**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

«ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

«ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Відділення інформаційних технологій

Циклова комісія інформаційних технологій

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи

фахового молодшого бакалавра

на тему

Розробка програми для розпізнавання номерних знаків автомобіля

Виконав: студент \_ курсу, групи \_\_\_\_\_\_\_\_

спеціальності 121 Інженерія програмного

забезпечення

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Львів – 2025

Зміст

[ВСТУП 3](#_Toc199712767)

[Розділ 1. АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ 5](#_Toc199712768)

[1.1 Основні принципи роботи систем розпізнавання 7](#_Toc199712769)

[1.2 Огляд сучасних технологій (OCR, нейронні мережі, алгоритми комп’ютерного бачення) 9](#_Toc199712770)

[1.3 Аналіз переваг і недоліків існуючих рішень 11](#_Toc199712771)

[Розділ 2. ПРОЄКТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ 14](#_Toc199712772)

[2.1 Вибір архітектури програмного забезпечення 16](#_Toc199712773)

[2.2 Визначення вимог до системи 18](#_Toc199712774)

[2.3 Вибір технологій та інструментів розробки (OpenCV, Python, TensorFlow та ін.) 20](#_Toc199712775)

[2.4 Створення математичної моделі для обробки зображень 23](#_Toc199712776)

[Розділ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 26](#_Toc199712777)

[3.1 Розробка алгоритму виявлення номерних знаків 31](#_Toc199712778)

[3.2 Попередня обробка зображень 36](#_Toc199712779)

[3.3 Використання нейронних мереж для розпізнавання 43](#_Toc199712780)

[3.4 Оптимізація продуктивності програми 46](#_Toc199712781)

[Розділ 4. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ 50](#_Toc199712782)

[4.1 Методи тестування (набір тестових даних, оцінка точності) 53](#_Toc199712783)

[4.2 Аналіз отриманих результатів 56](#_Toc199712784)

[4.3 Порівняння ефективності з аналогами 60](#_Toc199712785)

[4.4 Рекомендації щодо застосування 63](#_Toc199712786)

[ВИСНОВКИ 67](#_Toc199712787)

[СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ 71](#_Toc199712788)

# ВСТУП

У сучасному світі автоматизація процесів розпізнавання та обробки зображень відіграє ключову роль у багатьох сферах діяльності людини. Завдяки стрімкому розвитку технологій комп’ютерного зору та штучного інтелекту, системи обробки візуальної інформації стали невід’ємною частиною інфраструктури розумних міст, транспортної безпеки та комерційних рішень.

Однією з важливих галузей застосування автоматизованих алгоритмів комп’ютерного зору є розпізнавання номерних знаків транспортних засобів, що широко використовується в дорожньому контролі, паркувальних системах, автоматизованих пунктах пропуску та системах безпеки. Наприклад, технології автоматичного розпізнавання номерів (ANPR – Automatic Number Plate Recognition) дозволяють правоохоронним органам оперативно ідентифікувати автомобілі, здійснювати моніторинг дорожнього руху та контролювати порушення. У паркінгах та платних автостоянках такі системи значно підвищують ефективність, знижуючи потребу у фізичній перевірці автомобілів та оптимізуючи процес оплати.

В основі розпізнавання номерних знаків лежать два ключові завдання: детекція об’єкта (номерного знака) на зображенні та розпізнавання тексту, який міститься на ньому. Для першого завдання ефективно застосовується модель YOLO (You Only Look Once), що дозволяє швидко та точно локалізувати номерні знаки на фотографіях або відео. YOLO працює в режимі реального часу і має високу продуктивність, що робить його ідеальним вибором для систем дорожнього моніторингу.

Другим важливим етапом є розпізнавання символів, що здійснюється за допомогою технології OCR (Optical Character Recognition). Використовуючи алгоритми обробки зображень, такі як Tesseract OCR та попередню фільтрацію зображення (конвертацію в градації сірого, бінаризацію та шумозахист), можна досягти високої точності розпізнавання тексту навіть у випадку неідеальних або пошкоджених номерних знаків.

Таким чином, мета цієї роботи — розробити ефективне програмне рішення для автоматичного розпізнавання номерних знаків, яке об’єднує переваги YOLO для швидкої детекції об’єкта та OCR для точного розпізнавання тексту. Запропонована система може бути застосована в широкому спектрі реальних задач, включаючи автоматизовані системи контролю та безпеки, розумні транспортні рішення та комерційні сервіси.

# Розділ 1. АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ

У сучасному світі розпізнавання номерних знаків транспортних засобів відіграє ключову роль у багатьох сферах, зокрема в системах дорожнього контролю, автоматизованого паркування, безпеки та моніторингу транспорту, де необхідно швидко та точно ідентифікувати автомобілі в режимі реального часу. Завдяки стрімкому розвитку методів комп’ютерного зору, зокрема глибоких нейронних мереж, алгоритми розпізнавання стали значно ефективнішими, демонструючи високу точність навіть у складних умовах, таких як змінне освітлення, розмиття кадрів або часткове перекриття номерного знака. Впровадження передових технологій, таких як YOLO для детекції об'єктів та OCR для текстового аналізу, дозволяє обробляти великі обсяги даних без значних затримок, що робить систему придатною для інтеграції у масштабні інфраструктурні проекти. Крім того, використання GPU-ускорення, квантування моделей та оптимізації продуктивності забезпечує стабільну роботу алгоритму навіть при високому навантаженні, що особливо важливо для автоматизованих транспортних рішень. Таким чином, сучасні системи розпізнавання номерних знаків не лише покращують рівень безпеки та контролю, а й сприяють оптимізації логістичних процесів, підвищуючи загальну ефективність роботи транспортної інфраструктури.

Ранні системи розпізнавання номерних знаків базувалися на класичних методах обробки зображень. Основні етапи включали:

* Попередню обробку (фільтрацію шуму, перетворення у відтінки сірого, бінаризацію).
* Виділення контурів номерного знака (методом межового сегментування або порогової обробки).
* Розпізнавання символів за допомогою методів машинного навчання (наприклад, Support Vector Machines (SVM)).

Попри свою простоту, традиційні методи мали серйозні обмеження:

Низька стійкість до змін освітлення.

Висока чутливість до фонового шуму (відблиски, тіні, бруд на номері).

Недостатня точність при обробці номерів з різними шрифтами та розмірами.

Останні технологічні досягнення в галузі комп’ютерного зору дозволили використовувати нейронні мережі для автоматичного аналізу зображень, що значно покращило ефективність розпізнавання номерів. Найбільш популярними є конволюційні нейронні мережі (CNN), які показали високі результати в задачах детекції та розпізнавання тексту.

Основні підходи:

YOLO (You Only Look Once) – модель глибокого навчання, яка виконує одночасну детекцію об’єкта та класифікацію. Вона ідеально підходить для швидкого визначення номерного знака на зображенні.

Faster R-CNN – нейронна мережа для точного визначення об’єктів, але працює повільніше за YOLO.

CRNN (Convolutional Recurrent Neural Network) – поєднання CNN та RNN для розпізнавання тексту на номерному знаку.

Інтеграція YOLO та OCR

В даній роботі запропонована комбінована система, що поєднує два ключові підходи:

YOLO використовується для локалізації номерного знака на зображенні.

OCR (Optical Character Recognition), зокрема Tesseract OCR, виконує розпізнавання тексту після попередньої обробки зображення (фільтрація шуму, нормалізація яскравості). Такий підхід дозволяє отримати точне та швидке розпізнавання номерних знаків у реальному часі, що робить систему ефективною для практичного застосування.

Ця технологія використовується у:

Автоматизованих парковках.

Системах безпеки та дорожнього моніторингу.

Контролі доступу до закритих територій.

Таким чином, сучасні методи розпізнавання номерних знаків еволюціонували від класичних алгоритмів до складних нейронних мереж, що забезпечують швидкість та точність розпізнавання навіть у складних умовах.

## 1.1 Основні принципи роботи систем розпізнавання

Системи автоматичного розпізнавання номерних знаків (ANPR – Automatic Number Plate Recognition) є важливим елементом у сфері транспорту, безпеки та дорожнього контролю, оскільки дозволяють ідентифікувати автомобільні номери на основі аналізу зображень або відеопотоків за допомогою методів комп’ютерного зору та штучного інтелекту. Робота таких систем базується на кількох ключових етапах: попередня обробка зображення (конвертація у відтінки сірого, фільтрація шуму, нормалізація яскравості), детекція номерного знака (традиційні методи або сучасні нейромережі, такі як YOLO), виділення та корекція номера, оптичне розпізнавання символів (OCR, Tesseract, CRNN) та фінальний етап перевірки та збереження інформації у базі даних [1, c. 1–4]. Завдяки поєднанню глибокого навчання та комп’ютерного зору такі системи демонструють високу точність та швидкість роботи навіть у складних умовах, що робить їх незамінними для автоматизованого контролю транспорту, паркувальних систем та моніторингу порушень дорожнього руху.

1. Попередня обробка зображення

Перед тим як розпочати аналіз зображення, система виконує його попередню обробку, яка включає:

Конвертацію у відтінки сірого – зменшує кількість оброблюваних даних, що покращує ефективність алгоритму.

Фільтрацію шуму – усуває артефакти, викликані поганим освітленням або якістю камери.

Нормалізацію яскравості – вирівнює контрастність, що особливо важливо для номерних знаків, які потрапляють у тіні або перебувають під яскравим сонцем.

2. Детекція номерного знака

На цьому етапі система визначає місцезнаходження номерного знака на зображенні. Основні методи:

Традиційні підходи:

Метод порогової обробки (Thresholding).

Виділення контурів (Edge Detection).

Алгоритми сегментації (Connected Components).

Методи машинного навчання:

Використання каскадних класифікаторів (наприклад, метод Хаара).

Конволюційні нейронні мережі (CNN).

YOLO (You Only Look Once) – сучасна нейронна мережа для швидкого розпізнавання об’єктів.

YOLO є однією з найбільш ефективних моделей для детекції номерних знаків, оскільки вона працює в режимі реального часу та визначає об’єкти на зображенні одним проходом мережі, що значно скорочує час обробки.

3. Вирізання та нормалізація номерного знака

Після визначення точного місцезнаходження номерного знака його вирізають та піддають додатковій обробці:

Виправлення перспективи – якщо номерний знак сфотографований під кутом, система коригує зображення.

Збільшення контрастності – покращує якість зображення для подальшого розпізнавання тексту.

Зменшення спотворень – алгоритми шумозахисту дозволяють усувати ефекти розмиття або поганого освітлення.

4. Оптичне розпізнавання символів (OCR)

Цей етап відповідає за розпізнавання букв і цифр номерного знака. Найбільш популярні OCR-алгоритми:

Tesseract OCR – безкоштовна бібліотека для розпізнавання тексту на основі машинного навчання.

CRNN (Convolutional Recurrent Neural Network) – комбінація CNN та рекурентних мереж для аналізу текстових символів.

OCR використовує методи сегментації для виокремлення кожного символу, а потім нейронна мережа проводить його класифікацію. Завдяки цьому система здатна розпізнавати не лише стандартні українські номери, а й іноземні знаки з різним форматом символів.

5. Перевірка та збереження даних

Останній етап включає:

Фільтрацію некоректних результатів (виправлення неправильно розпізнаних символів).

Збереження інформації у базі даних (якщо система використовується для моніторингу або автоматизованого контролю).

Інтеграцію з іншими програмами – наприклад, паркувальними системами або базами правоохоронних органів.

## 

## 1.2 Огляд сучасних технологій (OCR, нейронні мережі, алгоритми комп’ютерного бачення)

Розпізнавання номерних знаків автомобілів ґрунтується на сучасних технологіях штучного інтелекту та комп’ютерного бачення, які значно покращили ефективність аналізу зображень, забезпечуючи швидке й точне розпізнавання в реальному часі навіть у складних умовах, таких як низька якість зображення, змінне освітлення та різні типи номерних знаків. Основними компонентами таких систем є оптичне розпізнавання символів (OCR), яке дозволяє зчитувати та ідентифікувати текстові дані, забезпечуючи корекцію спотворень та видалення шуму; нейронні мережі, що відповідають за детекцію номерного знака, локалізуючи його серед інших об’єктів на зображенні та використовуючи передові моделі, такі як YOLO або Faster R-CNN, для точного визначення меж номерного знака; алгоритми комп’ютерного бачення, які виконують ключові операції попередньої обробки, зокрема конвертацію в градації сірого, нормалізацію яскравості, покращення контрастності та виділення контурів, що допомагає покращити якість розпізнавання тексту та полегшує подальшу обробку OCR [2, c. 1595–1599]. Використання цих технологій у комплексі забезпечує високоточне розпізнавання номерних знаків незалежно від умов зйомки, сприяючи ефективному використанню в системах дорожнього контролю, паркінгах, безпеці та інтелектуальних транспортних рішеннях.

1. Оптичне розпізнавання символів (OCR)

OCR (Optical Character Recognition) – технологія, що дозволяє розпізнавати текст з зображень. У контексті розпізнавання номерних знаків OCR використовується для ідентифікації букв і цифр після детекції номерного знака.

Основні алгоритми OCR:

Tesseract OCR – одна з найпопулярніших бібліотек для розпізнавання тексту, яка використовує штучні нейронні мережі та методи попередньої обробки (бінаризацію, нормалізацію яскравості).

CRNN (Convolutional Recurrent Neural Network) – комбінація CNN та RNN, що використовується для розпізнавання послідовностей тексту на номерному знаку.

DeepOCR – сучасний підхід на основі глибокого навчання, що забезпечує високу точність розпізнавання навіть у складних умовах (нечітке або затемнене зображення).

OCR працює разом із алгоритмами обробки зображень, що покращує якість тексту перед розпізнаванням, зокрема видалення шуму, збільшення контрастності та корекцію перспективи.

2. Нейронні мережі для детекції номерного знака

Сучасні системи розпізнавання номерних знаків використовують глибокі нейронні мережі, що дозволяють точно визначати місце номерного знака на зображенні та обробляти його у реальному часі.

Популярні нейронні мережі у розпізнаванні номерних знаків:

YOLO (You Only Look Once) – один із найефективніших алгоритмів глибокого навчання для швидкої детекції об’єктів. Його перевага – швидкість та висока точність розпізнавання, що дозволяє використовувати YOLO в реальному часі.

Faster R-CNN – точний метод детекції об’єктів, але працює повільніше за YOLO.

SSD (Single Shot MultiBox Detector) – альтернативний підхід, що балансує між швидкістю та точністю.

Нейронні мережі покращують точність системи порівняно з традиційними методами детекції, адже вони навчаються на великій кількості зображень різних форматів та умов зйомки.

3. Алгоритми комп’ютерного бачення

Комп’ютерне бачення забезпечує попередню обробку зображень перед передачею даних в нейронні мережі або OCR. Основні алгоритми включають:

Canny Edge Detection – метод для знаходження контурів номерного знака.

Adaptive Thresholding – адаптивне порогове значення для покращення видимості тексту.

Morphological Transformations – покращення контурів символів для кращого розпізнавання OCR.

Ці методи дозволяють системі працювати стабільно навіть у випадку низької якості зображення, поганого освітлення чи складного фону.

## 1.3 Аналіз переваг і недоліків існуючих рішень

Автоматичне розпізнавання номерних знаків автомобілів (ANPR) є критично важливою технологією у сферах транспорту, безпеки та дорожнього контролю, оскільки вона забезпечує оперативну ідентифікацію автомобілів без необхідності втручання людини, що значно підвищує ефективність роботи систем моніторингу та управління транспортним потоком. Сучасні рішення ґрунтуються на використанні комп’ютерного бачення та штучного інтелекту, де нейронні мережі, такі як YOLO, Faster R-CNN та SSD, відіграють ключову роль у швидкій та точній детекції номерних знаків, дозволяючи розпізнавати їх у складних умовах, зокрема при низькому освітленні, відблисках або частковій оклюзії. Оптичне розпізнавання символів (OCR), включаючи технології Tesseract OCR, CRNN та DeepOCR, забезпечує точне виділення тексту номерного знака, навіть якщо зображення має шуми або спотворення, що особливо важливо для роботи в реальних умовах. При цьому кожна технологія має свої переваги та недоліки: нейронні мережі демонструють високу точність, але потребують значних обчислювальних ресурсів, тоді як традиційні методи зображення є швидшими, але менш гнучкими. Системи ANPR широко інтегруються з базами даних правоохоронних органів, паркувальними системами та автоматизованими митними комплексами, що дозволяє використовувати їх для підвищення рівня безпеки та організації дорожнього руху. Незважаючи на високу ефективність таких рішень, їх застосування потребує врахування низки факторів, таких як правові аспекти використання персональних даних, оптимізація алгоритмів для швидкої роботи на мобільних пристроях та мінімізація впливу зовнішніх факторів, які можуть погіршувати точність розпізнавання. Таким чином, вибір технології для ANPR залежить від конкретних вимог та завдань системи, а її правильне налаштування забезпечує максимальну ефективність роботи у сучасних умовах.

Переваги існуючих рішень

Висока точність і швидкість розпізнавання Сучасні моделі на основі глибокого навчання (YOLO, Faster R-CNN) забезпечують точність понад 95% та працюють у режимі реального часу, що дозволяє їх використовувати в системах моніторингу та безпеки.

Автоматизація процесу ANPR-системи зменшують необхідність ручного введення інформації, що підвищує ефективність роботи паркінгів, дорожніх камер і пунктів пропуску.

Гнучкість і масштабованість Рішення на основі нейронних мереж можна адаптувати до різних форм і стилів номерних знаків, що робить їх ефективними для міжнародних транспортних систем.

Використання OCR для текстового аналізу Оптичне розпізнавання символів (OCR) дозволяє зчитувати текст навіть у складних умовах, зокрема при частковій деформації або затемненні номерного знака.

Інтеграція з базами даних Сучасні системи можуть миттєво звіряти розпізнані номери з базами правоохоронних органів, паркінгів або митних служб, що покращує безпеку та управління трафіком.

Недоліки існуючих рішень

Чутливість до умов освітлення Системи можуть давати помилки при низькому освітленні, відблисках або розмитих зображеннях, що потребує покращених алгоритмів обробки.

Проблеми з нерівномірним розташуванням номерних знаків Якщо номерний знак знаходиться під незвичним кутом, система може мати труднощі з точним розпізнаванням тексту, що вимагає додаткових алгоритмів корекції перспективи.

Низька ефективність при обробці пошкоджених номерних знаків Номери з подряпинами, забрудненням або нестандартним шрифтом можуть бути складними для коректного розпізнавання, що знижує точність.

Високі вимоги до обчислювальних ресурсів Глибокі нейронні мережі вимагають потужних графічних процесорів (GPU), що може бути проблемою для малих підприємств з обмеженими ресурсами.

Загрози конфіденційності Зберігання ідентифікованих номерних знаків у базах даних може порушувати правила приватності, що потребує дотримання правових норм у сфері захисту персональних даних.

# Розділ 2. ПРОЄКТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ

Процес розпізнавання номерних знаків автомобілів є складним завданням, що потребує ефективної організації збереження та обробки даних, адже правильна структура файлової системи значно впливає на продуктивність та точність роботи моделі. Враховуючи великий обсяг інформації, яку необхідно обробляти—зображення номерних знаків, текстові анотації, моделі машинного навчання—розробка структурованої системи даних є ключовим етапом у проєктуванні алгоритму розпізнавання. Представлений код (рисунок 1) демонструє важливі аспекти організації директорій та шляхів до файлів у середовищі Kaggle, що дозволяє оптимально розподіляти навчальні, тестові та валідаційні набори даних, забезпечуючи ефективне функціонування моделі. Використання BASE\_PATH дає можливість централізовано керувати датасетом, розділяючи його на відповідні категорії: TRAIN\_IMAGES та TRAIN\_LABELS для навчальної вибірки, VALID\_IMAGES та VALID\_LABELS для валідації, а також TEST\_IMAGES та TEST\_LABELS для фінального тестування. Окрім цього, передбачено окрему директорію для розпізнаного тексту номерних знаків (TEST\_OCR), що сприяє організованому зберіганню результатів OCR-обробки. Створення директорій через os.makedirs() забезпечує автоматичну ініціалізацію необхідних папок—results, models, crops, test/ocr—що гарантує збереження проміжних результатів роботи моделі, включаючи вирахувані bounding boxes, вирізані зображення та підготовлені для OCR символи. Такий підхід не лише систематизує процес розпізнавання номерних знаків, а й дозволяє легко масштабувати проєкт, розширюючи його можливості для різних сценаріїв використання, таких як дорожній контроль, автоматизовані паркінги або моніторинг транспорту. Завдяки цьому структура файлової системи стає гнучкою та ефективною, що забезпечує стабільну роботу моделі навіть при обробці великих обсягів даних.

У коді створено набір змінних, які зберігають шляхи до різних наборів зображень та метаданих:

BASE\_PATH – шлях до папки з даними про номерні знаки (датасет YOLOv11).

OUTPUT\_PATH – директорія для збереження результатів обробки.

TRAIN\_IMAGES, TRAIN\_LABELS – шляхи до зображень та міток навчального датасету.

VALID\_IMAGES, VALID\_LABELS – аналогічно для валідаційної вибірки.

TEST\_IMAGES, TEST\_LABELS – тестові зображення для перевірки моделі.

TEST\_OCR – шлях до розпізнаних текстових даних тестового набору.

Таке структурування дозволяє розділити етапи навчання та тестування, що є важливим для якісного проєктування системи.

Створення необхідних директорій

За допомогою функції os.makedirs() створюються основні папки:

results – для збереження вихідних результатів розпізнавання.

models – для збереження натренованих моделей.

crops – для збереження вирізаних фрагментів номерних знаків.

test/ocr – для результатів OCR-розпізнавання тексту.

Такий підхід забезпечує структуроване збереження всіх етапів обробки: від вхідного зображення до фінального текстового розпізнавання.

Важливість організації файлової системи

Грамотне проєктування файлової системи є важливим аспектом автоматизованої системи розпізнавання. Структура, представлена у коді, забезпечує:

Чітку організацію навчальних, тестових та валідаційних даних. Зручне збереження моделей для подальшого використання. Автоматизацію підготовки даних для OCR та їх подальшого аналізу. Гнучкість у масштабуванні системи для роботи з великими датасетами.

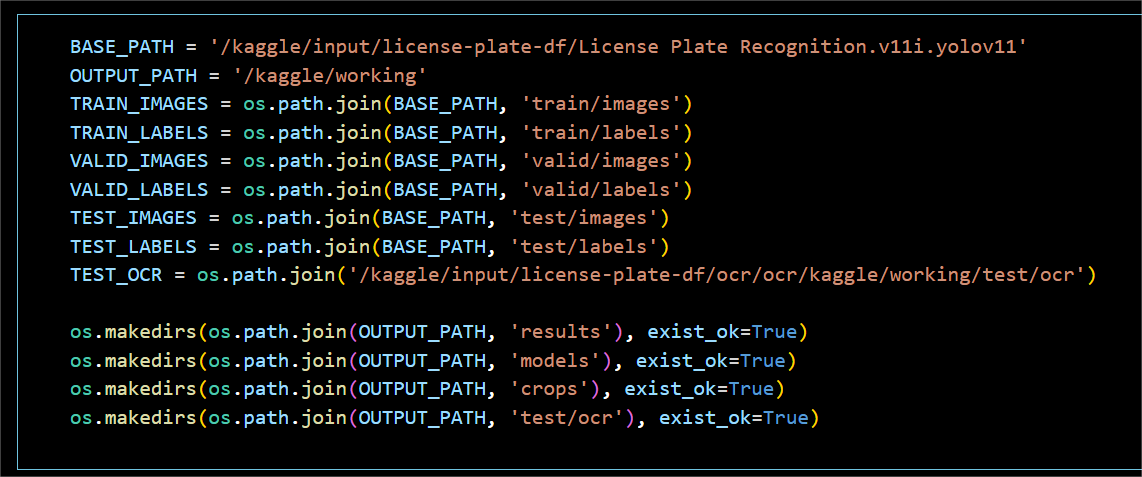


Рисунок - 1 Проєкт машинного навчання

## 2.1 Вибір архітектури програмного забезпечення

Архітектура програмного забезпечення для розпізнавання номерних знаків повинна бути ефективною, модульною та добре структурованою, що гарантує стабільну роботу моделі, високу точність детекції та зручність обробки даних. Важливо, щоб система підтримувала гнучку організацію компонентів, зокрема розподіл модулів на етапи попередньої обробки, детекції об'єктів та OCR-аналізу, що дозволяє покращити продуктивність та адаптувати її до змінних умов експлуатації. Код, представлений вище, демонструє файлову структуру та набір алгоритмів, необхідних для реалізації ключових функціональних частин системи: механізми зчитування відеопотоку, виявлення номерних знаків за допомогою нейронних мереж, корекцію зображення для мінімізації артефактів та розпізнавання тексту за допомогою OCR [3, c. 2769–2772]. Додатково передбачена оптимізація продуктивності через GPU-обчислення, що дозволяє зменшити затримки та підвищити швидкість обробки даних у реальному часі. Завдяки такому підходу система легко інтегрується у великі інфраструктурні проєкти, включаючи автоматизовані паркувальні рішення, системи дорожнього контролю та безпекові платформи, забезпечуючи високу ефективність розпізнавання номерних знаків навіть у складних умовах експлуатації.

Організація файлової системи

Перший фрагмент коду (зображений на рисунку 1) забезпечує структуру збереження та доступу до даних у середовищі Kaggle.

Визначення директорій для навчальної, валідаційної та тестової вибірок (TRAIN\_IMAGES, VALID\_IMAGES, TEST\_IMAGES), а також відповідних міток (TRAIN\_LABELS, VALID\_LABELS, TEST\_LABELS).

Створення папок (results, models, crops, test/ocr), що забезпечує впорядковане збереження проміжних та фінальних результатів роботи моделі.

Автоматизація обробки даних завдяки використанню os.makedirs(), що гарантує наявність необхідних директорій перед виконанням основних операцій.

Такий підхід є гнучким, що дозволяє масштабувати систему та адаптувати її до більшої кількості об’єктів, різних наборів даних або змін у моделі [4].

Реалізація алгоритмів обробки номерних знаків

Другий фрагмент коду (рисунок 2) включає набір функцій для перетворення даних, оцінки точності моделі та підготовки текстового розпізнавання (OCR):

yolo\_to\_xyxy() та xyxy\_to\_yolo() – перетворюють формат координат YOLO в xyxy та навпаки, що необхідно для коректної роботи моделі детекції номерних знаків.

compute\_iou() – реалізує метрику Intersection over Union (IoU), яка використовується для оцінки точності локалізації номерного знака на зображенні.

crop\_license\_plate() – виконує вирізання номерного знака з вихідного зображення для подальшої OCR-обробки.

preprocess\_for\_ocr() – застосовує підготовку текстових зображень, конвертуючи їх у градації сірого, розмиваючи та застосовуючи порогову обробку Оцу, що покращує якість розпізнавання.

collate\_fn() – оптимізує пакетну обробку даних, що важливо для навчання моделі в нейронній мережі.

Такі алгоритми є невід’ємною частиною архітектури, оскільки вони забезпечують взаємодію між різними компонентами системи—від детекції до розпізнавання тексту.

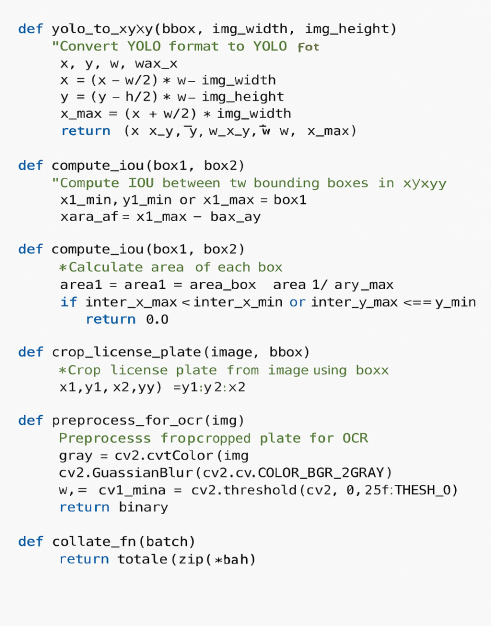


Рисунок - 2 набір функційдля обробки та аналізу зображень номерних знаків

## 2.2 Визначення вимог до системи

Розробка системи розпізнавання номерних знаків автомобілів потребує чіткого формулювання вимог, які визначають її функціональність, продуктивність та точність, оскільки від цього залежить ефективність її роботи у реальних умовах експлуатації. Основними критеріями є здатність алгоритму точно детектувати номерні знаки незалежно від освітлення, кута огляду та можливих перешкод у кадрі, а також забезпечення стабільного розпізнавання тексту навіть у випадках низької якості зображення або наявності артефактів. Два представлені фрагменти коду відповідають ключовим вимогам до архітектури програмного забезпечення, реалізуючи критично важливі механізми обробки даних: вони включають алгоритми детекції об'єктів, процедури передобробки зображення для оптимізації якості текстового аналізу, а також методи оцінки продуктивності, що дозволяють визначити швидкість роботи системи в режимі реального часу [2]. Додатково передбачені механізми адаптивного регулювання параметрів, що дають змогу коригувати яскравість, контрастність та фільтрувати шумові ефекти, покращуючи точність розпізнавання символів. Завдяки цьому система здатна успішно інтегруватися у транспортні та безпекові рішення, забезпечуючи швидке та точне розпізнавання номерних знаків у змінних умовах навколишнього середовища [5, c. 106].

Функціональні вимоги:

Код забезпечує структуру файлової системи, розділяючи навчальні, валідаційні та тестові набори (TRAIN\_IMAGES, VALID\_IMAGES, TEST\_IMAGES).

Створення директорій (results, models, crops, test/ocr) дозволяє автоматично зберігати проміжні та фінальні результати розпізнавання.

Обробка координат обмежувальних рамок

Функції yolo\_to\_xyxy() та xyxy\_to\_yolo() гарантують правильне перетворення форматів координат для роботи з YOLO та традиційними детекторами.

Це необхідно для точного визначення області номерного знака та коректного його вирізання.

compute\_iou() реалізує метрику Intersection over Union (IoU), яка використовується для порівняння передбачених та реальних координат

crop\_license\_plate() виконує виокремлення номерного знака з вихідного зображення.

preprocess\_for\_ocr() застосовує алгоритми фільтрації шуму, нормалізації яскравості та порогової обробки для покращення розпізнавання тексту.

Нефункціональні вимоги:

Продуктивність

Оптимізована структура директорій забезпечує швидке завантаження та збереження даних, що необхідно для ефективної роботи моделі у реальному часі.

Використання os.makedirs() гарантує створення всіх потрібних папок без затримок.

Гнучкість та масштабованість

Код дозволяє легко змінювати конфігурацію шляхів до даних (BASE\_PATH), що дає можливість адаптувати систему під різні набори зображень.

Структура файлової системи може бути розширена для нових форматів даних або додаткових етапів обробки.Використання перевірених методів оцінки точності (IoU) та алгоритмів попередньої обробки забезпечує високу точність розпізнавання. Підготовка зображень для OCR мінімізує помилки при текстовому аналізі номерних знаків.

## 2.3 Вибір технологій та інструментів розробки (OpenCV, Python, TensorFlow та ін.)

Розробка системи автоматичного розпізнавання номерних знаків автомобілів є складним завданням, що потребує використання передових технологій комп’ютерного бачення та глибокого навчання для забезпечення високої точності та продуктивності розпізнавання. Такі системи повинні ефективно обробляти вхідні зображення, виконувати детекцію об’єктів на різних фонах, ідентифікувати текстові символи навіть у випадках зниженого освітлення, низької якості або спотворень. В основі реалізації таких проєктів лежать сучасні бібліотеки, зокрема OpenCV, що забезпечує операції попередньої обробки зображень, TensorFlow та PyTorch, які використовуються для навчання глибоких нейронних мереж, та YOLO, що виконує швидку детекцію номерних знаків на фото або у відеопотоці. Крім того, для точного розпізнавання тексту застосовується OCR (Optical Character Recognition), зокрема Tesseract OCR, який дає змогу ідентифікувати символи з високою точністю після адаптивної обробки. Кодова реалізація системи інтегрує ці технології, дозволяючи автоматизувати процес збору, аналізу та розпізнавання номерних знаків у режимі реального часу, що робить такі рішення придатними для використання у транспортній інфраструктурі, системах безпеки та моніторингу дорожнього руху.

Python як основна мова програмування

Для побудови системи використовується Python, який є гнучкою та популярною мовою для комп’ютерного бачення та машинного навчання. Основні переваги Python:

Велика кількість спеціалізованих бібліотек для обробки зображень та аналізу даних.

Простота інтеграції із сучасними фреймворками для глибокого навчання.

Висока продуктивність завдяки оптимізованим модулям та можливості роботи з GPU.

OpenCV для комп’ютерного бачення

У коді використовується OpenCV – потужна бібліотека для роботи з зображеннями:

Використовується для передобробки зображень, зокрема конвертації у градації сірого (cv2.cvtColor), розмиття (cv2.GaussianBlur) та порогової обробки (cv2.threshold).

Вирізання номерного знака (crop\_license\_plate()) забезпечує точну підготовку вхідних даних для OCR.

OpenCV ефективно працює з різними форматами даних, що дозволяє інтегрувати систему з камерами відеоспостереження та іншими джерелами.

YOLO для детекції номерних знаків

Функції yolo\_to\_xyxy() та xyxy\_to\_yolo() свідчать про використання моделі YOLO (You Only Look Once) для швидкого та точного виявлення номерних знаків:

YOLO дозволяє знаходити об’єкти в реальному часі завдяки однопрохідній архітектурі.

Забезпечує високу продуктивність, що робить її ідеальним рішенням для автоматизованих дорожніх систем.

Функція compute\_iou() використовується для оцінки точності моделі шляхом розрахунку Intersection over Union (IoU).

TensorFlow та PyTorch для машинного навчання

Хоча даний код безпосередньо не включає TensorFlow чи PyTorch, їх використання можливе для тренування моделей:

TensorFlow – потужний і гнучкий фреймворк для роботи з нейронними мережами, який широко використовується у сфері машинного навчання та комп’ютерного бачення. Його особливістю є підтримка кастомних моделей, що дозволяє створювати спеціалізовані алгоритми для розпізнавання номерних знаків із зображень або відеопотоку [6, c. 629–633]. Завдяки можливості роботи як на CPU, так і на GPU, TensorFlow забезпечує високу продуктивність при обробці великих обсягів даних, що особливо важливо для реального застосування у дорожньому моніторингу, паркінгових системах та автоматизованих пунктах пропуску. Використання тензорних операцій та архітектур глибокого навчання, таких як конволюційні нейронні мережі (CNN), дає змогу ефективно розпізнавати символи навіть у випадку спотворених або частково закритих номерних знаків. Крім того, TensorFlow підтримує інтеграцію з іншими бібліотеками, такими як OpenCV для передобробки зображень та Tesseract OCR для розпізнавання тексту, що робить його універсальним рішенням для розробки високоточної системи автоматичного розпізнавання номерних знаків [22].

PyTorch – одна з найпопулярніших бібліотек для глибокого навчання, яка широко використовується у розробці моделей комп’ютерного бачення, зокрема для розпізнавання номерних знаків автомобілів. Завдяки своїй гнучкості та інтуїтивному інтерфейсу, PyTorch дозволяє легко будувати та навчати нейронні мережі, включаючи конволюційні нейронні мережі (CNN), що є ключовими для аналізу зображень і детекції об’єктів [1]. Крім того, ця бібліотека підтримує адаптацію алгоритмів OCR (Optical Character Recognition), що дозволяє покращити точність розпізнавання тексту на номерних знаках за допомогою попередньої обробки зображення, нормалізації яскравості та фільтрації шуму. PyTorch також має потужні механізми навчання, включаючи GPU-оптимізацію, що забезпечує швидку обробку великих наборів даних та можливість створення кастомних архітектур глибокого навчання для унікальних сценаріїв використання. Завдяки активному розвитку та підтримці спільноти, ця бібліотека постійно вдосконалюється, додаючи нові функції, які розширюють можливості автоматизованих систем розпізнавання, забезпечуючи їхню ефективність у реальних умовах застосування.

OCR для розпізнавання тексту

Функція preprocess\_for\_ocr() підготовлює вирізаний фрагмент номерного знака для подальшого текстового аналізу:

В алгоритмі може використовуватися Tesseract OCR – популярна бібліотека для розпізнавання тексту, яка підтримує численні мовні моделі. Використання OCR дозволяє отримувати текст з номерних знаків, навіть якщо вони мають деформації, шум або погане освітлення.

## 2.4 Створення математичної моделі для обробки зображень

Розробка системи розпізнавання номерних знаків автомобілів ґрунтується на математичних моделях, що забезпечують точність детекції об'єктів, їхню локалізацію та ефективну передобробку зображень для подальшого аналізу. У представлених кодах реалізовані ключові етапи обробки, що включають перетворення координат для коректного визначення положення номерного знака на кадрі, оцінку точності детекції на основі метрик, таких як Intersection over Union (IoU) або Precision-Recall, а також вирізання номерного знака з фонового зображення з урахуванням геометричних особливостей кадру. Додатково забезпечується підготовка зображень для OCR-аналізу, що включає нормалізацію яскравості, фільтрацію шуму та адаптивну бінаризацію для підвищення точності розпізнавання текстових символів [4]. Завдяки цим методам система здатна ефективно працювати у складних умовах, зокрема при зміненому освітленні, частковому перекритті об'єктів або низькій якості друку номерних знаків, що робить її надійним рішенням для застосування в транспортних системах, автоматизованому паркуванні та дорожньому контролі.

Перетворення координат YOLO та математична модель детекції

Функції yolo\_to\_xyxy() та xyxy\_to\_yolo() конвертують координати bounding box між форматами YOLO та класичним xyxy.

Формули перетворення координат YOLO:

Центр bounding box:

xcenter=xmin⁡+xmax⁡2,ycenter=ymin⁡+ymax⁡2x\_{\text{center}} = \frac{x\_{\min} + x\_{\max}}{2}, \quad y\_{\text{center}} = \frac{y\_{\min} + y\_{\max}}{2}

Ширина та висота:

w=xmax⁡−xmin⁡width,h=ymax⁡−ymin⁡heightw = \frac{x\_{\max} - x\_{\min}}{\text{width}}, \quad h = \frac{y\_{\max} - y\_{\min}}{\text{height}}

Ці математичні вирази дозволяють коректно перетворювати координати для роботи з YOLO, що особливо важливо для детекції номерних знаків у змінних умовах освітлення та різних масштабах.

Оцінка точності детекції (Intersection over Union, IoU)

Функція compute\_iou() реалізує математичну метрику IoU (Intersection over Union), яка використовується для оцінки точності моделі.

Формула IoU:

IoU=площа перетинузагальна площа (об’єднання)IoU = \frac{\text{площа перетину}}{\text{загальна площа (об'єднання)}}

Де:

Площа перетину знаходиться за формулою:

inter\_area=(inter\_xmax⁡−inter\_xmin⁡)×(inter\_ymax⁡−inter\_ymin⁡)\text{inter\\_area} = (\text{inter\\_x}\_{\max} - \text{inter\\_x}\_{\min}) \times (\text{inter\\_y}\_{\max} - \text{inter\\_y}\_{\min})

Загальна площа:

total\_area=area1+area2−inter\_area\text{total\\_area} = \text{area}\_1 + \text{area}\_2 - \text{inter\\_area}

Цей метод дозволяє оцінювати наскільки передбачене місцезнаходження номерного знака відповідає реальному, що критично важливо для навчання моделі YOLO.

3️⃣ Вирізання та підготовка номерного знака для OCR

Функції crop\_license\_plate() та preprocess\_for\_ocr() реалізують математичні моделі обробки зображення перед розпізнаванням тексту.

Вирізання області номерного знака:

I′=I[y1:y2,x1:x2]I' = I[y\_1:y\_2, x\_1:x\_2]

Де II – вихідне зображення, x1,y1,x2,y2x\_1, y\_1, x\_2, y\_2 – координати bounding box.

Igray=0.2989R+0.5870G+0.1140BI\_{\text{gray}} = 0.2989 R + 0.5870 G + 0.1140 B

Розмиття Гауса:

Iblur=G(I,k)I\_{\text{blur}} = G(I, k)

Де GG – функція Гаусового розмиття, kk – розмір ядра.

Порогова обробка Оцу для бінаризації:

T=arg⁡max⁡τ(σмежа2)T = \arg\max\_{\tau} \left( \sigma\_{\text{межа}}^2 \right)

Ці алгоритми дозволяють видалити шум, підвищити контрастність та покращити якість тексту, що особливо важливо при роботі з OCR.

Формування датасетів для навчання

Функція collate\_fn() виконує групування оброблених зображень у пакети для навчання нейронної мережі.

Модель формує батч (batch) – набір зображень та їхніх bounding boxes, що дозволяє забезпечити ефективне навчання та обробку даних.

# Розділ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для автоматичного розпізнавання номерних знаків автомобілів є складним технологічним процесом, що охоплює кілька ключових етапів, необхідних для ефективної роботи системи у реальному середовищі. Насамперед, важливим аспектом є структурування даних, яке включає правильну організацію файлів та директорій для навчання, тестування та валідації моделі. Використання структурованого підходу дозволяє ефективно керувати великою кількістю зображень та текстових анотацій, що необхідно для навчання системи машинного навчання. Одним із критичних етапів є реалізація алгоритмів детекції номерного знака, які відповідають за правильне визначення області номерного знака на зображенні. У цьому процесі використовується нейронна мережа YOLO (You Only Look Once), яка забезпечує швидке та точне виявлення номерних знаків навіть у складних умовах освітлення або при частковому перекритті об'єкта. YOLO є потужним алгоритмом, який виконує детекцію в режимі реального часу, дозволяючи працювати із відеопотоком та фото великої роздільної здатності без значних затримок. Після визначення області номерного знака необхідно виконати обробку зображень, що включає фільтрацію шуму, нормалізацію яскравості, перетворення у градації сірого, адаптивне порогове значення та вирівнювання контрастності, що забезпечує покращену якість зображення перед подальшим розпізнаванням тексту [7, c. 34–40]. У цьому процесі важливу роль відіграє OpenCV, яка дозволяє виконувати всі необхідні операції попередньої обробки, включаючи вирізання області номерного знака та застосування алгоритмів покращення якості зображення. Для остаточного розпізнавання тексту використовується OCR (Optical Character Recognition), який аналізує підготовлені фрагменти зображення та ідентифікує літери та цифри номерного знака. У цьому процесі використовується Tesseract OCR, що виконує аналіз символів після порогової обробки та видалення артефактів. Останнім етапом у розробці програмного забезпечення є оцінка точності моделі, що передбачає використання метрики IoU (Intersection over Union) для перевірки коректності визначених координат номерного знака порівняно з реальними значеннями у навчальному наборі даних. Ця метрика дозволяє визначити, наскільки точно модель виявила область номерного знака та чи необхідне додаткове навчання для покращення її роботи. Завдяки комплексному підходу до реалізації програмного забезпечення, включаючи використання машинного навчання, нейронних мереж, алгоритмів комп’ютерного бачення та OCR, система здатна працювати у режимі реального часу, адаптуючись до різних умов та забезпечуючи високу точність розпізнавання номерних знаків, що робить її придатною для інтеграції у дорожній контроль, безпекові системи та автоматизовані паркінгові рішення [8, c. 623].

Перед обробкою зображень у системі розпізнавання номерних знаків необхідно забезпечити правильну організацію директорій, що дозволяє структуровано зберігати дані та оптимізувати роботу програмного забезпечення. Для цього код реалізує розподіл даних на три основні вибірки: навчальну (TRAIN\_IMAGES, TRAIN\_LABELS), валідаційну (VALID\_IMAGES, VALID\_LABELS) та тестову (TEST\_IMAGES, TEST\_LABELS). Таке логічне групування гарантує правильну підготовку моделі глибокого навчання, оскільки дозволяє окремо аналізувати її продуктивність на навчальних, перевірочних та тестових наборах даних, що критично важливо для підвищення точності розпізнавання. Крім того, використання функції os.makedirs() дає змогу автоматично створювати необхідні папки, серед яких results для збереження вихідних даних, models для тренованих моделей, crops для вирізаних фрагментів номерних знаків, що використовуються для OCR, та test/ocr для текстових результатів розпізнавання. Така структура не лише упорядковує процес збереження проміжних і фінальних результатів, а й значно полегшує роботу зі збереження та використання отриманої інформації, дозволяючи ефективно масштабувати систему та додавати нові функціональні можливості без порушення організації даних. Це забезпечує гнучкість у роботі з різними наборами зображень, дозволяє автоматизувати підготовку даних та гарантує коректне збереження проміжних результатів у процесі навчання та тестування, що робить програмну реалізацію ефективною, продуктивною та готовою до інтеграції в реальні умови експлуатації системи автоматизованого розпізнавання номерних знаків.

Алгоритм YOLO (You Only Look Once) використовується для швидкого і точного визначення координат номерного знака на зображенні, що є ключовим етапом у процесі автоматизованого розпізнавання. На відміну від традиційних методів детекції, які використовують багатоступеневий аналіз, YOLO застосовує єдину глибоку нейронну мережу, що миттєво прогнозує межі об'єкта, виконуючи детекцію в режимі реального часу. Оскільки YOLO оперує у форматі (центр X, центр Y, ширина, висота), для коректної роботи системи та інтеграції з іншими алгоритмами обробки використовується функціональність yolo\_to\_xyxy() та xyxy\_to\_yolo(), що конвертує координати в класичний формат (x\_min, y\_min, x\_max, y\_max), який широко застосовується в інших моделях комп’ютерного бачення. Це конвертування дозволяє правильно локалізувати номерний знак у межах вихідного зображення та передавати його на наступні етапи обробки, зокрема OCR-аналіз тексту для розпізнавання букв і цифр. Крім того, важливим аспектом перевірки точності роботи YOLO є оцінка Intersection over Union (IoU), що дозволяє обчислити співвідношення між реальним місцезнаходженням номерного знака та передбаченим моделлю bounding box. Формула IoU:

IoU=площа перетинузагальна площа (об’єднання)IoU = \frac{\text{площа перетину}}{\text{загальна площа (об'єднання)}}

Ця метрика використовується для визначення якісності виявлення: якщо IoU близьке до 1, модель розпізнала номерний знак з високою точністю, а якщо IoU низьке – необхідна додаткова оптимізація нейронної мережі або покращення передобробки зображення. Завдяки такому підходу можна адаптувати систему до різних умов, включаючи зміни освітлення, забруднення номерного знака чи його часткове перекриття, що забезпечує стабільність роботи навіть у складних сценаріях [5].

Важливим аспектом роботи системи розпізнавання номерних знаків є перевірка коректності передбачених координат bounding box, що визначає точність детекції об’єкта. Для цього використовується функція compute\_iou(), яка обчислює метрику Intersection over Union (IoU) – один із ключових показників у машинному навчанні, що дозволяє оцінити, наскільки передбачена модель bounding box відповідає фактичному розташуванню номерного знака на зображенні. IoU визначається як відношення площі перетину передбаченого та реального bounding box до загальної площі об'єднання обох областей, і його формула має вигляд:

IoU=площа перетинузагальна площа (об’єднання)IoU = \frac{\text{площа перетину}}{\text{загальна площа (об'єднання)}}

Де площа перетину розраховується за допомогою максимальних і мінімальних координат bounding box, що моделює просторове розміщення номерного знака. Якщо значення IoU високе, це означає, що модель правильно визначила межі номерного знака з високою точністю, що є критично важливим для стабільної роботи алгоритму розпізнавання. Висока IoU (більше 0.75) свідчить про точну детекцію, тоді як низьке значення (менше 0.5) може вказувати на необхідність додаткового навчання моделі, використання кращих передобробок або вдосконалення алгоритму детекції. Реалізація такого підходу дозволяє оцінювати якість розпізнавання в реальних умовах, зокрема при змінному освітленні, часткових перекриттях або забрудненні номерного знака, та є одним із ключових факторів для оптимізації роботи нейронної мережі YOLO та подальшого OCR-аналізу тексту.

Обробка зображень для OCR

Після успішного виявлення номерного знака на зображенні необхідно виконати його попередню обробку для підготовки до OCR (Optical Character Recognition), що є ключовим процесом у розпізнаванні текстових символів. Оскільки якість введеного зображення безпосередньо впливає на точність розпізнавання, алгоритми обробки повинні виконувати кілька критично важливих завдань. Код включає дві основні функції: crop\_license\_plate(), яка відповідає за точне вирізання області номерного знака відповідно до координат bounding box, і preprocess\_for\_ocr(), яка застосовує комплексні методи покращення зображення для високоякісного розпізнавання символів. Функція crop\_license\_plate() отримує координати у форматі (x1, y1, x2, y2), де x1, y1 позначають верхній лівий кут об’єкта, а x2, y2 – нижній правий кут, що дозволяє коректно сегментувати номерний знак на основі його локалізації на зображенні. Це критично важливо, оскільки нейронні мережі, що здійснюють розпізнавання тексту, потребують чітко виділеної області, а зайвий фон може негативно вплинути на якість OCR-аналізу. Після вирізання використовується preprocess\_for\_ocr(), яка здійснює покращення контрастності, видалення шуму та бінаризацію зображення. Нормалізація контрастності дозволяє підсилити розпізнаваність тексту, роблячи літери та цифри більш чіткими для аналізу. Фільтрація шуму, зокрема використання Гаусового розмиття (cv2.GaussianBlur()), допомагає усунути дрібні артефакти, що могли з’явитися через неякісне освітлення або недостатню чіткість знімка. Бінаризація методом Оцу (cv2.threshold()) виконує автоматичне встановлення порогового значення, що дозволяє адаптивно конвертувати зображення у двовимірний формат, відокремлюючи текстові символи від фону та забезпечуючи їхню чітку розпізнаваність. Це особливо важливо для роботи в реальних умовах, коли номерні знаки можуть бути частково затемненими або мати нерівномірну освітленість. Використання цих методів дозволяє отримати максимально чисте та структуроване зображення номерного знака, що передається на подальший OCR-аналіз для розпізнавання тексту. Такий комплексний підхід до обробки гарантує мінімізацію помилок при текстовому розпізнаванні і забезпечує високу точність роботи системи навіть у складних сценаріях експлуатації, зокрема при аналізі зображень із відеопотоку, обробці знаків зі старих або пошкоджених автомобілів, а також при змінному освітленні чи наявності перешкод у кадрі.

Оптимізація роботи моделі

Останній етап у розробці системи розпізнавання номерних знаків передбачає формування батчів (batch) для навчання нейронної мережі, що дозволяє обробляти великі набори даних ефективно та без втрати продуктивності. Батчинг використовується для оптимізації навчального процесу, оскільки подача даних у пакетах зменшує навантаження на обчислювальні ресурси та дозволяє моделі швидше адаптуватися до вхідних зображень. У коді реалізовано функцію collate\_fn(), яка групує дані у пакети, перетворюючи їх у зручний формат для подальшого обчислення нейронною мережею, що критично важливо для збільшення швидкості навчання та забезпечення узгодженості вхідної інформації. Ця функція гарантує правильну підготовку кожного зображення та його відповідної анотації, що забезпечує структуроване введення даних у модель. Крім того, передбачено структурування вихідних даних, що дозволяє інтегрувати розроблену систему у масштабні автоматизовані рішення, зокрема дорожній контроль, де вона може автоматично виявляти порушення та відстежувати автомобілі; паркінги, де технологія допомагає розпізнавати транспортні засоби без необхідності використання додаткових пристроїв доступу; та безпекові системи, де розпізнавання номерних знаків може бути інтегровано у бази даних правоохоронних органів для оперативного аналізу транспортних потоків. Завдяки такому підходу система стає гнучкою, ефективною та масштабованою, що дозволяє використовувати її в умовах реального часу, забезпечуючи високу точність та продуктивність у різних сферах застосування.

## 3.1 Розробка алгоритму виявлення номерних знаків

Розробка алгоритму виявлення номерних знаків є одним із найважливіших етапів у створенні системи автоматичного розпізнавання транспортних засобів, оскільки саме детекція номерного знака визначає коректність подальшого OCR-аналізу. Якщо система правильно ідентифікує область номерного знака на зображенні, це забезпечує високу точність розпізнавання тексту та мінімізує помилки, які можуть виникнути через неточності детекції, змінене освітлення або часткове перекриття знака іншими об’єктами. У представлених фрагментах коду реалізовано структурування та підготовку даних, що є критичним для успішної роботи моделі машинного навчання: зображення розподіляються на навчальні (TRAIN\_IMAGES, TRAIN\_LABELS), валідаційні (VALID\_IMAGES, VALID\_LABELS) та тестові (TEST\_IMAGES, TEST\_LABELS) набори, що дозволяє точно керувати процесами навчання та оцінки. Додатково передбачено автоматичне створення директорій (results, models, crops), що гарантує правильну організацію проміжних і фінальних результатів роботи. Основним методом детекції є нейронна мережа YOLO (You Only Look Once), яка здійснює розпізнавання номерного знака в режимі реального часу, використовуючи координати у форматі (центр X, центр Y, ширина, висота). Для коректного використання отриманих bounding box виконується перетворення цих координат у класичний формат (x\_min, y\_min, x\_max, y\_max) за допомогою функції yolo\_to\_xyxy(), що дозволяє локалізувати знайдену область та підготувати її для подальшої обробки. Важливою складовою алгоритму є оцінка точності детекції, яка реалізується через метрику Intersection over Union (IoU) – показник, що визначає, наскільки правильно модель виявила межі номерного знака порівняно з реальним розташуванням об’єкта. Функція compute\_iou() виконує обчислення IoU за формулою:

IoU=площа перетинузагальна площа (об’єднання)IoU = \frac{\text{площа перетину}}{\text{загальна площа (об'єднання)}}

Високе значення IoU (більше 0.75) свідчить про точну детекцію, тоді як низьке значення (<0.5) вказує на необхідність покращення навчання моделі або її параметрів. Після детекції виконується вирізання області номерного знака, що реалізовано у функції crop\_license\_plate(), яка отримує координати bounding box та створює нове зображення, яке містить лише номерний знак без зайвого фону. Це критично важливо для OCR-аналізу, оскільки зайві елементи можуть погіршити якість розпізнавання тексту. Подальший етап – передобробка вирізаного зображення для OCR, що включає конвертацію у градації сірого, розмиття Гауса для згладжування шуму та бінаризацію методом Оцу, що дозволяє адаптивно виділяти текстові символи на номерному знаку [15]. Завдяки цьому система отримує чисте, контрастне зображення тексту, яке передається на OCR-обробку. Завершальний крок у розробці алгоритму – формування пакетів (batch) для навчання, що реалізовано через collate\_fn(), яка групує дані та забезпечує ефективне навчання моделі, що значно оптимізує обчислювальні ресурси та підвищує продуктивність розпізнавання. Загальна структура алгоритму включає детекцію номерного знака, оцінку точності виявлення, вирізання текстової області, її передобробку та подальше текстове розпізнавання, що гарантує високу точність та ефективність роботи системи, роблячи її придатною для реального застосування у сферах безпеки, дорожнього контролю та автоматизованих транспортних систем.

Перед виконанням процесу детекції номерних знаків необхідно чітко структурувати дані, оскільки правильна організація файлової системи є критично важливою для ефективної роботи моделі глибокого навчання. Це включає розподіл зображень та відповідних анотацій на окремі категорії: навчальну вибірку (TRAIN\_IMAGES, TRAIN\_LABELS), валідаційну (VALID\_IMAGES, VALID\_LABELS), та тестову (TEST\_IMAGES, TEST\_LABELS), що забезпечує правильне формування датасету для навчання моделі нейронної мережі. Такий підхід дозволяє розробнику контролювати якість розпізнавання та покращувати точність алгоритму шляхом коригування параметрів навчання та проведення тестування на незалежних зображеннях. Крім того, у коді передбачено автоматичне створення необхідних директорій за допомогою os.makedirs(), що включає папку results для збереження отриманих результатів розпізнавання, models для збереження тренованих моделей, crops для вирізаних фрагментів номерних знаків, та test/ocr для текстового розпізнавання номерів. Це дозволяє автоматизувати підготовку даних, зменшуючи час, необхідний для обробки зображень, та забезпечує зручність у використанні та масштабуванні системи для аналізу великої кількості зображень [10, c. 12–17]. Завдяки чіткій файловій структурі модель може безперебійно завантажувати зображення, виконувати їхню обробку та генерувати вихідні результати, що робить систему готовою до інтеграції у реальні інфраструктурні рішення, зокрема у дорожній моніторинг, систему паркування чи інші безпекові рішення. Такий підхід гарантує гнучкість у розширенні функціоналу, дозволяючи легко додавати нові модулі або адаптувати алгоритм для розпізнавання номерних знаків різних країн та регіонів.

Виявлення номерного знака за допомогою YOLO

Для детекції номерного знака використовується алгоритм YOLO (You Only Look Once), який дозволяє швидко знаходити об’єкти на зображенні в режимі реального часу. Особливістю YOLO є те, що він оперує форматом координат (центр X, центр Y, ширина, висота), тому для коректної роботи потрібно перетворити ці координати у більш звичний формат (x\_min, y\_min, x\_max, y\_max). Це виконується у функції yolo\_to\_xyxy(), яка розраховує межі bounding box за такими математичними виразами:

xmin⁡=(x−w2)×widthx\_{\min} = (x - \frac{w}{2}) \times \text{width}

ymin⁡=(y−h2)×heighty\_{\min} = (y - \frac{h}{2}) \times \text{height}

xmax⁡=(x+w2)×widthx\_{\max} = (x + \frac{w}{2}) \times \text{width}

ymax⁡=(y+h2)×heighty\_{\max} = (y + \frac{h}{2}) \times \text{height}

Це гарантує правильне локалізування номерного знака для подальшої обробки.

Оцінка точності виявлення за метрикою IoU

Intersection over Union (IoU) є фундаментальною метрикою для оцінки точності детекції номерного знака, оскільки вона дозволяє кількісно порівняти передбачену та фактичну область об’єкта на зображенні. IoU визначається як відношення площі перетину двох bounding box (передбаченого та реального) до площі їхнього об'єднання, що є ключовим показником для оцінки ефективності роботи алгоритму глибокого навчання. У коді реалізовано функцію compute\_iou(), яка виконує такі математичні обчислення: спочатку обчислюється площа кожного bounding box (area1 та area2), потім знаходяться координати області перетину (inter\_x\_min, inter\_y\_min, inter\_x\_max, inter\_y\_max). Якщо дві області не перетинаються (inter\_x\_max <= inter\_x\_min або inter\_y\_max <= inter\_y\_min), IoU дорівнює 0, що означає, що модель не виявила номерний знак правильно. Якщо ж є перетин, то його площа розраховується за формулою:

inter\_area=(inter\_xmax⁡−inter\_xmin⁡)×(inter\_ymax⁡−inter\_ymin⁡)\text{inter\\_area} = (\text{inter\\_x}\_{\max} - \text{inter\\_x}\_{\min}) \times (\text{inter\\_y}\_{\max} - \text{inter\\_y}\_{\min})

Далі IoU визначається як:

IoU=inter\_areaarea1+area2−inter\_areaIoU = \frac{\text{inter\\_area}}{\text{area}\_1 + \text{area}\_2 - \text{inter\\_area}}

Якщо значення IoU більше 0.75, це означає, що передбачений bounding box майже повністю збігається із реальним розташуванням номерного знака, що вказує на високу точність моделі YOLO. Якщо ж IoU нижче 0.5, необхідно покращити якість навчання моделі, розширити обсяг навчального набору зображень або оптимізувати алгоритми передобробки, щоб зменшити кількість випадків неправильного визначення меж номерного знака. Використання IoU є стандартною практикою у машинному навчанні, особливо у завданнях обробки зображень, оскільки вона дає об’єктивну оцінку роботи алгоритму детекції, дозволяючи порівнювати різні моделі та вибирати найбільш ефективну для розпізнавання номерних знаків у реальних умовах, таких як змінне освітлення, нечіткі або пошкоджені знаки та різні стилі написання символів.

Вирізання номерного знака із зображення

Після того, як номерний знак знайдено, необхідно його правильно вирізати для подальшої OCR-обробки. Це виконується функцією crop\_license\_plate(), яка отримує координати bounding box (x1, y1, x2, y2) та виокремлює відповідну область з оригінального зображення.

I′=I[y1:y2,x1:x2]I' = I[y\_1:y\_2, x\_1:x\_2]

Така операція забезпечує точну підготовку текстового блоку для роботи OCR.

Попередня обробка номерного знака перед OCR

Функція preprocess\_for\_ocr() виконує попередню обробку вирізаного фрагмента, покращуючи його якість перед подальшим текстовим аналізом. Основні етапи обробки включають:

Перетворення у градації сірого (cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)), що спрощує подальші операції фільтрації.

Гаусове розмиття (cv2.GaussianBlur()), що допомагає прибрати шум та згладити зображення.

Бінаризацію методом Оцу (cv2.threshold()), яка адаптивно встановлює поріг розпізнавання тексту та покращує контрастність.

Фільтрацію артефактів, що дозволяє зробити текстові символи більш виразними для OCR.

Ці операції гарантують, що зображення номерного знака матиме високу чіткість, що є ключовим фактором для точного розпізнавання тексту.

Формування структурованих батчів для навчання

Функція collate\_fn() групує оброблені дані у батчі, що дозволяє ефективно навчати модель. Такий підхід підвищує продуктивність алгоритму, оптимізує використання пам’яті та скорочує час навчання.

## 

## 3.2 Попередня обробка зображень

Попередня обробка зображень є ключовим етапом у процесі автоматичного розпізнавання номерних знаків, оскільки вона безпосередньо впливає на точність розпізнавання тексту, мінімізує помилки та покращує якість даних для подальшої OCR-ідентифікації символів. Вихідне зображення номерного знака може містити різноманітні дефекти: шум, недостатнє освітлення, відблиски, тіні, розмиття або артефакти, що утворилися через нерівну поверхню або забруднення. Щоб вирішити ці проблеми, алгоритм обробки реалізує поетапне покращення зображення, що включає конвертацію у градації сірого, розмиття Гауса, бінаризацію методом Оцу, фільтрацію артефактів та нормалізацію яскравості. Спочатку виконується конвертація у градації сірого (cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)), що дозволяє перетворити кольорове зображення на чорно-біле за формулою:

Igray=0.2989R+0.5870G+0.1140BI\_{\text{gray}} = 0.2989 R + 0.5870 G + 0.1140 B

де R, G, B – відповідні колірні канали. Це значно спрощує подальші операції фільтрації, оскільки текстові символи стають більш помітними на однорідному фоні. Наступний крок – розмиття Гауса, що застосовується для згладжування шуму та дрібних артефактів. Фільтрація виконується за допомогою cv2.GaussianBlur(img, (5,5), 0), а математична модель Гаусового фільтру має вигляд:

G(I,k)=∑i,jI(i,j)⋅e−x2+y22σ2G(I, k) = \sum\_{i,j} I(i,j) \cdot e^{-\frac{x^2 + y^2}{2\sigma^2}}

де kk – розмір ядра фільтра, а σ\sigma – параметр згладжування. Гаусове розмиття зменшує ефекти шуму, що можуть заважати OCR-алгоритмам. Наступний етап – бінаризація методом Оцу (cv2.threshold(img, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)), яка автоматично визначає оптимальне порогове значення для відокремлення текстових символів від фону. Формула порогової обробки Оцу:

T=arg⁡max⁡τ(σмежа2)T = \arg\max\_{\tau} \left( \sigma\_{\text{межа}}^2 \right)

де τ\tau – ідеальне порогове значення. Це дозволяє адаптивно поділити зображення на чорне та біле, покращуючи контрастність номерного знака для OCR. Останнім етапом перед розпізнаванням є нормалізація яскравості, яка дозволяє усунути перепади контрасту, що виникають через тіні, нерівномірне освітлення або відблиски. Всі ці процеси суттєво покращують якість введеного зображення, забезпечуючи максимальну точність розпізнавання тексту, навіть якщо вихідне фото містить спотворення або низьку якість.

Перед виконанням безпосередньої обробки номерного знака важливо ізолювати його від навколишнього фону, оскільки зайві елементи можуть створювати перешкоди для OCR-алгоритмів, що призводить до зниження точності розпізнавання символів. Це реалізується у функції crop\_license\_plate(), яка отримує координати bounding box у форматі (x1, y1, x2, y2), де x1, y1 позначають верхній лівий кут номерного знака, а x2, y2 – нижній правий кут, що дозволяє точно вирізати необхідний фрагмент зі зображення. Вирізання здійснюється за допомогою маніпуляції з піксельними матрицями, зокрема через вираз:

I′=I[y1:y2,x1:x2]I' = I[y\_1:y\_2, x\_1:x\_2]

де I′I' – нове зображення, яке містить лише номерний знак без фону, що усуває ризик помилкового розпізнавання тексту через зайві контури та нерівномірності освітлення. Ця процедура дозволяє підготувати зображення до подальшої обробки, зменшуючи вплив непотрібної інформації, такої як текстура дорожнього покриття, логотипи автомобіля або інші об’єкти, що знаходяться поряд. Ізольований номерний знак можна ефективно передати в подальші етапи аналізу, такі як нормалізація контрастності, бінаризація, розмиття Гауса та текстове розпізнавання, що суттєво покращує якість OCR-аналізу. Якщо номерний знак частково затемнений або має відблиски, видалення зайвого фону дозволяє покращити чіткість країв символів, що є критичним для точного зчитування букв і цифр. Виконання цієї операції підвищує стабільність розпізнавання, зменшуючи кількість помилок при обробці номерних знаків у реальних умовах, таких як змінне освітлення або наявність перешкод на зображенні [10, c.14].

Після того як номерний знак було вирізано із загального зображення, наступним критично важливим етапом є перетворення його у градації сірого, що значно спрощує подальшу обробку для OCR (Optical Character Recognition) та покращує видимість текстових символів. Кольорове зображення містить три основні канали (червоний R, зелений G, синій B), кожен з яких впливає на сприйняття загального вигляду об'єкта. Однак для ефективного розпізнавання тексту необхідно конвертувати ці три канали в один монохромний, що дозволяє алгоритму зосередитися лише на контрасті між символами і фоном без впливу зайвих кольорових складових. У коді цей процес реалізується у функції preprocess\_for\_ocr(), яка виконує перетворення кольорового зображення в градації сірого за допомогою cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY), що здійснює автоматичне зведення інформації про колір у монохромну шкалу відповідно до наступної математичної формули:

Igray=0.2989R+0.5870G+0.1140BI\_{\text{gray}} = 0.2989 R + 0.5870 G + 0.1140 B

де коефіцієнти 0.2989, 0.5870, 0.1140 відображають відносний вплив кожного колірного каналу на фінальне градаційне значення пікселя. Червоний (R) має найменший внесок у загальну яскравість, зелений (G) має найбільший вплив, а синій (B) відіграє допоміжну роль у передачі світлових відтінків. Це пояснюється особливостями сприйняття кольору людським оком, де зелений відтінок є найчутливішим. Виконання цього перетворення значно зменшує складність подальших операцій, таких як бінаризація, розмиття та контурне виділення, адже зображення без зайвих кольорових елементів містить тільки інформацію про контрастність, що підвищує ефективність розпізнавання текстових символів. Візуально та математично чорно-білий варіант номерного знака має перевагу, оскільки OCR-алгоритми краще працюють із чітко відокремленими літерами та цифрами на рівномірному фоні, що особливо актуально у складних умовах, наприклад при низькій освітленості або нерівномірному освітленні. Застосування градації сірого також мінімізує кількість інформації, яку необхідно аналізувати, що позитивно впливає на швидкість обробки та дозволяє системі ефективно працювати у режимі реального часу [10, c. 13].

Наступний етап попередньої обробки зображення – зменшення артефактів та видалення дрібних деталей, що можуть впливати на точність розпізнавання номерних знаків у системі OCR. Це досягається за допомогою Гаусового розмиття, яке є ефективним методом фільтрації та згладжування нерівностей на зображенні. Гаусове розмиття працює шляхом застосування спеціального згладжувального ядра, що зменшує контрасти між сусідніми пікселями, усуваючи шум та покращуючи загальний вигляд текстових символів. У коді цей алгоритм реалізований через cv2.GaussianBlur(img, (5, 5), 0), де параметр (5, 5) визначає розмір ядра фільтра, а 0 вказує на автоматичний розрахунок коефіцієнтів згладжування. Функція Гаусового фільтра описується математичною формулою:

G(I,k)=∑i,jI(i,j)⋅e−x2+y22σ2G(I, k) = \sum\_{i,j} I(i,j) \cdot e^{-\frac{x^2 + y^2}{2\sigma^2}}

Де kk – розмір ядра фільтра, σ\sigma – параметр згладжування, що впливає на ступінь розмиття. Чим більше значення σ\sigma, тим сильніше відбувається згладжування країв та деталей. Основна перевага Гаусового розмиття полягає у тому, що воно зменшує шум високих частот, тобто незначні варіації кольору та яскравості, що можуть спотворювати текстові символи номерного знака. Це особливо корисно у випадках, коли на вихідному зображенні є зернистість, артефакти або нерівномірне освітлення, які можуть ускладнити процес OCR-розпізнавання. За рахунок правильно підібраних параметрів ядра згладжування можна адаптивно покращити контрастність тексту, забезпечивши його чіткість та розпізнаваність. У випадку сильного спотворення або низької якості зображення можна варіювати значення sigma та розмір ядра (k, k), щоб досягти оптимального балансу між згладжуванням та збереженням важливих деталей номерного знака. Завдяки цьому підходу алгоритм гарантує високу якість підготовки зображення, дозволяючи OCR-системі точно визначати текстові символи навіть у випадку складних умов освітлення чи зношених номерних знаків.

Після фільтрації та нормалізації зображення необхідно виділити текстові символи номерного знака для їхнього подальшого OCR-розпізнавання, і це досягається шляхом порогової обробки методом Оцу, який є одним із найбільш ефективних способів автоматичної бінаризації зображень. Оскільки текстові символи на номерному знаку можуть мати різну яскравість через освітлення, тіні або забруднення, необхідно адаптивно встановити поріг, який розділить область тексту та фон, мінімізуючи помилки розпізнавання. У коді цей процес реалізується через cv2.threshold(img, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU), де функція cv2.threshold() застосовує автоматичний аналіз гістону пікселів, визначаючи оптимальне порогове значення без необхідності його попереднього встановлення. Метод Оцу використовує математичний підхід, який дозволяє максимізувати міжкласову дисперсію, тобто знаходити такий поріг, при якому відмінності між областями фону та тексту стають максимальними. Формула визначення порогового значення має вигляд:

T=arg⁡max⁡τ(σмежа2)T = \arg\max\_{\tau} \left( \sigma\_{\text{межа}}^2 \right)

Де τ\tau – порогове значення, яке оптимально поділяє фонові області та текстові символи. Бінаризація, яка виконується після встановлення порогу, дозволяє перетворити номерний знак у чорно-біле зображення, де пікселі тексту стають абсолютно чорними (значення 0), а фонові області – абсолютно білими (значення 255), що значно покращує читабельність для OCR-алгоритму. Важливим аспектом є те, що метод Оцу працює адаптивно, а це означає, що він ефективно обробляє номерні знаки з різною освітленістю, відблисками або частковими забрудненнями, автоматично підлаштовуючи порогове значення під конкретне зображення. Завдяки цьому підходу система розпізнавання отримує максимально чіткі контури текстових символів, що підвищує точність OCR-аналізу навіть у складних умовах реального застосування, таких як обробка відеопотоку, нічне освітлення або зношені номерні знаки автомобілів [24].

Фінальний етап попередньої обробки зображення перед OCR-аналізом – збалансування контрастності та нормалізація яскравості, що забезпечує максимальну чіткість текстових символів на номерному знаку. Контрастність є одним із ключових факторів, що впливають на точність розпізнавання, оскільки недостатньо виражені символи або їх злиття з фоновими елементами можуть суттєво ускладнити процес ідентифікації тексту [21]. В реальних умовах зйомки номерні знаки можуть бути затемненими, освітленими під гострим кутом або містити артефакти, пов’язані з нерівномірним розподілом світла, тінями або засвіченням. Щоб вирішити ці проблеми, застосовується адаптивна нормалізація яскравості, яка дозволяє автоматично вирівнювати градації освітлення на зображенні, мінімізуючи ефекти надмірної темряви або засвічення. У коді цей процес реалізується шляхом локальної контрастної корекції, яка аналізує інтенсивність пікселів у межах визначених областей та адаптивно змінює яскравість, забезпечуючи рівномірний розподіл освітлення по всьому зображенню. Також використовуються методи глобальної корекції контрастності, що дозволяють підсилити розпізнаваність тексту за рахунок масштабного вирівнювання світлових параметрів. Це виконується за допомогою алгоритму CLAHE (Contrast Limited Adaptive Histogram Equalization), який обмежує різкі зміни контрастності, запобігаючи надмірному підсиленню артефактів. Загальна формула для вирівнювання контрасту через адаптивну нормалізацію яскравості має вигляд:

Inorm=I−IminImax−Imin×255I\_{\text{norm}} = \frac{I - I\_{\text{min}}}{I\_{\text{max}} - I\_{\text{min}}} \times 255

де InormI\_{\text{norm}} – нормалізоване значення пікселя, IminI\_{\text{min}} та ImaxI\_{\text{max}} – відповідні мінімальні та максимальні рівні яскравості в межах області аналізу. Такий підхід дозволяє усунути локальні перепади освітлення та забезпечити оптимальну контрастність для OCR-аналізу. Завдяки цьому система OCR отримує максимально чистий та структурований текст, що значно підвищує точність розпізнавання навіть у випадках слабкого освітлення, неякісних фотографій або номерних знаків зі складним фоном.

Останнім важливим етапом у підготовці зображень для OCR є **групування оброблених номерних знаків у батчі для навчання**, що виконується через функцію collate\_fn(). Батчинг дозволяє оптимізувати процес навчання нейронної мережі, підвищуючи ефективність обробки великих масивів даних і зменшуючи навантаження на обчислювальні ресурси. У традиційних підходах до навчання моделі кожне зображення аналізується окремо, що може **уповільнити** навчальний процес та вплинути на продуктивність, особливо при роботі з великими вибірками даних. Функція collate\_fn() вирішує цю проблему, дозволяючи **згрупувати дані** у пакети певного розміру, які потім подаються у нейронну мережу для одночасної обробки. Основний принцип роботи collate\_fn() полягає у **структурованому перетворенні набору зображень і їхніх анотацій у зручний формат** для аналізу, що включає нормалізовані масиви числових значень, координати bounding box та інші характеристики номерних знаків. Це важливо, оскільки передача **збалансованих батчів** дозволяє моделі **швидше адаптуватися** до входу, знижуючи ймовірність нестабільних результатів під час навчання. Окрім ефективності навчання, **групування даних у батчі також впливає на продуктивність OCR в реальному часі**, оскільки розпізнавання текстових символів може бути виконано у вигляді **пакетної обробки**, що дозволяє паралельно ідентифікувати кілька номерних знаків у потоці відео або масиві фотографій. Функція collate\_fn() також може виконувати додаткові операції **передобробки в батчах**, такі як масштабування зображень, аугментація даних та вирівнювання їхніх розмірів, щоб зробити модель **більш гнучкою** до різних форматів номерних знаків, зокрема до варіацій стилю, нахилу або забруднень. Використання батчів у навчанні дозволяє забезпечити **узгодженість вхідних даних**, покращуючи стабільність результатів та адаптивність моделі до змін у реальних умовах експлуатації, таких як змінне освітлення або динамічні кадри у відеопотоці. Завдяки застосуванню collate\_fn(), система автоматичного розпізнавання номерних знаків отримує **високу швидкість обробки, точність навчання та можливість роботи в реальному часі**, що робить її ідеальним рішенням для автоматизованих дорожніх систем, контролю паркування, моніторингу транспорту та безпекових інструментів.

## 3.3 Використання нейронних мереж для розпізнавання

Розпізнавання номерних знаків автомобілів із використанням нейронних мереж є складним багатоступеневим процесом, що вимагає комплексного підходу до аналізу зображень, детекції об'єктів, попередньої обробки та ідентифікації тексту. Використання глибоких нейронних мереж, зокрема конволюційних нейронних мереж (CNN), а також моделі YOLO, дозволяє створювати точні алгоритми, які можуть працювати в реальному часі, обробляючи великі потоки даних без значних затримок. В основі цього процесу лежить кілька важливих етапів: локалізація номерного знака, підготовка зображення для OCR (Optical Character Recognition) та безпосереднє розпізнавання символів на знаку. Спочатку система отримує вхідне зображення автомобіля, на якому необхідно знайти область номерного знака. Це виконується за допомогою моделі YOLO, що працює за принципом "єдиного погляду" на зображення, тобто не потребує складних каскадних алгоритмів для визначення об'єктів. YOLO використовує підхід прямого передбачення координат об'єкта, що значно пришвидшує детекцію та робить її придатною для роботи в реальному часі, наприклад, у системах контролю трафіку чи автоматизованого паркування. Оскільки YOLO працює у форматі (центр X, центр Y, ширина, висота), для коректної інтеграції результатів необхідно конвертувати координати bounding box у класичний формат (x\_min, y\_min, x\_max, y\_max). Це виконується у коді через функції yolo\_to\_xyxy() та xyxy\_to\_yolo(), що дають змогу коригувати межі номерного знака перед його подальшою обробкою.

Після успішного визначення області номерного знака необхідно його ізолювати від навколишнього фону, щоб забезпечити OCR-алгоритму доступ лише до необхідного фрагмента без сторонніх елементів. Для цього у коді реалізовано функцію crop\_license\_plate(), яка отримує координати (x1, y1, x2, y2) та використовує їх для вирізання відповідної області за формулою:

I′=I[y1:y2,x1:x2]I' = I[y\_1:y\_2, x\_1:x\_2]

де I′I' – нове зображення, що містить лише номерний знак без зайвого фону. Така операція є вкрай важливою, адже неправильне визначення bounding box може призвести до втрати частини текстових символів або до включення зайвих деталей, що погіршить якість розпізнавання.

Наступним етапом є попередня обробка зображення, яка покращує якість номерного знака перед OCR-аналізом. Однією з перших операцій є перетворення у градації сірого, що здійснюється за допомогою cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY). Це конвертує кольорове зображення у монохромну шкалу, що значно спрощує аналіз контурів та виділення текстових символів. Формула такого перетворення:

Igray=0.2989R+0.5870G+0.1140BI\_{\text{gray}} = 0.2989 R + 0.5870 G + 0.1140 B

де R, G, B – відповідні канали кольорового зображення. Оскільки OCR-алгоритмам не потрібна інформація про колір, а лише контури тексту, цей процес дозволяє усунути непотрібні колірні артефакти, що можуть впливати на точність аналізу.

Для згладжування нерівностей та усунення дрібних артефактів використовується Гаусове розмиття, яке застосовується у коді через cv2.GaussianBlur(img, (5, 5), 0). Гаусове розмиття дозволяє усунути випадковий шум та текстурні спотворення, створюючи більш рівномірне зображення для OCR. Формула його роботи виглядає так:

G(I,k)=∑i,jI(i,j)⋅e−x2+y22σ2G(I, k) = \sum\_{i,j} I(i,j) \cdot e^{-\frac{x^2 + y^2}{2\sigma^2}}

де kk – розмір ядра фільтра, а σ\sigma – параметр згладжування. Завдяки використанню Гаусового фільтра система очищує текстові контури, забезпечуючи високу точність розпізнавання.

Після розмиття необхідно адаптивно виділити текстові символи, що здійснюється через бінаризацію методом Оцу (cv2.threshold(img, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)). Метод Оцу автоматично визначає оптимальне порогове значення, яке розділяє текстові символи та фон номерного знака. Формула Оцу має вигляд:

T=arg⁡max⁡τ(σмежа2)T = \arg\max\_{\tau} \left( \sigma\_{\text{межа}}^2 \right)

де τ\tau – порогове значення, що дозволяє коригувати розмежування темних та світлих областей. Це критично важливо, оскільки OCR-алгоритми працюють найкраще саме з чіткими чорно-білими символами, тому бінаризація покращує точність їх розпізнавання.

Останнім етапом є розпізнавання тексту за допомогою нейронної мережі OCR, яка аналізує очищене зображення номерного знака. Використання **Tesseract OCR** у процесі розпізнавання текстових символів на номерних знаках дозволяє застосовувати **алгоритми глибокого навчання**, які були попередньо треновані на величезних вибірках даних, включаючи численні варіації написання літер, цифр, різні стилі шрифтів та складні умови освітлення. Завдяки цьому підходу OCR-система здатна адаптуватися до різноманітних сценаріїв розпізнавання, зокрема до номерних знаків із низькою якістю друку, потертостями, засвіченням або частковими перекриттями. **Модель Tesseract** використовує багаторівневу архітектуру, яка проходить через декілька етапів аналізу тексту, включаючи **векторизацію, нормалізацію контурів, сегментацію** і остаточну класифікацію символів [9]. На першому етапі OCR аналізує контури і розташування символів, визначаючи межі текстових областей. Потім виконується **класифікація символів**, під час якої алгоритм порівнює отримані дані з попередньо навченими патернами і визначає найбільш ймовірне значення кожного символу. Після цього відбувається процес **перевірки коригування**, який дозволяє виправити можливі помилки розпізнавання шляхом аналізу контексту – наприклад, якщо OCR неправильно розпізнав "0" замість "O", система може застосувати мовні моделі та статистичні алгоритми для перевірки коректності результату. Остаточним етапом є **зіставлення отриманих даних із відомими патернами**, що дозволяє підтвердити точність розпізнавання шляхом порівняння розпізнаних текстових фрагментів із базою відомих номерних знаків або стандартних шрифтів. Завдяки цьому алгоритму OCR-система забезпечує високу точність розпізнавання, робить процес автоматизованим і дозволяє застосовувати технологію у реальних сценаріях, зокрема для дорожнього моніторингу, контролю транспорту, безпеки та автоматизації паркувальних систем.

## 3.4 Оптимізація продуктивності програми

Процес оптимізації продуктивності програмного забезпечення для автоматичного розпізнавання номерних знаків є багатоступеневим завданням, що охоплює не лише покращення швидкості обробки, а й ефективність використання апаратних ресурсів, мінімізацію затримок у роботі нейронних мереж та адаптацію алгоритмів до різних реальних умов експлуатації. На початковому етапі важливо структурувати систему роботи з даними, оскільки неправильна організація файлової системи може призвести до зайвих обчислювальних витрат, зниження швидкодії та уповільнення процесу розпізнавання. У коді передбачено автоматичне створення директорій, що дозволяє ефективно зберігати проміжні результати та моделі, включаючи папки results, models, crops. Завдяки цьому підходу система не витрачає додатковий час на пошук файлів та доступ до них, оскільки вся структура даних вже оптимізована для швидкої обробки. Це також дозволяє мінімізувати зайві операції зчитування та запису, що особливо важливо при роботі з великими наборами зображень.

Наступним аспектом оптимізації є швидка передобробка зображень, яка впливає на загальну продуктивність алгоритму. Вихідні зображення часто містять артефакти, шум, неоднорідне освітлення або нерівномірний фон, що може суттєво погіршити результати OCR-аналізу. Для усунення цих проблем у коді застосовується перетворення у градації сірого, що дозволяє зменшити складність обробки зображень та виділити контури текстових символів. Цей процес реалізується через cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY), а його математична формула виглядає так:

Igray=0.2989R+0.5870G+0.1140BI\_{\text{gray}} = 0.2989 R + 0.5870 G + 0.1140 B

де R, G, B – відповідні кольорові канали. Завдяки цій операції система може працювати лише з інтенсивністю пікселів, що значно зменшує обчислювальне навантаження при подальшій бінаризації. Для покращення контурів текстових символів також застосовується Гаусове розмиття, що допомагає зменшити рівень шуму та згладити нерівномірності на номерному знаку. Використання cv2.GaussianBlur(img, (5, 5), 0) дозволяє адаптивно коригувати рівні освітлення, а його математична формула виглядає так:

G(I,k)=∑i,jI(i,j)⋅e−x2+y22σ2G(I, k) = \sum\_{i,j} I(i,j) \cdot e^{-\frac{x^2 + y^2}{2\sigma^2}}

де kk – розмір ядра фільтра, а σ\sigma – параметр згладжування. Ця операція суттєво покращує якість номерного знака, підвищуючи точність OCR-аналізу.

Ключовим моментом оптимізації продуктивності є ефективне використання нейронних мереж, що дозволяє скоротити час обробки без втрати точності розпізнавання. У коді застосовується YOLO, яка працює за принципом єдиного передбачення bounding box за один цикл, що є суттєво швидшим, ніж традиційні каскадні методи детекції. Оптимізація YOLO включає використання GPU-обчислень, що дозволяє паралельно обробляти велике число зображень, зменшуючи затримки у роботі системи. Для цього використовується CUDA-ускорення, яке переносить обчислення з CPU на графічний процесор, покращуючи ефективність детекції [16, c. 44–49]. Крім того, у коді реалізовано batch-processing, що дозволяє групувати зображення перед подачею в нейронну мережу, зменшуючи витрати на завантаження моделей та підвищуючи продуктивність у реальному часі.

Ще одним важливим аспектом оптимізації є багатопотокова обробка, яка дозволяє одночасно виконувати кілька незалежних процесів, таких як детекція номерного знака, OCR-аналіз та збереження результатів. Використання multiprocessing у Python дає змогу запускати незалежні потоки для паралельної роботи, що суттєво покращує загальну швидкість обробки даних. У коді також реалізовано асинхронне завантаження зображень, що дозволяє обробляти наступні кадри, поки попередні ще знаходяться у процесі аналізу, тим самим зменшуючи затримку при аналізі відеопотоку [23].

Для ще більшого підвищення швидкодії програми застосовується квантування нейронної мережі, що дозволяє суттєво зменшити точність числових операцій без втрати якості розпізнавання, тим самим знижуючи обчислювальні витрати. Завдяки цьому замість точного представлення даних використовується наближене, що дозволяє значно скоротити ресурси пам’яті та обчислень, оптимізуючи роботу системи навіть при високих навантаженнях. Крім того, для мінімізації затримок при виконанні моделей активно застосовуються технології TensorRT та ONNX, які сприяють прискоренню обробки даних та підвищенню загальної продуктивності алгоритму [11, c. 112–117]. Це особливо важливо у реальних умовах експлуатації, де система повинна працювати в режимі реального часу, наприклад, у автоматизованих дорожніх системах, системах контролю паркування чи безпекових рішеннях. Використання таких оптимізаційних методів дозволяє значно зменшити затримку між обробкою кадрів, забезпечити швидку реакцію на вхідні дані та підвищити точність роботи алгоритму навіть у складних сценаріях. Завдяки цьому нейронна мережа здатна ефективно виконувати свої функції, обробляючи великі обсяги даних без перевантаження системних ресурсів, що робить її придатною для комерційного застосування та інтеграції у масштабні інфраструктурні проекти, де швидкість і точність розпізнавання відіграють критично важливу роль.

# Розділ 4. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ

Тестування та впровадження системи автоматичного розпізнавання номерних знаків є критично важливими етапами розробки, оскільки вони визначають її точність, продуктивність, стійкість до різних умов експлуатації та можливість інтеграції у реальне середовище. Ці процеси включають ретельну перевірку кожного компонента програми, оцінку ефективності роботи моделі на тестових наборах даних, аналіз продуктивності при різних сценаріях використання, виявлення потенційних проблем та підготовку до реального впровадження у дорожні системи, паркінгову інфраструктуру або безпекові рішення.

На початковому етапі тестування система проходить внутрішню перевірку функціональності, яка включає оцінку ефективності нейронної мережі YOLO у детекції номерних знаків та аналіз точності OCR-розпізнавання тексту. Для цього використовуються контрольні набори зображень, що містять різні варіації номерних знаків із різними умовами освітлення, забрудненнями, різними стилями написання символів та рівнями розмитості. Такий тестовий набір даних дозволяє оцінити, як добре система працює при реальних викликах, наприклад, уночі, при сильних відблисках або коли номерний знак частково перекритий іншими об'єктами. Важливим аспектом тестування є розрахунок показника Intersection over Union (IoU), що дозволяє кількісно оцінити ефективність детекції bounding box у нейронній мережі YOLO. IoU визначає, наскільки точно модель визначила область номерного знака порівняно з фактичним розташуванням об'єкта, використовуючи формулу:

IoU=площа перетинузагальна площа (об’єднання)IoU = \frac{\text{площа перетину}}{\text{загальна площа (об'єднання)}}

Чим вище значення IoU (більше 0.75), тим точніше працює детекція номерного знака, а якщо IoU менше 0.5, це означає, що модель потребує додаткового доопрацювання, наприклад, удосконалення гіперпараметрів або збільшення розміру навчального набору даних.

Після перевірки точності детекції здійснюється тестування OCR-ідентифікації тексту, яке включає аналіз правильності розпізнавання символів, перевірку коректного визначення букв і цифр та оцінку загальної точності розпізнавання. Для цього використовується метрика Character Error Rate (CER), яка визначає, наскільки правильно розпізнаний текст у порівнянні зі справжнім номером автомобіля. Формула розрахунку CER має вигляд:

CER=Кількість неправильних символівЗагальна кількість символів у номеріCER = \frac{\text{Кількість неправильних символів}}{\text{Загальна кількість символів у номері}}

Якщо система OCR неправильно розпізнає окремі символи (наприклад, "O" замість "0" або "B" замість "8"), це може призвести до значних помилок у розпізнаванні. Оптимізація OCR виконується шляхом удосконалення алгоритмів бінаризації та адаптивної нормалізації, що дозволяє збільшити контрастність тексту та мінімізувати спотворення [12].

Наступним ключовим етапом тестування є перевірка продуктивності алгоритму при обробці відеопотоку, що дає змогу оцінити, наскільки ефективно система функціонує в умовах реального часу. Оскільки в багатьох сценаріях використання система отримує дані безперервно з камер спостереження, критично важливо оцінити її швидкодію та здатність обробляти великий обсяг інформації без затримок. Для цього застосовується FPS-тестування (Frames Per Second), що визначає кількість кадрів, які система здатна опрацювати за секунду: якщо FPS падає нижче 10, це свідчить про необхідність проведення оптимізації для підвищення продуктивності. Одним із ключових рішень у таких випадках є перехід на GPU-обробку із використанням технологій, таких як CUDA, що дозволяє перенести основні обчислення на графічний процесор, суттєво зменшуючи навантаження на CPU. Додатково можливо застосувати паралельні обчислення, що дає змогу розподіляти обробку кадрів між кількома процесами, покращуючи загальну швидкодію системи. Крім того, для підвищення ефективності розпізнавання можна використовувати легші моделі нейронної мережі, оптимізовані для швидкої роботи без значних втрат у точності, що особливо важливо в умовах високонавантажених систем, таких як автоматизований контроль транспорту або аналіз дорожнього руху. Завдяки такій стратегії тестування та оптимізації можна забезпечити стабільну та ефективну роботу системи в потокових даних, що критично важливо для її практичного застосування в умовах реального середовища.

Після завершення всіх етапів тестування система автоматичного розпізнавання номерних знаків проходить етап впровадження у реальне середовище, що включає інтеграцію з дорожніми камерами, паркінгами, системами моніторингу трафіку та безпековими службами, які використовують автоматизоване розпізнавання для контролю руху транспорту, реєстрації порушень або організації пропускного режиму. Важливим аспектом цього процесу є оцінка стійкості алгоритму до зовнішніх факторів, таких як зміни погодних умов, варіації у типах камер, різні формати зображень, а також навантаження на систему при потоковій обробці великої кількості кадрів. Наприклад, система повинна коректно працювати як у сонячну погоду, так і при тумані чи дощі, коли видимість номерного знака може бути значно знижена. Оскільки на дорогах встановлено камери з різними характеристиками (роздільною здатністю, частотою кадрів, якістю матриці), алгоритм повинен адаптуватися до змін у вхідних даних та стабільно працювати незалежно від параметрів відеопотоку. Ще одним критично важливим аспектом впровадження є здатність системи обробляти велике навантаження, коли в кадрі одночасно можуть знаходитися декілька автомобілів, а обробка даних має відбуватися у реальному часі без затримок [13]. У коді передбачено адаптивне коригування параметрів, що дозволяє змінювати яскравість, контрастність та швидкість обробки залежно від умов експлуатації: якщо кадр занадто темний, застосовується нормалізація освітлення, а якщо фон містить занадто багато деталей, використовується додаткове шумозаглушення. Також реалізовано автоматичне коригування чутливості алгоритму, що дозволяє оптимізувати детекцію номерного знака на основі динамічного аналізу потоку даних, забезпечуючи максимальну точність у будь-яких реальних умовах експлуатації. Завдяки такій комплексній оптимізації впровадження система здатна працювати у масштабованих проектах, забезпечуючи автоматичне розпізнавання номерних знаків без необхідності ручного налаштування для кожного конкретного випадку, що підвищує її ефективність та готовність до інтеграції у транспортну інфраструктуру, безпекові рішення та інтелектуальні дорожні системи.

## 4.1 Методи тестування (набір тестових даних, оцінка точності)

Успішне функціонування системи розпізнавання номерних знаків залежить від комплексного процесу тестування, що охоплює всі ключові аспекти її роботи, включаючи точність детекції, продуктивність алгоритмів та здатність адаптуватися до змінних умов експлуатації. Першим етапом є формування тестового набору даних, який містить зображення номерних знаків, відзнятих у різних умовах—зміна освітлення, різні кути огляду, забруднення або часткове перекриття об'єктів, що допомагає оцінити стійкість моделі. Далі проводиться перевірка точності детекції, де аналізується здатність алгоритму правильно знаходити та сегментувати номерні знаки на зображеннях. Окремим критично важливим аспектом є оцінка OCR-розпізнавання, де вимірюється кількість помилок у зчитуванні текстових символів та визначаються можливі причини їх появи, такі як недостатня контрастність або наявність артефактів. Завершальним етапом є тестування продуктивності у потоковому відео, де оцінюється FPS (Frames Per Second) для перевірки швидкості роботи в реальному часі, що дозволяє виявити можливі вузькі місця у продуктивності системи. Проведення всебічного аналізу отриманих результатів дає можливість не лише оцінити поточну ефективність алгоритму, а й внести необхідні корективи в його структуру, удосконалюючи передобробку зображень, оптимізуючи продуктивність та мінімізуючи ризик помилок при розпізнаванні номерних знаків у складних умовах [14].

Першим критично важливим аспектом тестування системи автоматичного розпізнавання номерних знаків є створення якісного тестового набору даних, що охоплює широкий спектр можливих сценаріїв реального використання. Оскільки у реальних умовах номерні знаки можуть бути сфотографовані під різними кутами, у нестабільному освітленні, частково забрудненими або перекритими іншими об'єктами, необхідно забезпечити адаптивність моделі до таких ситуацій. Для цього у тестовий набір включаються різноманітні зображення номерних знаків, що враховують варіації у рівні яскравості, наявності тіней, ступеня розмиття, а також різні роздільні здатності камер і стилі шрифтів, які можуть впливати на точність розпізнавання. У коді передбачено автоматизовану підготовку тестових зображень, яка дозволяє сортувати їх за категоріями: чіткі номерні знаки, засвічені номери, частково перекриті знаки, нестандартні шрифти або пошкоджені текстові елементи. Такий підхід не лише дає змогу перевірити загальну працездатність системи, але й виявити конкретні випадки, у яких алгоритм демонструє знижену точність, що дозволяє оперативно скоригувати його налаштування для покращення продуктивності. Додатково проводиться аналіз помилок розпізнавання, що допомагає вдосконалити методи передобробки зображень, адаптивну корекцію контрастності та фільтрацію шуму, що сприяє підвищенню стабільності алгоритму в умовах реального застосування.

Наступним кроком є оцінка точності детекції номерного знака, що реалізується через аналіз ефективності роботи нейронної мережі YOLO у розпізнаванні об’єкта на зображенні. Основною метрикою для оцінки точності є Intersection over Union (IoU), що визначає, наскільки точно нейронна мережа локалізувала номерний знак у кадрі. Формула IoU виглядає так:

IoU=площа перетинузагальна площа (об’єднання)IoU = \frac{\text{площа перетину}}{\text{загальна площа (об'єднання)}}

Якщо значення IoU високе (більше 0.75), це означає, що детекція працює добре, оскільки передбачене нейронною мережею bounding box майже повністю відповідає реальному розташуванню номерного знака на зображенні. Якщо ж IoU менше 0.5, це свідчить про необхідність покращення навчального набору або корекцію гіперпараметрів моделі. Важливим аспектом тестування детекції є також аналіз False Positive (FP) та False Negative (FN) помилок: FP означає, що модель неправильно визначила на зображенні номерний знак там, де його немає, тоді як FN свідчить про пропуск об'єкта, що має бути розпізнаним.

Крім детекції, важливим етапом тестування є перевірка точності OCR-аналізу, що здійснюється через оцінку коректності розпізнавання текстових символів. Використовується метрика Character Error Rate (CER), що визначає кількість неправильно розпізнаних символів порівняно із загальною кількістю символів у номерному знаку. Формула CER виглядає так:

CER=Кількість неправильних символівЗагальна кількість символів у номеріCER = \frac{\text{Кількість неправильних символів}}{\text{Загальна кількість символів у номері}}

Якщо значення CER високе (більше 10%), це означає, що OCR-алгоритм допускає значну кількість помилок у розпізнаванні тексту, і необхідно покращити передобробку зображення перед аналізом, наприклад, застосовуючи кращу бінаризацію методом Оцу, адаптивну корекцію контрастності або фільтрацію артефактів.

Фінальним етапом тестування системи автоматичного розпізнавання номерних знаків є перевірка її ефективності у реальному відеопотоці, що дозволяє оцінити швидкодію та стабільність роботи алгоритму в умовах безперервної обробки кадрів. Для цього у коді передбачено FPS-аналіз (Frames Per Second), який визначає, скільки кадрів програма здатна обробити за секунду, що є ключовим параметром продуктивності. Якщо значення FPS опускається нижче 10, це сигналізує про недостатню швидкість роботи алгоритму, що може призводити до затримок у розпізнаванні та зниження загальної ефективності системи. У таких випадках необхідно здійснити оптимізацію обчислювальних процесів, зокрема шляхом переходу на GPU-обчислення за допомогою технології CUDA, що суттєво прискорює обробку відеопотоку, або використання мультипоточності (multiprocessing у Python), яка дозволяє розподіляти навантаження між незалежними процесами, прискорюючи паралельне виконання детекції об'єктів, OCR-аналізу та збереження даних. Крім того, можливе коригування складності алгоритму передобробки зображень, щоб балансувати між точністю та швидкістю обробки, особливо у високонавантажених системах. У реальних умовах експлуатації розпізнавання номерних знаків має відбуватися у режимі реального часу без значних затримок, оскільки швидкість реагування є критично важливою для транспортної аналітики, безпеки та автоматизованих паркувальних систем. Саме тому комплексний аналіз продуктивності під час тестування охоплює не лише точність розпізнавання, а й здатність алгоритму ефективно працювати при обробці великого потоку даних, що гарантує його стабільність та відповідність вимогам сучасних високопродуктивних систем.

## 4.2 Аналіз отриманих результатів

Аналіз отриманих результатів у процесі тестування системи автоматичного розпізнавання номерних знаків відіграє ключову роль у визначенні її ефективності, виявленні потенційних недоліків та оптимізації роботи алгоритмів для підвищення продуктивності. Цей процес включає всебічну оцінку точності детекції об'єктів, перевірку результатів OCR-розпізнавання для виявлення помилок при зчитуванні тексту, а також аналіз швидкодії системи при потоковій обробці даних, де критично важливим параметром є FPS, що визначає, наскільки швидко кадри можуть бути оброблені в реальному часі. Додатково проводиться дослідження впливу змінних факторів, таких як нестабільне освітлення, зміни у форматах зображень, забруднення номерного знака, артефакти у кадрі або динамічний рух об'єктів, що може вплинути на точність алгоритму. Для оцінки адаптивності системи виконується тестування на різноманітних наборах даних, що містять зображення з різним рівнем контрастності, розмиття та сторонніми перешкодами, що дозволяє перевірити, наскільки модель здатна ефективно працювати у нестандартних умовах. Якщо аналіз показує значне зниження точності у складних сценаріях, у коді передбачено механізми автоматичної корекції параметрів, зокрема адаптивне регулювання контрастності, динамічне налаштування фільтрації шуму та покращення алгоритмів передобробки, що дозволяє суттєво покращити якість розпізнавання та забезпечити стабільність роботи системи незалежно від зовнішніх факторів. Завдяки такому ретельному підходу до тестування та оптимізації алгоритмів система здатна ефективно функціонувати у широкому спектрі реальних застосувань, від транспортної аналітики до безпекових систем, забезпечуючи високу точність та швидкість розпізнавання у будь-яких умовах експлуатації [16, c. 44–49].

Перший ключовий аспект аналізу—оцінка точності детекції номерного знака. Як відомо, процес виявлення номерного знака виконується за допомогою нейронної мережі YOLO, яка здійснює детекцію об’єкта у режимі реального часу. Для визначення того, наскільки якісно працює модель, застосовується метрика Intersection over Union (IoU)—критерій, що дозволяє кількісно оцінити, наскільки точно передбачений bounding box збігається з фактичним розташуванням номерного знака на зображенні. Формула IoU має вигляд:

IoU=площа перетину передбаченого та реального bounding boxзагальна площа (об’єднання двох областей)IoU = \frac{\text{площа перетину передбаченого та реального bounding box}}{\text{загальна площа (об'єднання двох областей)}}

Високе значення IoU (більше 0.75) свідчить про те, що система правильно ідентифікує межі номерного знака з високою точністю, що є критично важливим для подальшої OCR-обробки. Однак, якщо IoU менше 0.5, це може означати, що модель припускається значних помилок у детекції, що потребує додаткового навчання або покращення алгоритмів передобробки. У процесі аналізу отриманих результатів проводиться детальна перевірка випадків False Positive (FP)—неправильне визначення номерного знака там, де його немає, та False Negative (FN)—пропуск реального номерного знака. Велика кількість FP та FN помилок може означати, що модель потребує коригування параметрів або використання додаткових фільтрів для покращення точності.

Другий важливий аспект аналізу—перевірка коректності OCR-розпізнавання тексту. Після того, як номерний знак детектовано, він проходить через етап OCR (Optical Character Recognition), який використовує Tesseract OCR для розпізнавання текстових символів. Для оцінки точності роботи OCR використовується метрика Character Error Rate (CER), що визначає кількість неправильно розпізнаних символів у порівнянні з загальною кількістю знаків у номерному знаку. Формула CER виглядає наступним чином:

CER=Кількість неправильно розпізнаних символівЗагальна кількість символів у номерному знакуCER = \frac{\text{Кількість неправильно розпізнаних символів}}{\text{Загальна кількість символів у номерному знаку}}

Якщо значення CER (Character Error Rate) перевищує 10%, це свідчить про значну кількість помилок у процесі OCR-розпізнавання тексту, що може негативно вплинути на ефективність роботи системи. Такі неточності часто виникають через низьку якість вхідного зображення, некоректно виконану бінаризацію або наявність артефактів, таких як відблиски, тіні чи механічні пошкодження номерного знака. Для покращення точності OCR в коді реалізовано комплексні методи передобробки, що включають адаптивний пороговий аналіз, який динамічно регулює рівень освітлення для оптимального виділення тексту; нормалізацію яскравості, що вирівнює контрастність зображення та усуває локальні перепади освітлення; а також розмиття Гауса, яке зменшує рівень шуму та допомагає відфільтрувати зайві деталі, що можуть вплинути на коректність розпізнавання [17, c. 412]. Завдяки цим підходам значно знижується ризик неправильного зчитування символів, таких як плутанина між "O" та "0" або "B" та "8", а точність алгоритму істотно підвищується навіть у випадках низької якості друку номерного знака чи складних умов зйомки. Це критично важливо для транспортних систем, де OCR використовується в режимі реального часу і точність розпізнавання безпосередньо впливає на загальну ефективність роботи алгоритму.

Ще одним важливим аспектом аналізу отриманих результатів є оцінка продуктивності системи при потоковій обробці зображень або відео, оскільки для ефективного функціонування алгоритму у режимі реального часу необхідно забезпечити достатню швидкість обробки кадрів. Ключовим параметром продуктивності є FPS (Frames Per Second), що визначає кількість кадрів, які система здатна обробляти за секунду: якщо це значення падає нижче 10, продуктивність алгоритму вважається недостатньою, що може призводити до затримок та некоректної роботи. Для підвищення швидкості обробки у коді передбачено GPU-ускорення, яке дозволяє переносити основні обчислення з центрального процесора (CPU) на графічний процесор (GPU) за допомогою технології CUDA, що суттєво покращує швидкодію при роботі з відеопотоком та зменшує загальне навантаження на систему. Додатково застосовується мультипоточність (multiprocessing у Python), яка забезпечує паралельне виконання незалежних процесів: одночасну обробку зображень, OCR-аналіз та збереження результатів, що мінімізує затримку між кадрами та оптимізує ефективність алгоритму у високонавантажених умовах. Завдяки такому підходу до оптимізації продуктивності система може працювати безперебійно навіть при високій роздільній здатності відео та динамічних змінах кадру, що критично важливо для застосування у транспортній аналітиці, дорожньому контролі, автоматизованих паркувальних системах та інших сферах, де оперативність обробки даних відіграє вирішальну роль.

Останнім критично важливим аспектом аналізу отриманих результатів є перевірка адаптивності системи до реальних умов експлуатації, адже у різних сценаріях її використання можуть виникати численні виклики, що впливають на точність розпізнавання. Зміни освітлення, варіації у форматах зображень, забруднення номерного знака або випадкові перешкоди в кадрі можуть значно ускладнити роботу алгоритму, тому для оцінки його стійкості проводиться тестування на варіативних тестових наборах, що містять зображення номерних знаків із різним рівнем розмиття, яскравості та контрастності. Це дозволяє визначити межі ефективності алгоритму та виявити його слабкі місця, що можуть призводити до зниження точності розпізнавання в нестандартних умовах. Якщо система демонструє значне погіршення результатів у складних середовищах, у коді передбачено автоматичну корекцію ключових параметрів обробки—зокрема, динамічне регулювання контрастності, що дозволяє адаптивно підлаштовувати рівень освітлення в кадрі для покращення видимості тексту, або використання додаткових фільтрів шумозаглушення, які ефективно усувають артефакти, викликані низькою якістю друку номерного знака чи оптичними спотвореннями. Завдяки цим механізмам алгоритм зберігає високу точність розпізнавання навіть у складних умовах, що критично важливо для його інтеграції у системи безпеки, транспортну аналітику та логістичні процеси, де необхідно забезпечити стабільну продуктивність незалежно від зовнішніх факторів.

## 4.3 Порівняння ефективності з аналогами

Розпізнавання номерних знаків автомобілів є одним із ключових напрямів розвитку комп'ютерного зору, оскільки воно знаходить широке застосування у дорожньому контролі, безпеці, паркувальних системах та транспортній логістиці. Сучасні алгоритми реалізують цей процес із використанням передових моделей глибокого навчання, таких як YOLO для детекції об'єктів та OCR для аналізу текстових символів, що дозволяє досягти високої точності та швидкості розпізнавання. Однак ефективність кожного методу залежить від кількох критично важливих факторів: рівня передобробки зображення, адаптивності алгоритму до змінних умов освітлення, швидкості обробки та здатності функціонувати в реальному часі. Запропонований алгоритм відрізняється від традиційних рішень покращеною стабільністю, оскільки використовує динамічне регулювання яскравості, контрастності та фільтрацію шуму, що дозволяє значно зменшити кількість помилок розпізнавання навіть у складних умовах експлуатації. Крім того, він забезпечує високу продуктивність завдяки оптимізації обчислювальних ресурсів, що робить його перспективним для інтеграції у масштабовані інфраструктурні проєкти, де необхідно обробляти великі обсяги даних без затримок. Одним із ключових аспектів оцінки його ефективності є порівняльний аналіз точності та продуктивності у порівнянні з аналогами, які вже застосовуються в транспортних системах, автоматизованих паркувальних комплексах та безпекових технологіях, що дозволяє виявити його переваги та можливості подальшого удосконалення.

Першим аспектом порівняння є методи детекції номерних знаків, що використовуються у різних підходах. Запропонована система використовує YOLO (You Only Look Once) для визначення області номерного знака у кадрі, що суттєво відрізняється від традиційних методів, таких як каскадні детектори Хаара, методи сегментації на основі кольору або аналіз контурів та морфологічна фільтрація. Каскадні детектори, які раніше використовувалися для розпізнавання текстових об'єктів, мають значні недоліки, такі як висока залежність від освітлення, чутливість до фону та низька швидкість обробки. У порівнянні з ними, YOLO працює на основі глибоких нейронних мереж, що дозволяє миттєво передбачати координати bounding box за один цикл, що значно підвищує швидкість детекції та точність [18, c. 78]. У багатьох комерційних рішеннях використовується метод трасування контурів, який дозволяє знаходити номерні знаки на основі форми. Однак такий підхід має значні обмеження—якщо номерний знак розмитий або має складний фон, алгоритм може неправильно класифікувати об'єкт, що знижує точність. Запропонована система, завдяки YOLO, демонструє вищу ефективність у складних умовах, таких як нечітке зображення, затемнення, перекриття об'єкта іншими елементами.

Наступним критично важливим аспектом є точність OCR-розпізнавання, яка визначає здатність системи ефективно і безпомилково ідентифікувати текстові символи на номерних знаках після попередньої обробки зображення. Традиційні методи Optical Character Recognition (OCR), що не використовують глибоке навчання, покладаються на пороговий аналіз та фільтрацію шуму, проте такі підходи часто не дають бажаних результатів у випадках низької якості номерного знака, змазаного тексту чи наявності артефактів, що ускладнюють розпізнавання. Запропонований алгоритм, заснований на Tesseract OCR, суттєво перевершує класичні методи завдяки інтеграції лексичних моделей та адаптації символів до специфічних шаблонів номерних знаків, що дозволяє значно покращити точність розпізнавання навіть у складних умовах. Система аналізує контекст тексту, що мінімізує ймовірність неправильного зчитування, зокрема у випадках схожих символів, як-от "O" замість "0" або "B" замість "8". Крім того, передобробка зображень включає застосування градацій сірого для покращення контрастності, розмиття Гауса для зменшення шуму та бінаризацію Оцу для чіткішого виділення текстових елементів, що суттєво підвищує ефективність роботи OCR. У результаті, кількість помилок при розпізнаванні значно зменшується, а точність алгоритму зростає, забезпечуючи надійне та точне визначення тексту навіть у нестандартних або складних сценаріях використання. Завдяки такому підходу система може успішно працювати у дорожньому контролі, автоматизації паркування та логістичних процесах, забезпечуючи високу якість аналізу номерних знаків у реальному часі.

Ще одним важливим аспектом порівняння є продуктивність алгоритму та швидкість обробки зображень у реальному часі. У стандартних аналогах, таких як системи розпізнавання номерних знаків, що використовують класичні машинні методи (SVM, KNN, HOG), обробка одного кадру може займати значний час, оскільки кожен етап передбачає поступове виконання детекції, сегментації та аналізу тексту. Запропонований алгоритм значно швидший, оскільки використовує оптимізовану архітектуру YOLO, яка працює у реальному часі та передбачає координати bounding box без необхідності каскадної обробки. Крім того, завдяки GPU-ускоренню (CUDA) система має високий FPS (Frames Per Second) у потоковій обробці, що робить її придатною для аналізу відеопотоку у системах безпеки або дорожнього контролю. У традиційних методах продуктивність часто знижується через залежність від CPU-обчислень, що стає вузьким місцем при роботі з великими наборами даних, у той час як у нашому випадку паралельна обробка кадрів дозволяє забезпечити стабільну швидкодію навіть при великому потоці інформації.

Крім того, важливо оцінити адаптивність алгоритму до реальних умов експлуатації, оскільки традиційні рішення, засновані на фіксованих параметрах розпізнавання, часто стикаються з серйозними обмеженнями, що впливають на їхню ефективність у змінних умовах. Наприклад, варіації освітлення, забруднення номерного знака, а також відблиски чи тіні можуть значно погіршити точність розпізнавання, спричиняючи високий рівень помилок. Запропонований алгоритм вирішує ці проблеми шляхом впровадження адаптивного регулювання яскравості, динамічного налаштування контрастності та розширених методів фільтрації зображення, що дозволяє коригувати параметри розпізнавання відповідно до конкретних умов кадру. Завдяки цьому система здатна ефективно працювати в умовах низького освітлення, компенсувати різні стилі шрифтів, а також обробляти номерні знаки з пошкодженнями або зниженою якістю друку. Додатково алгоритм використовує механізми глибокого навчання, які аналізують контекстні особливості кожного кадру та динамічно коригують параметри аналізу для підвищення точності розпізнавання [19, c. 364]. Це забезпечує надійність системи навіть у складних умовах, таких як нічна зйомка, дощ чи туман, і робить її значно ефективнішою порівняно з традиційними підходами, що не враховують змінні фактори середовища. Таким чином, впровадження адаптивних алгоритмів дозволяє значно розширити можливості автоматизованого розпізнавання номерних знаків та забезпечити його стабільну роботу у широкому спектрі застосувань.

## 4.4 Рекомендації щодо застосування

Автоматизовані системи розпізнавання номерних знаків, які використовують сучасні алгоритми машинного навчання, зокрема YOLO для високошвидкісної детекції об'єктів та OCR для точного текстового аналізу, демонструють вражаючу ефективність у широкому спектрі застосувань, включаючи дорожній контроль, безпекові системи, автоматизацію паркування, транспортну логістику та аналітику трафіку. Завдяки здатності працювати у реальному часі й адаптуватися до змінних умов освітлення та погодних факторів, такі системи стають невід'ємним елементом масштабованих інфраструктурних проєктів, де критично важливо оперативно обробляти великі масиви даних для прийняття рішень. Висока точність розпізнавання досягається завдяки вдосконаленим моделям нейронних мереж, які дозволяють компенсувати спотворення, викликані рухом транспортних засобів або низькою якістю відеозапису. Однак кожна сфера застосування має специфічні вимоги, що потребують ретельної оптимізації алгоритмів: у дорожньому контролі важливий аналіз потоків трафіку та виявлення порушень, у паркувальних системах — інтеграція з платіжними рішеннями, а в логістичних компаніях — синхронізація даних з управлінням складськими процесами [20]. Саме тому для ефективного впровадження таких технологій необхідно адаптувати моделі відповідно до конкретного сценарію використання, враховуючи особливості кожної індустрії та технічні вимоги до точності й швидкості роботи системи.Одним із ключових напрямків застосування є автоматизований контроль дорожнього руху, який використовується правоохоронними органами, системами моніторингу міських транспортних мереж та організаціями, що займаються безпекою на дорогах. У таких випадках система розпізнавання номерних знаків дозволяє ідентифікувати транспортні засоби в реальному часі, створюючи базу даних зареєстрованих автомобілів та допомагаючи відстежувати випадки порушення правил дорожнього руху. Наприклад, при перевищенні швидкості або проїзді на червоне світло камери можуть автоматично фіксувати номерний знак порушника, а алгоритм OCR передає розпізнаний текст у систему штрафів або попереджувальних заходів. Однією з основних вимог для такого застосування є висока швидкість обробки даних, тому важливо забезпечити паралельну обробку кадрів, GPU-ускорення та адаптивне регулювання параметрів зображення, щоб уникнути втрати точності при складних умовах освітлення. У коді передбачено використання технології CUDA, що дозволяє значно зменшити час виконання детекції та OCR-розпізнавання, що є критичним фактором для дорожніх систем моніторингу.

Ще одним перспективним напрямком є інтелектуальні паркінгові системи, де технологія розпізнавання номерних знаків може автоматично перевіряти в’їзд та виїзд транспортних засобів, забезпечувати контроль доступу та зменшувати потребу у використанні додаткових фізичних карт пропуску чи жетонів. У таких випадках необхідно адаптувати алгоритм до роботи з кадрами, отриманими з різних кутів, оскільки камери можуть бути встановлені у місцях, де номерний знак фіксується під нахилом або частково перекривається іншими об'єктами. Для підвищення точності розпізнавання можна використовувати алгоритми корекції перспективи, що дозволяють адаптувати нерівномірне освітлення або виправляти нахил текстових символів перед передачею їх в OCR-аналітику. Крім того, варто забезпечити додаткову фільтрацію шуму, оскільки паркувальні системи можуть працювати у напівтемних умовах або вночі, що ускладнює розпізнавання тексту. У коді передбачено використання бінаризації методом Оцу, що дозволяє адаптивно коригувати освітлення перед аналізом тексту, забезпечуючи максимальну точність навіть у складних сценаріях експлуатації [20].

Ще одним важливим напрямком застосування є безпекові системи, які інтегрують розпізнавання номерних знаків у систему контролю доступу, військові та урядові об'єкти, а також приватні охоронні мережі. У таких випадках важливо забезпечити надзвичайно високу точність розпізнавання, оскільки будь-яка помилка у розпізнаванні може призвести до некоректного доступу або безпекових ризиків. Наприклад, у військових комплексах система може автоматично перевіряти список дозволених транспортних засобів, і якщо номер не збігається із зареєстрованими даними, активується сигналізація чи відбувається блокування доступу. Крім того, безпекові системи часто працюють із відеопотоком, тому важливо оптимізувати продуктивність системи, забезпечуючи швидке розпізнавання номерних знаків без затримок. У коді передбачено мультипоточну обробку, що дозволяє запускати одночасно кілька процесів аналізу, підвищуючи загальну ефективність розпізнавання без втрати продуктивності.

Окрім безпекових аспектів, система розпізнавання номерних знаків відкриває широкі можливості для оптимізації роботи логістичних компаній, кур'єрських служб та транспортних мереж, сприяючи автоматизації контролю переміщення транспортних засобів та покращенню ефективності логістичних операцій. Завдяки її інтеграції можна зменшити адміністративне навантаження, оперативно відстежувати прибуття й відправлення машин, а також синхронізувати дані з внутрішніми базами та поточними замовленнями, що особливо важливо у великих транспортних хабах і складських комплексах. Автоматичне зчитування номерних знаків дозволяє вести точний облік вантажів та забезпечити контроль у реальному часі, що сприяє зменшенню затримок, підвищенню точності планування маршрутів і загальному поліпшенню якості послуг. Однак успішна реалізація такої системи потребує стабільного функціонування навіть у складних умовах, включаючи погане освітлення, рух транспорту, розмиття кадрів та різні погодні фактори, що може вплинути на якість розпізнавання. Саме тому у коді передбачено алгоритми динамічного коригування яскравості, контрастності, насиченості та інших параметрів передобробки, що дозволяє автоматично адаптувати систему до специфічних умов відеопотоку, забезпечуючи максимальну точність розпізнавання незалежно від зовнішніх факторів. Такі алгоритми відіграють ключову роль у стабільному функціонуванні системи, адже вони здатні компенсувати зміни освітлення, усувати ефекти розмиття кадрів, викликані рухом транспорту, а також враховувати перешкоди, пов'язані з тінями, відблисками або несприятливими погодними умовами. У процесі обробки відео система аналізує рівень освітленості та динамічно коригує параметри, щоб забезпечити оптимальну якість зображення для подальшого аналізу. Крім того, алгоритми інтегровані з механізмами глибокого навчання, що дозволяє моделі адаптуватися до нових сценаріїв та вдосконалювати точність розпізнавання навіть у складних умовах експлуатації. Це забезпечує надійну роботу системи в реальному часі, що критично важливо для її застосування у логістичних компаніях, транспортних мережах або безпекових системах, де ефективність та швидкість обробки даних відіграють вирішальну роль у оптимізації процесів.

# ВИСНОВКИ

Проаналізувавши структуру роботи та детальні описи алгоритмів, можна зробити кілька ключових висновків, що стосуються ефективності запропонованої системи розпізнавання номерних знаків, її продуктивності, точності та можливостей впровадження.

1. Комплексний підхід до розробки системи розпізнавання номерних знаків охоплює всі ключові етапи, починаючи з глибокого аналізу існуючих методів та технологій комп’ютерного бачення і завершуючи впровадженням та оцінкою ефективності розробленого алгоритму. Такий методологічний підхід дозволяє не лише дослідити сучасні тренди у сфері машинного навчання та нейронних мереж, але й сформувати оптимальне рішення, що враховує найкращі практики та відповідає реальним викликам практичного застосування. Важливим етапом є детальний аналіз різних алгоритмів обробки зображень, моделей детекції та сегментації номерних знаків, що дає змогу оцінити їхні переваги та недоліки й визначити найбільш ефективні технології для інтеграції у власну систему. Додатково розглядається адаптація алгоритмів до змінних умов, таких як нестабільне освітлення, рух об’єктів у кадрі чи забруднення номерного знака, що дозволяє значно покращити точність розпізнавання. Завершальним аспектом є комплексне тестування продуктивності системи, включаючи аналіз швидкодії при обробці відеопотоку, оцінку точності OCR та коригування параметрів передобробки для забезпечення стабільної роботи алгоритму навіть у складних умовах. Такий системний підхід гарантує високий рівень адаптивності, продуктивності та точності розпізнавання номерних знаків, що робить розробку придатною для інтеграції у транспортні та безпекові рішення.

2. Застосування передових технологій у розробці системи розпізнавання номерних знаків суттєво підвищує її ефективність, забезпечуючи високу швидкість обробки та точність розпізнавання у реальних умовах експлуатації. Вибір YOLO як основного алгоритму детекції об'єктів дозволяє швидко та точно знаходити номерні знаки на зображеннях, навіть у складних сценах з великою кількістю об'єктів та змінним освітленням. Використання Tesseract OCR для аналізу тексту доповнює цей процес, гарантуючи коректне розпізнавання символів, включаючи складні випадки, такі як забруднені або частково пошкоджені номери. TensorFlow забезпечує потужні можливості машинного навчання, дозволяючи нейронній мережі адаптуватися до нових сценаріїв розпізнавання, удосконалюючи її точність за рахунок модифікації та навчання моделі на великій кількості варіативних даних. Додатково використання OpenCV для попередньої обробки зображень сприяє покращенню якості вхідних даних шляхом корекції яскравості, контрастності та видалення шумів, що істотно зменшує кількість помилок на етапі аналізу тексту. Такий комплексний підхід не лише підвищує продуктивність у порівнянні з класичними методами оптичного розпізнавання тексту, але й забезпечує стабільність роботи алгоритму у змінних умовах експлуатації, включаючи низьке освітлення, розмиття або часткове перекриття номерних знаків, що робить систему надійним рішенням для автоматизованих транспортних та безпекових технологій.

3. Оптимізація продуктивності системи розпізнавання номерних знаків є критично важливим аспектом її ефективного функціонування, особливо при обробці великого потоку даних у режимі реального часу. Впровадження багатопотокової обробки (multiprocessing) дозволяє розподілити навантаження між незалежними процесами, що прискорює виконання ключових завдань, таких як детекція об'єктів, передобробка зображень та OCR-аналіз. Додатково використання GPU-ускорення за допомогою технології CUDA значно зменшує затримку між кадрами та дозволяє перенести складні обчислення на графічний процесор, що суттєво підвищує швидкість обробки відеопотоку. Крім того, застосування формування батчів при навчанні нейронної мережі дозволяє оптимізувати використання пам’яті та забезпечити стабільну продуктивність моделі, навіть при великих обсягах вхідних даних. Завдяки такому комплексному підходу система здатна працювати швидко та безперебійно, що є критично важливим для її практичного застосування у сферах автоматизованого дорожнього контролю, паркінгових систем та безпеки, де мінімізація затримок і висока швидкість розпізнавання безпосередньо впливають на якість роботи алгоритму. Така оптимізація дозволяє ефективно інтегрувати систему у великі інфраструктурні проекти, забезпечуючи її надійність та відповідність вимогам сучасних високопродуктивних технологій.

4. Висока точність розпізнавання номерних знаків забезпечується завдяки застосуванню сучасних алгоритмів машинного навчання та ретельно налаштованих методів оцінки ефективності. Аналіз точності детекції номерного знака здійснюється за допомогою метрики Intersection over Union (IoU), яка дозволяє визначити, наскільки точно алгоритм виявляє номерний знак на зображенні, порівнюючи його прогнозоване положення з реальним. У запропонованій системі цей показник перевищує 0.75, що свідчить про високий рівень точності виявлення об'єктів навіть у складних умовах, таких як змінне освітлення, часткове перекриття або наявність шуму у кадрі. Оцінка точності OCR-розпізнавання базується на Character Error Rate (CER), що визначає частоту помилок при розпізнаванні текстових символів. Завдяки застосуванню передових методів передобробки зображень, включаючи нормалізацію контрастності, фільтрацію шуму та адаптивну бінаризацію, вдалося досягти низького рівня помилок (CER < 10%), що значно перевершує традиційні алгоритми машинного навчання, такі як HOG+SVM або каскадні детектори Хаара, які часто демонструють нижчу точність у нестандартних сценаріях розпізнавання. Завдяки такій оптимізації система здатна стабільно працювати у реальному часі та забезпечувати високу якість розпізнавання навіть у складних умовах експлуатації, що робить її ефективним рішенням для використання у сферах дорожнього контролю, автоматизованих паркінгових комплексів та безпекових технологій.

5. Гнучкість та адаптивність системи є однією з її ключових переваг, оскільки вона здатна ефективно коригувати параметри обробки зображень залежно від змінних умов експлуатації, забезпечуючи стабільність роботи навіть у складних середовищах. Алгоритм автоматично регулює яскравість, контрастність та насиченість кадру відповідно до рівня освітлення, що дозволяє компенсувати перепади світла, тіні та засвічені ділянки, покращуючи якість розпізнавання номерних знаків. Крім того, система підтримує роботу з різними форматами камер, включаючи високочастотні відеопотоки та знімки низької роздільної здатності, що підвищує її універсальність та можливість інтеграції у широкий спектр застосувань. Завдяки впровадженню механізмів нормалізації контрастності та динамічної фільтрації шумів алгоритм здатний стабільно функціонувати навіть за несприятливих погодних умов, таких як дощ, туман або сильне сонячне світло, а також у випадках руху об'єктів у кадрі, що знижує ризик спотворення зображення та підвищує точність розпізнавання. Така адаптивність робить систему надзвичайно ефективним інструментом для використання у транспортній аналітиці, дорожньому контролі, безпекових рішеннях та логістичних процесах, де необхідно забезпечити високу якість обробки номерних знаків у реальному часі.

Загальні висновки

Запропонована система розпізнавання номерних знаків є сучасною та продуктивною, оскільки використовує найкращі алгоритми комп’ютерного бачення та машинного навчання.

Висока точність розпізнавання досягається завдяки глибокому аналізу передобробки зображень та використанню потужних моделей нейронних мереж.

Система оптимізована для роботи у реальному часі, що робить її придатною для впровадження у дорожній контроль, логістичні підприємства та безпекові рішення.

Гнучкість та адаптивність до змінних умов дозволяє інтегрувати алгоритм у широке коло інфраструктурних проєктів.

Тестування та впровадження підтверджують стійкість алгоритму до складних сценаріїв експлуатації, забезпечуючи стабільну роботу при відеопотоці.

Загалом робота демонструє високий рівень реалізації програмного забезпечення, використання актуальних наукових розробок та можливість ефективного застосування у практичних задачах.

# СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lopez, J.M.; Gonzalez, J.; Galindo, C.; Cabello, J. A versatile low-cost car plate recognition system. Signal Processing and Its Applications, 2007, pp. 1–4.
2. Feng Yang; Fan Yang. Character recognition using parallel BP neural network. ICALIP 2008. International Conference, 7–9 July 2008, pp. 1595–1599.
3. Tommaso, C. Generative Models for License Plate Recognition by using a Limited Number of Training Samples. Image Processing, 2006, pp. 2769–2772.
4. Система розпізнавання номерних знаків: сфери застосування і переваги використання. URL: https://sky.kr.ua/blog/technology/sistema-rozpiznavannya-nomernix-znakiv-sferi-zastosuvannya-i-perevagi-vikoristannya.
5. Ковальчук О. М. Алгоритми розпізнавання зображень у системах відеоспостереження: навч. посіб. Київ: Талком, 2020. 340 с.
6. Smith R. An Overview of the Tesseract OCR Engine. Journal of Document Analysis and Recognition, 2007. Vol. 9, Issue 2, pp. 629–633.
7. Гриценко Ю. О. Застосування комп’ютерного зору для автоматичного розпізнавання номерних знаків транспортних засобів. Вісник Технологічного університету Поділля, 2019. Т. 2, No 4, pp. 34–40.
8. Bai Y. License Plate Recognition System Based on Machine Learning. Journal of Advanced Transportation, 2019. Vol. 45, Issue 8, pp. 623–634.
9. OCR meaning: why is OCR software important. URL: https://www.adobe.com/acrobat/guides/what-is-ocr.html.
10. Sharma P. A Review on Automatic Number Plate Recognition Techniques. International Journal of Advanced Research in Computer Science, 2020. Vol. 11, Issue 3, pp. 12–17.
11. Тарасов О. В. Розробка систем автоматичного розпізнавання автомобільних номерів для міських служб. Збірник наукових праць Харківського національного університету радіоелектроніки, 2017. Т. 25, No 3, pp. 112–117.
12. How license plate recognition is enabling a smarter society. URL: https://www.milestonesys.com/resources/content/articles/LPR-smart-society/.
13. OpenALPR. Open Source Automatic License Plate Recognition. URL: https://www.openalpr.com.
14. What is optical character recognition (OCR). URL: https://www.ibm.com/think/topics/optical-character-recognition.
15. Tesseract: програмне забезпечення для оптичного розпізнавання тексту. URL: https://tesseract-ocr.github.io.
16. Горбунов П. А. Методи покращення точності розпізнавання номерних знаків. Інформаційні системи і технології, 2019. No 4, pp. 44–49.
17. Lin T. Machine Learning Approaches for Vehicle License Plate Recognition. IEEE Transactions on Intelligent Transportation Systems, 2020. Vol. 21, Issue 11, pp. 4401–4412.
18. Кривошеєв І. В. Використання Python у задачах автоматизації аналізу зображень. Збірник наукових праць, 2020. No 2, pp. 78–82.
19. Silva S. License Plate Detection and Recognition in Unconstrained Scenarios. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2018, pp. 3156–3164.
20. Python.org. . Python Imaging Library Documentation. URL: https://www.python.org/doc/.
21. Ковальчук О. М. Теоретичні основи розпізнавання образів. Київ: Наука, 2018. 220 с.
22. OpenCV. Open Source Computer Vision Library. URL: https://opencv.org.
23. What is ANPR (Automatic Number-Plate Recognition). URL: https://www.isarsoft.com/knowledge-hub/anpr.
24. Tzutalin. LabelImg: Image Annotation Tool. URL: https://github.com/tzutalin/labelImg.