КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Випускна кваліфікаційна робота

на здобуття ступеня магістра

за спеціальністю 121 Інженерія Програмного Забезпечення

на тему:

Проектування та розробка системи для автоматичного сканування та розпізнання номерного знаку транспортного засобу

Виконав студент 2-го курсу
Алі ШАНААХ _____
(підпис)

Науковий керівник:
к. т. н., доцент кафедри інтелектуальних програмних систем
Євген ДЕМКІВСЬКИЙ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«____ » _____ 202__ р.
протокол №____

Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ — 2021

РЕФЕРАТ

Обсяг роботи 50 сторінок, 25 ілюстрацій, 7 лістингів, 12 джерел посилань.

ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ ДЛЯ АВТОМАТИЧНОГО СКАНУВАННЯ ТА РОЗПІЗНАННЯ НОМЕРНОГО ЗНАКУ ТРАНСПОРТНОГО ЗАСОБУ

Об'єктом роботи є процес автоматичного сканування та розпізнання номерних знаків на транспортних засобах.

Предметом роботи є розробка алгоритму та програмного забезпечення для автоматичного розпізнання номерного знаку.

Метою дипломної роботи є розробка системи для розпізнавання номерного знаку без участі людини.

Методи розроблення: комп'ютерне моделювання, розробка програмного продукту на основі ітеративної моделі. Інструменти розроблення: інтегроване середовище розробки, мови програмування Python та Go, gRPC, бібліотеки OpenCV та PyTorch.

Результати роботи: проаналізовано та порівняно різні методи виявлення меж об'єктів на зображенні, алгоритми зменшення шумів та інші методи попередньої обробки вхідних зображень, розроблено алгоритм локалізації номерного знаку на основі оператора Кенні, розроблено глибинну нейронну мережу, що вміє розпізнавати деякі українські символи, імплементована система для розпізнання номерного знаку на зображенні.

Розроблений програмний продукт має відкритий код, та може використовуватися у комерційних цілях будь-якого підприємця чи компанії. Система була розгорнута, тож будь-який користувач чат-боту може її використовувати.

ЗМІСТ

ВСТУП

Зі швидким зростанням урбанізації автоматизоване розпізнавання номерних знаків відіграє важливу роль у повсякденному житті у місцях на кшталт автономних автостоянок та зон контролю безпеки. Нові технічні можливості можуть допомогти містам у розв'язання питання заторів та заощадження трудових ресурсів.

Актуальність дипломної роботи полягає у побудові системи, що може використовуватися для автоматичного розпізнання номерних знаків.

Метою дипломної роботи є розробка системи для розпізнавання номерного знаку без участі людини. Розпізнавання номерних знаків виступає ключовим фактором у відділі управління дорожнім рухом, а код на номерному знаці — це ідентифікаційний номер для учасників дорожнього руху.

Практична значущість дипломної роботи полягає в можливості застосування її результату на практиці з метою надання користувачам інструмента для розпізнання номерних знаків.

Низка чинників впливають на процес розпізнання номерного знаку транспортного засобу: освітленість, розмір, шрифт, колір, розташування на транспортному засобі. Дійсно гарний алгоритм повинен адаптуватися під усі ці чинники та демонструвати високу якість розпізнання. Саме тому, методи розпізнавання можуть відрізняється в різних місцях. Проте, існує декілька обмежень щодо розпізнання номерних знаків, а тобто: різний фон, фіксоване освітлення, швидкість руху, тощо [1].

Із розвитком комп'ютерного зору методи розпізнавання номерних знаків почали змінюватися. У 1963 р. було винайдено перший метод виявлення об'єкта з персептроном Д. Хубелем та Т. Візелем. Вони вивчали знання, пов'язане з біологією, і підтвердили що суть описує ключову інформацію. У 1970-х роках моделювання зображень, наприклад, 3D-моделювання та стереоскопічне бачення, набуло

великої популярності, штучний інтелектуальний контроль вперше застосували до комп'ютерного зору. Починаючи з 1980-х популярність комп'ютерного зору почала стрімко зростати. У 1990-х роках машинне навчання сало широко застосовуватися для розпізнавання, виявлення, сегментації, тощо. З того часу комп'ютерний зір імплементується не тільки за допомогою обробки сигналів, але й машинного навчання. У цій дипломній роботі методи обробки сигналів, такі як оператор Кенні, використовуються разом із методами машинного навчання, такими як глибинні нейронні мережі. Два методи поєднуються для комбінації ефективності традиційної обробки сигналів та точності машинного навчання.

РОЗДІЛ 1
ПОПЕРЕДНЯ ОБРОБКА ТА ЛОКАЛІЗАЦІЯ

Попередня обробка зображень забезпечує міцний фундамент для розпізнавання та адаптує зображення для подальшого розпізнання. Попередня обробка може ігнорувати такі ефекти, як світло, рух, положення зйомки та чіткість зображення. Коли світло опромінює нерівномірно, контраст зображення низький і вимагає балансування. Ця дипломна робота спрямована на послаблення обмежень за допомогою декількох методів оптимізації.

Як ми бачимо на рис. 1.1 номерні знаки в Україні мають фіксовану висоту та ширину. Розмір номерного знаку — 520 мм. на 112 мм., має 8 символів. Межа номерного знаку має прямокутну форму, складається з ліній і змінюється в певному діапазоні. Фактична ширина та висота можуть змінюватися залежно від кута зйомки.

Рис. 1.1. Український номерний знак

Номерний знак містить різні кольори, що мають різні значення гамми сірого. Кольорова межа номерного знаку призводить до мутації. Крайова сіра шкала номерного знаку з горизонтального і перпендикулярного напрямку показує безперервний низхідний, а потім висхідний тренд. Гістограма краю сірої гамми має два окремих центри. Ці характеристики використовуються для розпізнавання номерного знаку та сегментації символів.

1.1. Розмиття Гауса

На зображеннях завжди є якісь шуми. Гаусовий фільтр може прибрати шуми з номерного знака [2]. Гаусове розмиття містить згортку функції Гауса з пікселів зображення. Рівняння згортки у дискретному випадку визначається, як 1.1.

$$f * g[n] = \sum_{m=-\infty}^{\infty} f[m] \times g[n-m] \tag{1.1}$$

У двовимірному просторі, як на зображенні, функцію можна описати, як 1.2.

$$f(x,y) = M \times e^{-(\frac{(x-x_0)^2}{2\sigma x^2} + \frac{(x-y_0)^2}{2\sigma y^2})} \tag{1.2}$$

де — це амплітуда, (x_0, y_0) — центр пікселів, а σx, σy — стандартне відхилення. Дисперсія розподілу Гауса суттєво впливає на результати фільтрації.

1.2. Монохромний фільтр

Перетворення зображення в монохромне повинно пришвидшити наступні кроки, а також спростити процес. Кольорове зображення складається з трьох основних кольорів — червоного, зеленого та синього. Трансформація в монохромне просто перетворює зображення з кольорового на сіре, а вага цих трьох кольорів ідентична. Перетворення зазвичай застосовується шляхом досягнення інваріантності гамми сірого. При перетворенні зображення відбувається перерахування червоного, зеленого та синього компонентів з відповідною вагою рівняння 1.3.

$$c = (R * 299 + G * 587 + B * 114 + 500)/1000 \tag{1.3}$$

де c — остаточне значення гамми сірого для кожного пікселя. R, G, B — це червоний, зелений та сині компоненти значення пікселя. Результат перетворення зображення на монохромне зображено на рис. 1.2 та 1.3.

Рис. 1.2. Вхідне зображення

Рис. 1.3. Монохромне зображення

1.3. Інтенсифікація зображення

Після застосування чорно-білого фільтру межі зображення розмиваються, а тому їх стає важко виявити. Інтенсифікація допомагає збільшити контраст зображення для кращого локалізації номерного знаку на зображенні. Чорне-білий фільтр, згладжування зображення та лінійний фільтр — це методи, що частіше використовуються для інтенсифікації зображення.

1.4. Бінаризація зображення

Зображення називають *бінарним*, якщо кожен його піксель має лише одно з двох можливих значень. Зазвичай, використовується чорний і білий кольори, проте будь-які інші кольори можуть бути використані. Колір, що використовується для об'єкта на зображенні є кольором переднього плану, в той час, як інша частина зображення є кольором тла. У сфері розпізнавання зображень такі зображення називають *двотональними* [3].

Бінарні зображення також називають дворівневими, а це означає, що кожен піксель зберігається у вигляді одиничного біту, тобто 0 або 1. Назви чорно-білий або монохромний часто використовуються для цієї концепції, але також можуть використовуватися для будь-яких зображень, що мають лише один шаблон на кожен піксель, такі як зображення у відтінках сірого.

Процес бінаризації — це переведення кольорового зображення у бінарне. Головним параметром такого перетворення є поріг T — значення, з яким порівнюється яскравість кожного пікселя. За результатами порівнянь, піксель отримує значення 0 або 1. Підрахувати поріг можна за формулою 1.4.

Існують різні методи бінаризації, які умовно можна розділити на дві групи — глобальні та локальні. У першому випадку значення порогу залишається незмінним протягом усього процесу бінаризації. У другому зображення розбивається на область, у кожній з яких підраховується локальне значення порогу.

Головною метою процесу бінаризації є радикальне зменшення кількості інформації, з якою потрібно працювати. Вдала бінаризація сильно спрощує подальшу роботу із зображенням. З іншої сторони, невдачі в процесі бінаризації можуть привести до спотворень, такими, як розриви у лініях, втрата значущих деталей, порушення цілості об'єктів, шуми та непередбачуване спотворення символів через неоднорідність фону. Різні методи бінаризації мають свої слабкі місця: так, наприклад, метод Оцу може призвести до вставки невеликих деталей та склеюванню близьких символів, а метод Ніблека може створити хибні об'єкти у випадку неоднорідності фону з низькою рівнем контрастності. Таким чином, кожен метод повинен бути застосований у своїй галузі.

Приклад бінарного зображення можна побачити на рис. 1.4.

$$T = f_{max} - (f_{max} - f_{min})/3 \tag{1.4}$$

де T — порогове значення, f_{max} — максимальне значення, а f_{min} — мінімальне значення.

1.5. Виявлення меж

Процес виявлення меж номерного знака фокусується на трьох видах меж, об'єкт до об'єкта, фон до об'єкта і регіон до регіону (включаючи різні кольори), а також точно знаходить межі і стримує шуми. Диференціальний оператор має значний вплив на гамму сірого і змушує межі номерного знаку отримувати більше

Рис. 1.4. Бінарне зображення

значення в результатах. Оператори Собеля, Робертса та Лапласа найчастіше застосовуються для класифікації меж номерного знаку, саме тому в цій дипломній роботі проводиться порівняння між трьома операторами для кращих результатів виявлення ребер.

1.5.1. Оператор Робертса. Оператор Робертса досить точно знаходить межі, але не вміє видаляти шум. Межі з'являються, коли ступінь освітлення змінюється і має складні форми. Межі можна визначити найбільш точно за допомогою методу виявлення градієнта. Нехай $f(x, y)$ — функція розподілу гамми сірого на зображенні, $s(x, y)$ — значення градієнта зображення, а $p(x, y)$ — напрямок градієнта, тоді має місце рівняння 1.5 та 1.6.

$$s(x,y) = \sqrt{[f(x+n,y) - f(x,y)]^2 + [f(x,y+n) - f(x,y)]^2} \tag{1.5}$$

$$p(x,y) = tan\frac{f(x+n,y) - f(x,y))}{x(x,y+n) - f(x,y))} \tag{1.6}$$

Рівняння 1.5 можна переписати у 1.7, як значення та напрямок градієнта у точці (x, y).

$$g(x,y) = \sqrt{[\sqrt{f(x,y)} - \sqrt{f(x+1,y+1)}]^2 + [\sqrt{f(x+1,y)} - \sqrt{f(x,y+1)}]^2} \tag{1.7}$$

де $g(x,y)$ — оператор Робертса.

Розрахунок квадрата $f(x,y)$ робить процес схожим на систему комп'ютерного зору. Між іншим, цей алгоритм визначає межі за допомогою часткової диференціації шляхом обчислення різниці діагоналей сусідніх пікселів. Вирази можна записати у градієнтних формах, як у рівняннях 1.8 та 1.9. Приклад можна побачити на рис. 1.5.

$$\Delta_x f(x,y) = f(x,y) - f(x-1,y-1) \tag{1.8}$$

$$\Delta_y f(x-1,y) = f(x,y) - f(x,y-1) \tag{1.9}$$

Рис. 1.5. Оператор Робертса

1.5.2. Оператор Собеля. Оператор Собеля спрямований і визначає найсильніші межі з горизонтального вертикального напрямків [4]. Оператор Собеля є кращим як у виявленні країв, так і в контролі шумів. Оператором є згортка $g_1(x,y)$ та $g2(x,y)$ та вихідного $f(x,y)$. Оператор може бути виражений, як рівняння 1.10.

$$s(x,y) = max[\sum_{m=1}^{M}\sum_{n=1}^{N} f(m,n)g_1(i-m,j-n), \sum_{m=1}^{M}\sum_{n=1}^{N} f(m,n)g_2(i-m,j-n)] \quad (1.10)$$

Варто зазначити, що оператор Собеля спочатку вимірює середньозважене, а потім вимірює різницю. Градієнт виглядає як рівняння 1.11 та 1.12.

Рис. 1.6. Оператор Собеля

$$\Delta_x f(x,y) = [f(x1,y+1)+2f(x,y+1)+f(x+1,y+1)][f(x1,y1)+2f(x,y1)+f(x+1,y1)] \quad (1.11)$$

$$\Delta_y f(x,y) = [f(x1,y1)+2f(x1,y)+f(x1,y+1)][f(x1,y1)+2f(x+1,y)+f(x+1,y+1)] \quad (1.12)$$

Оператор Собеля визначає межі з горизонтального та перпендикулярного напрямку зображення, згодом вимірює максимальну згортку серед отриманих. В результаті виходить зображення з очевидними межами. Приклад можна побачити на рис. 1.6.

1.5.3. Оператор Лапласа. Оператор Лапласа вимірює межі з усіх боків, а також чутливий до тонких ліній та незалежних точок [5]. Однак має негативний

вплив на шуми та створює подвійні краї пікселів. Вимірює похідну 2D-функції, це можна побачити в формулі 1.13.

$$\nabla^2 f(x,y) = \frac{d^2 f(x,y)}{dx^2} + \frac{d^2 f(x,y)}{dy^2} \tag{1.13}$$

Рис. 1.7. Оператор Лапласа

Якщо замінити похідну диференційним рівнянням, то матиме вигляд, як рівняння 1.14.

$$\Delta^2 f(x,y) = f(x+1,y) + f(x1,y) + f(x,y+1) + f(x,y1)4f(x,y) \tag{1.14}$$

Особливість оператора Лапласа полягає в тому, що центральний коефіцієнт додатній, а інші — від'ємні, а сума коефіцієнтів дорівнює нулю. Оператор Лапласа чутливий до ліній, точок та шумів, але не до напрямків, тому зазвичай його не застосовують для визначення меж. Приклад можна побачити на рис. 1.7.

1.5.4. Оператор Кенні. Алгоритм Кенні — це оператор виявлення меж, який використовує багатоступеневий алгоритм для виявлення широкого діапазону країв на зображеннях. Він був розроблений Джоном Ф. Кенні в 1986 році. Варто зазначити, що Кенні також створив обчислювальну теорію виявлення меж, що пояснює як та чому працює цей метод [6].

Цей алгоритм широко застосовується в сучасних системах компютерного зору, тому що вміє витягувати корисну інформацію з різних об'єктів та сильно зменшує кількість даних для подальшої обробки. Автор виявив, що вимоги до задачі виявлення меж об'єктів у різноманітних системах зору досить схожі. Таким чином, алгоритм виявлення меж, що відповідає певному списку вимог може підійти до низки різних задач у сучасних системах компютерного зору. Приклад можна побачити на рис. 1.8.

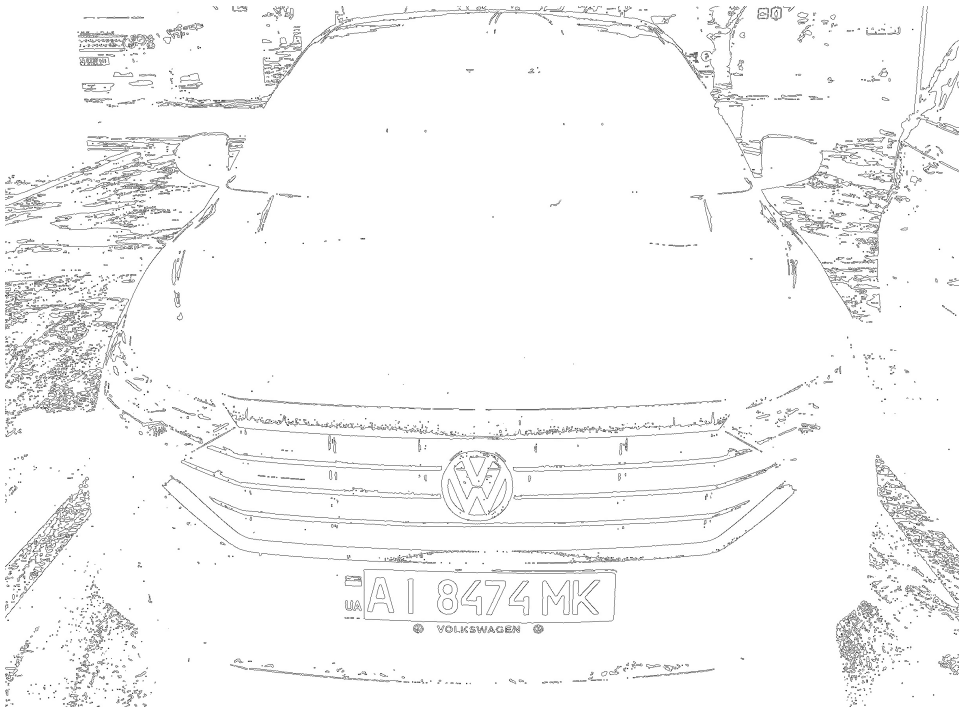

Рис. 1.8. Оператор Кенні

Загальні критерії виявлення країв включають:

1. Низький рівень помилок при виявленні меж, а тобто алгоритм повинен фіксувати якомога більше країв, що містяться на зображенні.
2. Точка ребра, виявлена оператором, повинна точно локалізуватися в центрі ребра.
3. Ребро на зображенні слід позначати лише один раз, і там, де це можливо, шуми не повинні створювати помилкові ребра.

Аби задовольнити усі вище перелічені вимоги, Кенні використовував варіаційне числення. Варіаційне числення — розділ аналізу, в якому вивчаються варіації функціоналів. Найбільш типова задача — знайти функцію, на якій заданий

функціонал досягає екстремального значення. Таким чином, варіаційне числення підбирає функцію, що знаходить краще рішення для заданого функціонала. Оптимальна функція в алгоритмі Канні описується сумою чотирьох експоненціальних доданків, але вона може бути апроксимована першою похідною за Гауссом.

Отже, серед розроблених до цього часу методів виявлення меж, алгоритм Кенні є одним із найбільш суворо визначених методів, що забезпечує якісне та надійне виявлення. Завдяки своїй оптимальності відповідати трьом критеріям виявлення країв і простоті процесу реалізації, він став одним з найпопулярніших алгоритмів виявлення меж.

Алгоритм Кенні можна розбити на 5 кроків:

1. Розмиття зображення.
2. Пошук градієнтів.
3. Придушення немаксімумов.
4. Подвійна порогова фільтрація.
5. Трасування області неоднозначності.

Розмиття зображення. Оскільки на всі результати виявлення країв легко впливають шуми на зображенні, важливо відфільтрувати їх, щоб запобігти помилковому виявленню, спричиненому ними. Щоб згладити зображення, ядро фільтра Гауса змішується із зображенням. Цей крок трохи згладить зображення, щоб зменшити вплив явних шумів на виявлення меж. Рівняння для ядра фільтра Гауса розміром $(2k + 1) * (2k + 1)$ визначається як рівняння 1.15. Для видалення шумів алгоритм Кенні використовує Розмивання Гауса з $\sigma = 1.4$. Приклад фільтра Гауса з розміром $5 * 5$, який використовується для створення зображення $\sigma = 1$ можна побачити на рівнянні 1.16. Необхідно розуміти, що вибір розміру ядра фільтра Гауса вплине на продуктивність алгоритму. Чим більше розмір, тим менша чутливість алгоритму до шуму. Варто зазначити, що похибка локалізації для виявлення меж трохи збільшується із збільшенням розміру ядра фільтра Гауса. 5*5 — це гарний розмір для більшості випадків, але він також буде відрізнятися залежно від конкретних ситуацій.

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \leqslant i, j \leqslant (2k+1) \quad (1.15)$$

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A} \quad (1.16)$$

Пошук градієнтів. Границі відмічають там, де градієнт набуває найбільшого значення. Вони можуть мати різні напрямки, тому алгоритм Кенні використовує чотири фільтри для визначення горизонтальних, вертикальних і діагональних ребер в розмитому зображенні.

Ребро на зображенні може бути направлене в різних напрямках, тому алгоритм Кенні використовує чотири фільтри для виявлення горизонтальних, вертикальних та діагональних країв на розмитому зображенні. Оператори виявлення меж повертають значення для першої похідної в горизонтальному напрямку (G_x) і вертикальному напрямку (G_y).

З цього можна визначити градієнт та напрямок ребра:

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2} \quad (1.17)$$

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x), \quad (1.18)$$

Кут напрямку ребра закруглений до одного з чотирьох кутів, що представляють вертикаль, горизонталь та дві діагоналі (0°, 45°, 90° та 135°).

Трасування області неоднозначності. На цьому етапі вирішується, які всі ребра насправді є ребрами, а які ні. Для цього нам потрібні два порогові значення, $minVal$ і $maxVal$. Будь-які ребра з градієнтом інтенсивності, що переви-

щують $maxVal$, обов'язково будуть ребрами, а ті, що нижче $minVal$, обов'язково будуть не-ребрами, а тому відкидаються. Усі інші, що лежить між цими двома порогами, класифікуються як ребра чи не-ребра на основі їх зв'язку. Якщо вони підключені до пікселів із вірогідним ребром, то вони вважаються частиною межі. В іншому випадку їх також відкидають. Дивіться зображення нижче:

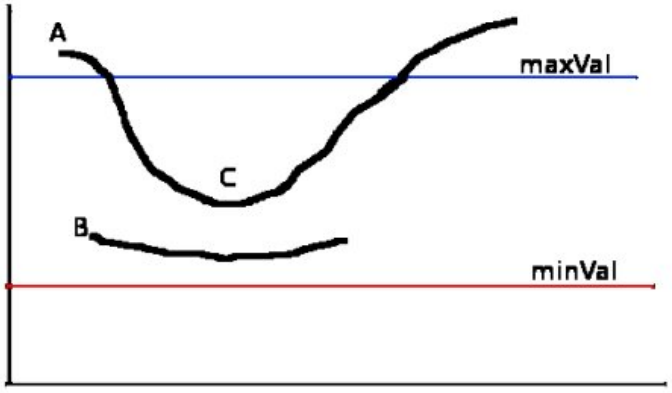

Рис. 1.9. Трасування області неоднозначності

Як ми бачимо на рис. 1.9, ребро A знаходиться вище $maxVal$, тому його розглядають, як достовірне ребро. Хоча ребро C нижче $maxVal$, воно з'єднане з ребром A, а тому воно також вважається достовірний, ми отримуємо повну межу. Але ребро B, хоча воно вище $minVal$ і знаходиться в тій самій області, що і ребро C, воно не з'єднане з жодним достовірним ребром, а тому його відкидають.

Порівняння. Результати чотирьох операторів ми можемо побачити на рис. 1.5, 1.6, 1.7 та 1.8.

Отже, після проведеного аналізу, можна підсумувати, що оператори Кенні та Собеля має найкращий результат у нашому випадку. Крім того, ми дізналися, що оператор Робертса теж досить не погано визначає межі, проте він не чутливий до шумів. Таким чином, оператор Собеля є середнім фільтром і може контролювати шуми. Оператор Лапласа добре виявляє межі, але не чутливий до шумів. Отже, надалі у цій дипломній роботі для визначення меж номерного знаку буде використовуватися оператор Собеля.

1.6. Локалізація номерного знака

Традиційні методи вилучення параметрів текстури зазвичай базується на монохромних зображеннях. Зображення попередньо обробляється і перетворюється з кольорового на монохромне. Алгоритм спочатку сканує зображення і виявляє кожен рядок, що містить щось подібне до номерного знаку та записує початкову точку та довжину. Якщо декілька рядків мають більше одного запису, а номери рядків перевищують порогове значення, підтверджується, що у горизонтальному напрямку був знайдений можливий регіон номерного знаку. Записуються можлива початкова точка та висота регіону. Далі сканується можливий регіон номерного знаку та підтверджується початкова точка та висота для справжнього регіону номерного знаку.

1.7. Локалізація методом математичної морфології

Математична морфологія — це нелінійний метод обробки зображень за допомогою двовимірних операцій згортки. Цей метод може бути використаний для виявлення контурів, сегментації зображень, ліквідації шумів, виділення ознак та інших задачах обробки зображень, активно використовується в області розпізнавання зображень.

Вперше теорія була запропонована студентом-науковцем Дж. Серра і його науковим керівником Г. Мазоном у 1964 році. Заснована на важкій роботі дослідників в інституті Фонтенбло і дослідників з інших країн, математична морфологія поступово розроблялася і стала самодостатньою наукою. У 1982 році, після публікації про «аналіз зображень і математичної морфології» Дж. Серра, математична морфологія стала всесвітньо відомою. Крім того, застосування математичної морфології призвело до значних поліпшень в галузі сільського господарства, а також варто зазначити, що теорія стрімко розвивається надалі.

Математична морфологія зображує та аналізує зображення на основі кутів множини, робить геометричну трансформацію для цільових об'єктів за допомогою *структурного елементу* для того, щоб відкинути необхідну інформацію. Те-

орія математичної морфології набула широкого розповсюдження для виявлення меж на зображеннях. У порівнянні з традиційними алгоритмами виділення меж зображення, морфологія досягає значно кращих результатів. Морфологічний метод виявлення меж на зображенні зберігає детальні характеристики зображення, а також розв'язує проблему точності виявлення меж.

Чжоу був першим, хто зробив обробку кольорового зображення за допомогою морфології в відтінках сірого, згодом використовував метод математичної морфології для виявлення меж, де структурним елементом був квадрат розмірів 3x3. Цей метод зміг розв'язати проблему ліквідації шумів та виявлення меж шкідників в зерні. У 2006 році було запропоновано розширений метод виявлення меж об'єктів за допомогою математичної морфології, аби покращити якість розпізнавання меж об'єктів.

Математична основа і мова — теорія множин, а операції *звуження*, *розширення*, *відкриття* та *закриття* — операції, що лежать в основі математичної морфології.

Звуження множини A за структурним елементом B визначається як вираз 1.19. Іншими словами, звуження множини A за структурним елементом B, це таке геометричне місце точок для всіх таких позицій точок центру z, при зсуві яких множина B цілком міститься в A.

$$A \ominus B = \{z \in E | B_z \subseteq A\} \tag{1.19}$$

Розширення множини A по множині B визначається як вираз 1.20. При цьому розширення множини A за структурним елементом B це множина всіх таких переміщень z, при яких множини A і B збігаються принаймні в одному елементі. Розширення є комутативною функцією, а тобто актуальний вираз 1.21.

$$A \oplus B = \{z \in E | (B^S)_z \cap A \neq \varnothing\} \tag{1.20}$$

$$A \oplus B = B \oplus A = \bigcup_{a \in A} B_a \tag{1.21}$$

Розкриття множини A за структурним елементом B позначається як $A \circ B$ і визначається виразом 1.22. Таким чином, розкриття множини A по структурному елементу B знаходиться як звуження A по B, результат котрої піддається розширенню за тим самим структурним елементом B. Зазвичай операція розкриття згладжує межі об'єкту, усуває вузькі перешийки й ліквідує виступи невеликої ширини.

$$A \circ B = (A \ominus B) \oplus B \tag{1.22}$$

Закриття множини A за структурним елементом B позначається як $A \bullet B$ і отримується шляхом виконання операції розширення множини A за структурним елементом B, за котрою слідує операція звуження множини за структурним елементом B. Закриття визначається виразом 1.23. В результаті операції закриття відбувається згладження меж, але, на відміну від розкриття, в загальному випадку заповнюються невеликі розриви й довгі заглибини малої ширини, а також ліквідуються невеликі отвори та заповнюються проміжки контуру. Результат операцій розкриття та закриття можна побачити на рис. 1.10 та 1.11.

$$A \bullet B = (A \oplus B) \ominus B \tag{1.23}$$

Морфологічні операції можна використовувати для виділення меж бінарного об'єкта. Це операція дуже важлива, тому що межа є повним, і разом з тим досить компактним описом об'єкта. Таким чином, $G_i(A)$ являє собою внутрішній градієнт, а $G_e(A)$ — зовнішній градієнт. $G(A)$ — це градієнт, а тобто різниця операцій розширення та звуження, що можна побачити у виразі 1.26. Приклад градієнту зображення можна побачити на рис. 1.12.

$$G_i(A) = (A \oplus B) - A \tag{1.24}$$

Рис. 1.10. Морфологічне розкриття

Рис. 1.11. Морфологічне закриття

Рис. 1.12. Морфологічний градієнт

$$G_e(A) = A - (A \ominus B) \tag{1.25}$$

$$G(A) = (A \oplus B) - (A \ominus B) \tag{1.26}$$

РОЗДІЛ 2
РОЗПІЗНАННЯ НОМЕРНОГО ЗНАКА

2.1. Традиційний метод

У цій роботі застосовано ефективний метод сегментації [7]. Він відрізняється від розпізнавання моделі нейронної мережі, але ефективніший, ніж традиційні методи. Метод має наступні кроки.

2.1.1. Сегментація області номерного знаку. Пошук вертикальних меж номерного знака виконується за допомогою оператор Собеля, як вже згадувалося раніше. Для номерного знака на рис. 2.1, ми можемо отримати вертикальну проекцію зображення, як показано на рис. 2.2. Після знаходження області, що обмежена верхньою та нижньою межами номерного знака, область над верхньою межею та під нижньою межею видаляється. Область між верхньою та нижньою межами номерного знака — це область сегментації символів [8].

Рис. 2.1. Номерний знак

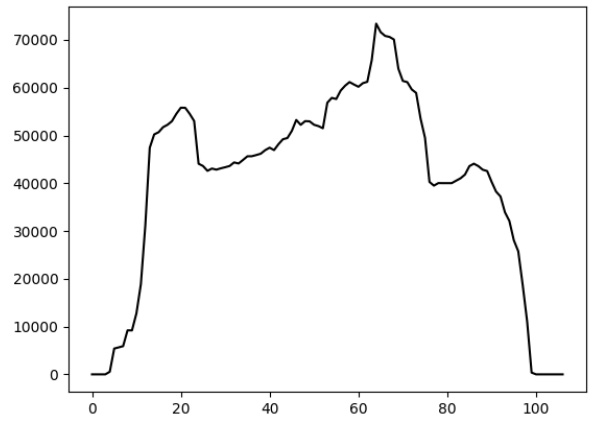

Рис. 2.2. Вертикальна проекція зображення

Наступним кроком будується проєкційна гістограма, щоб знайти розриви між символами на номерному знаці. Значення ряду гістограми — сума білих пікселів

вздовж лінії у вертикальному напрямку. Як ми можемо побачити на рис. 2.4, нульові проміжки на гістограмі збігаються з розривами між символами на рис. 2.3. Таким чином, можна досить легко сегментувати область номерного знаку.

Рис. 2.3. Номерний знак

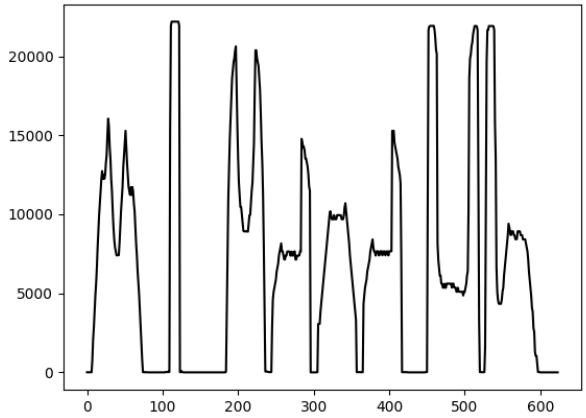

Рис. 2.4. Горизонтальна проекція

2.1.2. Процес сегментації та нормалізації символів. Сегментація символів є однією з найважливіших етапів у процесі оптичного розпізнавання символів, зокрема, при оптичному розпізнанні зображень документів. Сегментацією називається декомпозиція зображень, що містять послідовність символів, на фрагментах, що містять окремі символи [9].

Важливість сегментації обумовлених тим, що в основі більшості сучасних систем оптичного розподілу тексту лежать класифікатори окремих символів, а не слів чи фрагментів тексту. У таких системах помилки неправильного проставлення розривів між символами, як правило, є причиною левової частки помилок кінцевого розподілу.

Пошук меж символів умовно виходить із-за артефактів друку та зацифрування документа, що призводить до розсипання та склейки символів. У разі використання стаціонарних або мобільних малорозмірних відеокамер спектр артефактів істотно доповнюється: можливе дефокусування та змазування, деформація та пошкодження документа. При фільмуванні камер у природних сценах на зображеннях часто з'являються паразитні перепади яскравості, а також колірні

спотворення і цифровий шум в результаті низької освітленості.

У цій роботі кожен піксель зображення досліджується, аби не було спотворень і шумів, усі непотрібні частини зображення вирізаються. Поріг встановлюється за розміром зображення і визначає вісь X зображення. Якщо ширина більша за порогове значення, зображення вирізається. Нормалізований розмір зображення символу становить 20 * 10 і відповідає шаблону.

В автоматизованому процесі розпізнання номерних знаків, етап сегментації символів виконує роль об'єднання попереднього та наступного етапів, оскільки сегментація базується на попередньому етапі локалізації номерного знаку. Варто зазначити, що у цій роботі реалізовано спеціальний метод пошуку та сегментації символів. Якщо ширина символу перевищує обране порогове значення, то існує більш як двоє символів і сегмент вимагає повторної сегментації.

Зазвичай, у подібних системах, сегментований символ вимагає подальшої обробки перед розпізнаванням символу. Однак для розпізнавання номерного знаку не потрібен подальший процес обробки символів.

2.1.3. Розпізнавання символів. У процесі розпізнання номерних знаків існує два відомі методи розпізнавання символів: OCR на основі шаблону та метод, що базується на згортковій нейронній мережі. OCR на основі шаблону спочатку розпізнає символ і змінює його до розміру шаблону. Потім сегмент порівнюється із шаблонами та обираються найкращі результати. Цей метод є досить ефективний, коли зображення має дефекти, а також його можна досить легко реалізувати. Саме тому цей метод використовується для порівняння в цій роботі.

Спершу, алгоритм витягує деяку характеристику з області зображення $f(i, j)$ та порівняє її з шаблоном відповідно до області T(i, j) для стандартизованої перехресної кореляції. Кореляція найвищого значення має найбільшу подібність. Чим визе значення кореляції — тим більша подібність. Варто зазначити, що подібність також можна виміряти, обчисливши відстань характеристики між об'єктом та моделлю. Зазвичай зображення, що застосовуються для порівняння, мають різні умови формування та велику кількість шумів, або вони попередньо оброблені чи

стандартизовані, щоб змінився рівень сірого та розташування пікселів. При розробці реальної моделі враховуються внутрішні характеристики форми регіону, шуми та зміщення зображення. Шаблон побудований із власними характеристиками зображення, щоб уникнути інших впливів. Цей метод застосовує операцію різниці для пошуку найбільш схожого символу та виведення найбільшої подібності.

Сьогодні в Україні існує 12 типів і 25 підтипів номерних знаків. У цій роботі ми фокусуємось на розпізнанні більшої частини номерів, проте не на всіх. На восьми типах номерів використовуються українські літери, що мають відповідники в латиниці (загалом 12: A, B, C, E, H, I, K, M, O, P, T, X).

Отже, на більшій кількості українських номерів 8 символів, а також вони мають фіксований перелік символів, що може бути використано для них, приблизно дванадцять букв та десять цифр. Саме тому у цій роботі було побудовано шаблон із двадцяти двох символів. Зображення порівнюється із шаблоном і відтворюється разом із шаблоном. Чим більше нулів буде отримано, тим більше символів збігаються.

2.2. Згорткова нейронна мережа

Згорткові нейронні мережі (CNN) — це клас штучних нейронних мереж прямого поширення, що активно застосовувався для аналізу зображень. Згорткова нейронна мережа - це алгоритм глибокого навчання, який може приймати вхідне зображення, призначати важливість різним аспектам/об'єктам на зображенні та може відрізняти один від іншого. Попередня обробка, необхідна в CNN, набагато нижча порівняно з іншими алгоритмами класифікації. В той час, як в примітивних методах фільтри розробляються вручну, проте з достатнім рівнем тренування, CNN вміє вивчати ці фільтри/характеристики.

Архітектура CNN аналогічна структурі зв'язку нейронів в мозку людини і натхненна організацією зорової кори. Окремі нейрони реагують на подразники лише в обмеженій області зорового поля, відомому як рецептивне поле. Колекція

таких полів перекривається, щоб охопити всю зорову зону.

CNN складається з вхідних, вихідних та декількох прихованих шарів, як показано на рис. 2.5. Приховані шари зазвичай складаються з згорткових, агрегувальних, повноз'єднаних шарів та шарів нормалізації [10].

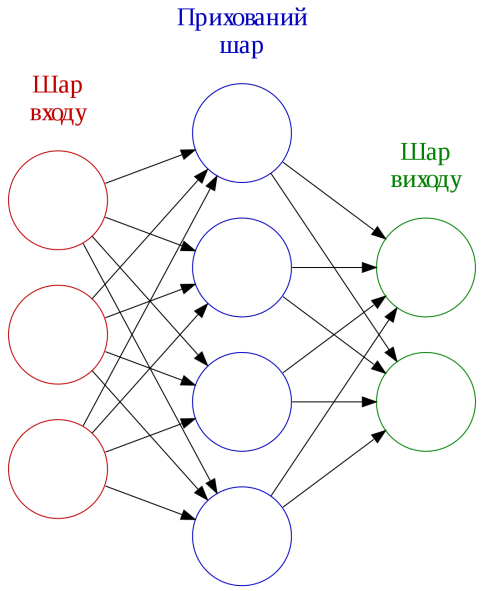

Рис. 2.5. Будова нейронної мережі

Згорткові шари. Згорткові шари застосовують до входу операцію згортки, передаючи результат до наступного шару. Згортка імітує реакцію окремого нейрону на зоровий стимул. Кожен згортковий нейрон обробляє дані лише для свого рецептивного поля. Хоча повноз'єднані нейронні мережі прямого поширення й можливо застосовувати як для навчання ознак, так і для класифікування даних, застосування цієї архітектури до зображень є непрактичним. Було би необхідним дуже велике число нейронів, навіть у поверхневій (протилежній до глибинної) архітектурі, через дуже великі розміри входу, пов'язані з зображеннями, де кожен піксель є відповідною змінною. Наприклад, повноз'єднаний шар для зображення розміром 100×100 має 10 000 ваг. Операція згортки дає змогу розв'язати цю проблему, оскільки вона зменшує кількість вільних параметрів, дозволяючи мережі бути глибшою за меншої кількості параметрів. Наприклад, незалежно від розміру зображення, області замощування розміру $5x5$, кожна з одними й тими

ж спільними вагами, вимагають лише 25 вільних параметрів.

Агрегувальні шари. Згорткові мережі можуть включати шари локального або глобального агрегування, які об'єднують виходи кластерів нейронів одного шару до одного нейрону наступного шару. Наприклад, максимізаційне агрегування використовує максимальне значення з кожного з кластерів нейронів попереднього шару. Іншим прикладом є усереднювальне агрегування, що використовує усереднене значення з кожного з кластерів нейронів попереднього шару.

Повноз'єднані шари. Повноз'єднані шари з'єднують кожен нейрон одного шару з кожним нейроном наступного шару. Це, в принципі, є тим же, що й традиційна нейронна мережа багатошарового перцептрону.

Ваги. CNN використовує спільні ваги в згорткових шарах, що означає, що для кожного рецептивного поля шару використовується один і той самий фільтр, а це зменшує обсяг необхідної пам'яті та покращує продуктивність.

2.2.1. Сегментація символів. Після вдалого знаходження прямокутного номерного знаку, наступний кроком треба вирізати прямокутні області символів, так само, як це робится в традиційниих методах сегмантації. Далі, потрібно нормалізувати зображення задля стандартизації формату символів.

2.2.2. Витяг ознак для тестового зображення. Зображення повинно бути розміщено в моделі штучної нейронної мережі для навчання. Перед цим, ознаки повинні бути витягнуті для навчання. Існує три типи ознак. Гістограма орієнтованого градієнта є своєрідною ознакою виявлення об'єкта шляхом обчислення розподілу регіонального градієнту зображення. Локальний двійковий шаблон є своєрідною частково-бінаризованою моделлю, чи оператором регіональних текстурних ознак зображення. Хаар має ознаки меж, лінійності, центральності та діагоналей. Витягнуті ознаки використовується для навчання.

2.2.3. Навчання. Найбільш простим та популярним способом навчання є метод навчання з вчителем - метод зворотного поширення помилки та його мо-

дифікації. Існує також ряд технік навчання згортковох мережі без вчителя. Наприклад, фільтри операції згортки можна навчяти окремо і автономно, подаючи на них вирізані випадковим чином шматочки вихідних зображень навчальної вибірки і застосовуючи для них будь-який відомий алгоритм навчання без вчителя (наприклад, автоассоціатор або навіть метод k-середніх) - така техніка відома під назвою patch-based тренування. Відповідно, наступний шар згортки мережі буде навчатися на шматочках від уже навченого першого шару мережі. Також можна скомбінувати згорткову нейромережу з іншими технологіями глибинного навчання. Наприклад, зробити згортковий авто-асоціатор, згорткову версію каскадних обмежених машин Больцмана, що навчаються за рахунок ймовірнісного математичного апарату, згорткову версію розрідженого кодування. Для поліпшення роботи мережі, підвищення її стійкості і запобігання перенавчання застосовується також виключення - метод тренування підмережі з викиданням випадкових одиничних нейронів.

Витягнуті функції передаються в бібліотеку OpenCV для навчання. У цій роботі ми застосували ядро RBF для навчання. Основна властивість RBF класифікується за коефіцієнтом штрафу та гаммою. Нам потрібно знайти найкращу комбінацію цих факторів, і найпростіший спосіб — це метод вичерпування.

РОЗДІЛ 3
ОСОБЛИВОСТІ РОЗРОБКИ СИСТЕМИ ДЛЯ АВТОМАТИЧНОГО СКАНУВАННЯ ТА РОЗПІЗНАННЯ НОМЕРНОГО ЗНАКУ ТРАНСПОРТНОГО ЗАСОБУ

3.1. Постановка задачі

Формалізація вимог є важливим аспектом у розробці програмного забезпечення, оскільки кінцевою метою цієї роботи є розробка системи для розпізнавання номерного знаку без участі людини.

Варто зазначити, що процес аналізу вимог є критичним для успішної розробки проєкт. Вимоги мають бути тестовними, вимірними, задокументованими, а також описаними з рівнем деталізації достатнім для проєктування системи. У загальному випадку, вимоги можуть бути архітектурними, структурними, поведінковими, функціональними, та не функціональними. Аналіз вимог полягає в визначенні потреб та умов, які висуваються щодо нового, чи зміненого продукту, враховуючи можливо конфліктні вимоги різних замовників, таких як користувачі чи бенефіціари. Перш за все, Аналіз вимог полягає в визначенні потреб та умов, які висуваються щодо нового, чи зміненого продукту, враховуючи можливо конфліктні вимоги різних замовників, таких як користувачі чи бенефіціари.

Таким чином, протягом етапу аналізу вимог, спершу було формалізовано функціональних вимог. Функціональні вимоги — це вимоги до програмного забезпечення, які описують внутрішню роботу системи та її поведінку. Отже, система повинна використовувати сучасні бібліотеки та алгоритми для роботи з зображенням. Сервіс, що займається розпізнанням не повинен використовувати бази даних чи інші засоби збереження даних. Єдине, що може бути збережено це модель нейронної мережі. А тому повинен бути розроблений інтерфейс для комунікації цього сервісу з іншими частинами системи. Пропонується використовувати низь-

корівневі протоколи для пришвидшення взаємодії, наприклад gRPC. Серед нефункціональних вимог: повинна вміти розпізнавати номерні знаки на зображенні з ймовірністю 95% відсотків, а також час опрацювання зображення повинен бути меншим за хвилину. Будь-який користувач повинен мати можливість розпізнання номерного знаку на зображенні. Таким чином, система повинна бути розгорнута в мережі Інтернет та доступна для використання.

Передбачається, що серверна частина повинна компілюватися на операційній системі Linux. Це обмеження дасть можливість використовувати інструмент для управління ізольованими Linux-контейнерами. Зокрема, «Docker» дозволяє не переймаючись вмістом контейнера запускати довільні процеси в режимі ізоляції, а потім переносити й клонувати сформовані для даних процесів контейнери на інші сервери, беручи на себе всю роботу зі створення, обслуговування та підтримки контейнерів.

Вся інформація між клієнтом та сервером повинна шифруватися за допомогою сучасних криптографічних протоколів. Прикладні програмні інтерфейси повинні працювати за gRPC-протоколом. Це система віддаленого виклику процесів з відкритим вихідним кодом, розроблена в Google. У якості транспортного протоколу використовується HTTP/2, у якості мови опису інтерфейсу — буфери протоколів. Ця технологія дає можливість використовувати такі функції, як аутентифікація, двонапрямна потокова передача та управління потоком, а також скасовування та тайм-аути. Генерує міжплатформові зв'язки для клієнта та сервера на багатьох сучасних мовах програмування. Складне використання протоколу HTTP/2 робить неможливу реалізацію клієнта gRPC у браузері — замість цього необхідно використовувати проксі.

3.2. Побудова архітектури

Взявши до уваги функціональні та нефункціональні вимоги до програмного забезпечення, необхідно спроєктувати архітектуру системи. Як зображено на рис. 3.1, архітектура системи буде складатися з декількох сервісів, а кожен сер-

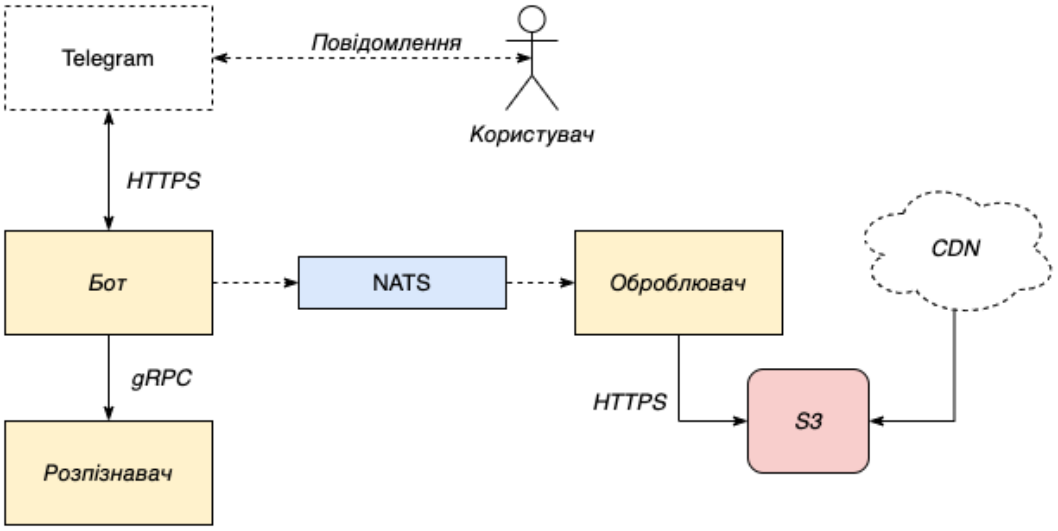

Рис. 3.1. Принципова схема роботи системи

віс відповідальний за певну частину функціоналу. Подібне розділення системи на сервіси дасть змогу сервісам працювати ізольовано один від одного, а це спрощує процес розробки та подальшої підтримки. Не менш важливим фактором є стійкість системи до атак та непередбачуваних помилок інфраструктури, ці питання також стає легше розв'язати маючи декілька копій кожного із сервісів.

Оскільки в першому та другому розділах цієї роботи детально досліджується теми сканування та розпізнання номерних знаків, залишається лише підсумувати наше дослідження. Розроблений алгоритм можна звести до трьох кроків:

1. Попередня обробка зображення;
2. Локалізація номерного знаку на зображенні;
3. Розпізнання символів на зображенні;

В рамках перших двох розділів було проведено детальний аналіз методів, їх сильних та слабих сторін. На кожному кроці є низка методів та алгоритмів, що можуть розв'язати задачу, яка стоїть. Необхідно розуміти, що будь-який метод має свої переваги та недоліки. Не менш важливо, що левова частина програмного забезпечення у цій сфері використовує традиційні методи розпізнання в комбінації з спеціалізованими камерами, а у цій роботі автор працює з зображеннями, що зроблені на смартфон.

Отже, розглянемо алгоритм більш детально. На першому кроці система на-

магається спростити собі роботу, а тобто прибирає усі зайві шуми за допомогою монохромного фільтра та фільтра Гауса. Це досить швидкий процес в порівнянні з іншими кроками, але не менш важливий, оскільки велика кількість шумів може завадити локалізувати об'єкт на зображення чи розпізнати символи.

На наступному кроці потрібно знайти координати номерного знака. Існує дуже багато метод для розв'язання цієї задачі, в більшості вони відрізняється фільтром, що використовується на початку цього кроку. Було виявлено декілька непоганих та досить відомих методів, а опрацювавши декілька десятків зображень найкращі результати продемонстрував оператор Кенні. Слід зазначити, що цей оператор має власний досить складний алгоритм, що базується на варіаційному численні, а тому не дивно що він має кращі результати. Після використанні фільтру для знаходження меж об'єктів на зображенні, треба виконати пошук усіх прямокутників на зображенні та перевірити їх на схожість номерному знаку за формою. Таким чином, отримуємо координати прямокутника з номерним знаком.

На останньому етапі виконується сегментація символів за допомогою гістограми горизонтальної проєкції номерного знаку. Далі використовується згорткова нейронна мережа для розпізнання символів. У результаті отримуємо текстову репрезентацію номерного знаку з вхідного зображення.

3.3. Технології

3.3.1. Технологія обміну даними NATS. Технологія, що дозволяє обмінюватися даними, які сегментовані на повідомлення між комп'ютерними програмами та послугами. Повідомлення адресовані суб'єктам та не залежать від розташування мережі. Це забезпечує рівень абстракції між додатком або службою та базовою фізичною мережею. Дані кодуються та формуються як повідомлення, а потім надсилаються відправником. Повідомлення приймає, декодує та обробляє один з клієнтів.

Подібні технології дозволяють програмам легко спілкуватися в різних середовищах, мовах, хмарних провайдерах та системах. Клієнти підключаються до

системи, як правило, за однією URL-адресою, а потім підписуються або публікують повідомлення. Завдяки цій простій конструкції NATS дозволяє програмам обмінюватися загальним кодом обробки повідомлень, ізолювати ресурси та взаємозалежності та масштабуватись, легко обробляючи збільшення обсягу повідомлень, незалежно від того, чи це запити на послуги, чи потокові дані.

Ядро цієї технології гарантує доставлення повідомлення щонайменше один раз. Якщо клієнт не слухає повідомлення або не активний під час надсилання, повідомлення не отримується. Це той самий рівень гарантії, який надає TCP/IP. NATS — це система обміну повідомленнями, яка зберігатиме лише повідомлення в пам'яті й ніколи не писатиме повідомлення безпосередньо на диск. Якщо вам потрібні вищі рівні обслуговування, ви можете використовувати NATS Streaming або вбудувати додаткову надійність у свої клієнтські програми, таких як ACK та порядкові номери.

3.3.2. Збереження даних за допомогою S3 протоколу. S3 — протокол, сумісний з об'єктним сховищем Amazon S3 (Simple Storage Service). Протокол дозволяє просто зберігати об'єкти, майже так само, як за допомогою файлової системи. Основною відмінністю S3 є можливість зберігати великий обсяг даних без розбивки їх на директорії. S3 — одне з найбільш популярних рішень для зберігання даних, бо його основна перевага — можливість доступу через API, що дозволяє організувати гнучку взаємодію зі сховищем. Так, рішення на протоколі S3 використовуються як файлові системи чи віддалені бази даних. S3 дозволяє налаштувати права доступу до окремих файлів і при цьому не проганяти всі файли через backend-сервер шляхом використання динамічних токенів.

Зберігання даних в хмарному сервісі об'єктного сховища AWS має декілька ключових переваг:

Надійність, доступність і масштабованість. Сервіси S3 розроблені з нуля і розраховані на надійність на рівні 99,9%. Дані автоматично розподіляються мінімум за трьома фізичними центрами, які географічно рознесені один від одного не менш ніж на 10 кілометрів в рамках регіону AWS. Крім того, можна автома-

тично копіювати дані в будь-який інший регіон AWS. За інформацією Gartner, Inc., S3 — це найбільший публічний хмарний сервіс зберігання об'єктів за кількістю обслуговуваних даних. У порівнянні з будь-яким іншим постачальником, AWS може надати більше аналітичних даних про те, як клієнти використовують публічні хмарні сервіси зберігання у відповідному масштабі.

Безпека і відповідність вимогам. S3 є єдиним хмарним сервісом зберігання, що підтримує три різних форми шифрування. Amazon Macie забезпечує сервіс безпеки зі штучним інтелектом, який виконує безперервний моніторинг використання даних для виявлення аномалій і своєчасного вжиття заходів для запобігання втрати або випадкової публікації даних. S3 підтримує більше стандартів безпеки та сертифікатів відповідності вимогам, ніж будь-який інший сервіс. Це забезпечує відповідність вимогам безпеки практично будь-якого наглядового органу у світі.

Гнучкість управління. AWS пропонує максимально гнучкий набір можливостей для керування сховищем і адміністрування. Адміністратори сховища можуть класифікувати та візуалізувати тенденції використання даних, випускати по ньому звіти, що дозволяє знижувати витрати й підвищувати рівень обслуговування. Об'єкти можуть бути позначені унікальними налаштованим метаданими, щоб клієнти могли бачити та контролювати використання, витрати та безпеку сховища окремо для кожного робочого навантаження. Інструмент S3 Inventory генерує плановані звіти про об'єкти і їх метадані для проведення технічного обслуговування, забезпечення відповідності вимогам і аналітики. Крім того, S3 може аналізувати шаблони доступу до об'єктів для створення політик життєвого циклу, які автоматизують багаторівневе зберігання, видалення та збереження даних. І нарешті, оскільки S3 сумісний з AWS Lambda, клієнти можуть реєструвати дії, визначати оповіщення і викликати робочі процеси — і все це без створення будь-якої додаткової інфраструктури.

Запити до даних без вилучення. S3 — це єдина платформа хмарного сховища, яка дозволяє виконувати складний аналіз великих даних, не вимагаючи їх

вилучення і переміщення в окрему аналітичну систему. Той, хто знає SQL, може використовувати Amazon Athena для аналізу на вимогу величезних обсягів неструктурованих даних в Amazon S3. Використовуючи Amazon Redshift Spectrum, клієнти можуть проводити складний аналіз даних в S3 і виконувати запити, які охоплюють дані, що зберігаються як в S3, так і в сховищах даних Redshift.

3.4. Реалізація програмного забезпечення

У цій роботі використовуються тільки сучасні технології, мови, алгоритми. Сучасний підхід для побудови архітектури, а тобто декомпозиція на декілька контекстів, кожний з яких представлений окремим мікросервісом. Для швидкої роботи використовується низькорівневі протоколи взаємодії.

Розглянемо більш детально як повинна функціонувати система.

Розпізнавач. Сервіс, що займається безпосередньо скануванням та розпізнанням номерного знаку повинен мати прикладний програмний інтерфейс для взаємодії з ним. Таким чином за допомогою gRPC протоколу клієнт відправляє запит на розпізнання номерного знаку з зображення на сервер та отримує відповідь з масивом координат номерних знаків та їх текстовими репрезентаціями.

Цей сервіс можна досить непогано горизонтально масштабувати, використовуючи сучасні інструменти для балансування вхідного трафіку. Використання gRPC відкриває широкі можливості для скасування запиту та використання потоків для розпізнання номерних знаків на потоковому відео.

Цей сервіс написаний на мові Python, оскільки ця мова усі потрібні бібліотеки для роботи з зображенням. OpenCV — бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом [11]. Бібліотека надає засоби для обробки й аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях. Для пришвидшення розробки нейронної мережі

було також використано PyTorch — це відкрита бібліотека машинного навчання, що використовують для комп'ютерного зору [12]. Розробкою бібліотеки займається група дослідження штучного інтелекту компанії Facebook. Бібліотека має декілька десятків підготовлених глибинних нейронних мереж, що одразу можуть використовуватися у проєктах. Варто зазначити, що в більшості випадків методи бібліотек реалізовані мовою C і мають лише програмні інтерфейси на мові Python, а це означає, що не повинно бути питань щодо їх швидкодії.

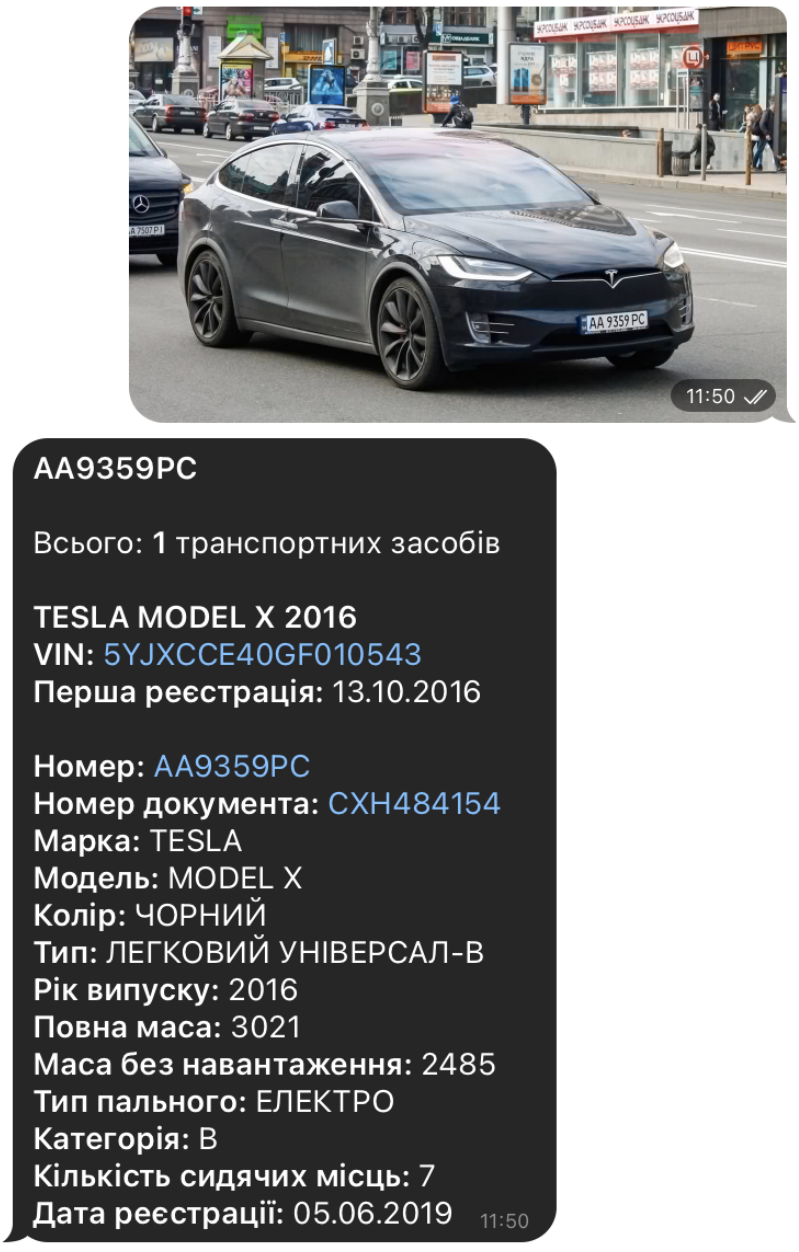

Рис. 3.2. Приклад сканування та розпізнання номерного знаку

Бот. Бот відіграє роль клієнта у взаємодії с системою розпізнання номерного знаку. Цей сервіс також представлений сервером, проте працює він за іншим, більш звичним протоколом — HTTPS. Усю логіку роботи можна звести до трансляції запиту користувача у формі зображення у gRPC запит до розпізнання номерного знаку. Це дає можливість контролювати взаємодії звичайних користувачів із досить враженою системою.

Як ми бачимо на рис. 3.2, в сервіс також інтегрована система, що була розроблена у рамках бакалаврської дипломної роботи. Таким чином, програмне забезпечення має змогу виконати пошук у реєстрі транспортних засобів після вдалого розпізнання зображення. Якщо ж, номерного знаку не було виявлено, бот інформує користувача, що номерний знак не було знайдено.

Оскільки є досить велика вірогідність того, що функціонал бути активно використовувати користувачами боту, було вирішено також розробити модуль, що бути зберігати зображення на якому було знайдено номерним знак. У майбутньому це дасть можливість покращити систему та навчати її на своїх помилках. Таким чином, після того, як зображення було розпізнано бот відсилає повідомлення до системи обробки розпізнання.

Цей сервіс написаний на мові Go, використовує бібліотеку TeleBot для взаємодії з прикладним програмний інтерфейсом системи Telegram.

Оброблювач. Ця частина системи має декілька простих завдань: компресія зображення до мінімальних розмірів, збереження інформації у базу даних, завантаження зображення у спеціалізоване сховище. У даному випадку використовується S3 сховище для збереження зображень та база даних PostgreSQL для збереження інших даних.

Цей сервіс написаний на мові Go, використовує низку стандартних бібліотек для компресії JPEG та PNG зображень, а також декілька інструментів для роботи з зовнішніми сховищами даних.

3.5. Тестування

У цій дипломній роботі для тестування розпізнання номерних знаків було застосовано 300 зображень. Точність обчислюється діленням числа символів і цифр, розпізнаних на загальну кількість символів і числа.

У такому випадку точність розпізнання обчислюється діленням кількості розпізнаних символів та чисел на загальну кількість символів та чисел. Деякі зображення для тренування було забрані власноруч, а також було зібрана зображення номерних знаків з мережі Інтернет. Кожне зображення було марковане вручну, змінюючи назву файлу на його номерний знак, щоб перевірити точність. Після навчання загальна точність розпізнання досягнула 97,05%, а точність розпізнавання номерного знака в нормальному стані становить 99.4%. Точність може бути обмежена алгоритмами відновлення спотворень, це означає, що коли номерний знак надто спотворений, номер неможливо відновити. Крім того, коли між символами номерного знаку є додаткові позначки, це впливає на точність. Точність розпізнавання можна покращити, змінивши моделі нейронних мереж на моделі з кращою точністю, але більш повільні, такі як швидкий R-CNN. Модель добре навчена та має допустимі втрати, а точність більша за 95%.

ВИСНОВКИ

Під час виконання роботи було розроблено алгоритм, що використовує найкращі напрацювання у сфері комп'ютерного зору, починаючи з покращеного процесу виявлення меж, що базується на варіаційному численні, закінчуючи нейронною мережею, що робить процес розпізнання символів більш інтелектуальним. Розроблена глибинна нейронна мережа має змогу навчатися на своїх помилках, а тому зі зростанням кількості оброблених зображень, буде покращуватися модель розпізнання. Для сегментації символів застосовується горизонтальна та вертикальна проєкція. Для розпізнавання символів використовуються оптичне розпізнавання символів та методи нейронних мереж. Загальна точність становить 97% із 300 перевірених знімків. Результати все ще можна поліпшити, регулюючи параметри нейронної мережі та проводячи обертання та відновлення спотворень.

Спроєктовано архітектуру, що ізолює незалежні між собою частини. Оскільки кожен сервіс має спеціальний прикладний програмний інтерфейс для взаємодії, система виглядає відмовостійкою, а також стійкою до атак на певні її частини. Архітектура системи була спроєктована таким чином, аби мінімізувати складність розгортання системи.

Спроєктовано та протестовано програмне забезпечення, що повністю автоматизує процес розпізнання. Існує велика кількість готових рішень, проте більшість з них досить застарілі, оскільки не застосовують нові технології та ідеї. Використання глибинних нейронних мереж не завжди є доцільним, але у цьому випадку це покращило результати розпізнання в порівнянні з традиційними методами. Слід зазначити, завдяки якісним бібліотекам для нейронних мереж на мові Python, процес розробки був досить швидким. Також було приділено увагу подальшої конфігурації системи, усі параметри, що відіграють важливу ролі у розпізнанні можна змінити, це дасть змогу підігнати алгоритми системи під номерні знаки інших країн. Отже, система вийшла досить гнучкою на випадок бажання змінити

деякі параметри розпізнання. У ході реалізації було приділено увагу використанню надійних технологій збереження даних. Також було враховано, що систему буде потребувати подальшого покращення розпізнання, а тому було розроблено додатковий модуль, що буде зберігати усі спроби розпізнання. На останньому етапі увесь функціонал був протестований в ручному режимі.

Отже, можна стверджувати, що мета дипломної роботи досягнена — система для розпізнавання номерного знаку без участі людини, була вдало розроблена, вимоги враховані й реалізовані. Таким чином, у результаті проведеної дипломної роботи ми отримали програмний продукт, що доступний для використання кожному, а також є простим у використанні при усій його технічній складності. Розроблений програмний продукт має відкритий код, та може використовуватися у комерційних цілях. Система була розгорнута, тож будь-який користувач системи Telegram може її використовувати.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Kamal N. N G. L. E.* License plate recognition using a set of classifiers / G. L. E. Kamal N. N. — Lambert Academic Publishing, 2013. — .
2. *D. S.* Digital image processing: a signal processing and algorithmic approach / S. D. — Springer, 2017. — . — С. 45–85.
3. *Chaki J. D. N.* A beginner's guide to image pre-processing techniques / D. N. Chaki J. — CRC Press, 2018. — .
4. *Jähne B. Scharr H. K. S.* Principles of filter design. in handbook of computer vision and applications. — 1999.
5. *A. G.* Modified Laplacian Filter and Edge Detection / G. A. — Lambert Academic Publishing, 2015. — . — С. 56.
6. *B. G.* Canny edge detection tutorial. — 2002. — .
7. *F. H.* The history of OCR, optical character recognition / H. F. — [Manchester Center, Vt.]: Recognition Technologies Users Association, 1982. — С. 21–30.
8. *Ye Q. D. D.* Text Detection and Recognition in Imagery: A Survey / D. D. Ye Q. — 2015. — Т. 37.
9. *Wood J. C. B. D. T.* Radon transformation of time-frequency distributions for analysis of multicomponent signals / B. D. T. Wood J. C. — 1994. — Т. 42.
10. *S. B.* Hierarchical neural networks for image interpretation / B. S. — Springer, 2003. — .
11. *Bradski G. K. A.* Learning OpenCV / K. A. Bradski G. — O'Reilly Media, Inc., 2008. — . — С. 21–30.
12. *Stevens E. Antiga L. V. T.* Deep Learning with PyTorch / V. T. Stevens E., Antiga L. — O'Reilly Media, Inc., 2020. — . — С. 520.

Додаток А

```
1  import cv2
2  import numpy as np
3  from matplotlib import pyplot as plt
4
5  img = cv2.imread('binarized-plate.jpg',0)
6
7  kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3 ,3 ))
8  closed = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
9  opened = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
10
11 cv2.imwrite('closed-plate.jpg', closed)
12 cv2.imwrite('opened-plate.jpg', opened)
13
14 tophat = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)
15 blackhat = cv2.morphologyEx(img, cv2.MORPH_BLACKHAT, kernel)
16
17 gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)
18
19 im_h = cv2.hconcat([tophat, blackhat, gradient])
20 cv2.imwrite('opencv_hconcat.jpg', im_h)
21
22 cv2.imwrite('tophat-plate.jpg', tophat)
23 cv2.imwrite('blackhat-plate.jpg', blackhat)
24 cv2.imwrite('gradient-plate.jpg', gradient)
```

Лістинг А.1. Побудова градієнта

```
import numpy as np
import cv2

img = cv2.imread('images/binarized-plate.jpg', 0)

edges = cv2.Canny(img, 100, 200)
not_canny = cv2.bitwise_not(edges)

cv2.imwrite('images/canny.jpg', not_canny)
```

Лістинг А.2. Оператор Кенні

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('images/plates.png', 0)

vertical = np.sum(img, axis=0).tolist()
horizontal = np.sum(img, axis=1).tolist()

f, ax = plt.subplots()
ax.plot(vertical, color='black')
plt.savefig('images/vertical.png')

f, ax = plt.subplots()
ax.plot(horizontal, color='black')
plt.savefig('images/horizontal.png')
```

Лістинг А.3. Проекції номерного знаку

```
import cv2
import numpy as np
import imutils
import easyocrd('images/binarized-plate.jpg', 0)

from matplotlib import pyplot as plt

img = cv2.imread('image4.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(cv2.cvtColor(gray, cv2.COLOR_BGR2RGB))


bfilter = cv2.bilateralFilter(gray, 11, 17, 17)
edged = cv2.Canny(bfilter, 30, 200)
plt.imshow(cv2.cvtColor(edged, cv2.COLOR_BGR2RGB))

keypoints = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours = imutils.grab_contours(keypoints)
contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]

location = None
for contour in contours:
    approx = cv2.approxPolyDP(contour, 10, True)
    if len(approx) == 4:
        location = approx
        break

mask = np.zeros(gray.shape, np.uint8)
new_image = cv2.drawContours(mask, [location], 0,255, -1)
new_image = cv2.bitwise_and(img, img, mask=mask)
plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
```

Лістинг А.4. Локалізація номерного знака

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('images/plate.jpg',0)

smallest = img.min(axis=(0, 1))
largest  = img.max(axis=(0, 1))
f        = largest - (largest - smallest) / 3

ret, thresh = cv2.threshold(img, f, 255, cv2.THRESH_BINARY)
cv2.imwrite('images/binarized-plate.jpg', thresh)
```

Лістинг А.5. Бінаризація зображення

```
(x,y)     = np.where(mask==255)
(x1, y1) = (np.min(x), np.min(y))
(x2, y2) = (np.max(x), np.max(y))
cropped_image = gray[x1:x2+1, y1:y2+1]
plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))

reader = easyocr.Reader(['en'])
result = reader.readtext(cropped_image)
text   = result[0][-2]

font = cv2.FONT_HERSHEY_SIMPLEX
res = cv2.putText(img, text=text, org=(approx[0][0][0], approx[1][0][1]+60),
    fontFace=font, fontScale=1, color=(0,255,0),
    thickness=2, lineType=cv2.LINE_AA)
res = cv2.rectangle(img, tuple(approx[0][0]), tuple(approx[2][0]), (0,255,0),3)

plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
```

Лістинг А.6. Роспізнавання номерного знаку за допомгою OCR

```
import asyncio
from nats.aio.client import Client as NATS
from nats.aio.errors import ErrConnectionClosed, ErrTimeout, ErrNoServers

async def run(loop):
    nc = NATS()

    await nc.connect("nats://127.0.0.1:4222", loop=loop)

    async def message_handler(msg):
        subject = msg.subject
        reply = msg.reply
        data = msg.data.decode()
        print("Received a message on '{subject} {reply}': {data}".format(
            subject=subject, reply=reply, data=data))


    await nc.subscribe("foo.*.baz", cb=message_handler)
    await nc.subscribe("foo.bar.*", cb=message_handler)




    await nc.subscribe("foo.>", cb=message_handler)


    await nc.publish("foo.bar.baz", b'Hello World')


    await nc.drain()

if __name__ == '__main__':
    loop = asyncio.get_event_loop()
    loop.run_until_complete(run(loop))
    loop.close()
```

Лістинг А.7. Приклад роботи з NATS

```go
package main

import (
    "log"

    "github.com/minio/minio-go/v7"
    "github.com/minio/minio-go/v7/pkg/credentials"
)

func main() {
    endpoint := "play.min.io"
    accessKeyID := "xxx"
    secretAccessKey := "yyy"
    useSSL := true

    // Initialize minio client object.
    minioClient, err := minio.New(endpoint, &minio.Options{
        Creds:  credentials.NewStaticV4(accessKeyID, secretAccessKey, ""),
        Secure: useSSL,
    })
    if err != nil {
        log.Fatalln(err)
    }

    log.Printf("%#v\n", minioClient) // minioClient is now setup

    file, err := os.Open("vehicle.jpg")
    if err != nil {
        fmt.Println(err)
        return
    }
    defer file.Close()

    fileStat, err := file.Stat()
    if err != nil {
        fmt.Println(err)
        return
    }

    uploadInfo, err := minioClient.PutObject(context.Background(), "mybucket", "myobject",
    if err != nil {
```

```
42          fmt.Println(err)
43          return
44      }
45      fmt.Println("Successfully uploaded bytes: ", uploadInfo)
```

Лістинг А.8. Приклад роботи з S3

```
1  import torch.nn as nn
2  import torch.nn.functional as F
3
4
5  class Net(nn.Module):
6      def __init__(self):
7          super().__init__()
8          self.conv1 = nn.Conv2d(3, 6, 5)
9          self.pool = nn.MaxPool2d(2, 2)
10         self.conv2 = nn.Conv2d(6, 16, 5)
11         self.fc1 = nn.Linear(16 * 5 * 5, 120)
12         self.fc2 = nn.Linear(120, 84)
13         self.fc3 = nn.Linear(84, 10)
14
15     def forward(self, x):
16         x = self.pool(F.relu(self.conv1(x)))
17         x = self.pool(F.relu(self.conv2(x)))
18         x = x.view(-1, 16 * 5 * 5)
19         x = F.relu(self.fc1(x))
20         x = F.relu(self.fc2(x))
21         x = self.fc3(x)
22         return x
23
24 net = Net()
```

Лістинг А.9. Створення нейронної мережі

```
for epoch in range(2):
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # get the inputs; data is a list of [inputs, labels]
        inputs, labels = data

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss += loss.item()
        if i % 2000 == 1999:
            print('[%d, %5d] loss: %.3f' %
                  (epoch + 1, i + 1, running_loss / 2000))
            running_loss = 0.0

print('Finished Training')
```

Лістинг А.10. Тренування нейронної мережі

```
dataiter = iter(testloader)
images, labels = dataiter.next()

# print images
imshow(torchvision.utils.make_grid(images))
print('GroundTruth: ', ' '.join('\%5s' % classes[labels[j]] for j in range(4)))
```

Лістинг А.11. Тестування нейронної мережі