НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

«До захисту допущено»
Завідувач кафедри
_____ Олег Чертов
«___» _____ 2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Наука про дані та математичне моделювання»
спеціальності 113 «Прикладна математика»
на тему: «Математичне та програмне забезпечення системи розпізнавання номерних знаків»

Виконала:
студентка IV курсу, групи КМ-92
Черниш Аліна Андріївна _____

Керівник:
Старший викладач, канд. техн. наук
Ліскін Вячеслав Олегович _____

Консультант з нормоконтролю:
Старший викладач,
Мальчиков Володимир Вікторович _____

Рецензент:
Доцент каф. СПіСКС , канд. техн. наук, доцент
Тарасенко-Клятченко Оксана Володимирівна _____

Засвідчую, що в цій дипломній роботі
немає запозичень із праць інших авторів
без відповідних посилань.
Студентка Черниш А.А. _____

Київ —2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 113 «Прикладна математика»

Освітньо-професійна програма «Наука про дані та математичне моделювання»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ ОлегЧертов
«___» _____ 2023 р.

ЗАВДАННЯ
на дипломну роботу студентці

Черниш Аліні Андріївні

1. Тема роботи: «Математичне та програмне забезпечення системи розпізнавання номерних знаків»,керівник роботи Ліскін Вячеслав Олегович, старший викладач, канд. техн. наук, затверджені наказом по університету від «31» травня 2023 р. № 2108-С.
2. Термін подання студенткою роботи: «12» червня 2023 р.
3. Вихідні дані до роботи: розроблювана система має детектувати та розпізнавати номерні знаки транспортних засобів, мінімальна точність розпізнавання – 80%.
4. Зміст роботи: виконати огляд існуючих рішень пошуку роботи та їх функціоналу, розробка концептуальної моделі детекції та розпізнавання номерних знаків, імплементація моделі детекції та розпізнавання номерних знаків, проведення верифікації та валідації системи детекції та розпізнавання номерних знаків.
5. Перелік ілюстративного матеріалу: структура композитної моделі детекції та розпізнавання, структура загорткової нейронної мережі, графіки метрик отриманих під час навчання, інтерфейс системи.

6. Дата видачі завдання: «06» лютого 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за тематикою та збір даних	12.11.2022	
2	Підготовка матеріалів першого розділу	14.12.2022	
3	Проведення порівняльного аналізу математичних методів	24.12.2022	
4	Підготовка матеріалів другого розділу роботи	01.02.2023	
5	Розроблення математичного забезпечення для розпізнавання номерних знаків	01.03.2023	
6	Підготовка матеріалів третього розділу роботи	15.03.2023	
7	Розроблення програмного забезпечення для розпізнавання номерних знаків	05.04.2023	
8	Підготовка матеріалів четвертого розділу роботи	15.04.2023	
9	Підготовка графічної частини	03.05.2023	
10	Оформлення пояснювальної записки	01.06.2023	

Студент _____ Аліна ЧЕРНИШ

Керівник роботи _____ Вячеслав ЛІСКІН

АНОТАЦІЯ

Дипломну роботу виконано на 73 аркушах, вона містить 2 додатки та перелік посилань на використані джерела з 21 найменувань. У роботі наведено 8 рисунки та 5 таблиць.

Метою даної роботи є розробка системи розпізнавання номерних знаків транспортних засобів.

Проведено порівняльний аналіз існуючих математичних та комерційних рішень розпізнавання українських номерних знаків. Для розв’язання задачі було обрано згорткову нейронну мережу для виявлення номерних знаків, а оптичне розпізнавання символів для видобутку тексту.Для підвищення точності розпізнавання були застосовані фільтри шумів.

Побудовано модель системи детекції та розпізнавання номерних знаків. Описано математичні методи, які використовуються окремими компонентами моделі. Розроблено програмне забезпечення системи. Проведено верифікацію та валідацію системи.

Ключові слова: детекція номерних знаків, розпізнавання символів, згорткова нейронна мережа, машинне навчання, системи Data Science, оптичне розпізнавання символів, сегментація символів.

ABSTRACT

The thesis is presented in 73 pages. It contains 2 appendixes and bibliography of 21 references. 8 figures and 5 tables are given in the thesis.

The purpose of this work is to develop a license plate recognition system. system for recognizing license plates of vehicles.

A comparative analysis of existing mathematical and commercial solutions for recognizing Ukrainian license plates has been conducted. To solve the problem, a convolutional neural network was chosen for license plate detection, and optical character recognition for extracting text. Noise filters were applied to improve the accuracy of recognition.

A model of the license plate detection and recognition system has been built. Mathematical methods used by individual components of the model have been described. The software of the system has been developed. Verification and validation of the system have been carried out.

Keywords: license plate detection, character recognition, convolutional neural network, machine learning, Data Science systems, optical character recognition, character segmentation.

ЗМІСТ

Перелік умовних позначень, скорочень і термінів

CCL – Connected Component Labeling

CNN – Convolution Neural Network

LPDR – License plate detection and recognition

LPR – License plate recognition

MSE – Mean Squared Error

OCR – Optical character recognition

PCA – Principal Component Analysis

ROI – Region of Interst

SOM – Self-Organized Map

SVM – Support Vector Machine

UML – Unified Modeling Language

МН – Машинне навчання

ШІ – Штучний інтелект

ВСТУП

Розпізнавання об'єктів на зображеннях є одним з найбільш розвинених та динамічно розвиваючихся напрямків у сфері інформаційних технологій. Його використання відчутно актуальне в різних галузях. Цей вид технології відіграє важливу роль у виявленні об'єктів, класифікації їх характеристик та ідентифікації їх на зображеннях, що сприяє поліпшенню безпеки, ефективності та точності відповідних систем і процесів.

Немаловажливою є система виявлення та розпізнавання номерних знаків в транспортних системах та не може бути переоціненою, оскільки номерний знак є основним аспектом серед більшості застосунків, пов'язаних із транспортною системою. Технологія розпізнавання номерних знаків демонструє низку переваг, серед яких підвищення рівня безпеки, поліпшення пропускної здатності транспорту та економія витрат на паркування і платні дороги. Більше того, правоохоронні органи використовують цю технологію для виявлення крадених автомобілів, відстеження підозрюваних осіб та пошуку зниклих.

Сучасні відеокамери спостереження використовують технологію інфрачервоного підсвічування, що дозволяє здійснювати розпізнавання номерів в будь-який час доби та погодних умовах, що робить цю технологію привабливою для використання в галузях, де важлива безпека периметра. З'явлення бездротового Інтернету також стало важливим моментом в розвитку технології, оскільки відбувається передача інформації в режимі реального часу, що сприяє впровадженню систем автоматичну ідентифікацію автомобільних номерів у правоохоронних органах, банках, корпоративних офісах та інших організаціях для забезпечення ефективної охорони периметра та контролю доступу.

Слід зазначити, що система виявлення та розпізнавання номерних знаків є значущою та підлягає інтенсивному дослідженню в галузі обробки зображень. Існує

велика кількість алгоритмів та технік аби успішно виявити та розпізнати номерні знаки транспортних засобів.

Попри різні погодні умови, забарвлення, шрифту та розміру номерних знаків, або навіть зміна законодавства, введення нових типів номерних знаків, зміна технічних характеристик транспортних засобів - усі ці фактори необхідні для подальшого розвитку процесу детекції та розпізнавання номерних знаків.

Система LPR використовує камери та спеціальне програмне забезпечення для захоплення зображень автомобільних номерних знаків та їх подальшого перетворення у текстовий формат.

Звичайно, процес LPDR складається з двох основних етапів: виявлення номерних знаків (LPD) та розпізнавання номерних знаків (LPR). Задачею LPD є ідентифікація та локалізація номерного знака, а також створення обмежувальної рамки навколо нього. Зазвичай також потрібний процес верифікації, щоб гарантувати правильність виявлення номерного знака. У LPR номерні знаки спочатку розбиваються на окремі символи, а потім кожен символ розпізнається за допомогою спеціалізованого розпізнавача.

1 Постановка задачі

Метою написання роботи є розробка системи для розпізнавання номерних знаків на основі машинного навчання.

При розробленні відповідного забезпечення потрібно розв’язати наступні завдання:

а) проведення порівняльного аналізу алгоритмів системи детекції та розпізнавання номерних знаків;

б) вибір існуючого методу для вирішення задачі розпізнавання номерних знаків;

в) створення математичного забезпечення

г) розробка програмного забезпечення на базі вибраного математичного методу;

д) тестування розробленої автоматизованої системи.

Реалізована система має задовольняти такі вимоги:

а) мати високі показники ефективності розпізнавання;

б) розпізнавання номерних знаків при різних ситуаціях;

в) надавати аналітичну інформацію про результати роботи програми.

2 Огляд та аналіз методів класифікації та виявлення номерних знаків

Першим та основним етапом є виявлення номерного знаку. Виявлення номерного знака є складним завданням через причини:

- Область номерного знаку є маленькою(менша за 80px) або занадто великою(більше 300px);
- Великий різновид номерних знаків, такі як довгі, короткі, високі, однорядкові чи дворядкові;
- Інші фактори як: розмите зображення, нерівномірне або слабке освітлення, рух автомобіля, низька роздільна здатність вхідного зображення, забруднення пластини, тінь і відображення, несприятливі погодні умови;

Тобто можна підсумувати, що розпізнавання автомобільних номерів залежить від чіткої детекції області надходження. Задача розпізнавання образів не має точного аналітичного рішення, що робить створення універсального алгоритму складним. Однак, було запропоновано багато методів та алгоритмів для вирішення цієї задачі.

2.1 Методи виявлення номерних знаків

Метод Віоли-Джонсона є одним з найвідомішим методом детекції об'єктів в реальному часі. Багато алгоритмів локалізації об'єктів на зображеннях базуються на ідеях, запропонованих Полом Віолою та Майклом Джонсом. Даний метод був запропонований у 2001 році та через свою швидкість та ефективність цей метод став широко використовуваним.

Метод складається з двох фаз: процесу навчання та процесу розпізнавання. Виконавча швидкість алгоритму навчання не має вирішального значення на практиці. Однак, критично важливою є продуктивність алгоритму розпізнавання.

Основні принципи:

1. Ознаки, подібні до Хаара, формується з двох або трьох «чорно» та «білих» прямокутних областей, які можна використовувати для опису областей зображення з різними текстурами та візерунками;
2. Використання інтегральних зображень через створення за допомогою суми значень яскравості вище та ліворуч від пікселя зображень;
3. AdaBoost – алгоритм підсилення машинного навчання, який використовує велику кількість базових класифікаторів, кожен з яких відповідає певній функції ознак. AdaBoost є адаптивним у тому розумінні, що наступні слабкі класифікатори перекручуються на користь тих випадків, які були неправильно класифіковані попередніми класифікаторами. Отже, відбір найкращих функцій відбувається на основі оцінки їхньої корисності для класифікації навчальних даних;
4. Складний класифікатор, який матиме вигляд багаторівневої структури, щоб забезпечити швидке виявлення об’єктів зосередження уваги на зоні зображення, яка ймовірніше містить об'єкт.

Використовуючи технологію ковзного вікна, метод визначає наявність детектованого об'єкту в аналізованому вікні за допомогою каскаду слабких класифікаторів. Рамка, розмір якої менший, ніж вихідне зображення, переміщується з деяким кроком по зображенню [17].

Переваги:

1. Швидкість та ефективність завдяки каскадному класифікатору.
2. Стійкість до змін, оскільки примітиви Хаара здатні виявляти шаблони при різних умовах.
3. Можливість навчити класифікатор визначати будь-який об'єкт.

4. Невелика кількість пам’яті використовується завдяки невеликій кількості функцій.
5. Можливе знаходження більше однієї області.

Недоліки:

1. Великий набір тренувальний даних;
2. Помилкові спрацювання, якщо недостатню різноманітна тренувальна вибірка (при збільшенні кута повороту об'єкта ефективність його розпізнавання значно падає).

Для створення набору даних використовуються позитивні та негативні дані. Позитивні дані містять зображення номерних знаків транспортних засобів, а негативні дані включають інші випадкові зображення, але без номерних знаків. Кількість позитивних та негативних даних, які використовуються, може змінюватися в залежності від проведених експериментів.

Метод опорних векторів є методом машинного навчання, який використовує моделі з певними алгоритмами для класифікації об'єктів, й Вапнік представив цей спосіб 1992 року. Алгоритм має широке застосування в різних галузях, таких як класифікація, регресія та відбір ознак. У процесі категоризації SVM знаходить оптимальну гіперплощину, використовуючи поняття зазору, яке є ключовою характеристикою цього методу.

Метод навчання МОВ будує модель для передбачення належності нового прикладу до однієї з двох категорій на основі заданого набору тренувальних прикладів, які вже мають відповідні мітки категорій [21].

Основні принципи:

1. МОВ — класифікує дані у два класи. Щоб визначити номерні знаки на зображеннях, декілька класифікаторів МОВ навчаються розпізнавати різні ознаки номерних знаків, такі як колір, форма і текстура.
2. Обирається гіперплощина, на якій відстань від кожної класу є максимально великою. Функція витрат описується так:

$$c(x, y, f(x)) = \begin{cases} 0 & y \times f(x) \geq 1 \\ 1 - y \times f(x) & y \times f(x) < 1 \end{cases}, \quad (2.1.1)$$

Функцію втрат позначають як $c(x, y, f(x))$, де x представляє вхідні ознаки точки даних, y – це клас точки даних (або -1, або 1). $f(x)$ – це вихід або прогноз моделі SVM для вказаного вхідного значення x.

Параметр регуляризації необхідний для збалансування максимізації маржі та витрат, тому функція витрат матиме наступний вигляд:

$$\min_w \lambda \|w\|^2 + \sum_{i=1}^{n} (1 - y_i < x_i w >) \quad (2.1.2)$$

- $\|w\|^2$ представляє квадратичну норму вектора вагів w,
- λ є параметром регуляризації, який контролює компроміс між максимізацією маржі і витрат,
- y_i – справжнє значення мітки класу для і-го навчального зразка,
- x_i – вектор ознак для і-го навчального зразка,
- $< x_i w >$ – внутрішній добуток між векторами ознак і вагами моделі.

Переваги:

1. Висока швидкість адаптації;
2. Стійкість через здатність знаходження шумів та відхилень, що відбуваються в зображеннях;
3. Швидка обробка даних та легка можливість перенавчання;
4. Ефективність у випадках, коли кількість вимірів більша за кількість зразків.

Недоліки:

1. Не під всі типи виявлення номерних знаків підійде;
2. Поступається точністю іншим методам;

3. Метод опорних векторів не є оптимальним для обробки великих обсягів даних;
4. Якщо дані містять багато шуму, тобто при наявності перетинів між цільовими класами, МОВ може не дати задовільних результатів;

Алгоритм щільності країв виявляє області з високою щільністю країв, які можуть вказувати на наявність номерних знаків. Такі області можна з'єднати, а менш щільні області можна видалити з бінарного зображення краю в кожному рядку та стовпці.

Основні 4 принципи:

1. Межі автомобільних номерів характеризуються високою щільністю.
2. Символи зазвичай друкуються горизонтально з майже однаковою висотою
3. Фон має протилежне двійкове значенняВідстань між номерними знаками значною, якщо їх декілька на зображенні.

Щільність краю de визначається частка крайових пікселів у рядку зображення [19]. Рішення базується на щільності краю de , що обчислюється як частка чорних і білих пікселів у кожному рядку та виконується:

$$\mathrm{de} = \frac{\mathrm{wl}}{\mathrm{wl} + \mathrm{bl}} \qquad (2.1.3)$$

Де wl та bl білі та чорні пікселі відповідно в кожному рядку, de−щільність краю.

Переваги:

1. Добре працює при різних кутах та змінні освітленні;
2. Працює в режимі реального часу та легкий в реалізації.

Недоліки:

1. Повільне виявлення області через високу обчислювальну складність;
2. Вузький діапазон розмірів та шрифтів,інакше можливі помилкові виявлення

Згорткова нейрона мережа є потужним методом виявлення номерних знаків у комп'ютерному зорі, який має вищу ефективність порівняно з класичними методами. У 2012 році Алекс Крижевський використав нейронні мережі для перемоги в конкурсі з машинного навчання ImageNet, знизивши помилки класифікації на 10%. Це стало проривом у використанні ЗНМ, і з того часу їм приділяється все більше уваги.

Сучасні компанії широко використовують глибинне навчання за допомогою нейронних мереж у різних сферах, зокрема, для автоматичного тегування фотографій, пошуку фотографій користувачів, рекомендацій товарів, персоналізації домашньої сторінки користувача та створення пошукової інфраструктури [19]. Оскільки ЗНМ має дуже велику впливовість, навчання такої нейронної мережі часто вимагає великої кількості даних. Але в реальній практиці дорого отримувати позначені зображення, особливо для виявлення об'єктів, де збір охоплюючих рамок для кожного об'єкта в тренувальному наборі потребує значного зусилля людини.

ЗНМ декомпонує зображення на прості, повторювані елементи (наприклад, лінії, градієнтні переходи, запам'ятовувані деталі), які потім збираються в більш складні елементи, що можуть бути подальшим чином об'єднані в більш складні ознаки. Завдяки цьому процесу, з простих елементів нарешті формується складна конструкція, наприклад, людина або машина [3].

Основні принципи:

1. Збір великого набору даних та маркуванням області їх;
2. Попередня обробка включає в себе зміну розміру зображень, перетворення їх у градацію сірого та нормалізацію;
3. Архітектура ЗНМ зазвичай з кількох рівнів згортки та об’єднання, які слідують за якими йдуть рівні повного зв'язку та вихідний рівень.

Використання ЗНМ для виявлення номерних знаків може бути хорошою ідеєю, оскільки воно ефективне у багатьох випадках. Якщо великий набір даних про анотовані номерні знаки, згорткову нейронну мережу можна навчити їх точно виявляти [3].

Переваги:

1. Висока точність та надійсність;
2. Здатність виявляти номерні знаки при неповних кадрах або частково закритих іншими автомобільними транспортами;

Недоліки:

1. Необхідність у великих наборах даних для виявлення різноманітних номерів.
2. ЗНМ може бути обчислювальна дорогою для навчання, оскільки вимагає спеціалізованого обладнання, такого як графічні процесори.

Таблиця 2.1 – Порівняльний аналіз методів виявлення номерних знаків

Метод/критерій	Метод Віоли Джонсона	Метод щільності країв	Метод опорних векторів	Згорткові нейронні мережі
Точність	Помірна	Помірна	Висока	Висока
Складність тренування	Низька	Низька	Помірна	Висока
Стійкість до варіацій	Помірна	Помірна	Помірна	Висока
Розмірі даних	Помірна	Помірна	Висока	Висока
Адаптабельність	Обмежена	Обмежена	Обмежена	Висока
Видобуток ознак	Вручну	Основані на границях	Вручну	Автоматично вивчені

Порівняльне дослідження цих методів із використанням деяких стандартних параметрів, таких як: точність виявлення, складність тренування, стійкість до варіацій, обробка складних шрифтів, адаптабельність, додаткові вимоги за наявністю.

2.2 Методи фільтрації цифрових зображень

Розпізнавання номерних знаків (LPR) включає три етапи: попередня обробка, сегментація та розпізнавання символів (CR).

Шум зображення - це непередбачувану варіацію яскравості або кольору пікселів на цифровому відтворенні, які зазвичай виникають під час зйомки в умовах обмеженого освітлення(нічна зйомка), неідеального обладнання для формування зображення(відеокамера) або в результаті обробки зображення.

Шум може вплинути на якість зображення, зробивши його менш чітким та менш деталізованим, та спричинити неточності та спотворення на етапі сегментації або розпізнавання. Наприклад, можливо, що система сприймає шуми як окремі об'єкти, що може викликати негативний вплив на подальші дії.

Окремо можна зустріти різні типи шумів на зображенні, але іноді можна зустріти комбіновані шуми, які складаються з кількох типів шумів. Можна послідовно використовувати різноманітні фільтри або комбінувати їх, щоб посилити їх переваги та зменшити недоліки [15]. Також можна використовувати гібридні фільтри, що поєднують різні методи, щоб отримати більш ефективний результат.

При імпульсному шуму функція щільності розподілу ймовірностей (біполярного) визначається як:

$$p(z) = \begin{cases} P_a, & \text{при } z = a \\ P_b, & \text{при } z = b \\ 0, & \text{інші випадки} \end{cases} \qquad (2.2.1)$$

Якщо піксель b перевищує піксель a, то піксель із яскравістю b виглядає як світла крапка на зображенні, в той час як піксель із яскравістю a виглядає, як темна крапка. Для 8-бітових зображень це означає, що a рівне 0 (чорне) і b рівне 255 (біле).

Якщо хоч одна з значень ймовірності буде дорівнювати нулю, то вони називаються уніполярними.

Адаптивний шум є типом шуму, який може бути спричинений різними факторами, такими як електромагнітні перешкоди, низька якість обладнання або слабкий сигнал.

Адаптивний шум може варіюватися за своєю інтенсивністю, формою та частотою, що може спричинити погіршення якості зображення, включаючи зменшення різкості, контрастності та точності кольорів.

Моделювання зображення з адаптивним шумом має наступний вигляд:

$$g(x, y) = f(x, y) + \eta(x, y) \qquad (2.2.2)$$

Де $\eta(x, y)$ – адаптивний шум;

$f(x, y)$ – початкове зображення;

$g(x, y)$ – вихідне зображення.

Білий шум - це тип шуму, де сила сигналу розподілена рівномірно по всьому діапазону частот. Його різновидом є білий гаусівський шум, що може виникати при поганих умовах прийому сигналу [10]. Математична модель білого шуму є простою і включає в себе властивості як у частотній, так і у просторовій області. Ймовірнісна щільність розподілу для гаусівської випадкової величини z задається виразом:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(z-\mu)^2/2\sigma^2} \qquad (2.2.3)$$

Де z – значення яскравості;

μ – середнє значення випадкової величини z;

σ – її средньоквадратичне відхилення;

σ^2 – називається дисперсією величини z.

Основні принципи:

1. Випадкова величина не повинна мати однакового розподілу або розподілятися за нормальним законом;
2. Потрібна велика кількість випадкових змінних, як у випадку термічного шуму, де термальні вібрації великої кількості крихітних електронів впливають на результат;
3. Випадкові змінні повинні бути незалежними, з таким впливом, що їх вплив на сумарний ефект є незначним
4. Кожна змінна повинна мати незначний вплив на сумарний ефект.

Мінусом такого розподілу є те що не можна виразити у закритій формі кумулятивну функцію розподілу, але вона є табульованою.

Методи попередньої обробки досить різноманітні і залежать від задач досліджень. Мета фільтрації: покращення зображень, підвищення точності вимірювальної інформації. Алгоритми шумозаглушення зазвичай спеціалізуються на ефективному пригніченні певного виду шуму. На сьогоднішній день не існує універсальних фільтрів, які б детектували та пригнічували всі види шумів. Сьогодні існує велика кількість різних методів фільтрації зображень, які можуть відрізнятися як за основними принципами дії, так і за деталями в алгоритмах, що вдосконалюють кожен метод. Вибір найбільш ефективного методу може бути складним завданням. Однак, деяку кількість шумів можна досить точно наблизити до моделі білого гаусівського шуму, тому більшість алгоритмів спрямована саме на пригнічення цього виду шуму.

Найбільш часто використовуються такі фільтри для просторового шумозаглушення:

Медіанний фільтр є нелинійним методом цифрової фільтрації, який широко використовується для збереження контурів та зменшення впливу імпульсного шуму, шляхом збереження яскравісних перепадів. У цифровій обробці зображень медіанна фільтрація має широке застосування, оскільки вона зберігає границі зображення під час видалення шуму відповідно до відповідних умов. Медіанні фільтри ефективні, коли щільність шуму невелика. Завдяки своїй нелінійній природі, цей метод є вдалим в контексті оптимізації.

Медіаний фільтр радіусом r означає вибір радіусу r в межах $[-r; r]$. Метод медіанного фільтрування полягає у використанні медіани набору пікселів, що оточують певний піксель на зображенні. Цей метод аналізує кожен піксель разом з його сусідами, з метою визначення, чи він є типовим для свого оточення. Потім цей піксель замінюється медіаною значень з цього набору. Медіана обчислюється шляхом впорядкування всіх значень пікселів у наборі за зростанням і вибору значення, що знаходиться посередині. У випадку, коли кількість пікселів у наборі є парною, використовується середнє значення двох центральних пікселів[10].

Медіаний фільтр простий та точний, але розраховується відносно довго, оскільки потребує часткового сортування масивів яскравості оточуючих пікселів. Аби прискорити обчислення можна застосувати алгоритми швидкого сортування або навпаки не виконувати сортування, а обрати гістограму околиці точки.

Переваги:

1. Не потрібно генерувати нові значення пікселів.
2. Легко реалізовувати
3. За допомогою медіани, більш ефективно можна видалити значення, які є екстремальними і відхиляються від середнього значення.

Недоліки:

1. Медіанний фільтр не є добрим для всіх типів шуму.

Фільтр Гауса є інструментом розмиття зображення, який використовується, коли на зображенні присутні дрібні деталі, які не потребують відокремлення від фону і можуть бути розмиті. Однією з особливостей цього лінійного фільтру є його низькочастотна характеристика. Фільтр Гауса використовує нормальний розподіл ймовірностей для шуму або даних. Нормальний розподіл є прийнятною моделлю для вагового розподілу. Графічно нормальний розподіл схожий на дзвонову криву, де значення величин ближче до центру є більш імовірними [10].

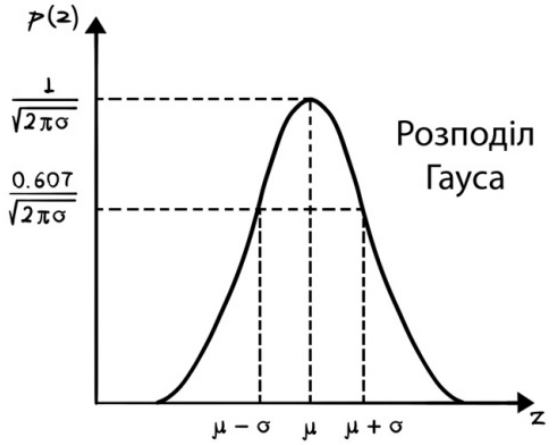


Рисунок 2.2 – Гаусове розподілення

Вплив розмиття Гауса часто досягається за допомогою ядра Гауса, значення якого конволюють зображення. При реалізації розмиття Гауса корисно скористатися властивістю розкладу на добуток двох одновимірних фільтрів для здійснення обробки у вертикальному та горизонтальному напрямках. Результат конвольювання з одновимірним ядром можна отримати за допомогою непарного прямого проходу, що потребує менше обчислень. Для дискретизації зазвичай використовують відбір зразків фільтра Гауса. На різних етапах цього процесу, ваги пікселів паралельні серединним точкам кожної складової. Хоча, при виборі зразків функції Гауса може виникати значна похибка. Точність функції Гауса в кожному пікселі простору є важливою.

Переваги:

1. Ефективне у видаленні гаусівського шуму;
2. Ваги надають більшого значення пікселям біля краю;
3. Обчислювально ефективний;
4. Має обертальну симетрію.

Недоліки:

1. Потребує багато часу на обчислення;
2. Зменшує деталізацію.

Метод головних компонент заснований на перетворенні Карунена-Лоєва. Даний метод є відносно новим та мало вивченим у сфері шумозаглушення та досягає найкращих результатів при обробці зображень з білим гаусівським шумом.

Початково використовуючись в статистиці, метод головних компонент (МГК) дозволяє скоротити розмірність ознак, зберігаючи при цьому значну кількість інформації. Його головна ідея полягає у пошуку базисних векторів у багатовимірному просторі, які найкращим чином відображають початкові дані. Основні принципи МГК включають:

1. За допомогою цього підходу можна виявити різні варіації в навчальній вибірці зображень номерних знаків і описати ці варіації у базисі кількох ортогональних векторів, які відомі як власні вектори.
2. Напрямок, вздовж якого розташовані дані, визначається вектором з найбільшим власним значенням, а зменшення розмірності початкових даних можливе шляхом використання векторів з найменшими власними значеннями.
3. Шляхом використання мінімальної кількості власних векторів можна отримати компактну апроксимацію вхідного зображення номерного знаку, яку можна зберігати у базі даних як ключ.

Отримані вектори за допомогою цього методу відповідають головним напрямкам візерунка (кордонів, деталей) на певній області зображення. Понадійно розподілений та непередбачуваний шум у зображенні відповідає власним векторам з

малими власними значеннями. З метою пригнічення шуму можна знизити амплітуди піксельних значень у даному напрямку, що сприятиме зменшенню його впливу. Тому метод головних компонент ефективний тільки при роботі з білим шумом: шум не повинен мати ніякої структури, інакше МГК його сприйме як візерунок і не зможе його придушити [11].

Переваги:

1. Зменшує розмірність даних;
2. Видаляє нерелевантні зображення.

Недоліки:

1. Працює тільки з білим шумом, адже МГК прийме його за візерунок;
2. При різних змінах, особливо з освітленням, точність падає.

2.3 Методи сегментування символів

У сучасний період, при розв'язанні багатьох завдань часто використовуються зображення, отримані оптико-електронними системами, щоб провести якісну обробку цих зображень необхідно розбити їх на елементи, або здійснити сегментування.

Сегментація – це процес розбиття цифрового зображення на його складові частини або об'єкти, в залежності від конкретного завдання, і вона має велике значення в обробці зображень. Цей етап створений з метою покращення точності розпізнавання символів та об'єктів, зменшення впливу фонового шуму та підвищення продуктивності шляхом зменшення кількості даних, що потребують обробки. Розділивши зображення на сегменти, можна ефективніше розпізнавати виділені об'єкти.

Сегментація зображення може відбуватися за допомогою різних ознак, таких як яскравість, координати, текстура, форма, глибина та характер руху, які можуть бути використані як роздільні ознаки. Алгоритми сегментації можуть використовувати розривність, яка базується на різких перепадах значень яскравості, або однорідність, яка розділяє зображення на однорідні області з використанням заданих критеріїв. Результатом сегментації є множина сегментів або контурів, які відображають всі області зображення [18].

Існує кілька методів сегментації зображень, також відомих як кластеризація, які включають методи на основі знаходження границь, порогові, регіонні, на нейронних мережах, кластеризаційні та гібридні методи.

Один з найпростіших і найефективніших методів - це порогова сегментація, де зображення ділиться на два класи за певним пороговим значенням. Всі пікселі, значення яких менше порогу, належать до одного класу, а все, що більше, - до

іншого. Важливо правильно вибрати порогове значення для отримання задовільного результату.

Метод сегментації на основі виявлення країв зображення полягає у пошуку однотипних ділянок та виявленні зон різкої зміни параметрів, які позначаються як краї. Далі ділянки країв об'єднуються, щоб утворити замкнену криву, по якій відбувається остаточне розділення зображення. Цей метод застосовується не тільки до відеозображень, але і до окремих фотографій.

При регіональній сегментації зображення так само, як і при виявленні країв, знаходяться однотипні ділянки зображення. Проте, на відміну від виявлення країв, у цьому методі визначається належність кожного пікселя до конкретного регіону на основі його параметрів. Після цього регіони об'єднуються, щоб утворити окремі області на зображенні.

Порогова сегментація є ключовим методом у задачі сегментації. Цей алгоритм також можна назвати методом за яскравістю, оскільки він використовує порогове значення яскравості для розділення пікселів на об'єкт та фон. Метод перетинів застосовується, коли на зображенні присутні кілька об'єктів, тому потрібно встановити не одне порогове значення, щоб виконати сегментацію пікселів, які мають яскравість в межах визначених значень.

Щоб належним чином розділити зображення на об'єкт і фон, важливо налаштувати поріг правильно. Наприклад, якщо встановити поріг на рівні 50, враховуючи, що значення пікселів зображення лежать в діапазоні від 0 до 255, тоді всі пікселі зі значеннями, меншими або рівними 50, будуть встановлені на 0 (чорний), а всі пікселі зі значеннями більше 50 будуть встановлені на 255 (білий) [20].

Використання бімодального зображення є необхідним при визначенні автоматичного порогового значеннякоторое має два виділені піки інтенсивності на графіку розподілу для об'єкту та фону.Недоліком цього типу порогу є є обмежена ефективність при низькій якості освітлення зображення.

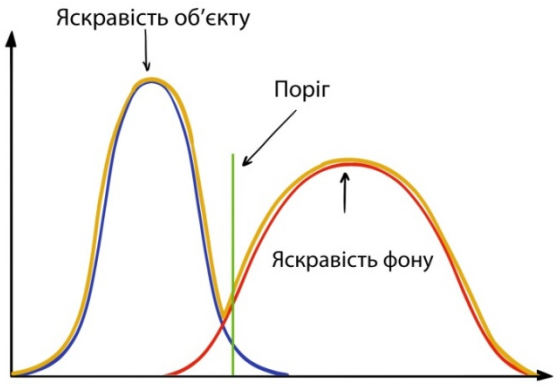


Рисунок 2.3 – Графічне представлення порогової сегментації

При самостійному визначенню порогового значення точно оцінити кількісні властивості об'єкта. Взагалі, цей підхід може бути оптимальним, при суттєвій різниці параметрів об'єктів на зображенні і параметри фону. Тому для визначення фону проводяться повторні експерименти з метою отримання оптимального результату, який відповідає потребам користувача.

Адаптивне порогове значення використовується для зменшення впливу ефекту освітлення на зображенні [20]. Цей метод включає розбиття зображення на різні субрегіони, кожен з яких сегментується з використанням свого порогового значення. За допомогою об'єднання окремих областей можна отримати завершене сегментоване зображення, що допомагає зменшити ефект освітлення до певної міри.Тобто сегментація таких частин відбувається враховуючи особливості освітлення кожної з них. Мінімізація неправильно класифікованих пікселей під час сегментації можлива завдяки використанню оптимальному пороговому значенню.

Переваги:

1. Простий та ефективний
2. Не вимагає попередньої інформації про зображення

Недоліки:

1. Низька ефективність роботи з слабо вираженою контрастністю зображеннями, при нерівномірному фоні
2. Складність у визначенні порогу в залежності від способу.

Основними етапами методу виділення границь в контурній сегментації є:

1. За наявності шумів відбувається фільтрація;
2. Акцентування в місцях з перепадами яскравості;
3. Додавання точок до границь при пороговому значенню;
4. Визначення місця розташування та напрямку.

Наприклад, алгоритм Кенні, що є одним із методів, складається з чотирьох фільтри для обчислення градієнта інтенсивності зображення в горизонтальному, вертикальному та діагональних напрямках [7]. Метод Кенні визначається низьким рівнем помилок, оскільки застосовується подвійний поріг, де верхнє та нижнє значення визначаються емпірично. Такий підхід дозволяє зменшити кількість помилково виявлених країв.

Отримане зображення, за алгоритмом Собеля, виділяє області з високими значеннями градієнта, які зазвичай відповідають контурам вихідного зображення.

Якщо розглядати роботу оператора Собеля, то можна виділити два основні моменти:

1. При застосуванні даного диференціального оператора в точці з постійною яскравістю, то результатом буде нульовий вектор.
2. Якщо точка лежить на межі областей з різною яскравістю, то вектор, що перетинає межу, буде спрямований в напрямку збільшення яскравості.

Нехай G_x і G_y – зображення, що в кожній точці містять наближення горизонтальної та вертикальної похідних, та A – початкове зображення, тоді наступні обчислення можна зобразити так:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A \qquad (2.3.1)$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \qquad (2.3.2)$$

Градієнт розраховується за формулою:

$$G = \sqrt{G_x{}^2 + G_y{}^2} \qquad (2.3.3)$$

Тобто, оператор Собеля згортає зображення з маленьким ядром для обчислення градієнта в кожному пікселі. Отримане зображення виділяє області з високими значеннями градієнта, які зазвичай відповідають контурам вихідного зображення.

Отже, існують методи порівняння та диференціально-градієнтні. Спільність у визначенні границь об’єкта при достатньо високому градієнту, в той час як різниця заключається у визначенні напрямку та оцінки градієнтного значення.

Переваги:

1. Результуюча інформація займає небагато місця.

Недоліки:

1. Використання методу тільки тоді, коли границі є достатньо чіткими;
2. Велика обчислювальна складність;
3. Проблеми при роботі з шумами.

Метод К-середніх відноситься до класу методів сегментування, що засноване на кластеризації та є одним з найбільш уживаних методів, який включає розділення набору даних на певну кількість груп.

Алгоритм розділяє зображення, попередньо перетворивши його в градації сірого, на К кластерів. Ідея полягає в тому, аби присвоїти об'єкт такому класу, який є найбільш розповсюдженим серед сусідів даного елементу. Існує багато методів для визначення відстані до класу (центру класу), проте найбільш поширеним залишається метод обчислення Евклідової відстані. Основні принципи:

1. Визначаємо число найближчих сусідів К, та обираємо їх випадковим чином, що будуть центрами шуканих кластерів.
2. Розраховуємо Евклідову відстань до кожного пікселя.

$$d_k^i = \left\| p^i - c_k \right\| \quad (2.3.4)$$

3. Розподіл елементів за принципом близькості до кожного елементу.
4. Обчислити та визначити новий центр кожного кластера

$$c_k = \frac{1}{|k|} \sum_{y \in c_k} \sum_{x \in c_k} p(x, y) \quad (2.3.5)$$

- c_k – центр кластеру k;
- $|k|$ – кількість пікселів, які відносяться до кластеур k;
- $p(x, y)$ – інтенсивність в точці (x,y).

5. Процес повторюється до тих пір, поки не досягнемо умови зупинки, тобто поки зміна центрів кластерів у порівнянні з обчисленим на попередній ітерації не стане досить малою.

$$\left| c_k^t - c_k^{t-1} \right| < \varepsilon \quad (2.3.6)$$

Де ε досить мала величина.

Мета – мінімізація суми відстаней між усіма точками та центром кластера:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \quad (2.3.7)$$

- k– кількість кластерів;
- n – кількість випадків;
- c – центр кластера;
- $\left\| x_i^{(j)} - c_j \right\|^2$ – відстань.

Вибір оптимального значення K дозволяє забезпечити гармонійну рівновагу між максимальним стисненням даних, отриманим за допомогою одного кластера, та досягненням найвищої точності шляхом привласнення кожній точці даних окремому кластеру. [4].

Переваги:

1. Гарантує збіжність
2. Легкий в реалізації
3. Легко адаптується до великих наборів даних, тобто можлива будь-яка кількість кластерів та ознак
4. Стійкість до аномальних викидів

Недоліки:

1. Помилки виникають, якщо дані складаються з кластерів різної форми, розмірності та щільності
2. Усе кластеризується, але це може бути плюсом, якщо необхідно виділяти усі області.
3. Існують параметри, що потребують ручного введення (ітерація, максимальне значення).

Рекурентний метод з використанням гістограм є ефективним підходом до сегментації:

1. Щоб виділити більш яскраві області від більш темних, можна застосувати порогове обмеження зображення, використовуючи мінімальне значення між модами гістограми яскравості.
2. Для кожної сегментованої частини створюється гістограма і отримані сегментами піддаються пороговому обмеженню, якщо гістограма неунімодальна.
3. Процес продовження процесу залежить від досягнення унімодальності для усіх сегментованих гістограм та рівня сегментації, який прагнемо досягти.

Переваги:

1. Вимагають тільки один прохід по пікселям
2. Мінімуми та максимуми використовуються для знаходження кластерів

Недоліки:

1. Важко знайти значні мінімуми та максимуми

Таблиця 2.3 – Експериментальні аналізи

Параметр	Пороговий метод	Метод виділення границь	Метод К-середніх	Метод з використанням гістограм
Швидкість	Висока	Середня	Низька	Середня
Шумостійкість	Низька	Середня	Висока	Висока
Точність	Висока	Висока	Висока	Середня
Складність обчислення	Низька	Низька	Середня	Низька

Порівняльне дослідження цих методів із використанням деяких стандартних параметрів, таких як: швидкість, складність обчислень, шумостійкість точність, проведено для методів на основі порогів, методів на основі виділення границь, К-середніх та метод з використанням гістограм.

Експериментальні результати показують, що:

- Якщо потрібна висока швидкість, пороговий метод може бути найкращим варіантом.
- Якщо треба метод з високою шумостійкістю, можливо, краще використовувати методи виділення границь або метод з використанням гістограм.
- Якщо необхідність у високій точності, метод виділення границь або метод К-середніх можуть бути більш підходящими варіантами.
- Якщо важлива низька складність обчислень, пороговий метод або метод з використанням гістограм можуть бути найкращими виборами.

2.4 Методи розпізнавання символів

Оптичне розпізнавання символів

Для визначення символів, які зображені на номерному знаку, використовується технологія оптичного розпізнавання тексту. Після розпізнавання, символи зберігаються у вигляді тексту у тому ж порядку, як на знаку.

Оптичне розпізнавання символів (OCR) – це технологія, що дозволяє комп'ютерам читати текст зображень. У випадку розпізнавання номерних знаків, програмне забезпечення OCR використовується для сканування зображень номерних знаків та отримання з них текстової інформації. Ця інформація може бути використана для різних цілей, наприклад, для ідентифікації транспортного засобу або його власника.

Існують два основні типи OCR: традиційний і глибокого навчання. Традиційні алгоритми OCR базуються на порівнянні шаблонів та евристиці, тоді як алгоритми OCR глибокого навчання засновані на нейронних мережах. Традиційні алгоритми OCR зазвичай менш точні, ніж алгоритми глибокого навчання, але вони також працюють швидше та вимагають менше обчислювальної потужності [1].

Зазвичай алгоритми OCR глибокого навчання навчаються на великих наборах позначених зображень. Процес навчання означає, що алгоритму надається велика кількість зображень номерних знаків разом з відповідними текстовими даними для кожного зображення (наприклад, марка і модель транспортного засобу). Потім алгоритм навчається розпізнавати патерни в зображеннях, що відповідають текстовим даним.

Після того, як алгоритм OCR навчений, його можна використовувати для сканування нових зображень номерних знаків та отримання з них текстової інформації.

Переваги:

- відсутність потреби вручну вводити дані.

- для автоматичного читання та оновлення баз даних. Це означає, що ваші записи завжди будуть актуальними, що є важливим для безпеки та ефективності.

Недоліки:

- у складнощах точного читання літер та цифр на номерному знаку, особливо якщо номерний знак брудний або погане освітлення
- не настільки вдосконалена, щоб читати всі типи номерних знаків, тому деякі з них можуть бути нечитабельними для системи

Морфологічна скелетизація

У процесі скелетизації, лінії на бінарному зображенні зводяться до мінімальної товщини одного пікселя, щоб отримати скелет. У задачах розпізнавання символів, збереження ключової інформації про графічні елементи вихідного символу під час скелетизації має велике значення. За методи, що використовують скелетизоване зображення, важливо уникнути випадкового пропуску жодного пікселя, який можна видалити без порушення топології графічного зображення, під час роботи алгоритму.

Тобто, морфологічна скелетизація це контурне (або медіальна вісь) зображення форми або бінарного зображення, створене за допомогою морфологічних операторів. Морфологічні скелети поділяються на два види:

- Ті, що визначаються за допомогою морфологічних виїмок, з яких можна модернізувати початковий символ
- Ті, що розраховуються за допомогою трансформації "попадання-промаху",які зберігають топологію форми.

Аналіз порожнин є ефективним, оскільки дозволяє класифікувати початкову групу, складену з 26 літер та 10 чисел, на три групи: елементи з двома, однією або без порожнин. Для виконання цієї задачі необхідно створити два алгоритми: один для виявлення порожнин, а другий - для підрахунку їх кількості. За допомогою аналізу порожнин легко впізнати символи "B" та "8", які мають дві порожнини.

Символи з однією порожниною складаються з 6 літер та 4 чисел, а символи з усіма порожнинами - з 19 літер та 5 чисел. Завершення аналізу полягає в визначенні кінцевих точок символу.

Отримання структурних елементів на скелетизованому бінарному зображенні є значно простішим у порівнянні з будь-яким іншим бінарним зображенням, оскільки такі зображення легше аналізувати. Це можливо завдяки можливості формулювати правила для визначення найбільш важливих елементів графічного представлення символу. Такий підхід дозволяє розробляти алгоритми для ефективного отримання цих елементів.

Структура включає ключові точки символу, які відображаються чорними пікселями на скелетизованому бінарному зображенні. Ці точки є важливими елементами структурної моделі символу. З'єднуючі ребра об'єднують дві ключові точки і описують послідовність чорних пікселів на скелетизованому зображенні. Ключова точка також може бути місцем стику двох з'єднуючих ребер, які мають кут менше 120 градусів. Вигин - це місце на скелетизованому зображенні, де лінія, що з'єднує дві ключові точки, різко змінює напрямок. Композитне ребро - це послідовність з'єднуючих ребер, яка починається і закінчується в ключових точках і не містить інших ключових точок, крім початкової і кінцевої. Чорні пікселі, які не є ключовими точками або вигинами, об'єднуються в з'єднуючі ребра, які, в свою чергу, утворюють композитні ребра [8].

SOM

Тево Кохонен вперше запропонував модель самоорганізаційної карти (SOM) в 1980 році. Вона створює низькомірне представлення простору введення номерних знаків. Кохоненська нейронна мережа використовує алгоритм SOM для навчання, де нейрони повинні конкурувати, щоб стати переможцем на карті шару. Таким чином, мінімальна евклідова відстань використовується для визначення переможця. У цьому дослідженні, гаусівська функція використовується для зменшення швидкості навчання та розміру оточення з часом. Для ітерації застосовується константа за замовчуванням на основі 1000 разів.

У самоорганізуючійся карті Кохонена кожен клас може містити кілька відповідних зображень. Ця карта автоматично розподіляє бібліотечні зображення на класи шляхом використання вагових коефіцієнтів, які зв'язують топографічні нейрони. Кількість класів є зовнішнім параметром, який залежить від необхідної точності кластеризації набору зображень з бібліотеки. У такій карті близькі нейрони в топографічному шарі відповідають схожим вхідним зображенням. Це дозволяє створювати топографічні карти Кохонена, які використовуються для візуалізації багатовимірних даних [2].

Метод послідовних наближень використовується для здійснення навчання мережі Кохонена. Спочатку матриця вагових коефіцієнтів ініціалізується випадковими значеннями, а потім навчальні вектори поступово подаються на вхід мережі. Для кожного вектора обчислюється відстань до кластерного елементу, і знаходиться переможний нейрон, у якого відстань є найменшою. Потім вагові коефіцієнти переможного нейрона та його сусідніх нейронів змінюються з урахуванням радіуса навчання. Під час навчання з кожною новою епохою поступово зменшується радіус навчання, і в кінці навчання лише один переможний нейрон отримує навички. Обсяг навчального набору повинен бути не менше, ніж у 5-10 разів більший за кількість вхідних параметрів і може містити помилки (шум).

Зазвичай, в розпізнаванні символів використовуються два методи: шаблонний пошук та нейронні мережі. Нейронна мережа навчається на великій кількості зразків символів. Шаблонний пошук потребує бібліотеки з великою кількістю різноманітних шрифтів та товщин, тому не є дуже практичним. SOM має застосування в широкому спектрі областей застосування, таких як оптичне розпізнавання символів. SOM - це процес, який буде виробляти подібні виходи для подібних вхідних даних. Звичайний SOM має наступні два шари: 1) вхідний шар і 2) обчислювальний шар [5]. Обчислювальний шар містить обчислювальні одиниці. Матриця ваг SOM обчислюється під час фази навчання. Апаратний засіб обчислює гамінгову відстань між матрицею ваг кожного нейрона та вхідним зображенням і приймає рішення щодо вихідного символу.

Таблиця 2.4 – Порівняльний аналіз методів розпізнавання символів

Метод	Оптичне розпізнавання символів	Морфологічна скелетизація	Самоорганізуючі карти
Точність	Висока	Помірна	Помірна до високої
Складність тренування	Помірна	Низька	Висока
Стійкість до варіацій	Висока	Помірна	Помірна
Розмір даних	Великий	Малий	Помірний
Обробка складних шрифтів	Так	Ні	Ні
Адаптабельність	Висока	Низька	Помірна
Додаткові вимоги	Маркування даних	Вхідні бінарні зображення	Структура топологічної мапи

Порівняльне дослідження цих методів із використанням деяких стандартних параметрів, таких як: точність розпізнавання тексту, складність тренування, стійкість до варіацій, розмір датасету, обробка складних шрифтів, адаптабельність, додаткові вимоги за наявністю.

2.5 Огляд комерційних рішень розпізнавання номерних знаків

1) Hikvision ANPR solutions

Рішення Hikvision ANPR є інтелектуальними системами, призначеними для визнання та обробки номерних знаків за допомогою передових технологій. Однією з ключових особливостей Hikvision ANPR є технологія низького освітлення DarkFighter, яка дозволяє точно зафіксувати номерні знаки навіть за умов поганого освітлення. Крім того, Blazer Express Станція iVMS є централізованою системою управління, яка надає повне рішення для управління автомобілями з простими елементами керування, такими як вікна перегляду в реальному часі, автоматичне керування бар'єром, спрацьовування тривог та просте звітування.

Технологія Hikvision ANPR здатна витягувати номерні знаки зі складного фону, відокремлювати та розпізнавати кожен символ на пластині та переформатувати пластину інформації. Ця технологія використовує визначення місцезнаходження пластини, розділення символів та розпізнавання символів для досягнення точного визнання номерних знаків. Hikvision розробила алгоритм, який поєднує вертикальну проекцію та CCL (Connected Component Labeling) для приблизного розділення символів, а також метод тонкого розділення символів та спеціальну стратегію для ідентифікації всіх меж символів на багатосекційних пластинах [9]. Отже, алгоритм Hikvision для розділення символів є придатним для глобальних застосувань.

2) AXIS License Plate Verifier

Аналітичне програмне забезпечення може автоматично розпізнавати номерні знаки в режимі реального часу, порівнювати їх з попередньо визначеним списком, додавати їх до списку або ініціювати різні дії, такі як відкриття воріт, нарахування платежу або подачу сигналу тривоги. Залежно від умов освітлення, можуть

знадобитися камери з вбудованим інфрачервоним підсвічуванням або додаткові джерела світла для оптимальної роботи системи.

Таке розпізнавання номерних знаків може бути корисним для повільного руху та контролю доступу транспортних засобів в таких місцях, як автостоянки, центри міст і закриті селища. Це дозволяє розумно зчитувати номерні знаки зі швидкістю до 70 км/год. Легке адміністрування дозволених і блокованих списків зберігає дорогоцінний час. За рекомендованими найкращими умовами, мінімальна ширина номерного знаку повинна складати 130 пікселів [2]. Для оптимальної роботи системи можуть знадобитися додаткові джерела світла або камери з вбудованим інфрачервоним підсвічуванням в залежності від умов освітлення.

3) Dahua Technology ANPR solution

Розробка Dahua ANPR включає в себе камери Dahua та програмне забезпечення, що може працювати на камері або на сервері. Це рішення автоматично визначає номерні знаки в режимі реального часу, застосовуючи велику кількість зображень та високоякісні моделі глибокого навчання [6]. ANPR підтримує такі функції:

- Розпізнавання номерних знаків на різних мовах.
- Висока точність розпізнавання
- Швидке визначення - процес займає близько 100 мілісекунд.
- Економічна технологія з відмінним співвідношенням ціни та продуктивності.
- ANPR підтримує номерні знаки з більш ніж 58 країн.
- Щоб отримати оптимальний результат, ширина номерного знаку повинна бути від 70 до 250 пікселів, в залежності від вимог країни, наприклад, в ЄС - від 130 до 250 пікселів, а в СНД від 150 до 250 пікселів.

4) Hanwha Techwin ANPR solution

Інноваційна система розпізнавання номерних знаків Hanwha Techwin ANPR використовує технологію IP-зв’язку між камерами для одночасного захоплення

та передачі даних до зручного інтерфейсу користувача. Це безсерверне рішення, що може використовуватись до чотирьох камер Wisenet ANPR, забезпечує точність розпізнавання номерних знаків на рухомих транспортних засобах, що рухаються зі швидкістю до 50 км/год, з точністю в 95% [8]. Система підходить для автостоянок, автозаправних станцій, малих житлових комплексів з кількома в'їздами/виїздами, шлагбаумів та доріг.

Крім того, безсерверне рішення ANPR дозволяє автоматично контролювати рух до 1000 автомобілів з «білих списків» через релейні виходи камери без необхідності встановлення та запуску програми на сервері. Для більшої гнучкості API доступний для інтеграції з програмним забезпеченням та системами сторонніх виробників за допомогою системних інтеграторів.

5) SecurOS

Рішення SecurOS Auto License Plate Recognition було розроблено з використанням передової технології штучної нейронної мережі ISS у поєднанні з технологією Intel Open Vino. Завдяки більш ніж 20-річному досвіду роботи з цим рішенням, воно розроблено для роботи незалежно або як модуль інтелектуальної відеоаналітики для систем керування відео ISS SecurOS, що забезпечує розпізнавання номерних знаків і порівняльний аналіз для програм корпоративного рівня.

Додатковою функцією, яка відрізняє SecurOS від конкурентів, є точність визначення кольору, марки та моделі споживчих транспортних засобів. Незалежний дослідник системи відеоспостереження IPVM виявив, що SecurOS точно розпізнає приблизно 500 автомобілів, вантажівок і позашляховиків протягом тижня польових випробувань, деякі з них за поганих погодних умов. Марка/модель/колір розпізнається лише після того, як спрацьовує виявлення номерного знака, і SecurOS працював ефективно, незважаючи на складні горизонтальні кути. Ця функція потребує встановлення та налаштування додаткових модулів, які зменшують виявлення додаткових написів на транспортних засобах, окрім самих номерних знаків [16].

2.6 Висновки до розділу

Успішне розпізнавання автомобільних номерів є складним завданням, що вимагає урахування різноманітних факторів, таких як освітлення, розмір та форма номерного знаку. Розробка універсальних алгоритмів стикається з викликами в цьому процесі. Одним з важливих етапів є фільтрація зображень від шумів, що відбувається між детекцією номерного знаку та сегментацією символів. Для досягнення кращих результатів може бути використана комбінація різних фільтрів.

Сегментація зображень є важливою складовою комп'ютерного зору, і вона знайшла застосування в багатьох галузях бізнесу. Наприклад, технології сегментації зображень використовуються для розпізнавання обличчя на вашому телефоні та в передових системах безпеки, що дозволяє ідентифікувати людину. Системи дорожнього руху використовують алгоритми сегментації зображень для розпізнавання номерних знаків.

Отже, кожен метод сегментації має своє застосування, яке залежить від вхідних даних, вимог до розпізнавання, об’ємів обчислювань, швидкості прийняття рішення та інших факторів. Зважаючи на переваги та недоліки усіх методів, можна стверджувати, що ідеального методу сегментації не існує. Вибір конкретного методу сегментації має залежати від вимог та потреб конкретної задачі.

3 Математичне забезпечення

3.1 Актуальність системи детекції та розпізнавання номерних знаків

Протягом останнього десятиліття світ пережив економічний бум, що призвів до значного зростання урбанізації та збільшення числа приватних автомобілів і відповідно, реєстраційних номерних знаків. Це створило проблему ідентифікації конкретного автомобіля серед багатьох. Завдання розпізнавання номерних знаків є складним, оскільки воно залежить від різних зовнішніх факторів.

Виділення області номерного знака є важливою складовою інтелектуальних транспортних систем, оскільки всі країни світу прийняли єдиний спосіб ідентифікації транспортних засобів за допомогою реєстраційних номерних знаків. Розмір номерного знака може варіюватися від автомобіля до автомобіля або від країни до країни. Для ідентифікації розміру символів необхідна сегментація зображення в інтелектуальній транспортній системі.

Кожна країна має свій власний метод видавання номерних знаків транспортним засобам, тому для правильної ідентифікації кожна країна потребуватиме унікального підходу до розпізнавання цих номерних знаків. Це допоможе зменшити хаос у процесі ідентифікації транспортних засобів. Розпізнавання номерних знаків є ефективним у різних областях застосування:

- Платні автостоянки/дороги: автоматизація прийому оплати, в'їзду та виїзду
- Організація руху в аеропортах: допуск лише уповноважених транспортних засобів на смуги для таксі та громадського транспорту
- Здійснювати контроль доступу шляхом відкриття воріт для автомобілів, що мають дозвіл, та автоматична реєстрація незареєстрованих автомобілів.
- Впізнання автомобілів: автоматичне оповіщення під час проїзду автомобіля, внесеного до контрольного списку

3.2 Модель системи. Опис функціональності системи

Структурне представлення системи у вигляді класів та зв’язків між ними:

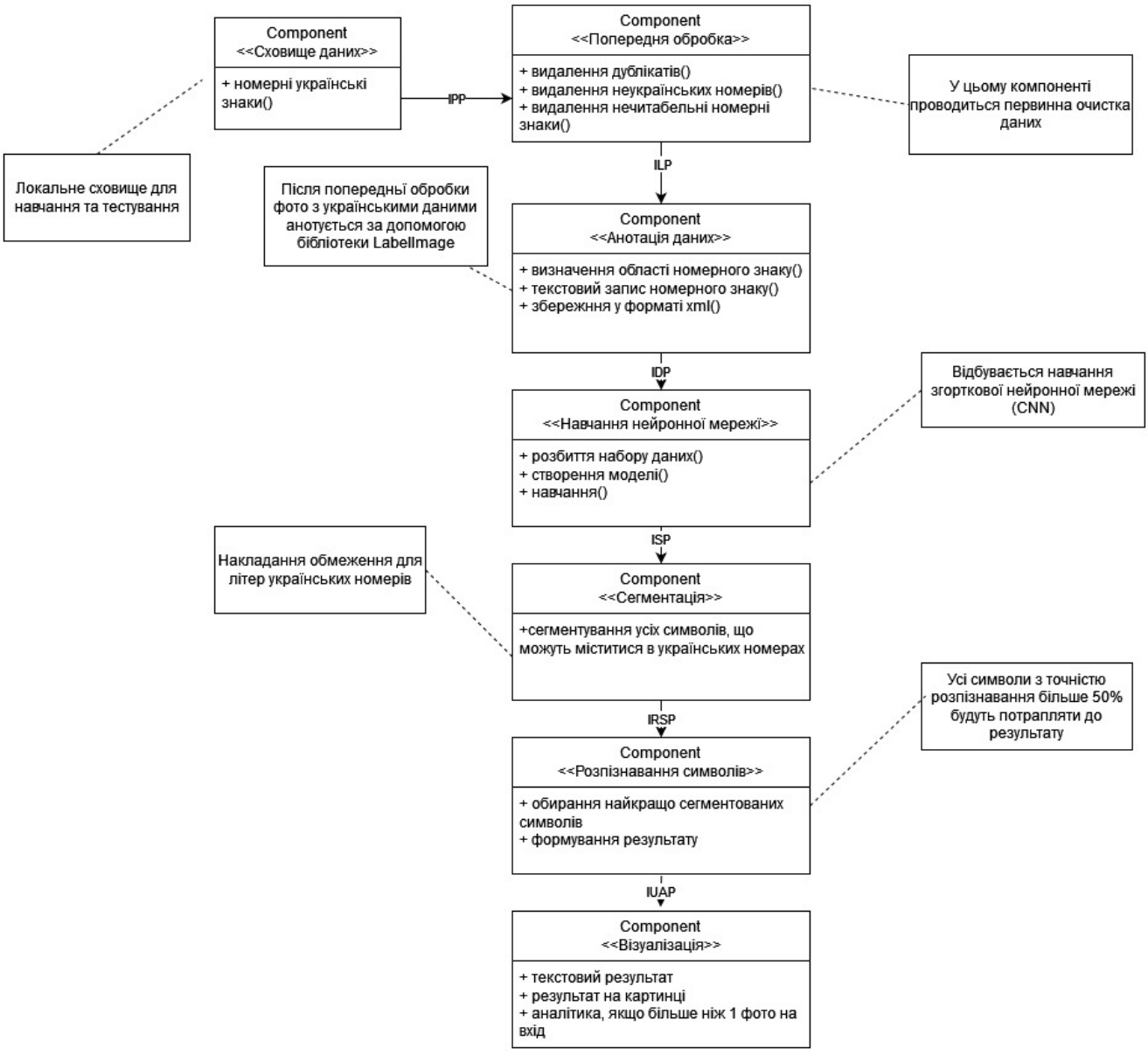


Рисунок 3.1 – Модель системи. Діаграма компонентів у нотації UML

Компонент "Сховище даних" призначений за зберігання наборів даних для навчання та тренування українських номерних знаків. У даному випадку сховище даних буде локальним, тобто розташовано на комп'ютері, на якому працює система.

Зображення для навчання моделі детекції номерних знаків можуть бути зібрані з різних джерел. У даному проекті зображення були взяті з відкритого джерела platesmania.com [14]. Цей сайт містить велику кількість фотографій автомобілів з номерними знаками з різних країн світу, але для навчання обирались номера з України у форматі jpg.

Навчальні дані використовуються для навчання моделі детекції та розпізнавання номерних знаків, а тестові дані - для перевірки точності роботи системи. Кількість зображень для навчання моделі складає близько 3000. Це достатня кількість зображень для тренування моделі, якщо дані мають якісну анотацію та репрезентують різні варіації номерних знаків, такі як кольори, шрифти, розміри, а також різні умови освітлення та кути зйомки. Для досягнення кращих результатів можна додати більше зображень до сховища даних та провести додаткову обробку та аугментацію даних.

Компонент "Сховище даних" повинен забезпечувати зручний та швидкий доступ до даних та забезпечувати захист даних від несанкціонованого доступу або втрати даних в результаті помилок чи випадкових втрат.

Компонент "попередня обробка" в системі детекції та розпізнавання номерних знаків відповідає за видалення непотрібної, зайвої для розв'язання задачі інформації з набору даних. Це включає в себе наступні етапи:

1. Видалення дублікатів у наборі даних допомагає зменшити розмір набору даних і покращити ефективність системи.
2. Виключити з набору даних зображення з номерами з інших країн або з іншими символами та залишити тільки українські номери.

3. Видалення нечітких номерних знаків тільки тих, які важко або неможливо прочитати або мають низьку якість. Даний етап відсіювання допомагає збільшити точність виявлення та розпізнавання номерних знаків.

Після виконання цих етапів набір даних буде містити лише українські номерні знаки, які можна прочитати та які потрібні для навчання моделі детекції та розпізнавання.

Компонент «Анотації даних» необхідний для створення набору даних для навчання. Для анотації даних зазвичай використовують спеціальні інструменти, такі як LabelImage.

LabelImage – відкритий API для анотації зображень [12]. Для даної системи це означає виділення області номерного знаку та позначення міткою, що це номерний знак. Вирішено замість тегу записувати сам номер, що дасть зручність у майбутньому для перевірки точності розпізнавання символів

Після анотації даних LabelImage зазвичай зберігає марковані зображення у форматі xml, який містить координати областей міток, що стосуються номерних знаків на зображеннях. Вони використовуються в наступних етапах системи, таких як компонент навчання нейронної мережі, для створення моделі та тренування.

Компонент «Навчання нейронної мережі» є важливою складовою системи детекції та розпізнавання номерних знаків. Його головна мета полягає в тому, щоб створити нейронну мережу, яка буде вірно класифікувати номерні знаки. Для локалізації області номерного знаку було прийнято рішення використовувати ЗНМ. Згорткова нейронна мережа може швидко обробляти вхідні зображення та виявляти в них патерни з високим рівнем точності.

Для навчання нейронної мережі використовуються набори даних, які були зібрані та попередньо оброблені. Кожен зображення номерного знаку має бути анотований для того, щоб навчальна мережа могла визначати, де саме зображений номерний знак та які символи в ньому містяться.

Після того, як дані були попередньо оброблені та анотовані, можна розпочати процес навчання. Для цього створюється архітектура нейронної мережі, в якій визначається кількість шарів та їх розміри.

Далі, навчальні дані піддаються процесу тренування, під час якого відбувається пошук оптимальних значень вагів мережі, що дозволяє мінімізувати помилки класифікації. Процес тренування зазвичай займає досить багато часу, але результати навчання дозволяють досягти високої точності розпізнавання номерних знаків. Після того, як нейронна мережа була навчена, її можна використовувати для розпізнавання номерних знаків в реальному часі.

Компонент «сегментації» в системі детекції та розпізнавання номерних знаків відповідає за виділення областей, де знаходяться символи номерного знаку. Цей компонент використовується для покращення якості розпізнавання символів та зменшення кількості помилок при розпізнаванні. Ідея полягає в тому, що сегментація дозволяє розділити зображення на окремі регіони, які містять символи, і використовувати ці області для подальшого розпізнавання. Далі сегментовані символи передаються на розпізнавання.

Компонент «Розпізнавання символів» призначений для розпізнавання найкращих сегментованих символів, отриманих з попередньої фази сегментації.

Для розпізнавання символів можна використовувати різні алгоритми, такі як машинне навчання, шаблонне відповідання, нейронні мережі тощо.

Визначення кожного символу, розділеного текстовим форматом, є останнім етапом розпізнавання номерного знака, який здійснюється за допомогою OCR Tesseract. Оптичне розпізнавання символів (OCR) - це метод, який дозволяє програмі ідентифікувати тексти або слова, що були написані в усній формі користувачами, без потреби людської участі. У галузі збору шаблонів та ШІ інтелекту, оптичне розпізнавання символів є одним з найбільш успішних застосувань цієї технології.

Tesseract – це відкрита бібліотека для OCR, яка розпізнає текст на зображеннях, що підтримує багато мов. Tesseract використовує нейронну мережу

для розпізнавання тексту на зображеннях, тому вона може розпізнавати текст набагато краще, ніж методи шаблоного відповідання.

Tesseract відомий своєю винятковою точністю та надійністю у розпізнаванні символів на номерних знаках. Завдяки процесу навчання, Tesseract вивчає ідентифікувати складні символи з вражаючою точністю, мінімізуючи помилки та хибні спрацювання. Здатність системи працювати в складних ситуаціях, таких як недостатнє освітлення, негативні погодні умови та часткове закриття, гарантує надійну продуктивність у реальних умовах.

Компонент розпізнавання символів приймає на вхід зображення, в якому виділені окремі символи, і використовує Tesseract OCR для розпізнавання тексту. Результат розпізнавання повертається у вигляді текстового рядка.

Після розпізнавання символів, текст можна додатково обробляти, наприклад, видаляти помилки та зайві символи, перевіряти чи отриманий текст відповідає формату номерного знаку тощо. Блок пост-аналітики допоможить покращити результат, наприклад розпізнався «0» замість «О», або «В» замість «8» і навпаки, але цей блок буде відпрацьовувати за правилами.

Компонент «Візуалізації» відповідає за представлення результатів розпізнавання номерних знаків у зручному та доступному форматі для користувача.

Для забезпечення зручного графічного інтерфейсу а інтуїтивно зрозумілого способу взаємодії з системою розпізнавання номерних знаків транспортних засобів, було розроблено веб-додаток. Цей додаток написано на мові програмування Python та використовує потужний фреймворк Flask для створення веб-інтерфейсу. Розробка проводилась у зручному середовищі Visual Studio Code через зручні інструменти для написання, відлагодження та тестування програмного коду. Користувачам надається можливість зручно вибирати та завантажувати зображення, а потім отримувати швидкі та точні результати розпізнавання номерних знаків.

Створено виведення результатів розпізнавання у текстовому форматі, де для кожного номерного знаку вказуються розпізнані символи та їх порядок. Цей формат

необхідний для візуальної перевірки користувачем розпізнаних номерних знаків та біля кожного номера є посилання на картинку, аби передивитися самостійно.

Використовуються також графічні компоненти для відображення результатів розпізнавання, такий як рендеринг зображення номерного знаку з розміщеними на ньому розпізнаними символами. Цей формат візуалізації є доцільним для однієї або невеликої кількості картинок.

Для більше ніж одного фото створено графіки та статистичні діаграми для відображення інформації про ефективність розпізнавання та якості даних. Також запроваджена аналітика після пост обробки. У загальному, компонент візуалізації дозволяє користувачеві зрозуміти та проаналізувати результати розпізнавання номерних знаків у зручний та інтуїтивно зрозумілий спосіб [13].

Список інтерфейсів:

- IPP (Interface Photo Processing) – інтерфейс передачі навчальних даних на оброблення, попередньо завантаживши їх в оперативну пам’ять.
- ILP (Interface Label Processing) – інтерфейс передачі вихідних (із модуля обробки зображень) даних на вхід процесу маркування області номерного знаку.
- IDP (Interface Detection Processing) – інтерфейс передачі вихідних(із модуля анотації зображень) даних на вхід процесу навчання детектора.
- ISP (Interface Segmention Processing) – інтерфейс передачі вихідних (із модуля навчання нейронної мережі) даних на вхід процесу сегментації символів.
- IRSP (Interface Recognition Symbols Processing) – інтерфейс передачі вихідних (із модуля сегментації символів) даних на вхід процесу розпізнавання символів.
- IUAP (Interface User Analytics Processing) - інтерфейс передачі вихідних(із модуля розпізнавання симовлів) на вхід процесу виведення результатів з наданням аналітики.

3.3 Формалізація структурного представлення моделі

Структуру системи моделюють як бізнес профіль Еріксона-Пенкера:

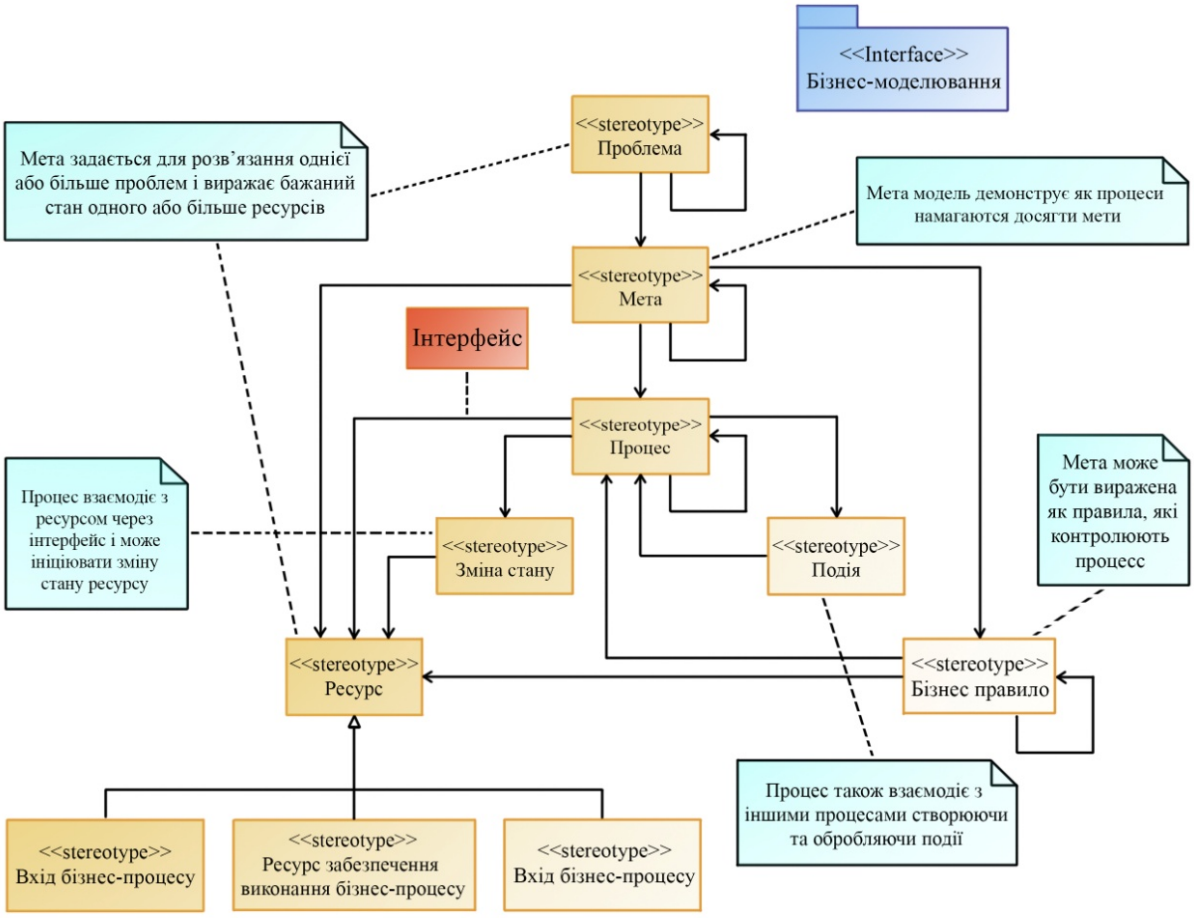


Рисунок 3.2 – Удосконалений бізнес-профіль Еріксона-Пенкера. Діаграма класів у нотації UML

Загальний опис елементів, що наведено на рисунку 3.3.

1. Проблема цієї роботи: значне зростання реєстраційних номерів в Україні та поява нейроних мереж на IP-камерах.

2. Мета цієї роботи: оптимізація нейронної мережі для розпізнавання номерних знаків на транспортних засобах.

3. Множина процесів склдається з компонентів як: збір даних українських номерних знаків; попередня обробка(чистка/маркування); розділення даних на тренувальний та тестувальні набори; проведення навчання на тренувальній вибірці; сегментація символів; розпізнавання тексту на номерному знаку; виведення результатів.

4. Зміна стану та певних ресурсів унаслідок роботи процесів: фото, що були необроблені → фотографії, що були оброблені; фотографії, що були оброблені → тренувальна вибірка; початкова модель→ навчена модель; навчена модель детекції →Розпізнані текстові дані.

5. Ресурси нижчого рівня ієрархії, які використовуються в бізнес-процесі:

Вихідні дані бізнес-процесу: локалізація номерного знаку та розпізнання текстових даних.

Ресурси, що забезпечують виконання бізнес-процесу: оброблені фотографії, навчена модель, система розпізнавання символів та візуалізація.

Вхідні дані бізнес-процесу: фотографії, які потребують обробки, та початкова модель.

6. Подія:

Виконання обробки зображень навчальних даних впливає на ефективність навчання моделі та має прямий вплив на якість результатів прогнозування;

Постійна ітерація над моделлю під час процесу навчання, що призводить до вдосконалення та оптимізації моделі, що в свою чергу позитивно впливає на результати, отримані під час подальшого використання моделі;

Автоматичне передбачення номерних знаків, яке забезпечує системі здатність самостійно прогнозувати та розпізнавати номерні знаки з високою точністю;

Використання методів розпізнавання символів, що базуються на передній сегментації, для забезпечення більш точного та надійного визначення окремих символів, що складають номерний знак;

Візуалізація отриманих результатів, що дозволяє представити розпізнані символи та номерні знаки у графічному вигляді, сприяючи зручному спостереженню та аналізу отриманої інформації.

7. Бізнес-правила

BR1. Кожне зображення, що використовується у навчальній вибірці, повинно мати відповідну обмежувальну рамку, яка точно охоплює об'єкт номерного знаку.

BR2. Зображення, що використовуються як у тестовій, так і у тренувальній вибірках, повинні бути унікальними та не повторюватись, щоб уникнути негативного впливу на процес навчання та прогнозування.

BR3. Навчальний набір повинен включати виключно українські номерні знаки, забезпечуючи спеціалізоване навчання моделі для розпізнавання цих конкретних знаків.

BR4. Навчальний датасет має містити не менше 3000 фотографій з різноманітними складними сценаріями, що дозволить моделі навчатися на широкому спектрі випадків і забезпечить високу якість розпізнавання.

BR5. Забезпечити швидкий час детекції та розпізнавання: система повинна ефективно виконувати процеси виявлення та ідентифікації номерних знаків.

BR6. Мінімізувати помилкове розпізнавання: система повинна приділяти особливу увагу уникненню неправильної ідентифікації та забезпечувати високу точність розпізнавання.

BR7. Забезпечити аналітику результатів: система має надавати аналітичні звіти та статистику щодо своєї роботи, що дозволяє оцінювати її ефективність, виявляти тенденції та проводити детальний аналіз результатів розпізнавання номерних знаків.

3.4 Висновки до розділу

Дана система автоматичного виявлення та розпізнавання номерних знаків складається з двох загальних компонентів: виявлення та розпізнавання номера. Загальна мета системи - знаходження номерного знака на транспортному засобі та оцінка популярності цього номерного знака. У системі можна виділити 7 важливих компонентів: сховище даних, попередня обробка, анотація даних, навчання нейронної мережі, сегментація та розпізнавання символів та візуалізація результатів.

«Сховище даних» зберігає набори даних для навчання та тренування українських номерних знаків. Компонент «Попередня обробка» очищує дані для навчання. Компонент анотації даних призначений для локалізації області номерного знаку та запису його у форматі xml. Компонент «Навчання нейронної мережі» розбиває набір даних на тренувальний та тестовий, створює модель та проводить навчання.

Компонент сегментації використовує фільтри для виділення області символа, а компонент розпізнавання символів використовує Tesseract для розпізнавання сегментованих символів. Завдяки використанню Flask, компонент візуалізації забезпечує зручний та ефективний веб-інтерфейс для виведення результатів у форматі тексту та графіки.

Усі компоненти системи детекції та розпізнавання номерних знаків взаємодіють між собою, щоб дати кінцевий результат. Кожен компонент грає важливу роль у процесі детекції та розпізнавання номерних знаків. Описані відношення між класами у вигляді 6 інтерфейсів.

4. Програмне забезпечення

4.1 Підготовка даних

4.1.1 Збір даних

Platesmania – це веб-платформа, яка надає відкритий доступ до великого обсягу зображень номерних знаків з різними типами та країнами [14]. Для навчання згорткової нейронної мережі було визначено наступні категорії даних:

1. звичайні приватні номерні знаки (2 598 зразків)
2. номерні знаки застарілого типу (104 зразків)
3. квадратні/дворядкові номерні знаки (71 зразків)
4. вантажівки (67 зразків)
5. дипломатичні (64 зразків)

Вибір України як країни для цього проекту мотивований бажанням зосередитись на специфіці українських номерних знаків і використати їх для тренування моделі. Ці номерні знаки мають різні варіації залежно від типу транспортного засобу. Наприклад, номерні знаки для легкових автомобілів мають чорні літери на білому фоні, тоді як для мотоциклів використовуються білі літери на чорному фоні. Цей фокус дозволить моделі ефективно розпізнавати українські номерні знаки та їх унікальні особливості.

Стандартні номерні знаки в Україні складаються з двох літер, за якими йдуть чотири цифри, а далі знаходяться ще дві літери. Перша пара літер вказують на регіон, в якому автомобіль зареєстрований. Цифри відображають порядковий номерний знак, а останні дві літери показують серію номерів.

Набір даних, який використовується для тренування моделі, постійно оновлюється. Якщо попередня модель не змогла правильно виявити або локалізувати нову область номерного знаку, такі дані включаються до тренувального набору даних. Це дозволяє моделі постійно покращуватись і стати

більш ефективною в розпізнаванні складних випадків та різних варіацій номерних знаків.

4.1.2 Маркування даних

Для ефективного навчання загорткової нейронної мережі (ЗНМ) з метою детекції автомобільних номерів, необхідно провести анотацію даних. Анотація має важливе значення, оскільки вона встановлює зв'язок між вхідними та вихідними даними, дозволяючи моделі розпізнавати й відокремлювати цільові об'єкти.

У випадку детекції автомобільних номерів, необхідно виділити область номерного знаку на зображенні. Це виконується за допомогою спеціального інструменту для анотації даних, наприклад, LabelImage [12]. Обирається прямокутник, який точно обведе номерний знак, щоб позначити його межі:

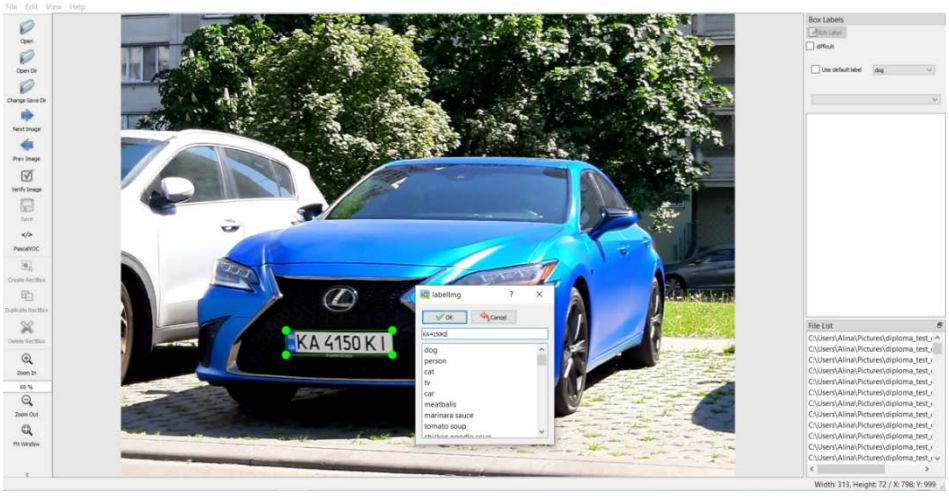


Рисунок 4.1 – Приклад маркування даних через LabelImage[12]

Програма LabelImage дає можливість зберігати дані у різних форматах(та (наприклад, формат YOLO або XML).

Структура маркованих даних включає в себе інформацію про шлях до зображення та координати обмежувальної рамки автомобільного номера за допомогою елементів `<xmin>`, `<ymin>`, `<xmax>`, `<ymax>` ,що задають координати лівого верхнього кута (`<xmin>`, `<ymin>`) та правого нижнього кута (`<xmax>`, `<ymax>`) обмежувальної рамки, що охоплює автомобільний номер на зображенні.

```xml
<annotation>
    <folder>1</folder>
    <filename>21598609.jpg</filename>
    <path>C:\Users\Alina\Pictures\diploma_test_cool\1\21598609.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>1806</width>
        <height>1356</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>KA4150KI</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>485</xmin>
            <ymin>927</ymin>
            <xmax>798</xmax>
            <ymax>999</ymax>
        </bndbox>
    </object>
</annotation>
```

Рисунок 4.2 – Приклад результату анотації даних

Одним з важливих аспектів розмітки даних для детекції автомобільних номерів є чіткість та точність розмітки. Якщо не вдалося чітко визначити границі номерного знаку на зображенні або виникають сумніви, краще утриматися від його розмітки.

Недостатня чіткість анотацій може призвести до неточностей під час навчання моделі.

Якщо границі номерного знаку неоднозначні або недостатньо видимі, то такі неточності може вплинути на якість навчання та результати прогнозування моделі.

4.1.3 Обробка зображень

Після збору та анотації даних, важливо підготувати дані для навчання. Функції для аугментації даних:

- Функція horizontal_flip_mask(dataset_train) виконує горизонтальне відображення зображення та відповідної маски. Це дозволяє створити додаткові екземпляри зображень та масок, де об'єкти можуть бути розташовані з іншого боку.
- Функція add_brightness(dataset_train) додає яскравість до зображень. Вона створює додаткові екземпляри зображень та масок, де яскравість змінюється на певну величину. Це допомагає моделі навчитися розпізнавати об'єкти в різних рівнях освітлення.
- Функція add_contrast(dataset_train) додає контраст до зображень. Вона створює додаткові екземпляри зображень та масок, де контраст змінюється на певний коефіцієнт. Це дозволяє моделі краще розрізняти деталі та вирізняти об'єкти на зображеннях з різним рівнем контрастності.
- Функція random_crop_mask(dataset_train) виконує випадкове обрізання зображень та масок. Вона створює додаткові екземпляри зображень та масок, в яких випадково вибирається область зображення та маски для обрізання. Це допомагає моделі розпізнавати об'єкти, які можуть знаходитися в різних частинах зображення

Функції аугментації застосовуються до набору даних dataset_train для покращення роботи моделі під час навчання. Після застосування аугментації дані змінюються, що допомагає моделі навчитися розпізнавати об'єкти в різних умовах та положеннях. Це покращує її загальну здатність до розпізнавання та забезпечує більш точні результати під час роботи з реальними зображеннями.

4.2 Опис обраного методу для виявлення області номерного знаку

Згорткова нейронна мережа є найкращим варіантом, серед інших методів виявлення номерних знаків. Кілька вагомих причин:

1. Спеціалізована архітектура: ЗНМ спеціально розроблені для виконання завдань, пов'язаних з обробкою зображень, і проявляють в цьому високу ефективність
2. Видобуття ознак: ЗНМ автоматично навчаються витягувати відповідні ознаки з вхідних зображень. Номерні знаки мають характерні особливості, такі як специфічні форми, візерунки та буквено-цифрові комбінації. ЗНМ можуть навчитися розпізнавати ці ознаки, що дозволяє їм точно виявляти номерні знаки.
3. Стійкість до варіацій: ЗНМ здатні справлятися з варіаціями, навчаючись виявляти і видобувати відповідні ознаки незалежно від різних типів, шрифтів і стилів номерних знаків.
4. Наявність навчальних даних: потреба великої кількості маркованих та різноманітних навчальних даних, що дозволяють ЗНМ покращувати точність та узагальнення.
5. Реально-часова продуктивність: ЗНМ можуть бути оптимізовані для роботи в реальному часі за допомогою ефективних архітектур моделей та технік апаратного прискорення.
6. Гнучкість та адаптабельність: ЗНМ можуть бути навчені й адаптовані для виявлення номерних знаків з різних країн або регіонів.
7. Постійне вдосконалення: ЗНМ можуть ітеративно навчатися та покращувати свою продуктивність при появі нових даних або коли мережа зустрічає нові сценарії.

Ефективність методу залежить від якості тренувальних даних, архітектури моделі та параметрвів навчання.

4.3 Архітектура моделі детекції автомобільних номерів

Опис структури згорткової нейронної мережі представлено у вигляді рисунка.

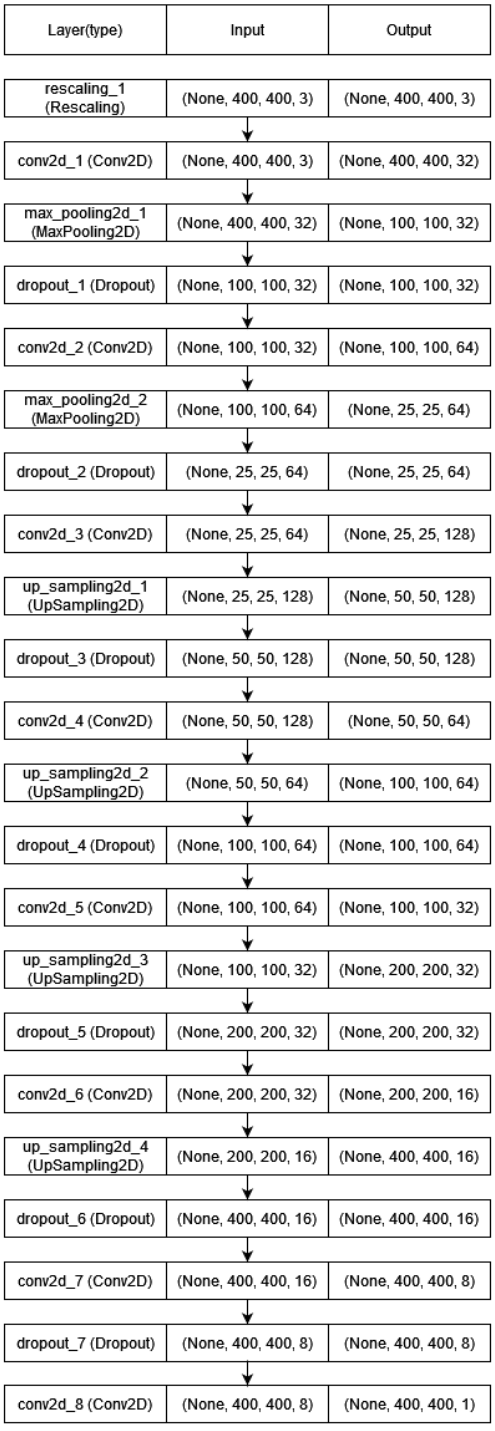


Рисунок 4.3 – Архітектура ЗНМ

Для початку тренування нейронної мережі необхідно розділити наявні дані на дві вибірки. Перша вибірка, яка становить приблизно вісімдесят відсотків усіх зображень, використовуватиметься для тренування моделі, тоді як друга вибірка буде використовуватися для тестування з метою перевірки правильності роботи системи.

Під час тренування нейронної мережі було проведено ретельне експериментування з гіперпараметрами з метою досягнення найкращих результатів. Важливими гіперпараметрами, які було оптимізовано, включалися кількість шарів та їх розмір, швидкість навчання, функції активації, метод оптимізації та регуляризаційні методи.

Основні параметри, що були використані, наведені нижче:

Таблиця 4.3 – Гіперпаратери для навчання ЗНМ

Гіперпараметр	Значення
Висота цільового зображення	400
Ширина цільового зображення	400
Розмір пакета	32
Кількість тренувальної вибірки	2 778
Кількість тестувальної вибірки	4 32
Коефіцієнт випадкового відключення (Dropout rate)	0.2
Оптимізатор	Adam
Функція втрат	Бінарна перехресна ентропія (Binary Crossentropy)
Метрики	Середня абсолютна похибка (MSE)
Кількість епох	30
Рання зупинка	Терпіння: 5, Відновлення кращих ваг: True
Зворотній виклик ModelCheckpoint	save_best_only: True

Правильний вибір цих гіперпараметрів мав значний вплив на процес навчання. Наприклад, відповідно налаштований шари та їх розмір можуть забезпечити належне виявлення та розпізнавання різних властивостей номерних знаків. Встановлення оптимальної швидкості навчання допомагає уникнути проблем недо- або перенавчання моделі. Вибір правильних функцій активації та методу оптимізації може сприяти швидкому збігу та покращенню точності прогнозування.

Процес оптимізації гіперпараметрів був ітеративним, з використанням різних комбінацій значень та оцінкою результатів навчання та перевірки. Цей підхід дозволив підібрати оптимальні значення гіперпараметрів, що забезпечило високу якість та ефективність моделі у процесі розпізнавання номерних знаків.

Після виконання налаштувань моделі, необхідно скомпілювати її з вказанням методу обчислення помилки, методу оптимізації. У даному випадку, оскільки ця модель є моделлю регресії, для обчислення помилки буде використовуватися метод MSE. Для ефективної оптимізації моделі буде використовуватися функція втрати категоріальної перехресної ентропії, яка дозволить оцінювати результативність моделі на кожній епосі. Метод оптимізації, що буде використовуватися Adam, оскільки він комбінує градієнтний спуск з моментумом і є одним з найкращих оптимізаторів для загальних завдань.

З метою запобігання перенавчанню, було використано Dropout – техніку, що допомагає зменшити надлишкове навчання і покращує швидкість виявлення номерних знаків на зображеннях.

Після успішної компіляції моделі, можна розпочати процес тренування, вказавши тренувальні дані, встановивши кількість епох навчання та обравши набір даних для подальшого тестування. По завершенні тренування важливо зберегти модель, щоб мати змогу безпосередньо завантажити її у майбутньому і використовувати без потреби повторного проведення тренування.

4.4 Система розпізнавання символів

Серед безлічі наявних варіантів Tesseract виходить на перше місце як найкраще рішення завдяки своїм винятковим можливостям, універсальності та високій продуктивності. Tesseract використовує передові алгоритми штучного інтелекту(ШІ), включаючи глибоке навчання та нейронні мережі, для розпізнавання символів на реєстраційних номерних знаках. У зв'язку з великим різноманіттям номерних знаків і зображень, які можуть відрізнятися залежно від умов оточуючого середовища, важко знайти специфічний набор даних для розпізнавання символів номерних знаків. Проте його передові моделі навчаються на обширних наборах даних, що дозволяє системі ефективно вивчати та узагальнювати шаблони.

З використанням методів глибокого навчання Tesseract може точно визначати та інтерпретувати різні символи незалежно від їх складності або унікальності. Здатність пристосовуватися до різних дизайнів та варіацій реєстраційних номерних знаків робить Терраскат надійним рішенням для розпізнавання символів.

Функція set_tesseract_cmd() призначена для встановлення шляху до команди Tesseract на основі розташування движка Tesseract OCR. Вона використовує бібліотеку pytesseract для встановлення змінної tesseract_cmd, що гарантує правильну конфігурацію Tesseract для розпізнавання символів.

Для розпізнавання символів використовується початкове зображення, а також початкові координати x_start і y_start, які визначають область інтересу (ROI) в межах цього зображення - зона, яка містить номерний знак. Таким чином, на цьому етапі відокремлюється та вибирається область, де ймовірно розташовані символи, тобто букви та цифри, і всі інші ділянки виключаються з подальшого аналізу. Це необхідно для успішного ідентифікування та зчитування текстової інформації з номерного знаку.

Перед початком вилучення символів, необхідно обробити зображення. Спочатку функція перетворює зображення на відтінки сірого за допомогою функції cv2.cvtColor(). Потім відтінкове зображення масштабується в 3 рази по обидва напрямки x та y за допомогою функції cv2.resize(). Використання фультрів, а саме: розмиття Гауса (cv2.GaussianBlur()) з розміром ядра (7, 7), та медіанне розмиття (cv2.medianBlur()) з розміром ядра 3, необхідне, аби згладити зображення та видалити залишковий шум відповідно.

Після етапу фільтрації виконується порогування методом Отсу (cv2.threshold()) на розмитому зображенні, щоб створити бінарне зображення з чорним текстом на білому фоні. Метод Отсу автоматично визначає оптимальне значення порогу на основі гістограми зображення для відокремлення переднього плану (тексту) від тла.

Створюється прямокутне ядро (cv2.getStructuringElement()) розміром (3, 3), що визначає сусідство для морфологічних операцій. Бінарне зображення потім розширюється (cv2.dilate()) за допомогою прямокутного ядра для заповнення прогалин в символах та зроблення їх більш виразними.

Після розширення відбуваються визначення контурів (cv2.findContours()), які потім сортуються за x-координатою їх обмежуючих прямокутників (cv2.boundingRect()), щоб забезпечити порядок зліва направо.

Функція ітерується по відсортованих контурах і виконує кілька перевірок(співвідношення висоти до ширини, ширини зображення з шириною обмежуючого зображення), щоб відфільтрувати непотрібні області і залишити лише потенційні символи номерного знаку.

Для залишених контурів, які пройшли всі перевірки, вилучається область інтересу (ROI) з бінарного зображення за допомогою координат обмежуючого прямокутника. Потім ROI інвертується за допомогою cv2.bitwise_not(), щоб мати чорні символи на білому фоні. Застосовується медіанне розмиття (cv2.medianBlur()) аби покращити видимість символів.

На завершення використовується функція pytesseract.image_to_data() для виконання OCR на обробленому ROI. Якщо не задати самостійно параметри, то будуть застосовані значення за замовчуванням, але вони не дуже якісно розпізнають символи на українських номерах.

Тож, параметри для розпізнавання символів були встановлені наступні:

Таблиця 4.4 – Параметри Tessaract

Параметр	Значення
Білий список tessedit_char_whitelist	Усі дозволені символи в українських номерних знаках ABCEHIKMNOPTXYZ0123456789
Чорний список tessedit_char_blacklist	Відсутній
Режим сегментації сторніки(psm)	6 (Припускається, що єдиний блок містить текст)
Режим роботи вдигуна(eom)	1 (Лише нейронна мережа LSTM)
Мова	English
Можливість пост обробки(nice)	0

Після виконання OCR, результати повертаються у вигляді масиву даних, який містить оцінки достовірності та визнаний текст. Далі, для фільтрації результатів, використовується метод dat.loc[dat["text"].notna()], який дозволяє вилучити рядки, де значення тексту відсутнє. Якщо після фільтрації залишаються результати OCR, координати та розміри обмежуючих прямокутників адаптуються згідно з масштабним множником, який був застосований для масштабування. Адаптовані результати OCR додаються до списку під назвою data_list.

Варто відзначити, що деякі регістраційні номерні знаки України повністю співпадають з номерними знаками Болгарії та попередніми номерними знаками Іспанії. Це означає, що швидко визначити, до якої країни належить номерний знак, є неможливим. Ця ситуація стосується чотирьох регіонів Іспанії з кодами BA, BI, CA, IB та п'яти регіонів Болгарії з кодами BT, BH, CA, CB, CH, KH

4.5 Аналіз результатів

Оцінка результатів нейронної мережі є надзвичайно важливою, оскільки вона дозволяє зрозуміти, наскільки добре модель працює і чи вона виконує свою поставлену задачу ефективно. Оцінка результатів допомагає зробити наступні кроки для поліпшення моделі і досягнення кращих результатів.

Оцінка результатів нейронної мережі дозволяє порівняти її з іншими моделями або альтернативними підходами, аби вибрати найкращу модель для вашої задачі. Це особливо важливо при виборі між різними архітектурами нейронних мереж або параметрами моделі.

Оцінка результатів нейронної мережі включає розгляд метрик ефективності, функції втрат і оптимізатора. Розглянемо їх важливість:

Метрики ефективності: Точність є поширеною метрикою для задач класифікації. Вона обчислює співвідношення правильно передбачених зразків до загальної кількості зразків у наборі даних. У контексті ЗНМ точність вимірює, наскільки добре модель класифікує зображення або точки даних. Максимізація точності під час навчання допомагає переконатися, що модель робить точні передбачення для задачі.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.5}$$

де TP (True Positive) – істинно-позитивне значення;

TN (True Negative) – істинно-негативне значення;

FP (False Positive) – хибно- позитивне значення;

FN (False Negative) – хибно-негативне значення.

При тестуванні на 235 зображеннях булла встановлена точність у 80,6%.

Оптимізатор визначає, як оновлюються ваги та зсуви моделі під час навчання для мінімізації функції втрат. Adam (адаптивна оцінка моменту) є популярним оптимізатором, який адаптує коефіцієнт швидкості навчання для кожного параметра моделі на основі перших та других моментів градієнтів. Він добре підходить для широкого спектра завдань глибокого навчання і часто забезпечує швидку та ефективну збіжність.

Функція втрат вимірює відхилення між передбаченими виходами вашої моделі та фактичними значеннями. Функція втрат середнього квадратичного відхилення (MSE) обчислює середнє значення квадратів різниць між передбаченими та справжніми значеннями. Мінімізуючи MSE під час навчання, модель краще наближається до цільових виходів.

В результаті навчання згорткової нейронної мережі, були отримані наступні результати:

Функція втрат ($loss\ function$):

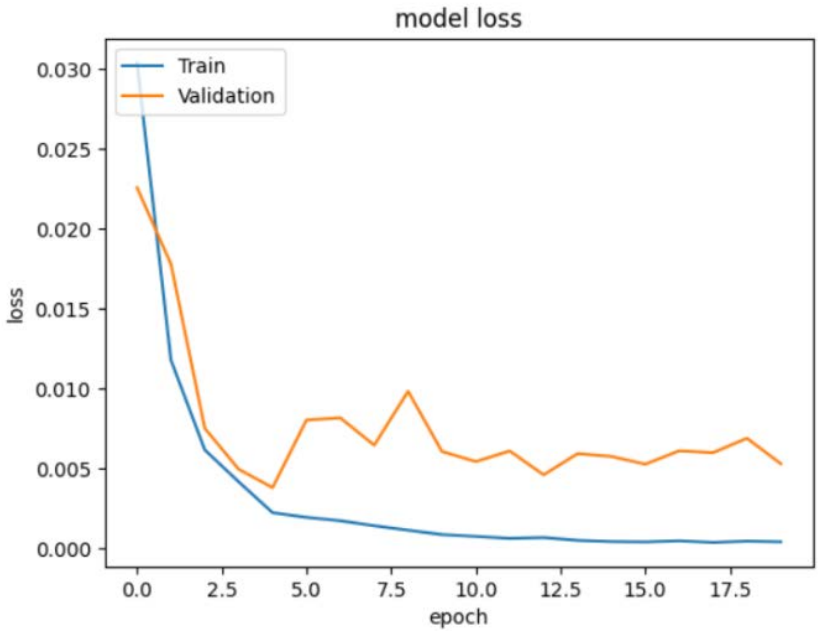


Рисунок 4.5 – Результат функції втрат

Результати вважаються задовільними, тому система є валідною.

4.6 Висновки до розділу

У цьому розділі було наведено увесь процес проектування системи розпізнавання номерних знаків, де розглянуті ключові аспекти, що включають застосування згорткової нейронної мережі для виявлення та локалізації та оптичного розпізнавання символів для індетифікації текстової інформації. Ці функціональності є вирішальними факторами у забезпеченні високої точності та ефективності системи.

Важливою складовою роботи системи є підготовка даних, включаючи обробку, маркування та розділення на тестовий та тренувальний набори. Цей процес допомагає створити відповідні навчальні дані, необхідні для навчання та тестування системи. Його правильна реалізація впливає на якість та результативність системи.

Аналіз результатів з метриками оцінки точності та функцій втрат є необхідним кроком для оцінки ефективності системи. Ці метрики надають об'єктивну інформацію про те, наскільки добре система виконує свої завдання. Це дозволяє зрозуміти, чи відповідають результати системи встановленим вимогам та чи потребуються подальші вдосконалення.

5 Верифікація та валідація системи детекції автомобільних номерів

5.1 Функціональність та особливість отриманої системи

До системи детекції та розпізнавання було висунуто певний список вимог до функціональності та особливості. Перелік:

- Точність розпізнавання складає більше 80%;
- Оптимізація роботи системи;
- Покращення розпізнавання українських номерів;
- Розпізнавання більше одного типу українських номерів;
- Зменшення розпізнавання хибних значень.
- Модель прогнозування повинна базуватися на основі використання згорткових нейронних мереж;
- Використання комбінації фільтрів для покращення зображень;
- Розпізнавання тексту має бути на основі оптичного розпізнання символів;
- Взаємодія з користувачем, надання обрати різні способи роботи програми та виведення зрозумілих результатів.

5.2 Верифікація та валідація

Бізнес-правила відіграють важливу роль у розробці будь-якої системи, оскільки вони визначають правила, політики та процеси, які керують діяльністю бізнесу. Перевірка виконання цих правил допомагає забезпечити якість та правильність функціонування системи. Було визначено основних 8 бізнес-правил для системи детекції та розпізнавання номерних знаків.

BR1.Кожне зображення у вибірці має мати свою обмежувальну рамку для об’єкту номерний знак

BR2.Виключити можливість «витоку даних» (Зображення у тестовій вибірці і тренувальній не мають повторюватись)

BR3. Навчальний набір має складатися тільки з українських номерних знаків,аби забезпечити спецалізавний тип навчання моделі.

BR4. Навчальний датасет має містити більше 3000 фотографій з різними складними випадками

BR5. Забезпечити швидкий час детекції та розпізнавання.

BR6. Мінімізувати помилки розпізнавання для запобігання неправильним ідентифікаціям.

BR7. Надання аналітики по результатам роботи системи

За допомогою програми LabelImage було анотовано більше 3000 номерних знаків, що задовольняє вимогу BR1.

Тренувальна та тестувальна вибірки формуються окремо, що унеможливлює повторення зображень і тим самим виконується вимога BR2.

Для розроблюваної системи було обрано українські номерні знаки, для навчання та тестування брались номерні знаки родом з України, що задовольняє вимогу BR3.

Після видалення дублікатів та невалідних номерних знаків, навчальна вибірка складалась з 3100 фотографій, що задовольняє вимогу BR4.

Важливо, щоб система швидко та правильно виявляла номерний знак, що задовольняє вимогу BR5.

Детекція номерних знаків на зображенні передбачає локалізацію розташування номерного знака на зображенні або прийняття рішення про його відсутність. В останньому випадку ніякі наступні дії не виконуються. Але важливо розпізнавати тільки номерні знаки. Потрібно уникнути неправильне розпізнавання, тобто номера , які були розпізнені там, де їх немає, наприклад: решітки радіаторі, написи на транспортах. Для вирішення цієї задачі було навчено ЗНМ на дата сеті з різноманітними зображеннями, яка з дуже мінімальної вірогідністю локалізує неправильну область. Це і виконує вимогу BR6.

Надання аналітики по результатам детекції та розпізнавання номерних знаків є корисним для користувачів, які мають мінімальні технічні здібності для оцінки роботи системи, що задовольняє умови BR7.

Основну мету розроблюваної системи було досягнуто.

5.3 Висновки до розділу

Було розглянуто функціональність та особливості отриманої системи. Вимоги до системи включали точність розпізнавання, оптимізацію роботи, використання згорткових нейронних мереж, оптичне розпізнавання символів та зрозумілу взаємодію з користувачем.

Було проведено перевірку виконання відповідних бізнес-правил (BR1-9) і перевірено реалізацію структурних представлень системи детекції та розпізнавання номерних знаків. Зокрема, було здійснено анотування більше 3000 номерних знаків, унеможливлено повторення зображень у тренувальній та тестовій вибірках, використано українські номерні знаки, створено достатньо об'ємний навчальний датасет, забезпечено швидку та правильну детекцію та розпізнавання номерних знаків, уникнуто неправильного розпізнавання та надано аналітику результатів.

Отже, система детекції та розпізнавання автомобільних номерів успішно пройшла верифікацію та валідацію, відповідаючи вимогам та бізнес-правилам, що були встановлені.

Висновки

1. У рамках виконаного проекту проведено детальний аналіз методів для детекції номерних знаків, фільтрації шумів, алгоритмів сегментації символів та способів їх розпізнавання. Створені порівняльні таблиці, що базуються на важливих аспектах, для прийняття обґрунтованого рішення. Також проведено огляд існуючих комерційних рішень для порівняння.
2. У результаті роботи розроблена модель системи виявлення та зчитування автомобільних номерних знаків у вигляді діаграми компонентів. Встановлені зв'язки, які були представлені у вигляді інтерфейсів. Основним результатом є програмне забезпечення, яке здатне локалізувати область номерного знаку та розпізнати текст.
3. Система детекції імплементована на основі навченої згорткової нейронної мережі, а розпізнавання тексту номерних знаків використовувало оптичне розпізнання символів. Дані для навчання були попередньо оброблені згідно вимог та анотовані з урахуванням правил. Для оцінки роботи системи були використані метрики точності та функції втрат. Розроблено аналітичний модуль для користувача, який дозволяє візуально сприймати результати системи.
4. У результаті валідації та верифікації, поставлені функціональні вимоги та бізнес-правила виконані. Результати відповідають очікуваній точності, як вище80%. Система оптимізована і не показує помилкових значень.

Отже, проект демонструє успішну розробку математичного та програмного забезпечення для системи детекції та розпізнавання номерних знаків. Результати випробувань і аналізу свідчать про ефективність та надійність системи, а використання відповідних метрик та функцій втрат дозволяє об'єктивно оцінити її роботу.

Перелік посилань

1. Agarwal A., Goswami S. An Efficient Algorithm for Automatic Car Plate Detection & Recognition. 2016 Second International Conference on Computational Intelligence & Communication Technology (CICT), Ghaziabad, India, 12–13 February 2016. 2016
2. Axis license plate verifier [Електронний ресурс] – Режим доступу до ресурсу: https://www.axis.com/products/axis-license-plate-verifier/
3. Bensouilah M., Zennir M.Z., Taffar M., “An ALPR System-based Deep Networks for the Detection and Recognition”.
4. Bhargava D., Vyas S., Bansal A. Comparative analysis of classification techniques for brain magnetic resonance imaging images. Advances in Computational Techniques for Biomedical Image Analysis. 2020. P. 133–144.
5. Caner H., Gecim H. S., Alkar A. Z. Efficient Embedded Neural-Network-Based License Plate Recognition System. IEEE Transactions on Vehicular Technology. 2008. Vol. 57, no. 5. P. 2675–2683.
6. Dahua technology anpr solution [Електронний ресурс] – Режим доступу до ресурсу: https://www.dahuasecurity.com/ceen/products/keyTechnologies/352
7. Du Sh., Ibrahim M., Shehata M. Automatic License Plate Recognition (ALPR): A State-of-the-Art Review / S. Du et al. IEEE Transactions on Circuits and Systems for Video Technology. 2013. Vol. 23, no. 2. P. 311–325.
8. Hanwha techwin anpr solution [Електронний ресурс] – Режим доступу до ресурсу: https://hanwhavision.eu/launching-the-serverless-anpr-solution-for-traffic-management/
9. Hikvision ANPR solutions [Електронний ресурс] – Режим доступу до ресурсу: https://www.hikvision.com/europe/products/software/HikCentral-Professional-series/HikCentral-ANPR/.

10. Jyothi S., Azeez S., Anil M., Srinu P., Krishna V. Automatic license plate recognition using pre-processing methods / International Journal of Engineering and Applied Sciences, 2017, March.
11. Kosasih K. K., Astuti W., Oey E. License plate recognition system based on principal component analysis and one-against-one multi-class support vector machine. IOP Conference Series: Earth and Environmental Science. 2020.
12. LabelImage recognition [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/labelImg/1.4.0/.
13. Maslianko, P. P., Sielskyi, Y. P., "МЕТОД СИСТЕМНОЇ ІНЖЕНЕРІЇ СИСТЕМ НЕЙРОННОГО МАШИННОГО ПЕРЕКЛАДУ", KPI Science News, 2021, August 31.
14. O. Martinsky, "Algorithmic and mathematical principles of automatic number plate recognition systems," B.Sc. thesis, department of intelligent systems, faculty of information technology, Brno University of Technology, 2007.
15. Platesmania [Електронний ресурс] – Режим доступу до ресурсу: https://platesmania.com/
16. Sanjib D., Khan J., Iqbal M. S. U. D. A comparative study of Different Denoising Techniques in Digital Image Processing. 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO), Manama, Bahrain, 15–17 April 2019.
17. Securos auto license plate recognition [Електронний ресурс] – Режим доступу до ресурсу: https://issivs.com/securos-auto/
18. Viola P., Jones M. Rapid object detection using a boosted cascade of simple features. 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA.
19. Wang S., Lee H.L. Detection and recognition of license plate characters with different appearances. 2003 IEEE International Conference on Intelligent Transportation Systems, Shanghai, China.

20. Yuan, Y., Zou, W., Zhao, Y., Wang, X., Hu, X.,Komodakis, N A Robust and Efficient Approach to License Plate Detection / Y. Yuan et al. IEEE Transactions on Image Processing. 2017. Vol. 26, no. 3. P. 1102–1114.
21. Yuheng S., Hao Y.Image Segmentation Algorithms Overview. The ACM/IEEE Joint Conference on Digital Libraries in 2020, Virtual Event China. New York, NY, USA, 2020.

Додаток А

Лістинги програм

Лістинг файлу app_new.p — основний модуль

```
import os
import tempfile
from flask import Flask, render_template, request, flash, redirect, url_for, current_app
from werkzeug.utils import secure_filename
from prediction_main import make_prediction
import secrets
from license_plate_extraction.reading import upload_img
from license_plate_extraction.graph import display_result, generate_prediction_chart, save_bounding_box
from pathlib import Path
from license_plate_extraction.region import get_region, update_last_symbol
import re
import  matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg')

pattern_last_digit = r'^[A-Z]{2}\d{4}[A-Z]{1}\d{1}$'
# suppress tensorflow's verbose logging
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"
pattern = r'^[A-Z]{2}\d{4}[A-Z]{2}$'
pattern_di = r'^[A-Z]{2}\d{6}$'

app = Flask(__name__)
app.secret_key = secrets.token_hex(16)

# Set UPLOAD_FOLDER and ALLOWED_EXTENSIONS as app config variables
app.config['UPLOAD_FOLDER'] = tempfile.gettempdir()
app.config['ALLOWED_EXTENSIONS'] = {'jpg', 'jpeg', 'png'}



@app.route('/')
def index():
    return render_template('index.html')

@app.route('/analytics')
def analytics():
    static_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'static')
    chart_filename = os.path.join(static_path, 'results.png')
    chart_url = url_for('static', filename='results.png')
    if os.path.isfile(chart_filename):
        return render_template('analytics.html', chart_url=chart_url)
    else:
        return render_template('analytics.html', chart_url=None)

@app.route('/detect', methods=['GET', 'POST'])
def detect():
    # Dictionary to map license plate region codes based on first two letters of the license plate

    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files and 'dir' not in request.files:
            flash('No file part')
            return redirect(request.url)

        visualize = request.form.get('visualize')
        option = request.form.get('option')

        # if user uploaded a single image
        if option == 'single':
            file = request.files['file']
            if not file:
                return render_template('index.html', message='No file selected.')
            path = Path(file.filename)
```

```
            file.save(path)
            if path.is_file():
                bounding_box, prediction, average_confidence = make_prediction(path)

                if len(prediction) ==8:
                    prediction = update_last_symbol(prediction)
                if len(prediction) == 6 and prediction.isdigit():
                    prediction = 'DP' + prediction

                image = upload_img(path)
                display_result(image, bounding_box, prediction)

                static_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'static')
                chart_filename = os.path.join(static_path, 'result.png')
                plt.savefig(chart_filename)
                more_region = get_region(prediction)

                return render_template('result.html', prediction=prediction, region=more_region, average_confidence = average_confidence)

        # if the user uploaded a directory of images
        elif option == 'directory':
            dir_files = request.files.getlist('dir')
            if len(dir_files) == 0:
                flash('No selected directory')
                return redirect(request.url)

            # create a temp dir to store uploaded images
            temp_dir = Path(tempfile.mkdtemp())
            for file in dir_files:
                if allowed_file(file.filename):
                    filename = secure_filename(file.filename)
                    file_path = temp_dir.joinpath(filename)
                    file.save(str(file_path))

            image_path_list = list(temp_dir.glob('*'))
            predictions = []
            num_correct_before = 0
            num_correct_after = 0
            num_extra_before = 0
            num_extra_after = 0
            num_missed_before = 0
            num_missed_after = 0
            not_detected = 0
            all_detected = 0
            for more_image_path in image_path_list:
                more_image_name = more_image_path.name
                more_bounding_box, more_prediction, more_average_confidence = make_prediction(str(more_image_path))

                # Before update
                if (len(more_prediction) == 8 and re.match(pattern, more_prediction))or (len(more_prediction) ==8 and re.match(pattern_di, more_prediction)):
                    num_correct_before += 1
                if len(more_prediction) == 6 and more_prediction.isdigit():
                    more_prediction = 'DP' + more_prediction
                    num_correct_before += 1
                elif len(more_prediction) > 8 and more_prediction != 'Not detected':
                    num_extra_before += 1
                elif len(more_prediction) < 8 and len(more_prediction) > 0 and more_prediction != 'Not detected':
                    num_missed_before += 1
                if more_prediction == 'Not detected':
                    not_detected += 1
                elif len(more_prediction) > 0 and more_prediction != 'Not detected':
                    all_detected += 1

                # After update
                if len(more_prediction) == 8 and not re.match(pattern_di, more_prediction):
                    more_prediction = update_last_symbol(more_prediction)

                if (len(more_prediction) == 8 and re.match(pattern, more_prediction))or (len(more_prediction) ==8 and re.match(pattern_di, more_prediction)):
                    num_correct_after += 1
                elif len(more_prediction) > 8 and more_prediction != 'Not detected':
                    num_extra_after += 1
                elif len(more_prediction) < 8 and len(more_prediction) > 0 and more_prediction != 'Not detected':
                    num_missed_after += 1

                more_region = get_region(more_prediction)
```

```
            filename = secure_filename(more_image_path.name)
            filename = os.path.splitext(filename)[0]  # Remove the file extension

            # Specify the desired bounding box image filename
            bounding_box_filename = f"{filename}_bounding_box.jpg"

            # Generate the save path for the bounding box image
            save_path = os.path.join(current_app.root_path, 'static', 'bounding_box_images', bounding_box_filename)
            # Save the bounding box image
            save_bounding_box(upload_img(more_image_path), more_bounding_box, save_path)

            predictions.append((more_image_name, bounding_box_filename, more_prediction, more_region, more_average_confidence))

        generate_prediction_chart(num_correct_before, num_extra_before, num_missed_before,
                                  num_correct_after, num_extra_after, num_missed_after,
                                  predictions, not_detected, all_detected)
        # Save the figure to a file
        static_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'static')
        chart_filename = os.path.join(static_path, 'results.png')
        plt.savefig(chart_filename)

        # Close the figure
        plt.close()

        return render_template('result.html', predictions=predictions)

    # if the file type is not allowed
    else:
        flash('File type not allowed')
        return redirect(request.url)

# if the method is GET
else:
    return redirect('/')
@app.route('/home')
def home():
    return redirect(url_for('index'))

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in app.config['ALLOWED_EXTENSIONS']

if __name__ == '__main__':
    app.run(debug=True)
```

Лістинг файлу model.py — основний модуль

```
import tensorflow as tf
import numpy as np
from reading import get_image_paths_from_directory, make_dataset_from_image_paths_with_masks
from prepearing import covert_box_to_percent, bounding_box_in_pixel, resize_bounding_box, bounding_box_to_binary_mask, mask_to_bounding_box
import datetime
import matplotlib.pyplot as plt
import os

from pathlib import Path

BASE_DIR = Path(__file__).parent.parent.parent
DATA_DIR = BASE_DIR / "data"
LOG_DIR = BASE_DIR / "logs"
dir_ukraine = DATA_DIR / "train"
dir_ukraine_old = DATA_DIR / "ua_old"
dir_ukraine_square = DATA_DIR / "square"
dir_ukraine_diplomatic = DATA_DIR / "ua_diplomatic"
dir_ukraine_pickup = DATA_DIR / "ua_pickup"

# make a list with all the image paths to use during training
image_paths_train = np.array(
    [

        *get_image_paths_from_directory(dir_ukraine_old),
        *get_image_paths_from_directory(dir_ukraine),
        *get_image_paths_from_directory(dir_ukraine_square),
        *get_image_paths_from_directory(dir_ukraine_diplomatic),
        *get_image_paths_from_directory(dir_ukraine_pickup),
```

```
    ]
)
# Create a dictionary to store the counts
image_counts = {
    "ua_square": len([file for file in os.listdir(dir_ukraine_square) if file.endswith('.jpg')]),
    "ua_pickup": len([file for file in os.listdir(dir_ukraine_pickup) if file.endswith('.jpg')]),
    "ua": len([file for file in os.listdir(dir_ukraine) if file.endswith('.jpg')]),
    "ua_old": len([file for file in os.listdir(dir_ukraine_old) if file.endswith('.jpg')]),
    "ua_diplomatic": len([file for file in os.listdir(dir_ukraine_diplomatic) if file.endswith('.jpg')])
}
# Sort the image_counts dictionary by values in descending order
sorted_counts = {k: v for k, v in sorted(image_counts.items(), key=lambda item: item[1], reverse=True)}

# Calculate the total number of images
total_count = sum(sorted_counts.values())

# Plot the counts
plt.figure(figsize=(10, 6))  # Set the figure size
bars = plt.bar(sorted_counts.keys(), sorted_counts.values(), color='skyblue')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha='right')

# Add value labels on top of each bar
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height, height,
             ha='center', va='bottom')

# Add the total count to the legend
plt.legend(['Total: {}'.format(total_count)])

plt.xlabel("Image Directory")
plt.ylabel("Count")
plt.title("Number of JPG Images in Different Directories")
plt.tight_layout()  # Adjust the spacing
plt.show()

dir_test = DATA_DIR / "test"

image_paths_test = np.array([*get_image_paths_from_directory(dir_test)])


TARGET_IMG_HEIGHT = 400
TARGET_IMG_WIDTH = 400
BATCH_SIZE = 32

dataset_train = make_dataset_from_image_paths_with_masks(
    image_paths_train,
    target_img_height=TARGET_IMG_HEIGHT,
    target_img_width=TARGET_IMG_WIDTH,
)

# Function to flip bounding box coordinates
def flip_bounding_box_coords(bounding_box_percent: np.ndarray) -> np.ndarray:
    x_min_percent, y_min_percent, width_percent, height_percent = bounding_box_percent

    # Calculate new coordinates
    x_min_new = 1 - x_min_percent - width_percent

    return np.array([x_min_new, y_min_percent, width_percent, height_percent])


# Function to adjust contrast augmentation
def adjust_contrast_augmentation(cur_image, cur_bounding_box):
    img_list = []
    bounding_box_list = []

    for cur_contrast in [4]:
        adjusted_image = tf.image.adjust_contrast(cur_image, contrast_factor=cur_contrast)
        adjusted_image = tf.clip_by_value(adjusted_image, clip_value_min=0, clip_value_max=255)
        img_list.append(adjusted_image)
        bounding_box_list.append(cur_bounding_box)

    return np.array(img_list), np.array(bounding_box_list)
```

```
# Function to apply flip augmentation
def flip_augmentation(cur_image, cur_bounding_box, mask=False):
    img_list = [cur_image, tf.image.flip_left_right(cur_image)]

    if not mask:
        bounding_box_list = [cur_bounding_box, flip_bounding_box_coords(cur_bounding_box)]
    else:
        # Convert bounding box coordinates to percent and perform flipping
        bounding_box_coords_pixel = mask_to_bounding_box(cur_bounding_box.numpy())
        bounding_box_percent = covert_box_to_percent(
            bounding_box_coords_pixel, img_height=cur_image.shape[0], img_width=cur_image.shape[1]
        )

        flipped_bounding_box_percent = flip_bounding_box_coords(bounding_box_percent)
        flipped_bounding_box_pixel = bounding_box_in_pixel(
            flipped_bounding_box_percent, cur_image.shape[0], cur_image.shape[1]
        )
        bounding_box_list = [
            cur_bounding_box,
            bounding_box_to_binary_mask(
                flipped_bounding_box_pixel, cur_image.shape[0], cur_image.shape[1]
            ),
        ]

    return np.array(img_list), np.array(bounding_box_list)


# Function to apply flip augmentation with mask
def flip_augmentation_mask(cur_image, cur_bounding_box):
    return flip_augmentation(cur_image, cur_bounding_box, mask=True)


# Function to adjust brightness augmentation
def adjust_brightness_augmentation(cur_image, cur_bounding_box):
    img_list = []
    bounding_box_list = []
    delta = 0.5

    for cur_brightness in [-delta, delta]:
        adjusted_image = tf.image.adjust_brightness(cur_image, delta=cur_brightness)
        adjusted_image = tf.clip_by_value(adjusted_image, clip_value_min=0, clip_value_max=255)
        img_list.append(adjusted_image)
        bounding_box_list.append(cur_bounding_box)

    img_list.insert(0, cur_image)
    bounding_box_list.insert(0, cur_bounding_box)

    return (np.array(img_list), np.array(bounding_box_list))


# Function to apply random crop augmentation
def random_crop_augmentation(cur_image, cur_bounding_box, mask=False):
    img_list = []
    bounding_box_list = []

    img_height, img_width, _ = cur_image.shape

    if mask:
        bounding_box_coords_pixel = mask_to_bounding_box(cur_bounding_box.numpy())
    else:
        bounding_box_coords_pixel = bounding_box_in_pixel(cur_bounding_box, img_height, img_width)

    x_min, y_min, width, height = bounding_box_coords_pixel

    rng = np.random.default_rng()
    random_offset_height = rng.integers(low=0, high=y_min + 1, size=1)[0]
    random_offset_width = rng.integers(low=0, high=x_min + 1, size=1)[0]
    random_target_height = rng.integers(
        low=height + (y_min - random_offset_height),
        high=img_height - random_offset_height + 1,
        size=1,
    )[0]
    random_target_width = rng.integers(
        low=width + (x_min - random_offset_width),
        high=img_width - random_offset_width + 1,
```

```
        size=1,
    )[0]

    cropped_image = tf.image.crop_to_bounding_box(
        cur_image,
        random_offset_height,
        random_offset_width,
        random_target_height,
        random_target_width,
    )
    cropped_image = tf.image.resize(cropped_image, size=(img_height, img_width), antialias=True)
    img_list.append(cropped_image)

    # update bounding box
    x_min_new = x_min - random_offset_width
    y_min_new = y_min - random_offset_height
    width_new = width
    height_new = height
    bounding_box_new = np.array([x_min_new, y_min_new, width_new, height_new])
    bounding_box_new = resize_bounding_box(
        bounding_box_new,
        img_height=random_target_height,
        img_width=random_target_width,
        target_img_height=img_height,
        target_img_width=img_width,
    )

    if not mask:
        bounding_box_new = covert_box_to_percent(bounding_box_new, img_height, img_width)
        bounding_box_list.append(bounding_box_new)
    else:
        bounding_box_mask = bounding_box_to_binary_mask( bounding_box_new, img_height, img_width)
        bounding_box_list.append(bounding_box_mask)

    return (np.array(img_list), np.array(bounding_box_list))


# Function to apply random crop augmentation with mask
def random_crop_augmentation_mask(cur_image, cur_bounding_box):
    return random_crop_augmentation(cur_image, cur_bounding_box, mask=True)


# Function to apply augmentation function to the dataset
def apply_augmentation_to_dataset(dts, augmentation_fun):
    dts = dts.map(lambda x, y: tf.py_function(augmentation_fun, inp=[x, y], Tout=(tf.float32, tf.float32)))
    return dts.flat_map(lambda x, y: tf.data.Dataset.from_tensor_slices((x, y)))


# Function to add contrast augmentation to the dataset
def add_contrast_augmentation(dataset: tf.data.Dataset) -> tf.data.Dataset:
    return apply_augmentation_to_dataset(dataset, adjust_contrast_augmentation)


# Function to add brightness augmentation to the dataset
def add_brightness_augmentation(dataset: tf.data.Dataset) -> tf.data.Dataset:
    return apply_augmentation_to_dataset(dataset, adjust_brightness_augmentation)


# Function to apply horizontal flip augmentation with mask to the dataset
def horizontal_flip_augmentation_mask(dataset: tf.data.Dataset) -> tf.data.Dataset:
    return apply_augmentation_to_dataset(dataset, flip_augmentation_mask)


# Function to apply random crop augmentation with mask to the dataset
def random_crop_augmentation_mask(dataset: tf.data.Dataset) -> tf.data.Dataset:
    return apply_augmentation_to_dataset(dataset, random_crop_augmentation_mask)


# Apply the augmentations to the dataset
dataset_train = horizontal_flip_augmentation_mask(dataset_train)
dataset_train = add_brightness_augmentation(dataset_train)
dataset_train = add_contrast_augmentation(dataset_train)
dataset_train = random_crop_augmentation_mask(dataset_train)
dataset_train = horizontal_flip_augmentation_mask(dataset_train)

# set batch size
```

```
dataset_train = dataset_train.shuffle(1024).batch(BATCH_SIZE)

dataset_test = make_dataset_from_image_paths_with_masks(
    image_paths_test,
    target_img_height=TARGET_IMG_HEIGHT,
    target_img_width=TARGET_IMG_WIDTH,
)
dataset_test = dataset_test.batch(BATCH_SIZE)
def cnn_model(input_height, input_width, num_channels, dropout_rate=0.2):
    model = tf.keras.Sequential(
        [
            tf.keras.Input(shape=(input_height, input_width, num_channels)),
            tf.keras.layers.experimental.preprocessing.Rescaling(1.0 / 255,),
            tf.keras.layers.Conv2D(32, 3, padding="same", activation="relu"),
            tf.keras.layers.MaxPool2D(4),
            tf.keras.layers.Dropout(dropout_rate),
            tf.keras.layers.Conv2D(64, 3, padding="same", activation="relu"),
            tf.keras.layers.MaxPool2D(4),
            tf.keras.layers.Dropout(dropout_rate),
            tf.keras.layers.Conv2D(128, 3, padding="same", activation="relu"),
            tf.keras.layers.UpSampling2D(),
            tf.keras.layers.Dropout(dropout_rate),
            tf.keras.layers.Conv2D(64, 3, padding="same", activation="relu"),
            tf.keras.layers.UpSampling2D(),
            tf.keras.layers.Dropout(dropout_rate),
            tf.keras.layers.Conv2D(32, 3, padding="same", activation="relu"),
            tf.keras.layers.UpSampling2D(),
            tf.keras.layers.Dropout(dropout_rate),
            tf.keras.layers.Conv2D(16, 3, padding="same", activation="relu"),
            tf.keras.layers.UpSampling2D(),
            tf.keras.layers.Dropout(dropout_rate),
            tf.keras.layers.Conv2D(8, 3, padding="same", activation="relu"),
            tf.keras.layers.Dropout(dropout_rate),
            tf.keras.layers.Conv2D(1, 1, padding="same", activation="sigmoid"),
        ]
    )

    return model

model = cnn_model(input_height=TARGET_IMG_HEIGHT, input_width=TARGET_IMG_WIDTH, num_channels=3)
model.summary()
model.compile(optimizer="adam", loss=tf.keras.losses.BinaryCrossentropy(), metrics=[tf.keras.metrics.MeanAbsoluteError()])

# add Tensorboard callback
log_dir = LOG_DIR / ("training_" + datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1, update_freq="batch")

model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint("checkpoint_callback_{val_loss}_e_{epoch}.tf", save_best_only=True)

# Add EarlyStopping callback
early_stopping_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model
history = model.fit(
    dataset_train,
    epochs=10,
    validation_data=dataset_test,
    callbacks=[tensorboard_callback, model_checkpoint_callback, early_stopping_callback],
)

loss, accuracy = model.evaluate(dataset_test, verbose=1)
loss_v, accuracy_v = model.evaluate(dataset_train, verbose=1)
print("Validation: accuracy = %f  ;  loss_v = %f" % (accuracy_v, loss_v))
print("Test: accuracy = %f  ;  loss = %f" % (accuracy, loss))

# Save the model with the best weights
model = model.save("ukr_model_3005_with_dropout")
```

Додаток Б

Ілюстративний матеріал

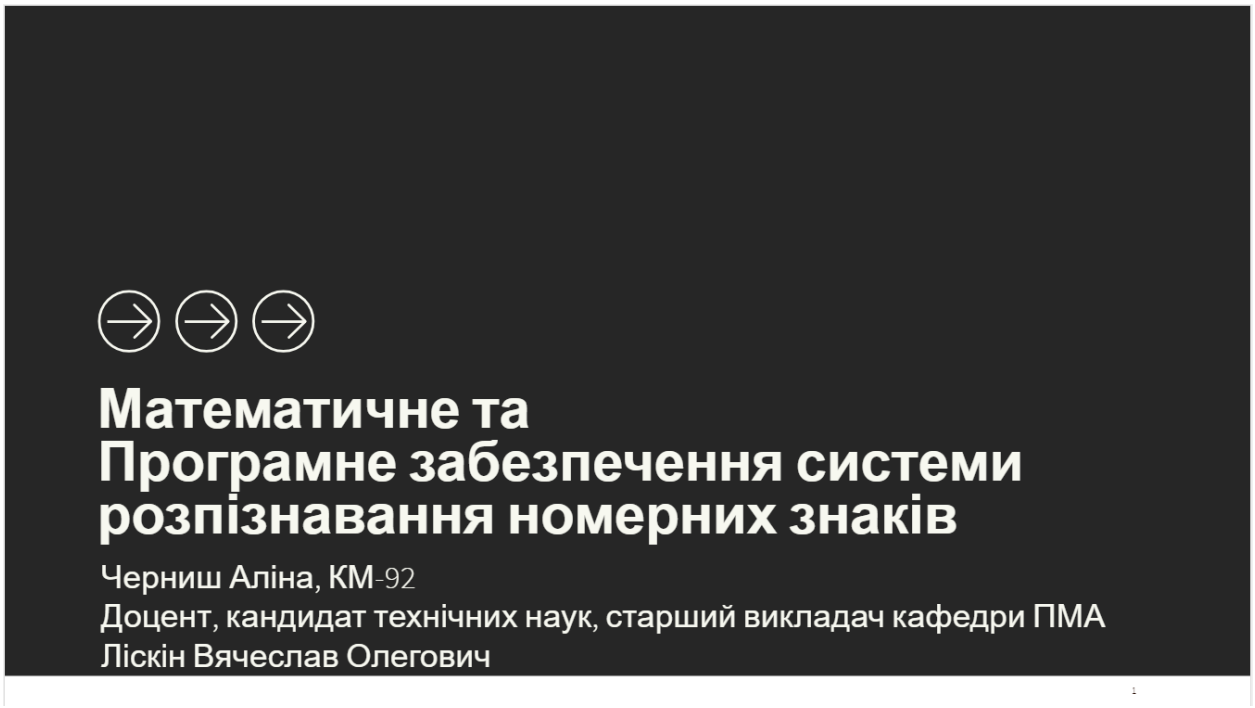


Рисунок Б.1 – Слайд 1

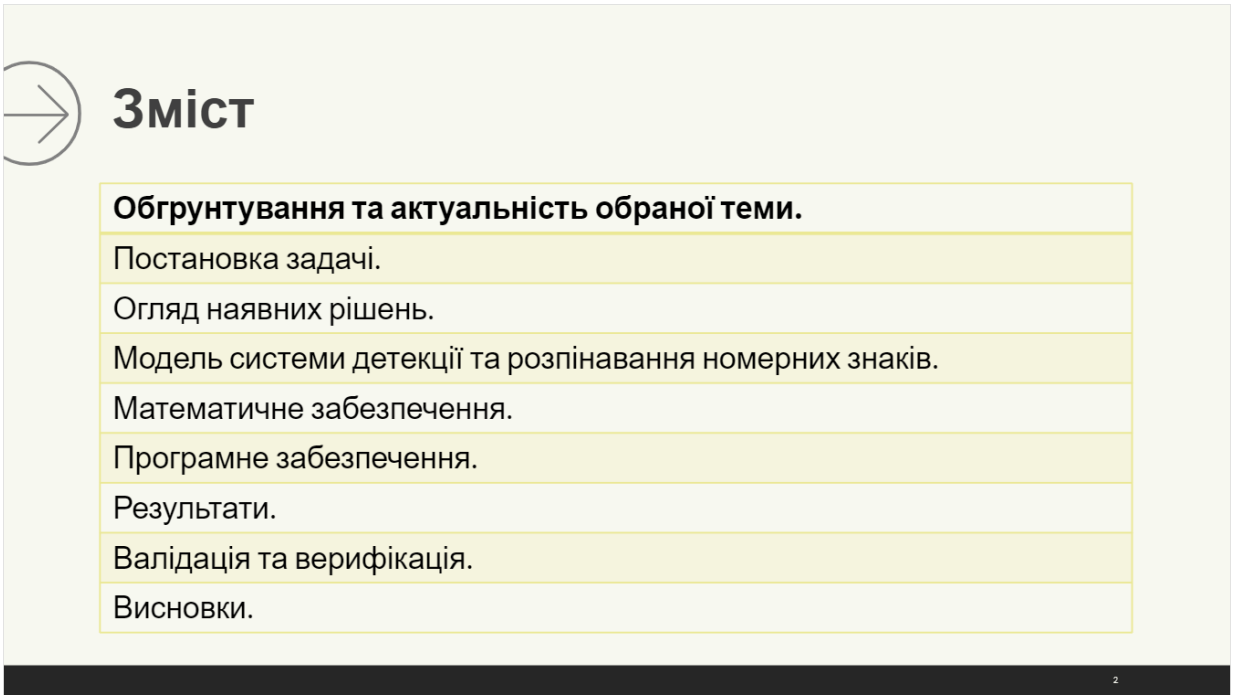
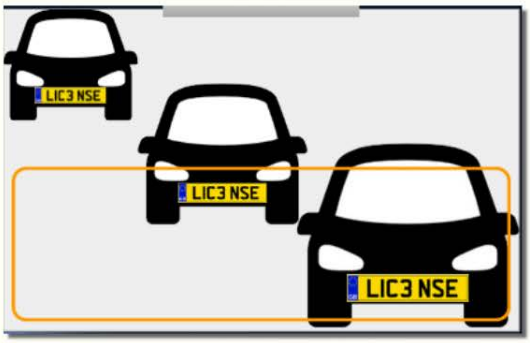


Рисунок Б.2 – Слайд 2

Актуальність обраної теми та практична цінність



Оптимізація роботи виявлення та розпізнавання номерних знаків на основі ЗНМ

Адаптація до різних станів українських номерних знаків

Покращення розпізнавання через семантичні правила

Практична цінність: зменшення розпізнавання False значень

3

Рисунок Б.3 – Слайд 3

Постановка задачі

Метою написання роботи є розробка системи для розпізнавання номерних знаків на основі машинного навчання.

При розробленні відповідного забезпечення потрібно розв'язати наступні завдання:
а) проведення порівняльного аналізу алгоритмів системи детекції та розпізнавання номерних знаків;
б) вибір існуючого методу для вирішення задачі розпізнавання номерних знаків;
в) створення математичного забезпечення
г) розробка програмного забезпечення на базі вибраного математичного методу;
д) тестування розробленої автоматизованої системи.

Реалізована система має задовольняти такі вимоги:
а) мати високі показники ефективності розпізнавання;
б) розпізнавання номерних знаків при різних ситуаціях;
в) надавати аналітичну інформацію про результати роботи програми.

4

Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

Рисунок Б.6 – Слайд 6

Порівняльна таблиця методів детекції номерних знаків

Метод/критерій	Метод Віоли Джонсона	Метод щільності країв	Метод опорних векторів	Згорткові нейронні мережі
Точність виявлення	Помірна	Помірна	Помірна	Висока
Складність тренування	Низька	Низька	Помірна	Висока
Стійкість до варіацій	Помірна	Помірна	Помірна	Висока
Розмірі даних	Помірна	Помірна	Висока	Висока
Адаптабельність	Обмежена	Обмежена	Обмежена	Висока
Видобуток ознак	Вручну створені ознаки Haar	Основані на границях	Вручну створені ознаки	Автоматично вивчені ознаки

8

Рисунок Б.7 – Слайд 7

Порівняльний аналіз методів розпізнавання номерних знаків

Метод/критерій	Оптичне розпізнавання символів	Морфологічна скелетизація	Самоорганізуючі карти
Точність розпізнавання тексту	Висока	Помірна	Помірна до високої
Складність тренування	Помірна до високої	Низька	Висока
Стійкість до варіацій	Висока	Помірна	Помірна
Розмірі даних	Великий	Малий	Малий до помірного
Обробка складних шрифтів	Так	Ні	Ні
Адаптабельність	Висока	Низька	Помірна
Додаткові вимоги	Маркування тренувальних даних	Вхідні бінарні зображення	Структура топологічної мапи

9

Рисунок Б.8 – Слайд 8

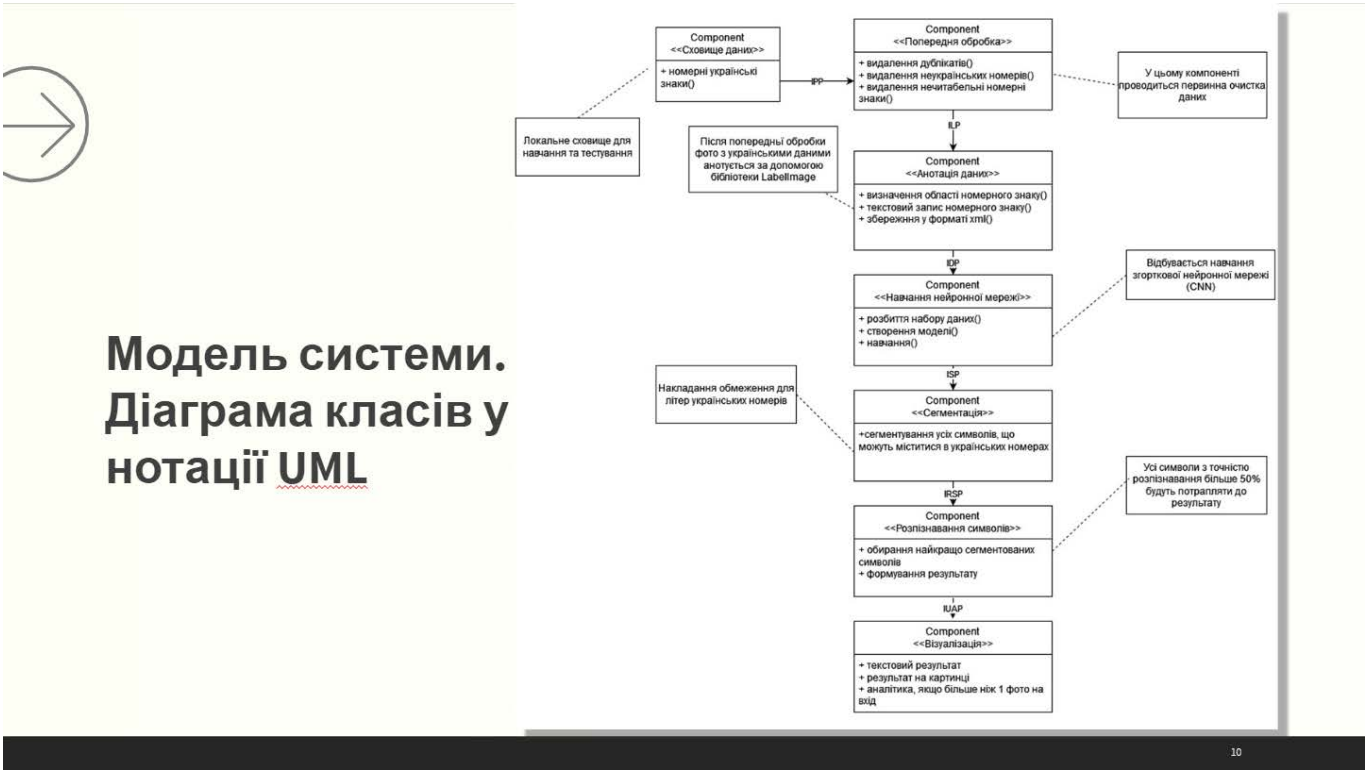
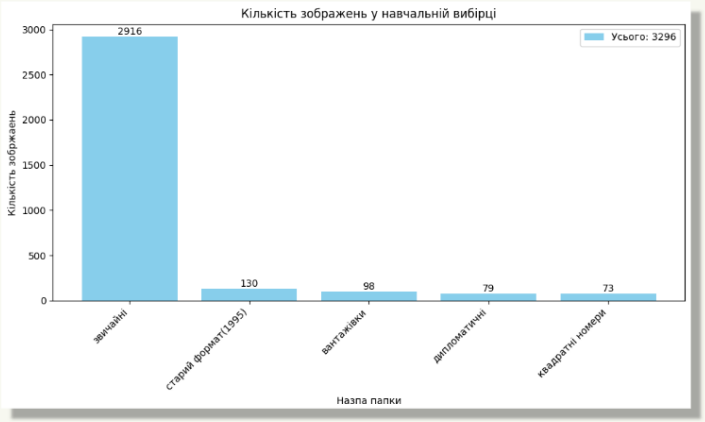


Рисунок Б.9 – Слайд 9

Рисунок Б.10 – Слайд 10

Огляд навчальної вибірки

1. Звичайний/стандартний тип
2. Старий тип
3. Квадратні/доврядкові
4. Вантажівки/автобуси
5. Дипломатичний тип



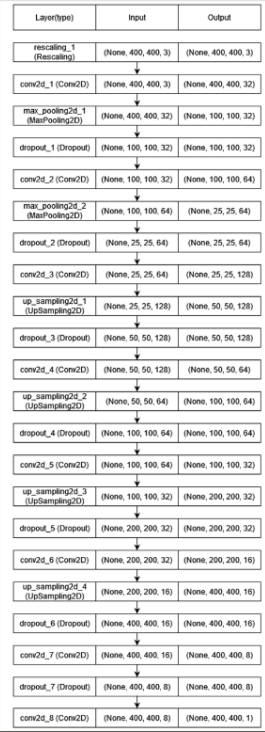
12

Рисунок Б.11 – Слайд 11

Детекція автомобільних номерів. Згорткова нейронна мережа

Білбліотеки:
tensorflow

Гіперпараметр	Значення
Висота цільового зображення	400
Ширина цільового зображення	400
Розмір пакета	32
Кількість тренувальної вибірки	2 778
Кількість тесту вальної вибірки	4 32
Коефіцієнт випадкового відключення (Dropout rate)	0.2
Оптимізатор	Adam
Функція втрат	Бінарна перехресна ентропія (Binary Crossentropy)
Метрики	Середня абсолютна похибка (Mean Absolute Error)
Кількість епох	30
Рання зупинка	Терпіння: 5, Відновлення кращих ваг: True
Зворотній виклик ModelCheckpoint	save_best_only: True



13

Рисунок Б.12 – Слайд 12

Оптичне розпізнавання символів (Optical character recognition)

Білбліотеки:
Pytesseract
numpy

Особливості:

До результуючого масиву даних потравляють тількі ті символи, у яких точність розпізнавання більше 50%.

Інші символи немають сенсу.

Параметр	Значення
Білий список tessedit_char_whitelist	Усі дозволені символи в українських номерних знаках ABCEHIKMNOPTXYZ0123456789
Чорний список tessedit_char_blacklist	Відсутній
Режим сегментації сторніки(psm)	6 (Припускається, що єдиний блок містить текст)
Режим роботи вдигуна(eom)	1 (Лише нейронна мережа LSTM)
Мова	English
Можливість пост обробки(nice)	0

14

Рисунок Б.13 – Слайд 13

Інтерфейс користувача

15

Рисунок Б.14 – Слайд 14

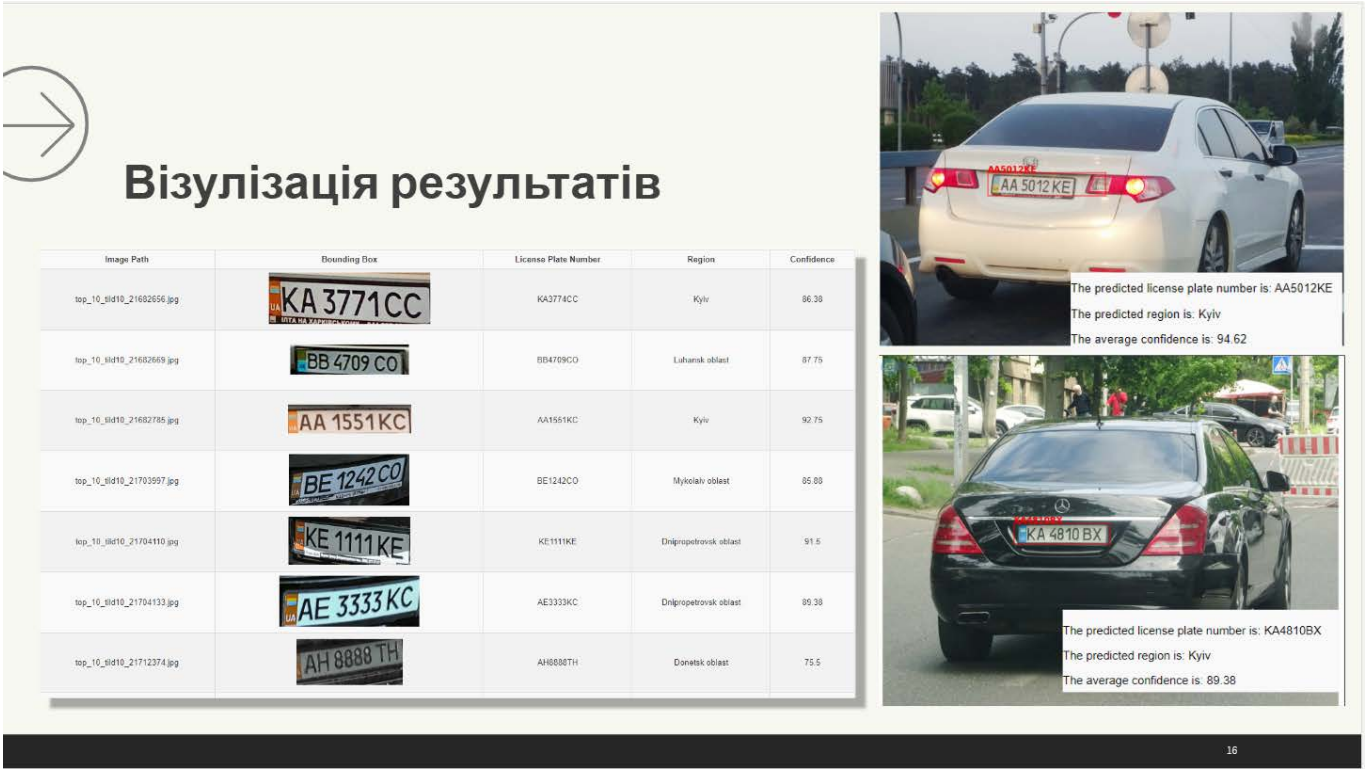


Рисунок Б.15 – Слайд 15

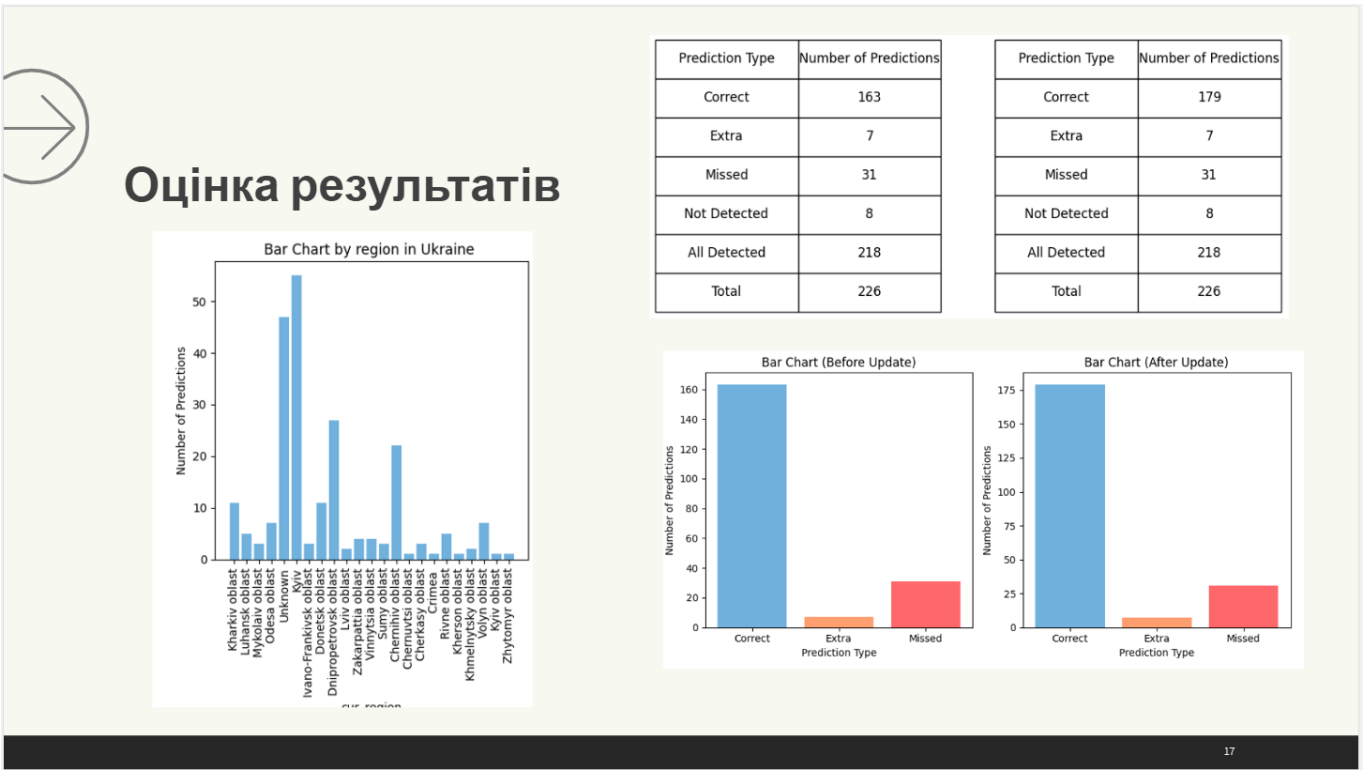


Рисунок Б.16 – Слайд 16

Верифікація та валідація

BR1.Кожне зображення у вибірці має мати свою обмежувальну рамку для об'єкту номерний знак

BR2.Виключити можливість «витоку даних» (Зображення у тестовій вибірці і тренувальній не мають повторюватись)

BR3. Навчальний набір має складатися тільки з українських номерних знаків.

BR4. Навчальний датасет має містити більше 3000 фотографій з різними складними випадками

BR5. Час детекції та розпізнавання не більше 5 секунд

BR6. Уникнути неправильного розпізнавання

BR7. Надання аналітики по результатам роботи системи

18

Рисунок Б.17 – Слайд 17

Висновки

У рамках виконаного проекту проведено детальний аналіз методів для детекції номерних знаків, алгоритмів сегментації символів та способів їх розпізнавання. Також було проведено огляд існуючих комерційних рішень для порівняння.

У результаті роботи розроблена модель системи виявлення та зчитування автомобільних номерних знаків у вигляді діаграми компонентів. Були встановлені зв'язки, які були представлені у вигляді інтерфейсів. Основним результатом є програмне забезпечення, яке здатне локалізувати область номерного знаку та розпізнати текст.

Система детекції імплементована на основі навченої згорткової нейронної мережі, а розпізнавання тексту номерних знаків використовувало оптичне розпізнання символів. Дані для навчання були попередньо оброблені згідно вимог та анотовані з урахуванням правил. Також був розроблений аналітичний модуль для користувача, який дозволяє візуально сприймати результати системи.

Система піддана верифікації та валідації. Було перевірено виконання поставлених функціональних вимог та бізнес-правил. Результати розпізнавання номерних знаків транспортних засобів відповідають очікуваній точності, яка не знижується нижче 80%. Система була оптимізована і мінімізує помилкові значення.

19

Рисунок Б.18 – Слайд 18

Перспективи подальших досліджень

SOON Реалізація класифікації регіонів за формами номерних знаків

SOON Реалізація класифікації типів номерних знаків за зовнішнім виглядом

20

Рисунок Б.19 – Слайд 19

Перелік посилань

1.Hikvision ANPR solutions [Електронний ресурс] – Режим доступу до ресурсу: https://www.hikvision.com/europe/products/software/HikCentral-Professional-series/HikCentral-ANPR/.

2. Axis license plate verifier [Електронний ресурс] – Режим доступу до ресурсу: https://www.axis.com/products/axis-license-plate-verifier/

3.Dahua technology anpr solution [Електронний ресурс] – Режим доступу до ресурсу: https://www.dahuasecurity.com/ceen/products/keyTechnologies/352

4.Hanwha techwin anpr solution [Електронний ресурс] – Режим доступу до ресурсу: https://hanwhavision.eu/launching-the-serverless-anpr-solution-for-traffic-management/.

5. Platesmania [Електронний ресурс] – Режим доступу до ресурсу: https://platesmania.com/.

6. Securos auto license plate recognition [Електронний ресурс] – Режим доступу до ресурсу: https://issivs.com/securos-auto/

7. Agarwal A., Goswami S., "An Efficient Algorithm for Automatic Car Plate. Second International Conference on Computational Intelligence & CommunicationTechnology", 2016.

8. Bensouilah M., Zennir M.Z., Taffar M., "An ALPR System-based Deep Networks for the Detection and Recognition".

9. Caner H., Selcuk Gecim H., Alkar A.Z. Efficient Embedded Neural-Network-Based License Plate Recognition System, 2008, Ocrober.

10. Maslianko, P. P., Sielskyi, Y. P., "МЕТОД СИСТЕМНОЇ ІНЖЕНЕРІЇ СИСТЕМ НЕЙРОННОГО МАШИННОГО ПЕРЕКЛАДУ", KPI Science News, 2021, August 31.

21

Рисунок Б.20 – Слайд 20