може не дати результату.

До переваг методу можна віднести наступне:

- виявлення більш ніж одного об'єкта;
- використання простих класифікаторів;
- висока швидкість пошуку;
- можливість використання у відеопотоці.

Розглянемо експериментальні дані за методом:

Загальна кількість символів $x = 1620$

Кількість сегментованих областей $s = 2247$

Кількість правильно сегментованих $y = 1348$

Середня точність сегментації

$$\frac{y}{x} * 100\% = \frac{1297}{1608} * 100\% = 80\% \quad (1.1)$$

Середня помилка сегментації

$$\left| 1 - \frac{y}{s} \right| = \left| 1 - \frac{1348}{2247} \right| = 0.4 \quad (1.2)$$

У таблиці 1.1 подано звіт за окремими групами чисел. Точність визначається відсотком правильно розпізнаних символів по відношенню до загальної кількості символів, детальніше див. [4].

Розрахунок групової точності

Група	Точність сегментації Метод Віоли-Джонса, %
Нормальний	90
Під кутом	80
Спотворений	73

1.1.2 Машини опорних векторів

Машина опорного вектора (SVM) — це контрольований алгоритм навчання, пов'язаний з лінійними класифікаторами. SVM використовується для вирішення завдань регресійного аналізу та класифікації. Регуляризацію Тихонова можна також розглядати як окремий випадок SVM [5]. Машина опорного вектора є методом класифікатора з максимальним розривом, оскільки він постійно зменшує емпіричну помилку класифікації та збільшує розрив.

Допоміжна векторна машина, ідея якої полягає в тому, що вихідні вектори перетворюються у високовимірний простір і здійснюється пошук розділової гіперплощини. Отримана гіперплощина має дві паралельні гіперплощини з боків, а також максимальний зазор у цьому просторі. Площиною, яка має максимальну відстань до цих паралельних гіперплощин, буде роздільна площина. Середня похибка результатів методу залежить від різниці або відстані між двома паралельними гіперплощинами, зі збільшенням відстані похибка зменшується [6].

Класифікація даних, проблема, яка часто зустрічається в алгоритмах машинного навчання. Кожен об'єкт даних представлений у вигляді вектора в просторі з послідовністю ρ чисел. Кожен об'єкт належить лише один із двох класів. Необхідно розділити об'єкти гіперплощиною розмірності $\rho - 1$. Для цього розрив максимально збільшується, що сприятиме більш впевненій класифікації. Для цього ми знаходимо відстань гіперплощини, від якої до

найближчого об'єкта буде максимально. Коли таку гіперплощину буде знайдено, її будемо називати оптимальною розділяючою гіперплощиною, а лінійний класифікатор, який їй відповідатиме, — оптимальним розділяючим.

Розглянемо експериментальні дані за методом:

Кількість правильно сегментованих $y = 1561$

Кількість SVM, розпізнаних алгоритмом $svm = 1601$

Середня точність розпізнавання для алгоритму

$$SVM = \frac{y}{svm} * 100\% = \frac{1561}{1601} * 100\% = 97.7\% \quad (1.3)$$

У таблиці 1.2 наведено звіт про окремі групи чисел для алгоритму SVM.

Таблиця 1.2

Розрахунок групової точності

Група	Точність сегментації Метод Віоли-Джонса, %
Нормальний	99
Під кутом	99
Спотворений	98

1.2.3 Використання штучних нейронних мереж

Людський мозок вирішує проблему розпізнавання образів набагато краще, ніж будь-який цифровий комп'ютер. Це пов'язано з тим, що спосіб обробки інформації мозком людини принципово відрізняється від методів, які використовуються звичайними цифровими комп'ютерами. Мозок людини є надзвичайно складним, нелінійним, паралельним комп'ютером (системою обробки інформації). Він має можливість організувати свої структурні компоненти так, щоб вони могли виконувати конкретні завдання (наприклад, розпізнавання образів) у багато разів швидше, ніж найшвидші комп'ютери [7].

У цій роботі пропонується використання штучних нейронних мереж (ШНМ). Нейронну мережу згідно з [7] можна описати так:

«Нейронна мережа — це величезний розподілений паралельний процесор, що складається з елементарних блоків обробки інформації (нейронів), які накопичують експериментальні знання та надають їх для подальшої обробки».

Переваги NN описані в [7] та в [8]. Доречно було б перерахувати деякі з них:

- нелінійність;
- відображення вхідної інформації у вихідну;
- адаптивність;
- очевидність відповіді;
- контекстна інформація;
- відмовостійкість;
- одноманітність аналізу та проектування.

Навчання мережі (налаштування) — це процес, в якому вільні параметри нейронної мережі налаштовуються шляхом моделювання середовища, в яке ця мережа вбудована. Тип навчання, у свою чергу, визначається способом вибору цих параметрів [7].

Це визначення процесу налаштування передбачає наступну послідовність:

- 1) Мережа отримує стимули ззовні.
- 2) В результаті змінюються внутрішні параметри мережі.
- 3) Після зміни внутрішньої структури нейронна мережа по-іншому реагує на збудження.

Алгоритм зворотного поширення. Існує багато алгоритмів навчання. Найвідомішим з них є алгоритм зворотного поширення. Цей алгоритм використовується для мінімізації відхилення реальних значень вихідних сигналів нейронної мережі (ANN) від необхідних. Вона детально описана в [7], [8], [9], [10], [11] та в [12].

Суть навчання зворотного поширення полягає в кодуванні відображення вхід-вихід (представленого набором позначених прикладів) у синаптичних вагових і порогових значеннях багатoshарового персептрона [7]. Проблему налаштування мережі можна розглядати як завдання вибору найкращої моделі з набору «кандидатів» структур з урахуванням певного критерію.

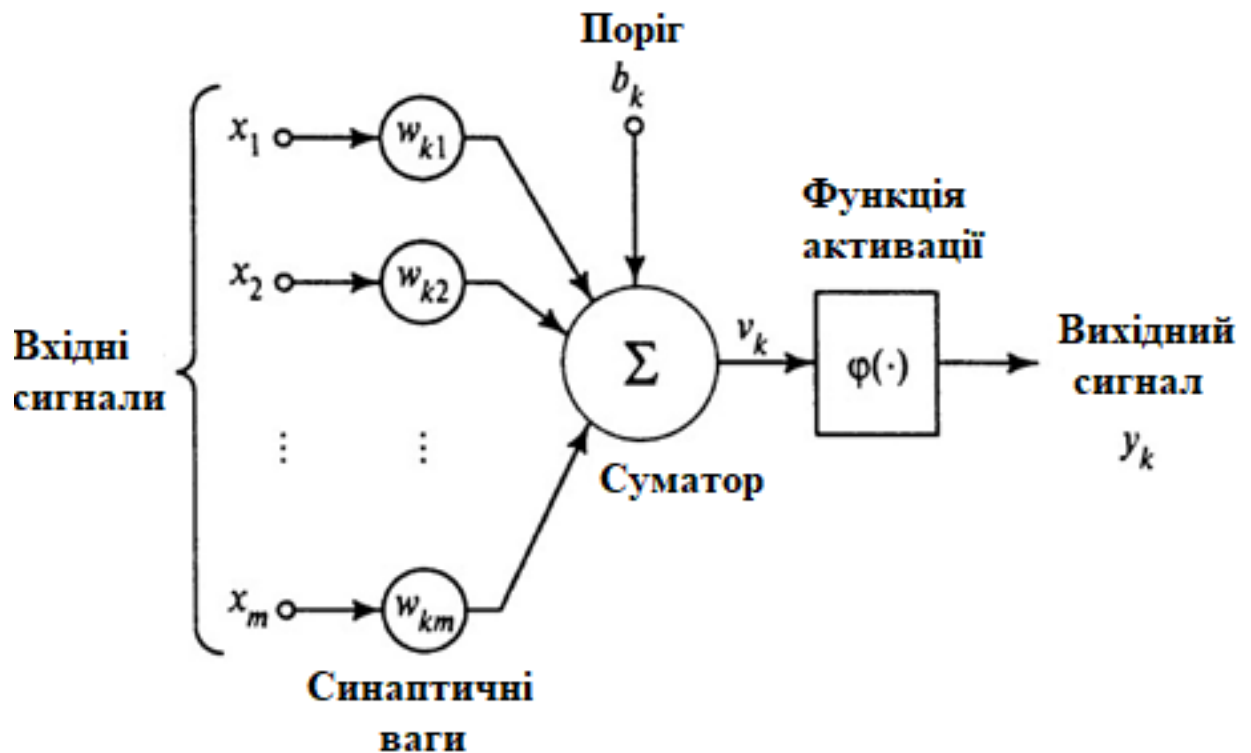


Рис. 1.2. Нелінійна модель нейрона [7]

Нейрон - це одиниця обробки інформації в нейронній мережі. На рис. 1.2 показана модель нейрона, що лежить в основі штучних нейронних мереж. У цій моделі є три основні елементи.

1. Сукупність синапсів (синапс) або зв'язків (сполучна ланка), кожен з яких характеризується своєю вагою (масою) або силою (силою).
2. Суматор додає вхідні сигнали, зважені відносно відповідних синапсів нейрона.
3. Функція активації обмежує амплітуду вихідного сигналу нейрона.

Модель нейрона, містить пороговий елемент (зміщення), який позначається символом b_k . Це значення відображає збільшення або зменшення вхідного сигналу, що застосовується до функції активації.

Математично нейрон можна представити наступними рівняннями [7]:

$$u_k = \sum_{j=1}^m w_{kj} x_j \quad (1.4)$$

$$y_k = \varphi(u_k + b_k) \quad (1.5)$$

де $x_1, x_2, \dots, x_m$ — вхідні сигнали; $w_{k1}, w_{k2}, \dots, w_{km}$ — синаптичні ваги нейрона k ; u_k — лінійна комбінація вхідних дій (вихід лінійного комбінатора); b_k - поріг; $\varphi(\cdot)$ — функція активації; y_k є вихідним сигналом нейрона.

Постсинаптичний потенціал розраховується так:

$$v_k = u_k + b_k \quad (1.6)$$

Функції активації визначають вихідний сигнал нейрона залежно від його індукованого локального поля. Розглянемо кілька функцій активації.

Сигмовидна функція. Це швидко зростаюча функція, яка підтримує баланс між лінійною та нелінійною поведінкою. Прикладом сигмовидної функції є логістична функція, задана таким виразом:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (1.7)$$

Іншою функцією активації є лінійний випрямляч (ReLU), заданий таким виразом:

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \quad (1.8)$$

Згорткові нейронні мережі (CNN) — архітектура ШНМ, запропонована Яном ЛеКуном у 1988 році, метою якої було підвищення ефективності розпізнавання різних об'єктів на зображеннях, і є частиною технологій глибокого навчання. Ці мережі є одними з найвпливовіших нововведень у

комп'ютерному зорі. Сьогодні глибоке навчання є центром послуг багатьох компаній, але класичний і, можливо, найпопулярніший варіант використання мереж - це обробка зображень [13].

Структура CNN представлена наступним чином: зображення пропускається через різні типи шарів, згорткові, нелінійні, об'єднані та повністю зв'язані, після чого генерується відповідь. Ця відповідь може бути класом або ймовірністю класів, які найкраще описують потрібне зображення. CNN – це односпрямована, тобто без зворотного зв'язку, багат шарова мережа. При навчанні мережі використовуються стандартні методи, найчастіше це метод зворотного поширення помилки. Функція активації може бути будь-якою з відомих на вибір дослідника [14].

Для сегментації використовується архітектура U-Net, ця CNN вважається однією із стандартних при вирішенні задач сегментації, коли вам потрібно сегментувати області за необхідними класами, створюючи маски, які поділяють зображення на класи. Архітектура складається з стискаючого шляху для захоплення контексту та симетричного шляху розширення, що дозволяє тонку локалізацію [15], [16], [17].

Ця архітектура перевершує попередній найкращий метод, згорткову мережу з ковзним вікном, навчаючись наскрізь лише з невеликою кількістю даних.

У конкурсі ISBI на сегментацію нейронних структур у стеках електронної мікроскопії, використовуючи ту саму мережу, яка була навчена зображенням мікроскопії пропускання світла (фазовий контраст і DIC), ця архітектура здобула перше місце відстеження клітин у 2015 році у цих категоріях, і вражає автора значним успіхом. Головна перевага цієї архітектури – швидкість. Сегментація зображення розміром 512×512 займає менше секунди на сучасному GPU [15], [16], [17].

Таблиця 1.3

Результати сегментації (IOU) на конкурсі відстеження клітин ISBI 2015

Назва	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
Second-best (2015)	0.83	0.46
U-Net	0.9203	0.7756

U-Net використовувався в задачі сегментації клітин на зображеннях оптичного мікроскопа. Це завдання сегментації є частиною змагань ISBI зі стеження за клітинками 2014 та 2015 років, подробиці див. у [15]. U-Net показує хороші результати в порівнянні з іншими мережами.

1.2 Огляд систем розпізнавання номерних знаків

Існує велика кількість промислового програмного забезпечення для розпізнавання номерних знаків. Цей розділ пропонує короткий аналітичний огляд деяких програмних продуктів. Опис кожного продукту містить інформацію про назву, область застосування та використовувані технології розпізнавання.

1.2.1 «Centurion»

«Centurion» — це система, призначена для розпізнавання номерних знаків. Він використовується для контролю проїзду автомобілів, перевірки їх за різними базами даних та обмеження доступу автомобілів на територію [18] (рис. 1.3).

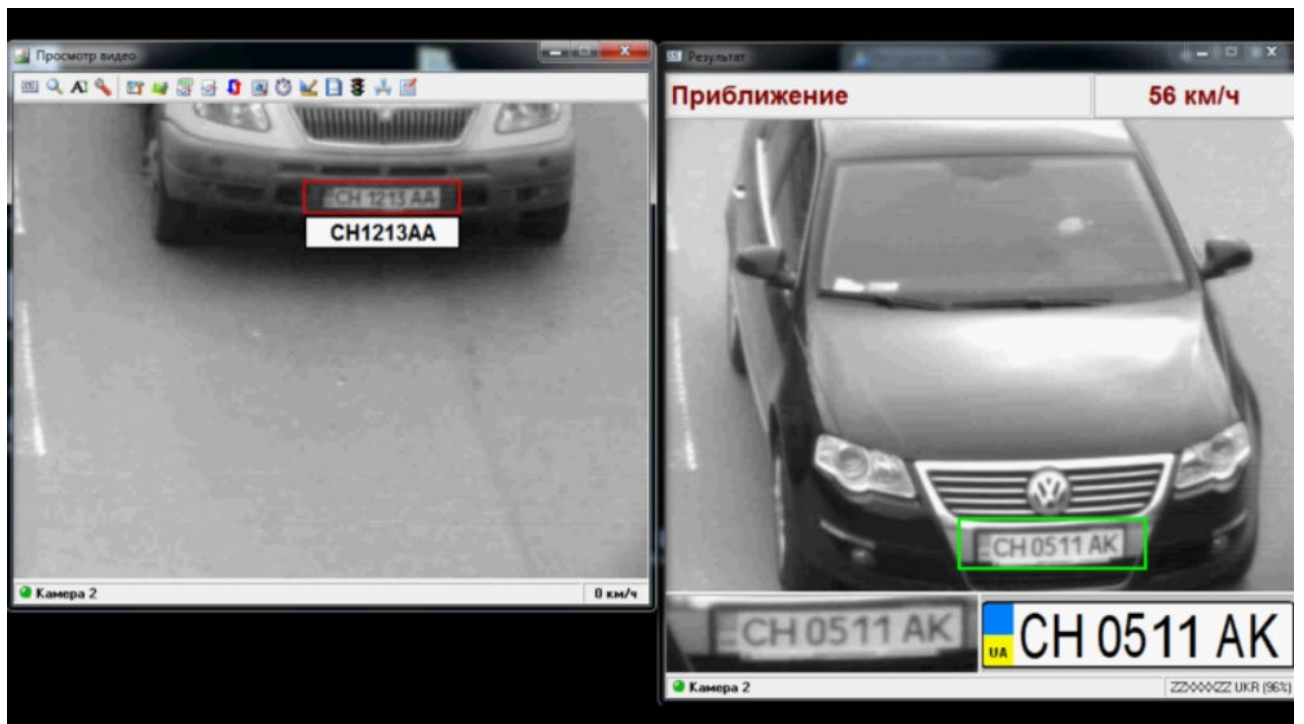


Рис. 1.3. Centurion

Область застосування:

- пункти пропуску підприємств;
- платні стоянки;
- паркування біля торгових центрів;
- закриті зони;
- житлові комплекси;
- котеджні селища;
- автомийки;
- автоматичний пошук автомобіля.

Можливості:

- виявлення транспортних засобів у зоні контролю та визначення напрямків їх руху;
- визнання державних реєстраційних номерів автомобілів;
- ведення бази проїжджаючих транспортних засобів;

- контроль шлагбаумів і світлофорів в автоматичному та автоматизованому режимах при організації проїзду транспортні засоби на закриті території (підприємства, автостоянки, котеджні селища тощо);
- пошук розпізнаних номерів автомобілів у базах даних користувача та, якщо вони збігаються, автоматична відповідь за сценарієм, визначеним користувачем;
- передача даних розпізнавання до сторонніх облікових систем;
- здійснення контролю за переміщенням транспортних засобів на території підприємства та контроль за дотриманням встановлених регламентів такого переміщення;
- диференціація доступу різних категорій користувачів до згенерованих даних за паролями або картками;

Методи і технології розпізнавання

«Centurion» для максимально надійної роботи в різних умовах (день, ніч, опади, яскраве сонце, фари) використовує два алгоритми, які працюють паралельно. Перший алгоритм базується на нейронних мережах, другий алгоритм є класифікатором на основі багатовимірної інтерполяції та апроксимації [18].

1.2.2 «Plate Recognizer»

Plate Recognizer — це сервіс для розпізнавання номерних знаків автомобілів за допомогою передових технологій комп'ютерного зору та машинного навчання. Нижче розглянемо деталі його функціональності:(рис. 1.4).

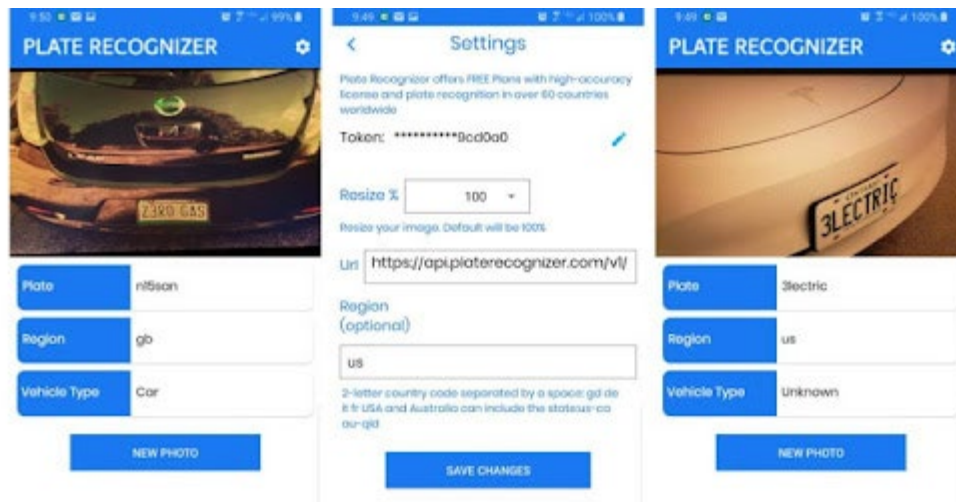


Рис. 1.4. Plate Recognizer

Можливості:

- API. Plate Recognizer надає API для інтеграції з іншими системами. Це означає, що ви можете використовувати його функції у ваших власних додатках або сервісах, що вимагає розпізнавання номерних знаків.
- Технології розпізнавання тексту. В основі Plate Recognizer лежать високоточні алгоритми машинного навчання та нейронні мережі, призначені для ефективного розпізнавання тексту на номерних знаках. Ці технології дозволяють системі точно виділяти символи навіть на зображеннях невеликої роздільної здатності або при різноманітних умовах освітлення.
- Інтеграція з камерами. Plate Recognizer може бути легко інтегровано з відеокамерами або системами відеоспостереження для автоматичного розпізнавання номерних знаків на відеозаписах чи в режимі реального часу. Це робить його відмінним рішенням для використання в системах безпеки або автоматизованого контролю транспортних потоків.
- Автоматична обробка зображень. Plate Recognizer автоматично обробляє та аналізує зображення з камер для визначення номерних знаків. Це включає в себе не тільки розпізнавання тексту, але й різні аспекти обробки зображень, такі як фільтрація шуму, корекція кольорів тощо
- Застосування в транспортних системах та безпеці. Plate Recognizer знаходить застосування в різних сферах, таких як транспортна безпека, контроль

паркування, автоматизований облік автотранспорту, а також в сферах, де важлива ідентифікація автомобілів за їхніми номерними знаками.

- Обробка інформації. Система збирає та обробляє інформацію про розпізнані номерні знаки. Це може включати в себе зберігання цих даних та генерацію звітів для подальшого аналізу чи використання. Plate Recognizer має широкий спектр областей застосування через свої можливості розпізнавання номерних знаків автомобілів.

Область застосування:

- Транспортна Безпека. Plate Recognizer може використовуватися для підвищення рівня транспортної безпеки, надаючи системам відеоспостереження можливість автоматичного розпізнавання та відстеження автомобільних номерних знаків. Це може бути корисно для виявлення порушень правил дорожнього руху, автоматичної ідентифікації транспортних засобів тощо.

- Контроль Паркування. В області паркування Plate Recognizer може допомагати автоматизувати процеси контролю за паркуваннями, виявляючи номерні знаки автомобілів та надаючи інформацію про час та тривалість паркування.

- Автоматизований Облік Транспорту. Plate Recognizer може використовуватися для автоматизованого обліку автотранспорту на дорогах чи в обмежених територіях. Це може бути корисно для збору статистики транспортного потоку, планування інфраструктури тощо.

- Безпека Комплексів та Територій. В області загальної безпеки Plate Recognizer може використовуватися для ідентифікації автомобілів, які в'їжджають на територію комплексу чи об'єкта. Це може бути важливо для забезпечення доступу або контролю за безпекою.

- Міське планування та інфраструктура. У контексті міського планування Plate Recognizer може допомагати збирати дані про рух транспорту для оптимізації інфраструктури та вирішення транспортних проблем.

– Логістика та управління транспортом. В галузі логістики та управління транспортом Plate Recognizer може служити інструментом для відстеження та ідентифікації автотранспорту під час перевезень.

– Системи безпеки на підприємствах. У великих підприємствах Plate Recognizer може використовуватися для створення систем безпеки, які включають розпізнавання номерних знаків для контролю за в'їздом автотранспорту на територію підприємства.

Ці області застосування вказують на універсальність технологій, які використовує Plate Recognizer для різноманітних завдань в сфері транспорту, безпеки та управління.

1.2.3 «SecurOS Auto»

Система інтелектуального відеоаналізу SecurOS Auto забезпечує розпізнавання номерних знаків транспортних засобів. SecurOS Auto працює на базі інтеграційної платформи керування відео SecurOS, що дозволяє створити комплекс безпеки з необхідним замовником функціоналом (рис.1.5).



Рис. 1.5. SecurOS Auto

Область застосування:

- Сфера діяльності «SecurOS Auto» охоплює значний спектр завдань від забезпечення безпеки на стоянках до контролю потоків автомобілів по місту.

Можливості

- розпізнавати номерні знаки понад 50 країн світу;
- розпізнавати номерні знаки за окремими кадрами без використання відео;
- швидко налаштувати оновлення для розпізнавання ряду нових стандартів;
- сформувати базу даних розпізнаних номерних знаків із збереженням відповідної інформації про дату, час і місце виявлення
- транспортний засіб, швидкість і напрямок його руху, а також посилання на відеоролик;
- організувати пошук раніше розпізнаних номерних знаків за заданими параметрами;
- підтримувати одночасно зовнішні та внутрішні списки номерів (інформаційний, білий, чорний);
- автоматично звіряти визнаний знак державної реєстрації за внутрішніми та зовнішніми списками (базами даних);
- налаштувати необхідні реакції системи на розпізнавання номерних знаків, на результати пошуку;
- швидко інформувати оператора (підтримується також голосове сповіщення) та/або надсилати сповіщення (e-mail, SMS тощо) зовнішнім службам про результати розпізнавання номерних знаків та/або результати порівняння даних транспортного засобу зі списками номерів;
- створити необхідну кількість робочих місць оператора;
- автоматично формувати звіти різного типу за результатами розпізнавання, пошуку в базі даних.

Методи і технології розпізнавання

Модуль локалізації генерує гіпотези про ймовірну присутність номерного знаку на зображенні (у кадрі) і правильно обробляє ситуації, коли в кадрі є кілька номерних знаків. Аналіз базується на пошуку ділянок кадру, які мають текстову структуру різниць яскравості. У налаштуваннях режиму розпізнавання можна вказати, чи здійснювати пошук номерних знаків у фіксованих ділянках рамки.

Положення номерного знаку в кадрі прогнозується не тільки на основі аналізу поточного зображення, а й шляхом екстраполяції результатів розпізнавання на попередні кадри.

Масштабування та бінаризація зображення. Розпізнавання GRZ здійснюється в кілька етапів аналізу двоградацийних (бінаризованих) і сірих зображень. Область кадру, задану локалізатором, масштабується, а потім бінаризується за допомогою власного алгоритму. В результаті виходить зображення, зручне для швидкого аналізу і складається лише з білих і чорних пікселів.

Розпізнавання чисел. На етапі аналізу двійкового зображення відбувається виявлення та розпізнавання знаків номерних знаків, їх вирівнювання в послідовність символів, а також пошук відсутніх символів на основі використаних номерних шаблонів певної країни.

Порівняння зображення з набором шаблонів дозволяє уникнути помилкового розпізнавання «артефактів», визначення неправильної кількості символів у номерному знаку, неправильного розпізнавання знаків, близьких один до одного тощо.

SecurOS Auto використовує шаблони, які враховують геометрію розташування символів на номерних знаках. Як правило, для правильного накладання шаблону на зображення достатньо розпізнавання трьох-чотирьох символів у числі. Це дозволяє визначити розташування решти символів і розпізнати їх з високим ступенем ймовірності. Шаблони містять інформацію про дозволені послідовності букв і цифр, їх кількість, фізичні розміри цифр -

співвідношення довжини і ширини кадрів, наявність зображення державного прапора або інших графічних елементів. SecurOS Auto дозволяє одночасно працювати з практично необмеженою кількістю наборів шаблонів з різних країн, однак для надійного розпізнавання за швидкістю надходження відео рекомендується використовувати шаблони не більше п'яти-шести країн.

Реалізований аналіз напівтонового зображення дозволяє значно підвищити точність розпізнавання. Це особливо важливо, якщо необхідно вибрати між кількома «конкуруючими» варіантами, які мають схожі варіанти написання, ступінь відмінності яких може стати ще меншим через малий розмір символів, низьку контрастність, шум тощо.

Уточнення результату розпізнавання, тобто повторне розпізнавання виконується залежно від результатів першого проходу. Початкове локалізоване зображення повторно обробляється з різними налаштуваннями контрастності, масштабуванням та іншими типами нормалізації, залежно від виявленої проблеми.

Міжкадрове злиття результатів розпізнавання та видача кінцевого результату. Розпізнавання номерного знаку здійснюється на всіх кадрах, де його розмір і контрастність знаходяться в заданих межах, а результати передаються в модуль «Відстеження» для міжкадрового порівняння, прогнозування траєкторії руху номерного знаку на наступних кадрах відеоряду, а також видача кінцевого результату (фіксація числа). Модуль «Відстеження» може одночасно «відстежувати» всі числа, знайдені в кадрі.

Для стабільно хороших результатів достатньо, щоб номерний знак був видимий не менше трьох-п'яти кадрів. Наприклад, одна камера 25-30 кадрів в секунду, яка контролює ділянку проїжджої частини довжиною 8 метрів, дозволяє отримати чотири і більше кадрів автомобіля, що рухається зі швидкістю 120-180 км/год. Такої кількості кадрів зазвичай достатньо для впевненого розпізнавання всіх символів номерного знаку.

Якість і швидкість розпізнавання залежать від таких показників, як якість оптики та експозиції камери, достатність освітленості зони спостереження, потужність обробки сервера відеоаналітики тощо. Рецензування дизайну та лабораторне тестування перед створенням специфікації проекту зберігає рівень розпізнавання на рівні 96%-99,9%.

Остаточний результат розпізнавання видається у вигляді «події», яка доступна всім модулям в «SecurOS». Залежно від налаштувань користувача, SecurOS зберігає весь фрагмент проїзду транспортного засобу або його «найкращу раму», визначену на основі проміжних результатів розпізнавання.

«SecurOS Auto» має кілька режимів видачі результатів розпізнавання, які залежать від налаштувань користувача. Двоє з них - «дорога» і «парковка» припускають, що камера встановлена стаціонарно, а третя, «мобільна», що камера встановлюється в рухомий, наприклад, патрульний автомобіль.

У режимі «дорога» розпізнаються тільки рухомі номери, а результат видається, коли номерний знак виходить з рамки або якщо він «загублений» у кадрі (зазвичай в результаті блокування).

У режимі «паркування» також розпізнаються фіксовані номери, а результат також видається у разі зупинки транспортного засобу, який раніше був у русі. В останньому випадку «SecurOS Auto» визначає, в який момент автомобіль з розпізнаним номерним знаком «вийшов з рамки»; цю подію можна перенести в систему управління шлагбаумом.

Збереження результатів розпізнавання в базі даних і збіг зі списками номерних знаків і перевірка на перевищення швидкості

Подія розпізнавання обробляється, серед іншого, в модулі «Контроль» номерний знак належить тій чи іншій смузі, швидкість транспортного засобу, що проїжджає, визначається за даними радара, результат розпізнавання зберігається в базі даних SecurOS Auto і перевіряється за внутрішнім і зовнішнім списками. Результат перевірки, наприклад, наявність цього номера в базі даних крадіжок або перевищення дозволеної швидкості на цій смузі, також зберігається у власній

базі даних, і, крім того, відправляється як подія в SecurOS. В результаті в інтерфейсі користувача «SecurOS Auto» з'являється рядок «тривожний».

1.3 Перевірка систем розпізнавання номерних знаків

Співробітники випробувальної лабораторії протестували системи розпізнавання номерних знаків, надані різними компаніями.

Методика проведення тестів, достатня кількість тестового матеріалу, можливість присутності представників фірм-конкурентів під час тестування - виключають суб'єктивність або неоднозначність в інтерпретації результатів.

У статті, де були представлені результати аналізу, вводиться поняття "якість системи", K . Якість системи пропорційна кількості правильно розпізнаних номерних знаків T і обернено пропорційна кількості нерозпізнаних номерних знаків $(100 - T)$ і "сміття", що складається з помилкових і неправильних розпізнавання - F . Таким чином, показник якості системи можна розрахувати за формулою:

$$K = \frac{T}{(100 - T)} \times 100 \quad (1.9)$$

1.4 Вимоги до розробленого алгоритму

За результатами випробувань на ринку систем розпізнавання номерних знаків можна сформулювати основні вимоги до алгоритму розпізнавання. Основним показником якості алгоритму є ймовірність розпізнавання.

Середня ймовірність розпізнавання становить понад 80%. Таким чином, розроблений алгоритм розпізнавання повинен давати результати з ймовірністю розпізнавання 80% і вище.

Швидкість розпізнавання одного номерного знаку в існуючих програмних системах становить 30–150 мс. Розроблений алгоритм повинен мати швидкість розпізнавання, порівнянну з заданим діапазоном, або працювати швидше.

Елемент класифікації повинен бути виготовлений на основі апарату штучних нейронних мереж (ШНМ).

Висновки до розділу 1

В даному розділі було розглянуто основні методи розпізнавання номерних знаків, детально розглянуто аналоги подібного програмного забезпечення відповідного темі кваліфікаційної роботи. Завдяки аналізу аналогів було сформовано план роботи. Було обрано найоптимальніший метод для розпізнавання номерних знаків спираючись на співставлення найбільш поширених методів.

РОЗДІЛ 2 АЛГОРИТМ РОЗПІЗНАВАННЯ АВТОМОБІЛЬНИХ НОМЕРІВ НЕЙРОМЕРЕЖАМИ

2.1 Загальне рішення задачі

Для розпізнавання автомобільних номерів широко використовуються нейромережі - мережі нейронів, які можуть виконувати складні обчислення та виявляти закономірності у великих наборах даних. Загальне рішення задачі розпізнавання автомобільних номерів за допомогою нейромереж включає кілька кроків:

1. Збір даних: Спершу необхідно зібрати велику кількість зображень автомобільних номерів з різних куточків та в різних умовах освітлення. Ці дані будуть використовуватися для навчання нейромережі та перевірки її точності.

2. Передпроцесинг даних: Перед подачею даних на вхід нейромережі їх необхідно підготувати. Це може включати зміну розміру зображень, видалення шуму, нормалізацію яскравості та контрастності тощо.

3. Побудова нейромережі: Для розпізнавання автомобільних номерів можна використовувати різні типи нейромереж, такі як згорткові нейронні мережі (CNN) або рекурентні нейронні мережі (RNN). Нейромережі будується за допомогою спеціальних програмних бібліотек, які надають можливість налаштування параметрів моделі та тренування.

4. Навчання нейромережі: Для навчання моделі нейромережі використовуються зібрані дані. Нейромережа навчається розпізнавати зображення автомобільних номерів із відповідними позначками. Процес тренування може зайняти значну кількість часу в залежності від розміру набору даних та складності моделі.

5. Тестування та оцінка результатів: Після завершення тренування моделі потрібно перевірити її ефективність. Це включає використання тестового набору даних, які не брали участь у тренуванні, для перевірки точності розпізнавання

номерів автомобілів. Результати можуть бути оцінені за допомогою метрик, таких як точність, відновлення та F-міра.

Застосування неймереж для розпізнавання автомобільних номерів дозволяє автоматизувати цей процес і значно збільшити ефективність роботи системи. Однак, варто враховувати, що ефективність такої системи може залежати від якості тренувальних даних, параметрів моделі та розміру набору даних. Налаштування цих параметрів може вимагати досліджень та експериментів для досягнення оптимальних результатів.

Процес розв'язання задачі розпізнавання автомобільного номерного знаку при в'їзді до автомийки, в загальних рисах можна представити у послідовності наступних кроків, показаних на рис. 2.1.

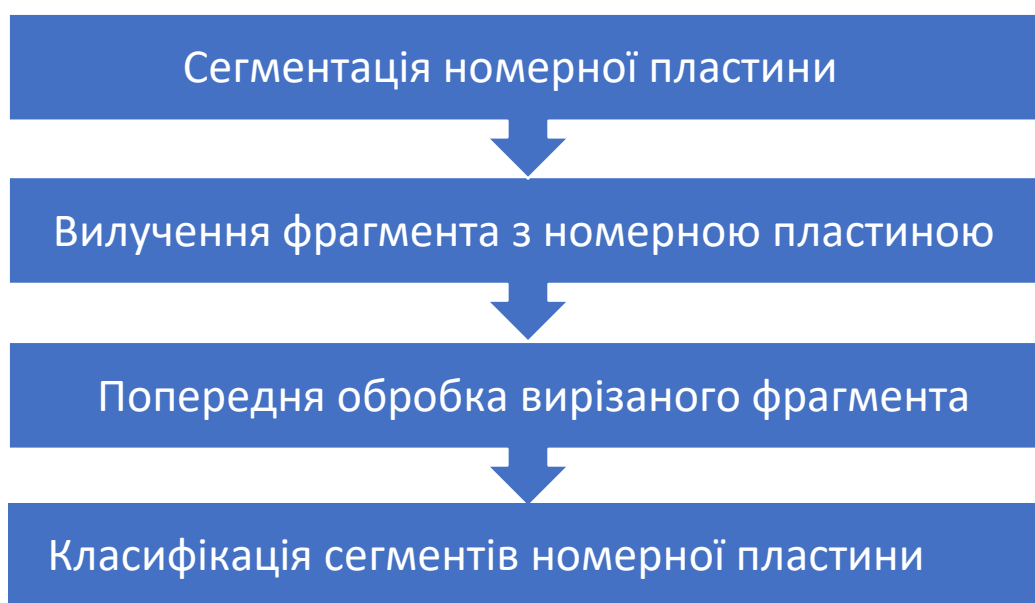


Рис. 2.1. Загальна схема розв'язання задачі розпізнавання автомобільного номерного знаку

У цій роботі ми пропонуємо алгоритм вилучення та розпізнавання цифрового зображення номерного знаку.

2.2 Поетапна робота з зображенням

Розглянемо більш детально етапи розв'язання задачі.

2.2.1 Сегментація номерних знаків

Для сегментації номерних знаків використовується U-Net SNS, архітектура мережі показана на рис. 2.2.

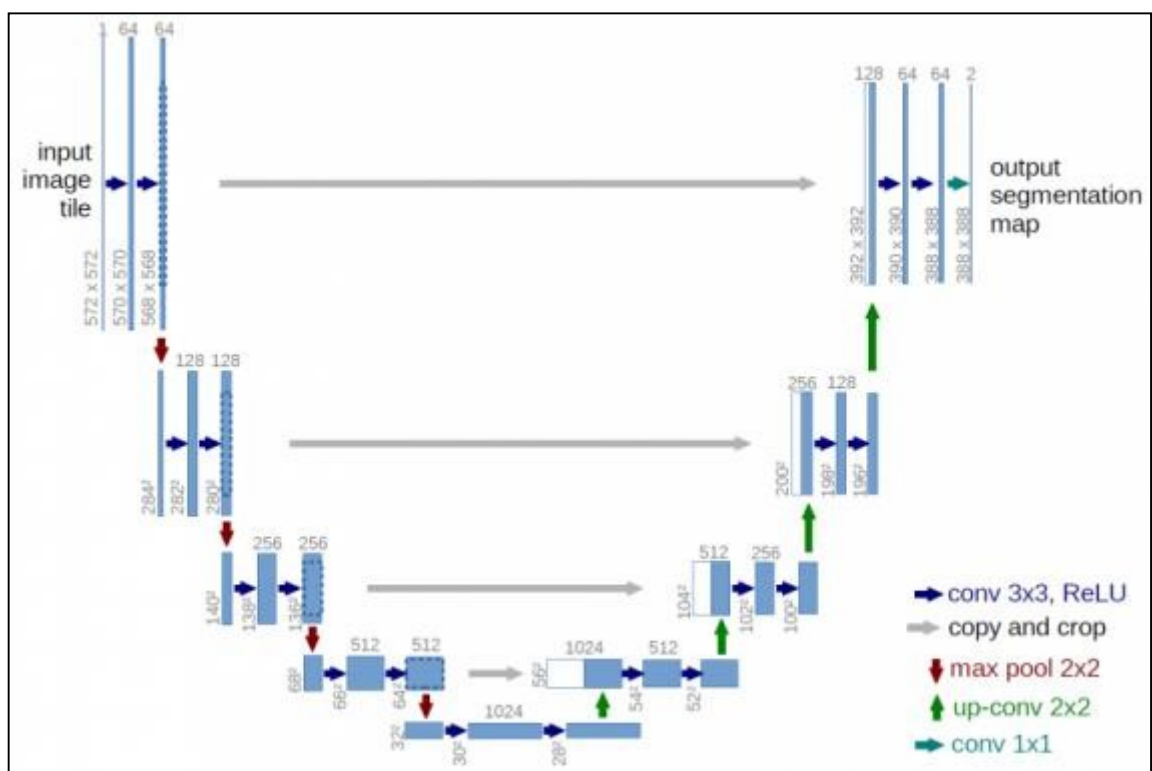


Рис. 2.2. Архітектура мережі U-Net.

На рис. 2.2 кожен синій квадрат відповідає карті багатоканальних властивостей. Кількість каналів показано у верхній частині квадрата. Розмір вказано в нижньому лівому краю квадрата. Білі квадрати є копіями карти власності. Стрілки представляють різні операції.

Архітектура мережі складається з шляху звуження (ліворуч) і шляху розширення (праворуч). Шлях звуження є типовою архітектурою згорткової

нейронної мережі [13]. Він складається з повторного застосування двох згорток 3×3 з наступною функцією активації ReLU і максимальним об'єднанням (2×2 потужність 2) для зниження роздільної здатності.

На кожному кроці зниження дискретизації функціональні канали подвоюються. Кожен крок у шляху розширення складається з операції підвищення дискретизації карти властивостей, за якою слідує:

- згортка 2×2 , що зменшує кількість функціональних каналів;
- об'єднано з відповідно обрізаною картою властивостей із скорочення шляху;
- дві складки 3×3 , а потім ReLU.

Обрізка необхідна через втрату граничних пікселів з кожною згорткою.

Останній шар використовує згортку 1×1 для відображення кожного 64-компонентного вектора ознак на бажану кількість класів [15], [16], [17].

Налаштування мережі здійснюється шляхом обчислення коефіцієнта Дайса (також званого коефіцієнтом Соренсена-Дайса [23]) або коефіцієнта подібності Жаккара [24], який показує міру подібності [22] — у цьому випадку показує міру площі правильно позначені відрізки (відношення площі перетину до площі об'єднання), надалі IoU (Intersection over union) [25].

Вхід нейронної мережі — це зображення реальної сцени, на якій є номерний знак автомобіля. Приклад вхідного зображення на рис. 2.3.



Рис. 2.3. зображення, що надходить на вхід нейронної мережі.

Для кожного зображення була створена маска, яка закриває номерний знак. Приклад маски для зображення на рис. 2.3 показано на рис. 2.4.



Рис. 2.4. Приклад маски, нанесеної на вхід нейронної мережі

На виході нейронної мережі отримуємо маску, потім, накладаючи її на вихідне зображення, отримуємо зображення з сегментованим номерним знаком, рис. 2.5.



Рис. 2.5. Зображення, отримане на виході нейронної мережі

2.2.2 Вилучення фрагмента з номерним знаком

Для подальшої роботи сегментований номерний знак необхідно вирізати, отримавши його координати.

Для отримання координат далі будемо працювати з сегментованим фрагментом, маскою, отриманою на виході нейронної мережі (рис. 2.6).



Рис. 2.6. Маска, що покриває номерний знак, отримана на виході нейронної мережі

До маски застосовується двосторонній фільтр, який є нелінійним фільтром згладжування зображень, що зберігає краї. Він замінює інтенсивність кожного пікселя середньозваженим значенням інтенсивності сусідніх пікселів. Ця вага може бути заснована на розподілі Гаусса. Важливо, що ваги залежать не тільки від евклідової відстані в пікселях, а й від радіометричних відмінностей (наприклад, різниці в діапазоні, такі як інтенсивність кольору, відстань глибини тощо). Це дає можливість зберегти гострі краї [26], [27], [28], зміни практично непомітні для ока.

Потім до відфільтрованої маски застосовують метод знаходження особливих точок — детектор Харріса, який є оптимальним детектором зв'язаних кутів [29, 30]. Потім створюється нове зображення і отримані точки накладаються на знайдені координати, результат на рис. 2.7.

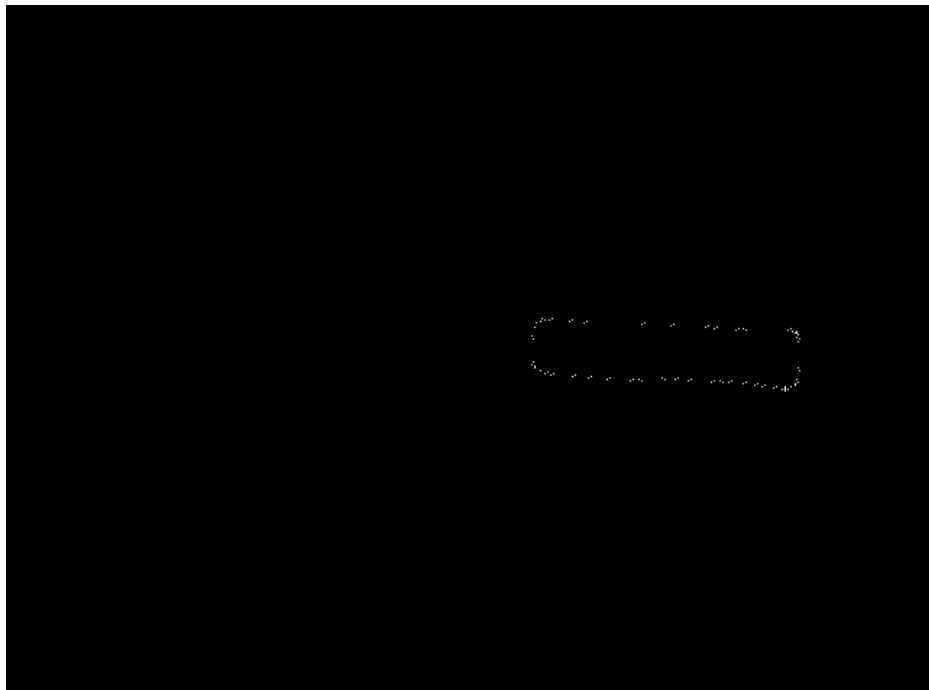


Рис. 2.7. Результат пошуку особливих точок

Білі пікселі — це області можливих кутів. Ви можете знайти багато кутів, що примикають один до одного. Щоб знайти чотири необхідні, обчислюється відстань кожного кута один до одного. Зберігаються лише ті кути, відстань яких

перевищила певний поріг, який змінюється до тих пір, поки не буде знайдено лише чотири необхідні точки.

Після отримання кутів вони можуть бути не в правильному порядку, сортуємо їх у правильному порядку та будуємо нове зображення, використовуючи отримані точки – перспектива [31], рис. 2.8, за допомогою набору інструментів бібліотеки OpenCV [33], [33].



Рис. 2.8. Вирізаний номерний знак

2.2.3 Попередня обробка вирізаного фрагмента

Далі потрібно отримати кожен окремий буквено-цифровий символ.

Перетворити в градації сірого. Grayscale (grayscale, grayscale [англ. Grayscale]) - колірний режим зображень, які відображаються у відтінках сірого, розміщені у вигляді таблиці як еталони яскравості білого кольору. Найчастіше використовується ступінчасте зображення рівномірного ряду оптичних густин нейтральних сірих полів. Він широко використовується в кольорознавстві та колористиці, для оцінки та вимірювання якості відтворення тону у фотозйомці, процесах сканування, копіювання та друку (друк).

Для перетворення кольорового зображення в градації сірого використовується набір інструментів бібліотеки OpenCV [34]. Результат перетворення показано на рис. 2.9.



Рис. 2.9. Результат зміни колірного простору (зображення у відтінках сірого)

Порогова обробка та бінаризація.

Далі зображення обмежується та бінаризується. Результат бінаризації зображення номерного знаку наведено на рис. 2.10.



Рис. 2.10. Результат порогового значення

Виключення неінформативних областей.

Не всі області представляють інтерес для проблеми розпізнавання номерних знаків. За допомогою методу зв'язаних компонентів ми відокремлюємо сегменти один від одного, замінюючи значення пікселів кожного окремого сегмента певними значеннями [35], а потім видаляємо об'єкти з площею, меншою за задане значення, рис. 2.11.



Рис. 2.11. Результат виключення неінформативних областей

Витягнення сегментів.

На наступному етапі всі окремі сегменти вирізаються для подальшого розпізнавання. На рис. 2.12 показаний результат вилучення сегментів із зображення номерного знаку.



Рис. 2.12. Результат вилучення відрізків

Кожен витягнутий сегмент зменшується до розміру 28×28 пікселів, щоб сформувати вектор ознак кожного буквено-цифрового символу номерного знаку.

2.2.4 Класифікація сегментів номерних знаків

Завданням класифікації є завдання віднесення зображення до однієї з поданих множин[36]. Кожна область зображення номерного знаку, отримана в результаті сегментації, є алфавітно-цифровим зображенням з алфавіту дійсних символів.

Архітектура штучної нейронної мережі (ШНМ) для класифікації буквено-цифрових символів представлена таким чином:

- 1 шар згортки з вікном згортки 3×3 , з функцією активації ReLU
- 1 прихований повністю підключений рівень з функцією активації ReLU
- повністю підключений рівень, з функцією логістичної активації на виході мережі.

На вхід ШНМ подається набір буквено-цифрових зображень, на виході кожне з яких відноситься до певного класу, потім формується кінцевий результат, який представляє собою номер автомобіля в текстовому форматі.

Для підвищення точності використовується ансамбль штучних нейронних мереж (ШНМ).

Для навчання ANS (комітету) використовується алгоритм змішування. Алгоритм є одним із основних алгоритмів, у якому рішення комітету приймається шляхом усереднення рішень окремих класифікаторів.

Для навчання класифікаторів використовується метод розбиття навчальної вибірки (bootstrap). Суть методу полягає в тому, щоб розділити навчальну вибірку на M рівних частин, що не перекриваються, відповідно до кількості використаних класифікаторів, причому кожен класифікатор навчається без

однієї частини даних. Таким чином, перший класифікатор навчається без першої частини, другий — без другої тощо. Такий підхід гарантує навчання кожного класифікатора на різних даних.

Вихідний сигнал ансамблю ШНМ при використанні алгоритмом бегінга розраховується за формулою:

$$F(n) = \frac{1}{M} \sum_{i=1}^M F_i(n) \quad (2.1)$$

де $F_i(n)$ - вихід i -ї ІНС, M - кількість ІНС, що утворюють ансамбль.

У цільову функцію вводиться додатковий штраф за корелювання вихідних сигналів ІНС:

$$E_i = \frac{1}{N} \sum_{n=1}^N E_i(n) = \frac{1}{N} \sum_{n=1}^N \frac{1}{2} (E_i(n) - d(n))^2 + \frac{1}{N} \sum_{n=1}^N \lambda p_i(n) \rightarrow \min \quad (2.2)$$

де $E_i(n)$ – помилка i -ї ІНС на n -му навчальному прикладі, $d(n)$ – необхідний вихідний сигнал для n -го навчального прикладу, N – розмір навчальної вибірки, λ – параметр, що регулює вплив штрафу.

Другий доданок обчислюється наступним чином:

$$p_i(n) = (F_i(n) - F(n)) \sum_{i \neq j}^N F_j(n) - F(n) \quad (2.3)$$

Мінімізація цільової функції (ЦФ) для ансамблю відповідає мінімізації цільової функції кожної мережі окремо. Формування ансамблю відбувається із використанням однорідних архітектур.

Висновки до розділу 2

В другому розділі було описано загальні алгоритми необхідні для вирішення поставленої задачі. Розглянуто такі процеси як сегментація та описана поетапна робота з фото, а саме обробка зображень для використання в розпізнаванні автомобільних номерів.

Також було виведено формули які в подальшому допоможуть в роботі з ШІ, а саме в класифікації сегментів та формуванні ансамблю.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Мова розробки

На сьогоднішній день існує величезне розмаїття мов програмування та програмних бібліотек для роботи зі штучними нейронними мережами та генетичними алгоритмами. Ці бібліотеки використовуються на різних мовах програмування, таких як C++, C#, Python та інші.

Була вибрана мова програмування Python як інструмент для вирішення завдань та реалізації алгоритмів. Python є високорівневою мовою загального призначення, спрямованою на підвищення продуктивності розробника і зрозумілість коду. Синтаксис ядра Python має мінімалістичний характер. Водночас, стандартна бібліотека містить велику кількість корисних функцій [15].

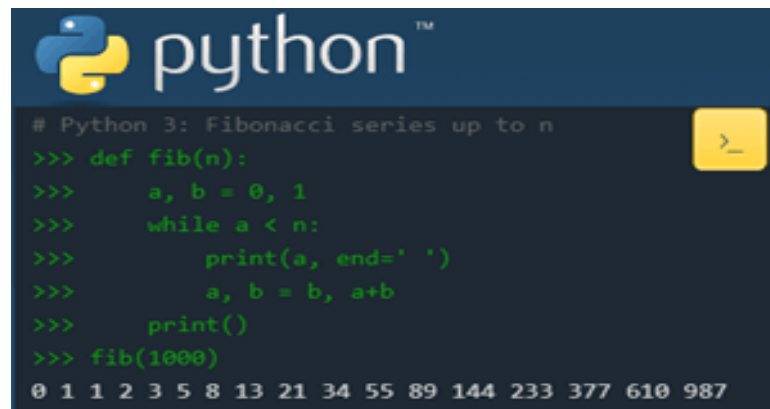
Python підтримує різноманітні парадигми програмування, такі як структурне, узагальнене, об'єктно-орієнтоване, функціональне та аспектно-орієнтоване програмування. Він має ряд архітектурних особливостей, таких як динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопотокових обчислень та високорівневі структури даних.

Можливість розбиття програм на модулі, які можуть бути об'єднані в пакети, також є однією з ключових функцій Python. Розробка мови Python була почата наприкінці 1980-х років Гвідо ван Россумом, співробітником голландського інституту cwi. Він розпочав розробку Python як хобі для розподіленої операційної системи атосеба, яка потребувала розширюваної скриптової мови. Під час написання Python, Гвідо використав деякі ідеї з мови abc, в якій він також брав участь у розробці. Перший вихідний код Python був опублікований в групі новин alt.sources у лютому 1991 року.

Розвиток мови відбувається шляхом виконання чітко визначеного процесу створення, обговорення, відбору та реалізації пропозицій щодо розвитку Python,

відомих як документи PEP (Python Enhancement Proposal). Варто зауважити, що назва цієї мови програмування не походить від назви родини плазунів. Вона отримала свою назву на честь відомого британського комедійного телешоу 1970-х років під назвою "Літаючий цирк Монті Пайтона" [17]. Проте, все ж таки назву "Python" частіше пов'язують зі змією, а не з телешоу - піктограми файлів в середовищі KDE та Microsoft Windows, а також логотип на сайті python.org (до версії 2.5) зображують зміїні голови. Одна з важливих мет розробників Python - створити цікаву мову програмування для використання, що відображено в її назві, натхненні Монті Пайтоном.

Приклад програми на мові програмування Python наведено на рис. 3.1.

A screenshot of a Python 3 interpreter window. The window has a dark blue header with the Python logo and the word "python" in white. Below the header, the text "# Python 3: Fibonacci series up to n" is displayed. The code being executed is as follows:

```
>>> def fib(n):  
>>>     a, b = 0, 1  
>>>     while a < n:  
>>>         print(a, end=' ')  
>>>         a, b = b, a+b  
>>>     print()  
>>> fib(1000)
```

The output of the program is displayed at the bottom: "0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987".

Рис. 3.1. Приклад програми на мові Python

Бібліотека `opencv` є відкритою системою комп'ютерного зору, обробки зображень і числових алгоритмів, що має відкритий код. Вона розроблена для створення загальної інфраструктури для програм, що працюють з комп'ютерним зором, і для прискорення використання машинного сприйняття у комерційних продуктах. Ліцензований за ліцензією BSD, `opencv` дозволяє підприємствам легко використовувати і модифікувати його код. [18]

В бібліотеці `OpenCV` доступно понад 2500 оптимізованих алгоритмів, які включають класичні та сучасні методи комп'ютерного зору та машинного навчання. Вона може бути використана для різноманітних завдань, таких як

виявлення облич, ідентифікація об'єктів, класифікація дій людини на відео, відстеження руху камери та об'єктів, створення 3D-моделей об'єктів, отримання точок 3D хмар зі стереокамер, пошук схожих зображень у базі даних, видалення червоних очей на фотографіях зі спалаху, відстеження рухів очей, розпізнавання пейзажів та встановлення маркерів для доповненої реальності, а також багато іншого. OpenCV має понад 47 000 користувачів у спільноті та була завантажена понад 18 мільйонів разів. Ця бібліотека широко використовується компаніями, науковими групами та державними установами.

Поряд із відомими компаніями, такими як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda і Toyota, які використовують бібліотеку, наявні численні стартапи, такі як Applied Minds, VideoSurf і Zeitera, що широко використовують OpenCV. OpenCV підтримує інтерфейси C++, Python, Java і Matlab, а також робочі системи Windows, Linux, Android і macOS. Головним застосуванням є реальні часові програми перегляду, які використовують інструкції MMX та SSE, якщо вони доступні. Наразі активно розробляються повноправні інтерфейси CUDA і OpenCL. В OpenCV налічується понад 500 алгоритмів і близько в 10 разів більше функціональності, що об'єднує або підтримує ці алгоритми. OpenCV реалізований на C++ і має шаблонний інтерфейс, який добре сумісний з контейнерами STL [12].

Бібліотека os

Модуль os, який входить до стандартної бібліотеки мови програмування Python, зазвичай використовується для взаємодії з операційною системою і файловою системою комп'ютера. Він надає багато корисних функцій для роботи з файлами і папками на жорсткому диску. Програми, що використовують модуль os, можуть працювати на будь-якій операційній системі і легко переноситися на іншу платформу. У Python модуль os є бібліотекою функцій для роботи з операційною системою [12]. Його методи дозволяють визначати тип операційної

системи, отримувати доступ до змінних оточення, управляти директоріями і файлами. Наприклад, можна перевіряти наявність об'єкта за заданим шляхом, визначати його розмір в байтах, видаляти, перейменовувати та виконувати інші дії. При виклику функцій `os` варто пам'ятати, що деякі з них можуть не підтримуватися поточною операційною системою.

Для використання методів з `os` необхідно підключити бібліотеку, наприклад, за допомогою команди імпорту `import os` в Python. Цю інструкцію слід додати до файлу перед першим використанням методів модуля `os`. Рекомендується розмістити цю інструкцію в початковій частині файлу з вихідним кодом. Приклад реалізації модуля представлено на рис. 3.2.

```
import os
print(os.name)

nt
```

Рис. 3.2. Приклад отримання інформації про ОС

3.2 Середовище розробки

Для реалізації платформи для створення програми з метою розпізнавання державних номерів для врахування часу стоянки та оплати було обрано середовище розробки `PyCharm`. `PyCharm` - це надійна та потужна інтелектуальна IDE для розробки на мові програмування Python, яка має повний набір інструментів для ефективного розробки застосунків. Цей інструмент доступний у двох варіантах: безкоштовна версія `PyCharm Community Edition` і розширена версія підписки - `PyCharm Professional Edition`, яка має більш широкі можливості. `PyCharm` виконує перевірку коду у режимі реального часу, включає автодоповнення, засноване на виконаному коді, навігацію по коду, а також надає безліч можливостей для рефакторингу [17].

PyCharm, як інтегроване середовище розробки для мови програмування Python, надає всі необхідні інструменти для аналізу коду, відладки за допомогою графічного відладчика, запуску юніт-тестів та підтримки веб-розробки на Django. Це програмне забезпечення розроблене компанією JetBrains, базуючись на IntelliJ IDEA. За більше ніж 10 років розвитку платформи IntelliJ, PyCharm отримала більше 50 різноманітних плагінів, включаючи підтримку додаткових систем контролю версій, інтеграції з різними інструментами та фреймворками, а також можливість використання редактора оновлень, що підтримує емуляцію Vim.



Рис. 3.3. PyCharm

Ключові можливості середовища розробки:

- потужний та функціональний редактор коду з підсвічуванням синтаксису, авто-форматуванням та авто-відступами для підтримуваних мов;
- проста й потужна навігація в коді;
- допомога при написанні коду, що включає в себе автодоповнення, авто-імпорт, шаблони коду, перевірка на сумісність версії інтерпретатора мови, та багато іншого;

- швидкий перегляд документації для будь-якого елемента прямо у вікні редактора, перегляд зовнішньої документації через браузер, підтримка docstring – генерація, підсвічування, автодоповнення та багато іншого;
- велика кількість інспекцій коду;
- потужний рефакторинг коду, який надає широкі можливості щодо виконання швидких глобальних змін у проекті [17].

3.3 Розробка програми

Для початку роботи були підключені наступні бібліотеки

```
import os
from datetime import timedelta
from pathlib import Path
import matplotlib.pyplot as plt
import cv2
import _pickle as cPickle
import numpy as np
import re
import imutils
import pytesseract
import pytesseract as tess
tess.pytesseract.tesseract_cmd = r'Tesseract-OCR\tesseract.exe'
from PyQt6 import uic
from PyQt6.QtGui import QPixmap
from PyQt6.QtWidgets import QApplication, QFileDialog
from imutils import contours
from moviepy.editor import VideoFileClip
import json
import sqlite3
import easyocr
```

Для розбиття відео на фрагменти було створено наступну частину коду

```
def split():
    video_clip = VideoFileClip(video_file)
    filename, _ = os.path.splitext(video_file)

    if not os.path.isdir(filename):
        os.mkdir(filename)

    saving_frames_per_second = min(video_clip.fps, SAVING_FRAMES_PER_SECOND)
    step = 1 / video_clip.fps if saving_frames_per_second == 0 else 1 /
    saving_frames_per_second

    for current_duration in np.arange(0, video_clip.duration, step):
        frame_duration_formatted =
        format_timedelta(timedelta(seconds=current_duration)).replace(":", "-")
        frame_filename = os.path.join(filename,
        f"frame{frame_duration_formatted}.jpg")
```

```
video_clip.save_frame(frame_filename, current_duration)
print("split")
```

Після того як відео розіб'ється на багато зображень, для розпізнавання зображення було створено окрема функція.

Після обробки зображень, для виводу інформації було написана наступна частина коду

```
def recognize():
    load_file()
    global result
    image = cv2.imread(image_file)
    height, width, _ = image.shape
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU)[1]
    cnts = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    cnts, _ = contours.sort_contours(cnts[0])

    for c in cnts:
        area = cv2.contourArea(c)
        x, y, w, h = cv2.boundingRect(c)
        if area > 20000:
            img = image[y:y+h, x:x+w]
            gray2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            thresh2 = cv2.threshold(gray2, 0, 255, cv2.THRESH_OTSU)[1]
            res = tess.image_to_string(thresh2)
            if len(res) > 7:
                form.label_7.setText(res)
            result = res
```

В залежності від кількості відвідувань розроблена відповідна схема знижок

```
count = 0;
for i in range(len(arr)):
    if arr[i][0] == result:
        count += 1
if count == 0:
    arr.append([result, 1])
else:
    for i in range(len(arr)):
        if arr[i][0] == result:
            arr[i][1] += 1
for i in range(len(arr)):
    if arr[i][0] == result:
        form.label_2.setText(str(arr[i][1]))
        if (arr[i][1] < 5):
            form.label_5.setText('0%')
        elif (arr[i][1] >= 5 and arr[i][1] < 10):
            form.label_5.setText('5%')
        elif (arr[i][1] >= 10 and arr[i][1] < 20):
            form.label_5.setText('10%')
        else:
            form.label_5.setText('15%')
```

На кожну окрему кнопку була створена відповідна подія

```
form.pushButton.clicked.connect(load);
form.pushButton_2.clicked.connect(split);
form.pushButton_3.clicked.connect(choose);
form.pushButton_4.clicked.connect(recognize);
```

Також для по кадрового розбиття була створена наступна функція

```
def format_timedelta(td):
    result = str(td)
    try:
        result, ms = result.split(".")
    except ValueError:
        return result + ".00".replace(":", "-")

    ms = round(int(ms) / 10000)
    return f"{result}.{ms:02}".replace(":", "-")
```

3.4 Робота з базою даних

Розробка корпоративних проєктів має на увазі створення БД, які вирішують питання пошуку, класифікації та обліку корисних відомостей, роблячи робочий процес автоматизованим. Без цього жодна ПО і веб-система не зможуть справно і оперативно функціонувати.

В процесі задіяні спеціальні програмні засоби - системи управління БД (СУБД), що дозволяють:

- створювати БД;
- виробляти доступ до інформації;
- вести роботу з інформацією (додавати, редагувати, видаляти) і забезпечувати безпеку даних.

Розробка баз даних здійснюється для оперування інформацією в реляційної БД за допомогою СУБД. Якісно структуровані сховища, які функціонують безперебійно, здатні вирішувати питання без залучення допоміжних ресурсів з боку персоналу.

В автоматизованих сховищах передбачаються допоміжні опції:

- облік клієнтів;
- відстеження обліку коштів і товарів;

- формування детальної статистики;
- ведення документообігу та ін.

Розробка програми бази даних дозволяє власнику бізнесу мати доступ до актуальних відомостей про функціонування всієї організації. При залученні якісної системи управління можна скоротити критичні навантаження на сервери підприємства, організовуючи рівномірний розподіл, гарантуючи зменшення витрат на електрику, а також обслуговування техніки.

Етапи розробки бази даних

Проектування і розробка бази даних здійснюється в кілька етапів:

- уточнюється технічне завдання;
- формується функціонуючий приклад БД;
- створений зразок запускається в дію;
- в ході кваліфікованого використання коригуються і уточнюються вимоги до бази даних;
- виходячи з наданої інформації, створюється БД;
- отриманий продукт проходить тестування, в ході якого усуваються допущені помилки;
- після база даних запускаються, і підтримується надалі.

При створенні програм БД клієнти можуть вирішити такі завдання:

1. Формування загального інформпростору для працівників.

Розглядається можливість підключення до бази і клієнтів, і партнерів.

2. Поділ доступу до даних.
3. Інформування про реальну картину розвитку бізнесу.
4. Скорочення тимчасових витрат на пошук необхідних відомостей.

Вистачить і кількох секунд з моменту введення запиту.

5. Гарантується розробка і захист баз даних від імовірної втрати відомостей, злому і т.д.

6. Припустимо звернення до БД через будь-який пристрій, підключений до Інтернету.

Фізична модель даних, навпаки, залежить від конкретної БД, фактично будучи відображенням системного каталогу. У фізичній моделі міститься інформація про всі об'єкти БД. Оскільки стандартів на об'єкти БД не існує (наприклад, немає стандарту на типи даних), фізична модель залежить від конкретної реалізації СУБД. Отже, однією і тією ж логічної моделі можуть відповідати кілька різних фізичних моделей. Якщо в логічній моделі не має значення, який саме тип даних має атрибут, то у фізичній моделі важливо описати всю інформацію про конкретних фізичних об'єктах - таблицях, колонках, індексах, процедурах і т.д. Поділ моделі даних на логічні і фізичні дозволяє вирішити кілька важливих завдань.

Всі дані зберігаються у базі даних Sqlite 3. Підключення до бази даних відбувалося завдяки наступному коду

```
import sqlite3
db = sqlite3.connect('server.db')
sql = db.cursor()
```

Створення таблиці було реалізовано наступним чином:

```
sql.execute("""CREATE TABLE IF NOT EXISTS carwash (number TEXT, countvisit
INT) """)
```

А сама база даних виглядає таким чином (рис. 3.4)

```
('XX 7777 AA\n', 4)
('~ AA 0000 AA\n', 1)
```

Рис. 3.4. Структура БД(cars)

3.5 Результат роботи програми

Для запуск програми необхідно відкрити файл, та запустити у PyCharm. Перед вами з'явиться наступна картина (рис. 3.5)

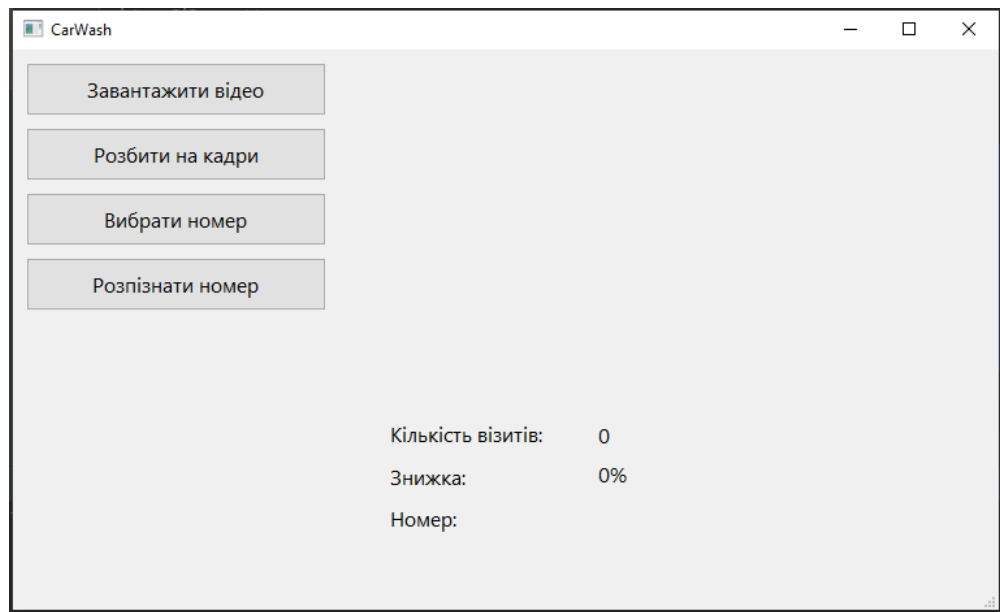


Рис 3.5. початок програми

Далі натиснувши на кнопку «Завантажити відео», та вибрати необхідне відео, пройде завантаження відео (рис. 3.6)

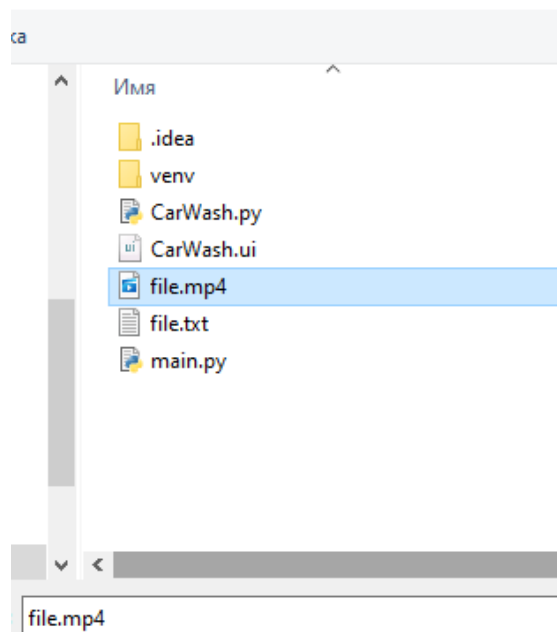


Рис. 3.6. завантаження відео

Потім натиснувши на кнопку «Розбити на кадри», програма розіб'є відео на кадри та збереже їх у відповідну папку (рис. 3.7)

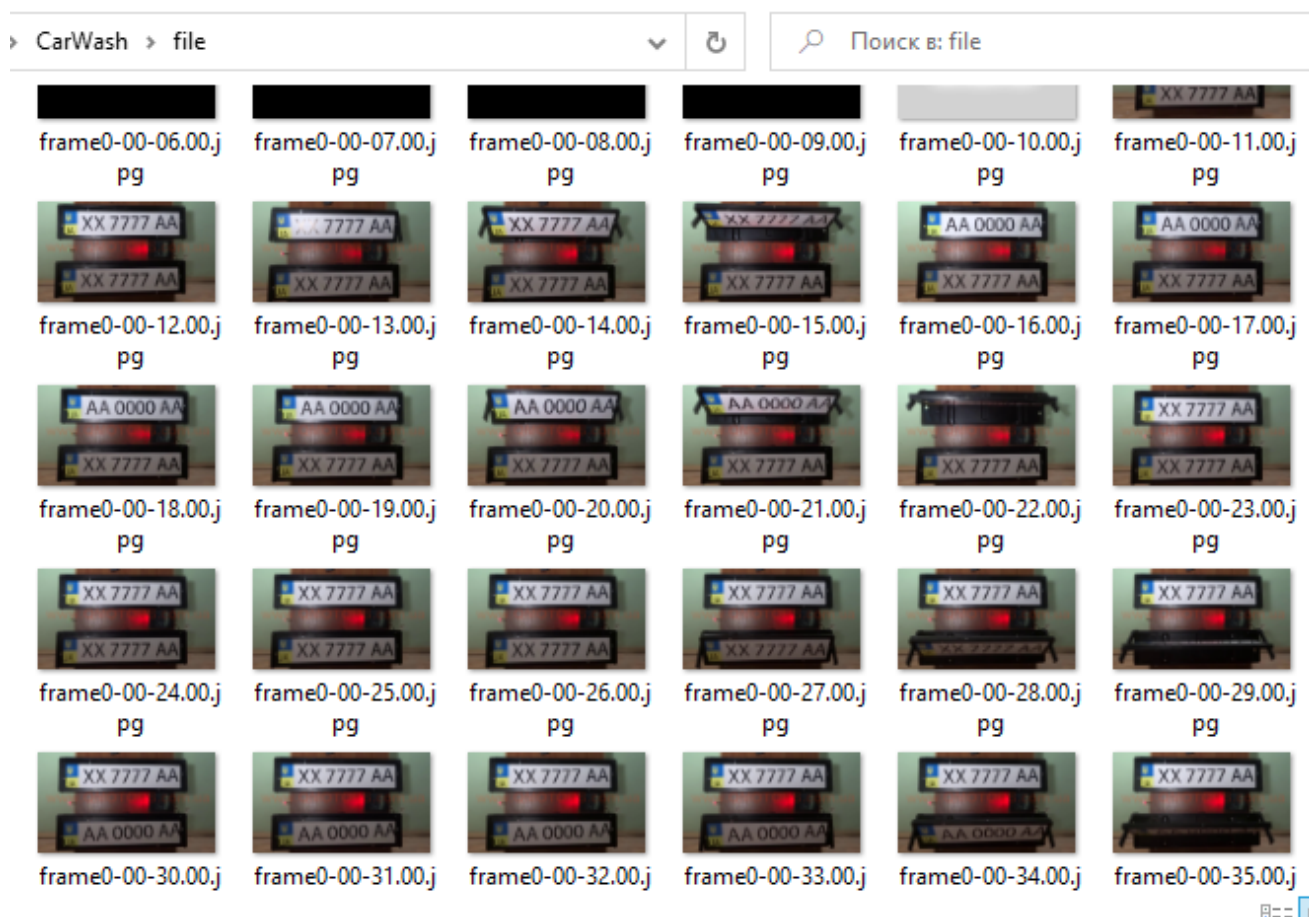


Рис. 3.7. Розбиття на кадри

Після чого можна зайнятися розпізнаванням зображень. Для цього натисніть на кнопку «Вибрати номер» та виберіть зображення. Перед вами з'явиться вибране вами зображення (рис. 3.8)

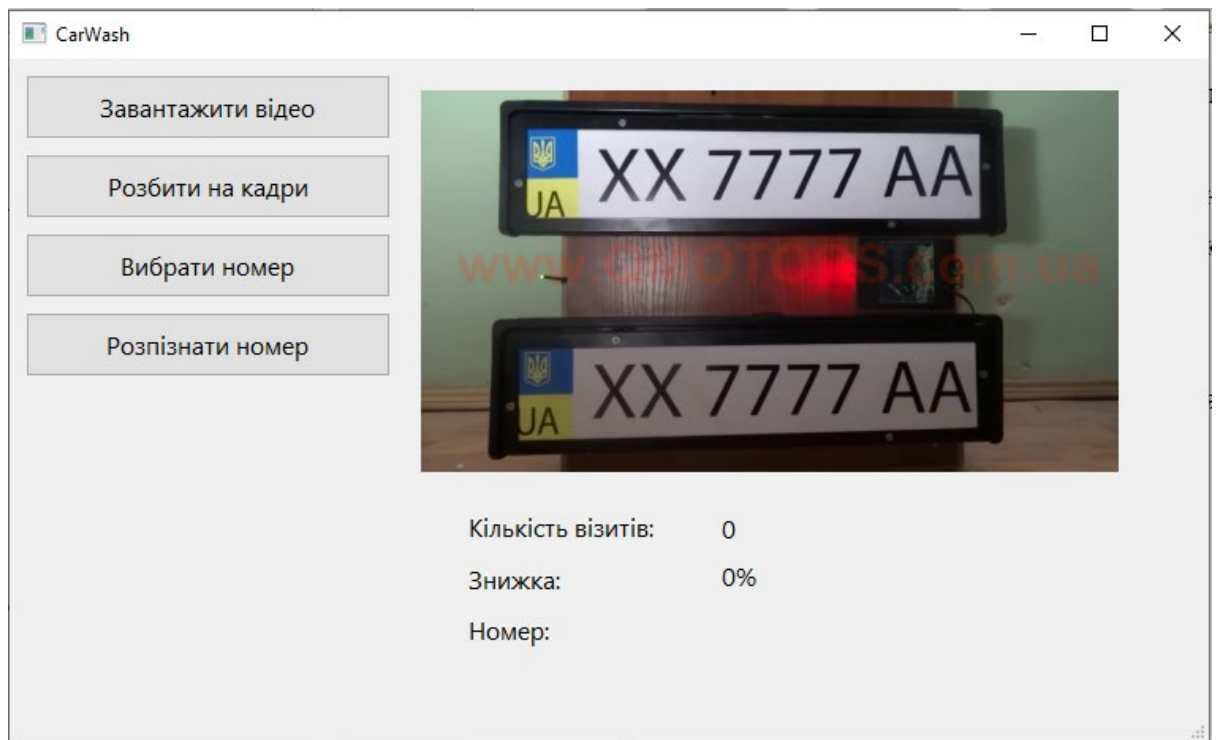


Рис. 3.8. завантаження зображення

Далі натиснувши на кнопку «Розпізнати номер» перед вами з'явиться наступна інформація (рис. 3.9)

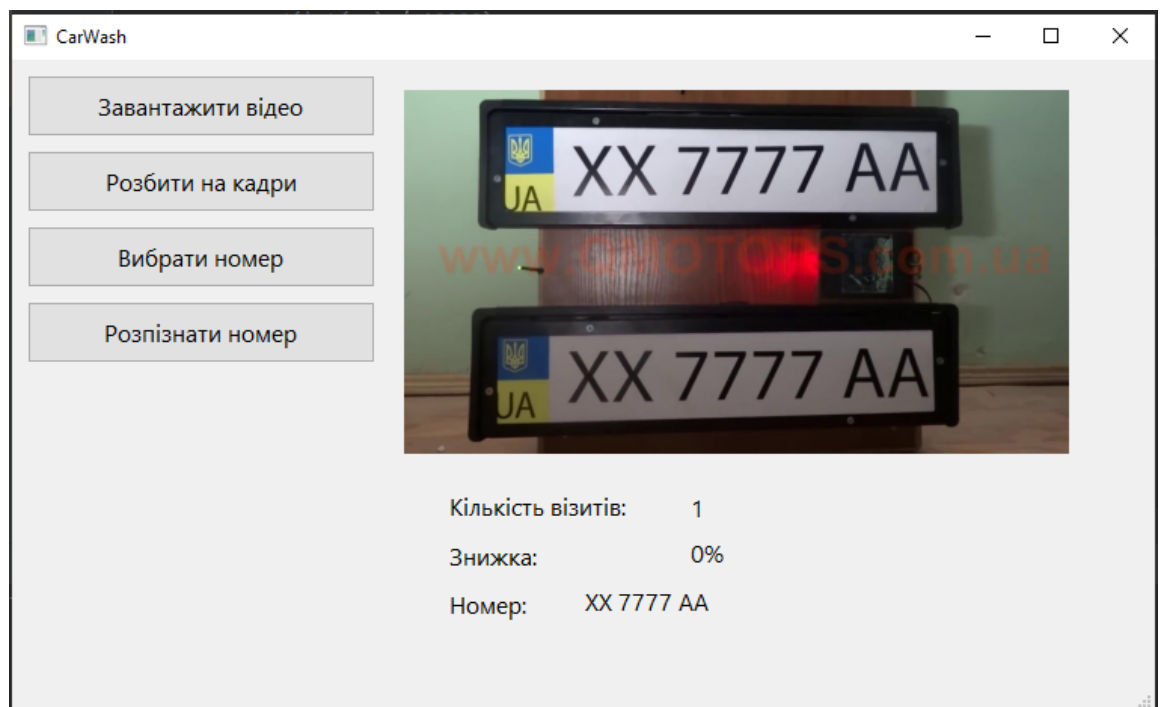


Рис. 3.9. Інформація про авто

У випадку, якщо авто було на мийці багато разів, то інформація відповідно покаже про це, та покаже знижку (рис. 3.10)

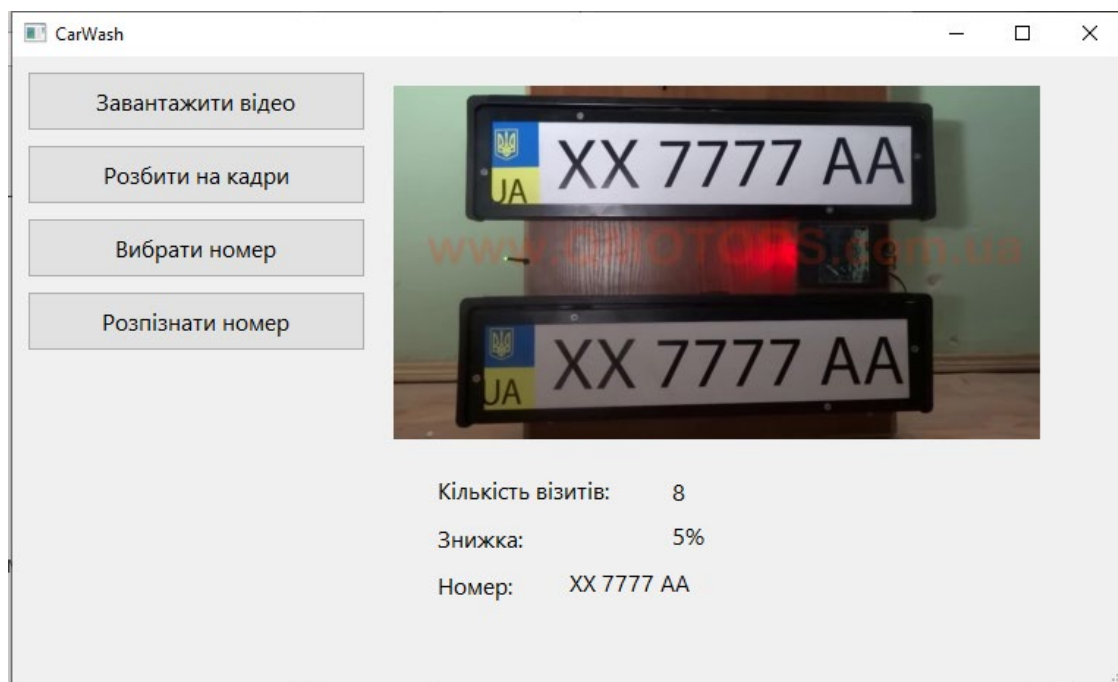


Рис. 3.10. Кількість візитів 8, знижка 5%

Нижче наведено декілька прикладів роботи програми (рис 3.11-3.13)

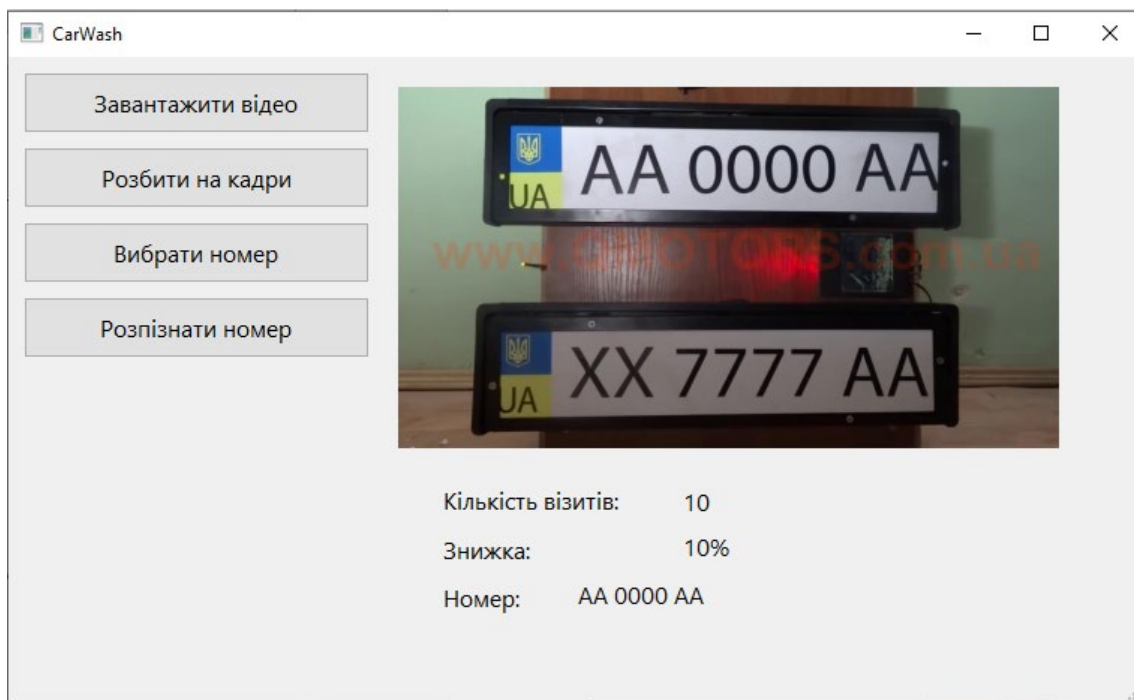


Рис. 3.11. тестування програми

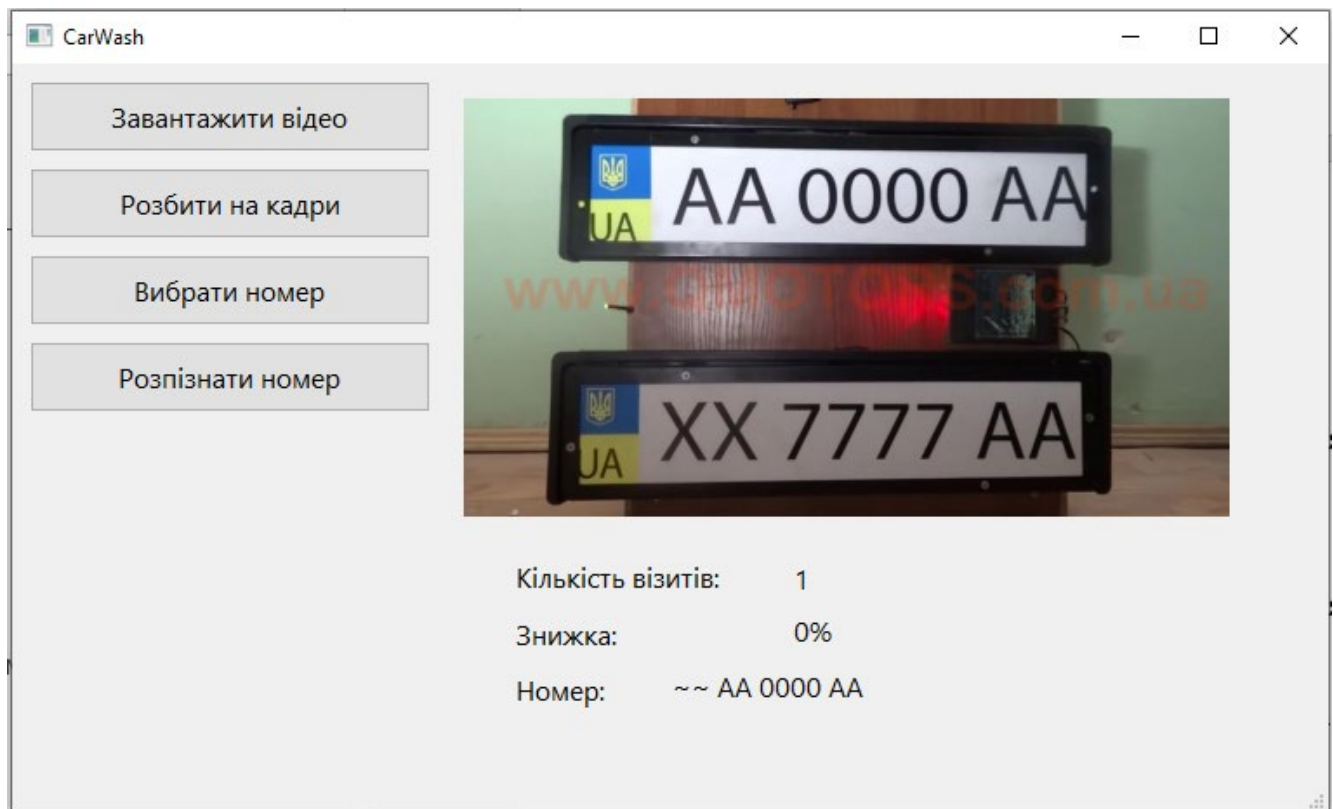


Рис. 3.12. тестування програми

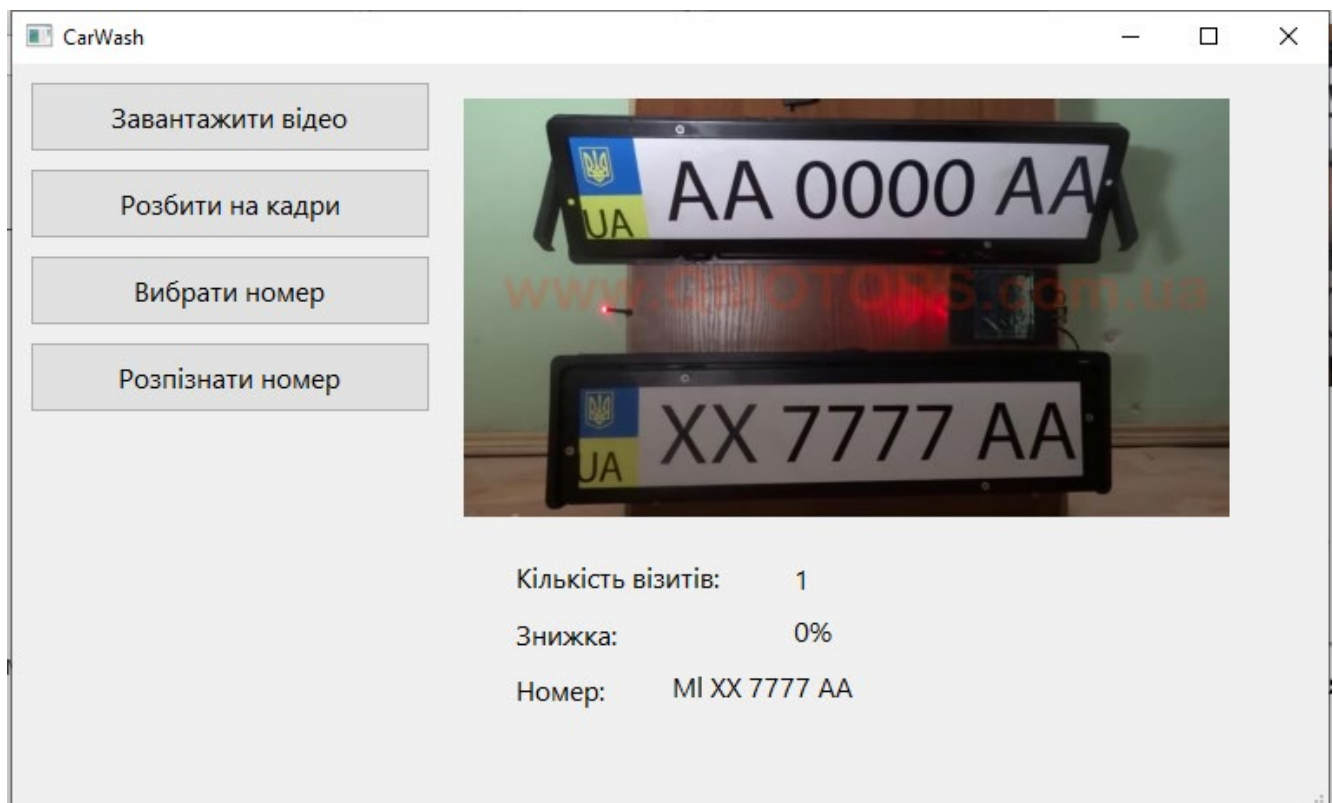


Рис. 3.13. тестування програми

3.6 Тестування роботи програми

3.6.1 Набір даних

Сегментація номерних знаків

Налаштування та тестування NN для сегментації номерних знаків проводилося на даних, згенерованих вручну із зображень реальних сцен, які містять не тільки номерні знаки автомобіля, а й сторонні предмети.

Зразок даних містить 100 зображень розміром 640x480 пікселів. Зразок поділено на 2 частини. Перша частина містить 80 зображень, що становить 80% від усієї вибірки, друга містить 20 зображень, що становить 20% від усієї вибірки. Перша частина використовується для навчання нейронної мережі, друга для тестування.

Класифікація сегментів номерних знаків

Налаштування та тестування класифікатора NN проводилось на даних, також вручну згенерованих із зображень алфавітно-цифрових зображень з алфавіту, що містить 22 допустимих символу.

Вибірка даних для роботи містить 15650 зображень розміром 28x28 пікселів. Аналогічно зразок поділено на 2 частини. Перша частина містить 12520 зображень, що становить 80% від усієї вибірки, друга містить 3130 зображень, що становить 20% від усієї вибірки. Перша частина використовується для навчання нейронної мережі, друга для тестування.

3.6.2 Результати тестування

Налаштування мережі проводилося на 20 епохах і займало 100 хвилин часу, в середньому 5 хвилин для кожної епохи.

Всього NN має 23 згорткові шари, повна архітектура мережі описана раніше.

Хоча навчання проводилося шляхом розрахунку коефіцієнта Dice, який показує міру подібності, тобто IoU, для більшої наочності якість сегментації номерних знаків порівнюється за такими показниками якості: помилка за модулем (MAE), помилка в квадраті (MSE), відсоткова точність (Accuracy), Intersection over union (IoU) [26]. Де MAE і MSE — це середня помилка сегментації пікселів за модулем і квадрат, відповідно, точність — це кількість правильно сегментованих пікселів (у відсотках), а IoU — це міра площі правильно позначених сегментів (відношення площі перетину до союзний район).

Результати експериментів із сегментацією номерних знаків представлені в таблиці 3.1.

Таблиця 3.1

Результати сегментації

Вибірка	MAE	MSE	Accuracy %	IoU
Навчаюча	0.101	0.085	97.3%	0.984
Тестова	0.116	0.099	94.8%	0.927

Класифікація сегментів номерних знаків

Отримані експериментальні дані щодо різних структур класифікаторів на основі ШНС та ансамблів нейронних мереж (АНМ) наведено в таблицях 3.2, 3.3.

У таблиці 3.2 наведено результати виконання під час виконання для різних архітектур однієї й тієї ж нейронної мережі.

Таблиця 3.2

Результати використання однієї ШНМ

	Архітектура	MAE	MSE	Accuracy %	Середній час розпізнавання, мс
Навчаюча	100-80-22	0.231	0.211	77.4	91.7

Продовження таблиці 3.2

Навчаюча	100-50-22	0.209	0.198	79.9	75.6
	100-20-22	0.194	0.179	83.7	51.4
	100-22	0.125	0.109	89.1	16.3
Тестова	100-80-22	0.251	0.263	75.5	91.7
	100-50-22	0.248	0.237	77.4	75.6
	100-20-22	0.224	0.196	82.2	51.4
	100-22	0.156	0.142	87.6	16.3

У серії експериментів найкращий результат розпізнавання досягається з використанням архітектури єдиної ШНМ «100-22», з максимальною точністю розпізнавання 87,6%.

У таблиці 3.3 наведено результати виконання робочих процесів для ансамблю нейронних мереж (ANN).

Таблиця 3.3

Результати використання ANS

Архітектура	Вибірка	MAE	MSE	Ассурасу %	Середній час розпізнавання, мс
100-22	Навчаюча	0.101	0.068	93.9	92.7
	Тестова	0.112	0.083	92.1	

ANS складається з 5 ANN. Кожен із 5 класифікаторів, що входять до ансамблю, має архітектуру ШНМ «100-22». У серії експериментів ANS середній отриманий результат розпізнавання досягає 92,1%.

3.7 Висновки за результатами експерименту

Час налаштування елемента класифікації не має значення і не розглядається як визначальний показник у запропонованому алгоритмі. Важливий показник, який слід враховувати відсоток точності розпізнавання, заданий класифікатором. Отже, з серії експериментів можна зробити наступні висновки:

- компоненти нейронних мереж демонструє значно вищу точність класифікації;
- час, необхідний для налаштування ANS, значно перевищує час навчання окремого класифікатора;
- якщо час налаштування не є критичним, то в якості класифікатора рекомендується вибрати ANS.

Як класифікуючий елемент в алгоритмі використовується ансамбль нейронних мереж однотипної архітектури.

Висновки до розділу 3

В третьому розділі було аргументовано вибір технологій для написання додатку, а також наведений опис взаємодії користувача з програмою та описаний графічний інтерфейс. Проведене тестування та наочно наведені статистичні дані продуктивності тестування.

ВИСНОВКИ

Проблему автоматичного розпізнавання номерних знаків у цифрових зображеннях вирішували багато дослідників. У роботі розглянуто власний алгоритм розв'язання задачі. Системи розпізнавання номерних знаків використовуються для контролю за в'їздом і виїздом транспортних засобів з території підприємств, автостоянок, контролю за транспортним потоком.

Існуючі комерційні системи розпізнавання номерних знаків не завжди надають користувачам інформацію про методи штучного інтелекту, які використовуються для розпізнавання буквено-цифрових символів. У представленій роботі було виконано ряд аналітичних завдань для визначення алгоритмів, що використовуються в системах. Дані системи отримані з відкритих незалежних джерел.

Зроблено аналітичний огляд існуючих методів та програмних систем для розпізнавання автомобільних номерів. Проведено їх порівняльний аналіз, визначено перелік програмних продуктів, придатних для вирішення певного кола цільових завдань щодо розпізнавання номерних знаків.

Для сегментації номерних знаків використовується навчена модель, яка показала хороші результати сегментації в інших областях. Вдалось досягти точності 94,8% при сегментації номерного знаку на наборі даних тесту.

У роботі досліджується можливість використання штучних нейронних мереж, а також їх ансамблів як елементів класифікації. Порівняльний аналіз різних архітектур та розмірів комітетів дає змогу зробити обґрунтований вибір класифікуючого елемента в запропонованому алгоритмі розпізнавання.

Розроблений нейромережевий алгоритм розпізнавання номерних знаків дозволяє досягти точності розпізнавання 92,1%.

Результати отримано на обраних зразках автомобільних номерних знаків, які є у відкритому доступі. Отримана точність розпізнавання порівнянна з існуючими аналогами.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Вакалюк Т.А., Власенко О.В., Василенко М.К. Аналіз методів розпізнавання номерних знаків. Тези VI Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення», м. Житомир, 29-30 листопада 2023 р. – Житомир: Житомирська політехніка, 2023. С. 19-21.
2. Желтов С. Ю. Обробка та аналіз зображень у проблемах машинного зору. // Київ.: Фізматкнига, 2010. - 672 с.
3. Віола П., Джонс М. Дж. Надійне виявлення обличчя в реальному часі / Міжнародний журнал комп'ютерного зору. – 2014. – Т. 57. – С. 137-154.
4. Silva S. M., Jung C. R. License plate detection and recognition in unconstrained scenarios. Computer vision – ECCV 2018. Cham, 2018. С. 593–609. URL: https://doi.org/10.1007/978-3-030-01258-8_36 (дата звернення: 18.10.2023).
5. Ільїн В. А., Позняк Є. Г. Лінійна алгебра // Київ Физматлит 2019.
6. Cristianini N., Shawe-Taylor J. Kernel methods for pattern analysis. Cambridge University Press, 2004.
7. Хайкін С. Нейронні мережі: повний курс: пер. з англ. // С. Хайкін. – 2-е вид., переп. – М. [та ін.]: Вільямс, 2016. – 1103 с.
8. Bengio Y., Courville A., Goodfellow I. Deep learning. MIT Press, 2016. 800 с.
9. Джонс М. Програмування штучного інтелекту в додатках // М. Джонс: Пер. з англ. Осіпов А.І. – Харків.: ДМК Прес, 2013. – 312 с.
10. Каллан Р. Основні поняття нейронних мереж // Пер. з англ. – Видавництво М. Вільямса, 2011.
11. Тадеусевич Р. Елементарне введення в технологію нейронних мереж з прикладами програм: Пер. з польської // Р. Тадеусевич [та ін.]. - М.: Гаряча лінія-Телеком, 2011. - 408 с.

12. Осовський С. Нейронні мережі для обробки інформації: пер. з польської. // С. Осовський. – М.: Фінанси і статистика, 2012. – 344 с.
13. Дж. Лонг, Е. Шелхамер, Т. Даррелл: Повністю згорткові мережі для семантичної сегментації 2014 р.
14. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Incorporated, 2022.
15. U-Net / T. von Eicken та ін. The fifteenth ACM symposium, м. Copper Mountain, Colorado, United States, 3–6 груд. 1995 р. New York, New York, USA, 1995. URL: <https://doi.org/10.1145/224056.224061> (дата звернення 18.10.2023).
16. Олаф Роннебергер, Філіп Фішер, Томас Брокс U-Net: згорткові мережі для сегментації біомедичних зображень
17. Глибоке навчання U-Net. [Електронний ресурс]: - Режим доступу: <http://deeplearning.net/tutorial/unet.html> (дата звернення 18.10.2023)
18. Centurion [Електронний ресурс] : уч. веб-сайт. 2012. – Режим доступу: <http://centurions.com.ua/>, вільний. Заголовок з екрану (дата звернення 18.10.2023).
19. Automatic License Plate Recognition – High Accuracy ALPR [Електронний ресурс] : оф. веб-сайт – Режим доступу: <http://platerecognizer.com/>, (дата звернення 18.10.2023).
20. Сватюк Д., Сватюк О., Белей О. «Застосування згорткових нейронних мереж для безпеки розпізнавання об'єктів у відеопотоці», Кібербезпека: освіта, наука, техніка, вип. 4, вип. 8, с. 97–112, Чер 2020.
21. Automatic number plate recognition system / K. Kiran Kumar та ін. Indian journal of science and technology. 2017. Т. 10, № 4. URL: <https://doi.org/10.17485/ijst/2017/v10i4/110674> (дата звернення 19.10.2023).
22. Introduction to algorithms / Т. Н. Cormen та ін. MIT Press, 2009. 1320 с.
23. Fleiss J. L., Levin B., Paik M. C. Statistical methods for rates and proportions. Wiley & Sons Australia, Limited, John, 2001.

24. Manning C. D., Schütze H., Raghavan P. Introduction to information retrieval. Cambridge University Press, 2012.
25. Evaluating image segmentation models [Електронний ресурс] / Blog Jeremy Jordan – Режим доступу: <https://www.jeremyjordan.me/evaluating-image-segmentation-models> (дата звернення 19.10.2023)
26. C. Tomasi, R. Manduchi Bilateral Filtering for Gray and Colour Images // Proceedings of the 1998 IEEE International Conference on Computer Vision, Bombay, India
27. Франческо Бантерле, Массіміліано Корсіні. Форум КОМП'ЮТЕРНОЇ ГРАФІКИ, том 31 2012 р., n 1, стор. 19–32.
28. OpenCV – Двосторонній фільтр [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/opencv/opencv_bilateral_filter.htm (дата звернення 19.10.2023).
29. Howse J. OpenCV computer vision with python. Packt Publishing, 2013. 122 с.
30. Peterson B. F. Understanding exposure. New York : Amphoto, 1990. 144 с.
31. Soille P. Morphological image analysis: principles and applications. 2-ге вид. Berlin : Springer, 2003. 391 с.
32. Гонсалес Р. Цифрова обробка зображень: пер. з англ. / Р. Гонсалес, Р. Вудс. - 3-є вид., переп. і доп.. - М.: Техносфера, 2012. — 1104 с.
33. A. Vishnu, S. Gopakumar "Comparative Analysis of Deep Learning Frameworks: A Review", 2019.
34. Matplotlib [Електронний ресурс] – Режим доступу: <https://matplotlib.org/>, вільний. - Загл. з екрану. (дата звернення 20.10.2023).
35. Google Colab [Електронний ресурс] – Режим доступу: <https://colab.research.google.com/notebooks/intro.ipynb/> вільний. - Загл. з екрану. (дата звернення 20.10.2023).

ДОДАТКИ

Додаток А(CarWash.py)

```
from PyQt6 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(758, 431)
        font = QtGui.QFont()
        font.setPointSize(12)
        MainWindow.setFont(font)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(10, 10, 231, 41))
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2.setGeometry(QtCore.QRect(10, 60, 231, 41))
        self.pushButton_2.setObjectName("pushButton_2")
        self.pushButton_3 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_3.setGeometry(QtCore.QRect(10, 110, 231, 41))
        self.pushButton_3.setObjectName("pushButton_3")
        self.pushButton_4 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_4.setGeometry(QtCore.QRect(10, 160, 231, 41))
        self.pushButton_4.setObjectName("pushButton_4")
        self.pushButton_5 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_5.setGeometry(QtCore.QRect(10, 210, 231, 41))
        self.pushButton_5.setObjectName("pushButton_5")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(290, 280, 141, 31))
        self.label.setToolTipDuration(-1)
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(450, 290, 47, 13))
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(290, 350, 71, 21))
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(290, 320, 71, 16))
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(self.centralwidget)
        self.label_5.setGeometry(QtCore.QRect(450, 320, 47, 13))
        self.label_5.setObjectName("label_5")
        self.pushButton_6 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_6.setGeometry(QtCore.QRect(10, 260, 231, 41))
        self.pushButton_6.setObjectName("pushButton_6")
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
```

```

MainWindow.setWindowTitle( _translate("MainWindow", "CarWash"))
self.pushButton.setText( _translate("MainWindow", "Завантажити відео"))
self.pushButton_2.setText( _translate("MainWindow", "Розбити на кадри"))
self.pushButton_3.setText( _translate("MainWindow", "Вибрати номер"))
self.pushButton_4.setText( _translate("MainWindow", "Розпізнати номер"))
self.pushButton_5.setText( _translate("MainWindow", "Занести номер до
бази даних"))
self.label.setText( _translate("MainWindow", "Кількість візитів:"))
self.label_2.setText( _translate("MainWindow", "0"))
self.label_3.setText( _translate("MainWindow", "Номер:"))
self.label_4.setText( _translate("MainWindow", "Знижка:"))
self.label_5.setText( _translate("MainWindow", "0%"))
self.pushButton_6.setText( _translate("MainWindow", "Розрахувати
знижку"))

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec())

```

Додаток Б(main.py)

```

import os
from datetime import timedelta
from pathlib import Path
import matplotlib.pyplot as plt
import cv2
import _pickle as cPickle
import numpy as np
import re
import imutils
import pytesseract
import pytesseract as tess
tess.pytesseract.tesseract_cmd = r'Tesseract-OCR\tesseract.exe'
from PyQt6 import uic
from PyQt6.QtGui import QPixmap
from PyQt6.QtWidgets import QApplication, QFileDialog
from imutils import contours
from moviepy.editor import VideoFileClip
import json
import sqlite3
import easyocr

SAVING_FRAMES_PER_SECOND = 1

Form, Window = uic.loadUiType("CarWash.ui")

app = QApplication([])
window = Window()
form = Form()
form.setupUi(window)
window.show()

video_file = ''
image_file = ''
result = ''

```

```

arr=[]

db = sqlite3.connect('server.db')
sql = db.cursor()

db = sqlite3.connect('server.db')
sql = db.cursor()
sql.execute("""CREATE TABLE IF NOT EXISTS parking (number TEXT, countvisit
INT) """)
db.commit()

def format_timedelta(td):
    result = str(td)
    try:
        result, ms = result.split(".")
    except ValueError:
        return result + ".00".replace(":", "-")

    ms = round(int(ms) / 10000)
    return f"{result}.{ms:02}".replace(":", "-")

def load_file():
    global arr

    global db
    global sql

    db = sqlite3.connect('server.db')
    sql = db.cursor()

    sql.execute("SELECT * FROM carwash")
    arr = sql.fetchall()
    arr = list(map(lambda x: list(x), arr))

    print('load')

def save_file():
    global arr

    global db
    global sql

    db = sqlite3.connect('server.db')
    sql = db.cursor()
    sql.execute("""CREATE TABLE IF NOT EXISTS carwash (number TEXT, countvisit
INT) """)
    db.commit()

    sql.execute("""DELETE FROM carwash""")
    db.commit()
    for i in range(len(arr)):
        sql.execute('INSERT INTO carwash(number, countvisit) VALUES(?, ?)',
arr[i])
        db.commit()

    for value in sql.execute("SELECT * FROM carwash"):
        print(value)

    print('save')

def load():
    global video_file
    video_file= QFileDialog.getOpenFileName()

```

```

path=Path(video_file[0])
video_file=path.name
print("load")

def split():
    video_clip = VideoFileClip(video_file)
    filename, _ = os.path.splitext(video_file)

    if not os.path.isdir(filename):
        os.mkdir(filename)

    saving_frames_per_second = min(video_clip.fps, SAVING_FRAMES_PER_SECOND)
    step = 1 / video_clip.fps if saving_frames_per_second == 0 else 1 /
saving_frames_per_second

    for current_duration in np.arange(0, video_clip.duration, step):
        frame_duration_formatted =
format_timedelta(timedelta(seconds=current_duration)).replace(":", "-")
        frame_filename = os.path.join(filename,
f"frame{frame_duration_formatted}.jpg")

        video_clip.save_frame(frame_filename, current_duration)
    print("split")

def choose():
    global image_file
    image_file = QFileDialog.getOpenFileName()
    image_file = image_file[0]
    form.label_6.setPixmap(QPixmap(image_file))
    form.label_6.setScaledContents(True)
    print("choose")

def carplate_text():
    global image_file
    img = cv2.imread(image_file)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    img_filter = cv2.bilateralFilter(gray, 11, 15, 15)
    edges = cv2.Canny(img_filter, 30, 200)

    cont = cv2.findContours(edges.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    cont = imutils.grab_contours(cont)
    cont = sorted(cont, key=cv2.contourArea, reverse=True)

    pos = None
    for c in cont:
        approx = cv2.approxPolyDP(c, 10, True)

        if len(approx) == 4:
            pos = approx
            break

    mask = np.zeros(gray.shape, np.uint8)
    new_img = cv2.drawContours(mask, [pos], 0, 255, -1)
    bitwise_img = cv2.bitwise_and(img, img, mask=mask)

    (x, y) = np.where(mask == 255)
    (x1, y1) = (np.min(x), np.min(y))
    (x2, y2) = (np.max(x), np.max(y))
    crop = gray[x1:x2, y1:y2]

```

```

text = easyocr.Reader(['en'])
text = text.readtext(crop)

res = text[0][-2]

return res

def recognize():
    load_file()
    global result
    image = cv2.imread(image_file)
    height, width, _ = image.shape
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU)[1]
    cnts = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    cnts, _ = contours.sort_contours(cnts[0])

    for c in cnts:
        area = cv2.contourArea(c)
        x, y, w, h = cv2.boundingRect(c)
        if area > 20000:
            img = image[y:y+h, x:x+w]
            gray2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            thresh2 = cv2.threshold(gray2, 0, 255, cv2.THRESH_OTSU)[1]
            res = tess.image_to_string(thresh2)
            if len(res) > 7:
                form.label_7.setText(res)
                result = res

    count = 0;
    for i in range(len(arr)):
        if arr[i][0] == result:
            count += 1

    if count == 0:
        arr.append([result, 1])
    else:
        for i in range(len(arr)):
            if arr[i][0] == result:
                arr[i][1] += 1

    for i in range(len(arr)):
        if arr[i][0] == result:
            form.label_2.setText(str(arr[i][1]))
            if (arr[i][1] < 5):
                form.label_5.setText('0%')
            elif (arr[i][1] >= 5 and arr[i][1] < 10):
                form.label_5.setText('5%')
            elif (arr[i][1] >= 10 and arr[i][1] < 20):
                form.label_5.setText('10%')
            else:
                form.label_5.setText('15%')

    save_file()
    print("recognize")

form.pushButton.clicked.connect(load);
form.pushButton_2.clicked.connect(split);
form.pushButton_3.clicked.connect(choose);
form.pushButton_4.clicked.connect(recognize);

app.exec()

```