МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

БУКЕВИЧ Ілля Ігорович

Програмний модуль розпізнавання транспортних засобів на відео за допомогою нейронних мереж / Software Module for Vehicle Recognition in Videos Using Neural Networks

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-41
І. І. Букевич

Науковий керівник:
к.т.н., доцент П. Є. Биковий

Кваліфікаційну роботу допущено до захисту
«___»_____ 2025 р.

В.о. завідувача кафедри
_____ Н. М. Васильків

Тернопіль – 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н. М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
БУКЕВИЧ Ілля Ігорович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Програмний модуль розпізнавання транспортних засобів на відео за допомогою нейронних мереж / Software Module for Vehicle Recognition in Videos Using Neural Networks
керівник роботи Биковий П. Є. к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- проаналізувати існуючі методи розпізнавання транспортних засобів, визначити їх переваги й обмеження в реальних умовах;
- сформувати концепцію побудови програмного модуля, яка дозволяє автоматизувати процес розпізнавання транспортних засобів шляхом аналізу відеоданих;
- розробити архітектуру програмного модуля розпізнавання транспортних засобів на відео, яка має поєднувати високу продуктивність, адаптивність до різноманітних умов і можливість роботи на стандартному обладнанні для практичного застосування;
- реалізувати нейромережеву модель виявлення транспортних засобів, на основі YOLO, з урахуванням забезпечення стабільної роботи за умов мінливого освітлення, погодних факторів та високої динаміки руху;
- провести експериментальні дослідження ефективності запропонованої моделі.
5. Перелік графічного матеріалу в роботі:
- структура програмного модуля.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент_____ Букевич І.І.
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ Биковий П.Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль розпізнавання транспортних засобів на відео за допомогою нейронних мереж» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 55 сторінок і містить 9 ілюстрацій, 4 таблиці, 2 додатки та 28 використаних джерел.

Метою кваліфікаційної роботи є створення програмного модуля розпізнавання транспортних засобів на відео за допомогою нейронних мереж, який забезпечить високу точність, швидкість і адаптивність до реальних умов експлуатації.

Об'єктом дослідження є процеси розробки, навчання, тестування та оптимізації модуля розпізнавання.

Предмет дослідження – програмний модуль розпізнавання транспортних засобів на відео за допомогою нейронних мереж.

Розроблено програмний модуль розпізнавання транспортних засобів на відео за допомогою нейронних мереж, який інтуїтивно зрозумілий, інформативний, мінімалістичний, адаптивний та має хорошу швидкість, високу точність і гнучкість, що дозволило подолати обмеження існуючих рішень, зокрема чутливість до зовнішніх умов і труднощі з розпізнаванням у щільних сценах.

Ключові слова: ПРОГРАМНИЙ МОДУЛЬ, РОЗПІЗНАВАННЯ ТРАНСПОРТНИХ ЗАСОБІВ, YOLOv8, ВІДЕОПОТІК.

ANNOTATION

The qualification work on the topic "Software module for vehicle recognition in video using neural networks" for obtaining the degree of "bachelor" in specialty 122 "Computer Science" of the educational program "Computer Science" is written in a volume of 55 pages and contains 9 illustrations, 4 tables, 2 appendices and 28 sources used.

The purpose of the qualification work is to create a software module for recognizing vehicles in videos using neural networks, which will provide high accuracy, speed, and adaptability to real operating conditions.

The object of research is the processes of development, training, testing, and optimization of the recognition module.

The subject of the study is a software module for recognizing vehicles in video using neural networks.

A software module for recognizing vehicles in video using neural networks has been developed, which is intuitive, informative, minimalistic, adaptive, and has good speed, high accuracy, and flexibility, which allowed overcoming the limitations of existing solutions, in particular, sensitivity to external conditions and difficulties with recognition in dense scenes.

Keywords: SOFTWARE MODULE, VEHICLE RECOGNITION, YOLOv8, VIDEO STREAM.

ЗМІСТ

ВСТУП

Сучасний етап розвитку технологій характеризується інтенсивним впровадженням комп'ютерного зору та штучного інтелекту в різні сфери життя. Одним із перспективних напрямів є розпізнавання транспортних засобів на відео, що має широке застосування в транспортній логістиці, системах безпеки дорожнього руху та інтелектуальних транспортних системах [1]. Автоматизація таких процесів сприяє підвищенню ефективності аналізу відеоданих, зменшенню помилок, пов'язаних із людським фактором, та забезпеченню оперативного реагування на події.

Актуальність теми зумовлена зростаючим попитом на автоматизовані системи моніторингу транспортних потоків, які здатні швидко й точно ідентифікувати об'єкти на відео. Розробка програмних рішень у цій сфері спрямована на подолання технічних обмежень і створення інструментів, що відповідають сучасним вимогам до швидкості, точності та гнучкості.

Метою кваліфікаційної роботи є створення програмного модуля розпізнавання транспортних засобів на відео за допомогою нейронних мереж, який забезпечить високу точність, швидкість і адаптивність до реальних умов експлуатації. Для досягнення поставленої мети, було поставлено такі завдання:

- проаналізувати існуючі методи розпізнавання транспортних засобів, визначити їх переваги й обмеження в реальних умовах;

- сформувати концепцію побудови програмного модуля, яка дозволяє автоматизувати процес розпізнавання транспортних засобів шляхом аналізу відеоданих;

- розробити архітектуру програмного модуля розпізнавання транспортних засобів на відео, яка має поєднувати високу продуктивність, адаптивність до різноманітних умов і можливість роботи на стандартному обладнанні для практичного застосування;

- реалізувати нейромережеву модель виявлення транспортних засобів, на основі YOLO, з урахуванням забезпечення стабільної роботи за умов мінливого освітлення, погодних факторів та високої динаміки руху;

- провести експериментальні дослідження ефективності запропонованої моделі.

Об’єктом дослідження є процеси розробки, навчання, тестування та оптимізації модуля розпізнавання транспортних засобів.

Предметом дослідження є програмний модуль розпізнавання транспортних засобів на відео за допомогою нейронних мереж.

Для вирішення поставлених завдань використано модель YOLO [2], яка відома своєю здатністю швидко й точно розпізнавати об’єкти. На основі цієї архітектури було проведено навчання власної моделі з використанням спеціально підготовленого набору даних, що дозволило адаптувати систему до специфічних умов і потреб розпізнавання транспортних засобів [4]. Такий підхід демонструє потенціал сучасних нейронних мереж для створення ефективних рішень у цій сфері.

Структура роботи включає вступ, три розділи, висновки, список використаних джерел та додатки. У першому розділі розглянуто стан предметної області, огляд існуючих рішень і теоретичні основи розв’язання задачі. Другий розділ присвячений проектуванню модуля, вибору та підготовці даних, розробці моделі машинного навчання та метрикам оцінки точності та оптимізації моделі. Третій розділ охоплює реалізацію програмного забезпечення, розробку інтерфейсу користувача, тестування та оптимізацію продуктивності. Додатки містять фрагменти коду та копії опублікованих результатів.

Результати дослідження опубліковано в матеріалах науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR – 2025), м. Тернопіль, 27–29 травня 2025 р. (додаток Б).

1 СТАН ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

У сучасному світі автоматизація процесів обробки відеоданих відіграє важливу роль у розвитку інтелектуальних систем, зокрема в транспортній сфері. Розпізнавання транспортних засобів на відео застосовується в таких галузях, як управління дорожнім рухом, логістика, безпека та автономні транспортні системи. Ця технологія дозволяє не лише ідентифікувати транспортні засоби, але й отримувати додаткову інформацію, наприклад, тип транспортного засобу чи його ідентифікаційні ознаки, що сприяє підвищенню ефективності моніторингу та аналізу транспортних потоків. У контексті розумних міст розпізнавання транспортних засобів слугує основою для адаптивного управління світлофорами, планування маршрутів та запобігання заторів, а в автономних транспортних засобах – для прийняття рішень у реальному часі.

Розпізнавання транспортних засобів на відео базується на технологіях комп’ютерного зору, які за останні роки значно еволюціонували завдяки розвитку глибинних нейронних мереж [1]. Традиційні методи, такі як використання алгоритмів виділення контурів чи шаблонної відповідності [5], мали суттєві обмеження, зокрема низьку точність у змінних умовах і високу обчислювальну складність. Поява згорткових нейронних мереж (CNN) відкрила нові можливості для обробки зображень і відео, забезпечивши високу точність і швидкість розпізнавання [6]. Серед сучасних підходів особливу увагу привертають архітектури, такі як YOLO [2], які оптимізовано для швидкої обробки відео із мінімальними втратами точності, а також інші моделі, як-от SSD (Single Shot MultiBox Detector) і Faster R-CNN, що адаптуються до специфічних задач детекції.

Актуальність предметної області зумовлена кількома факторами. По-перше, зростання обсягів транспортних засобів у містах вимагає автоматизованих систем для їхнього контролю та оптимізації. По-друге, розвиток автономних транспортних засобів спирається на здатність систем розпізнавати оточуючі об’єкти, зокрема інші автомобілі, для забезпечення безпеки руху. По-третє, у сфері безпеки та правоохоронної діяльності автоматичне розпізнавання транспортних засобів сприяє

швидкому виявленню порушень чи ідентифікації об’єктів інтересу. Проте, реалізація таких систем стикається з низкою викликів, серед яких: мінливі умови освітлення (день, ніч, туман, дощ), висока динаміка об’єктів, різноманітність типів транспортних засобів і наявність перешкод у кадрі.

Основним інструментом для вирішення цих завдань є нейронні мережі, які здатні навчатися на великих наборах даних і адаптуватися до різноманітних умов. Зокрема, модель YOLO [2] вирізняється своєю архітектурою, яка дозволяє одночасно прогнозувати розташування об’єктів (bounding boxes) і класифікувати їх за один прохід мережі. Навчання таких моделей на спеціалізованих наборах даних дає змогу підвищувати точність розпізнавання, враховуючи специфіку конкретного застосування, наприклад, ідентифікацію певних типів транспортних засобів чи їхніх номерних знаків. Крім того, інтеграція з апаратними платформами, такими як GPU, дозволяє реалізувати інференс у реальних умовах для практичного використання.

Предметна область розпізнавання транспортних засобів на відео поєднує в собі передові досягнення комп’ютерного зору та машинного навчання, спрямовані на створення ефективних і гнучких систем. Подальший розвиток у цій сфері потребує вдосконалення алгоритмів, оптимізації обчислювальних ресурсів і адаптації моделей до реальних умов експлуатації, що відкриває широкі перспективи для досліджень і практичного впровадження.

Важливим напрямком у розпізнаванні транспортних засобів є інтеграція з іншими технологіями, такими як системи IoT (Інтернет речей) та датчики LIDAR, що дозволяє підвищувати точність і надійність детекції в складних умовах, наприклад, у густонаселених районах чи при поганій видимості. Крім того, розвиток хмарних обчислень відкриває можливості для обробки великих обсягів відеоданих у реальному часі, що особливо актуально для моніторингу.

1.2 Огляд і аналіз існуючих рішень

Розпізнавання транспортних засобів на відео є активно досліджуваною областю, яка поєднує комп’ютерний зір і машинне навчання для створення систем,

здатних ідентифікувати об’єкти. За останні роки було розроблено низку рішень, що базуються як на традиційних методах обробки зображень, так і на сучасних нейронних мережах. Огляд існуючих підходів дозволяє оцінити їхню ефективність, виявити обмеження та визначити напрями для подальшого вдосконалення.

Традиційні методи розпізнавання транспортних засобів базуються на ручних алгоритмах обробки зображень і відео. До таких належать методи виділення контурів, а також шаблонне узгодження, яке порівнює шаблони транспортних засобів із кадрами відео.

Переваги традиційних методів включають простоту реалізації та низькі обчислювальні вимоги, що дозволяє використовувати їх на пристроях із обмеженими ресурсами. Однак, їхніми недоліками є низька точність у мінливих умовах і обмежена здатність розпізнавати різноманітні типи транспортних засобів. Через це традиційні підходи поступово витісняються методами глибокого навчання, особливо для задач із високими вимогами до точності та швидкості.

Серед сучасних рішень існують двоступеневі методи глибокого навчання, такі як R-CNN, Fast R-CNN та Faster R-CNN. Ці підходи спочатку генерують пропозиції регіонів інтересу (Region Proposal Network, RPN), а потім класифікують об’єкти в межах цих регіонів за допомогою згорткових нейронних мереж.

Перевагами Faster R-CNN є висока точність детекції, особливо для складних умов з великою кількістю об’єктів і здатність адаптуватися до різних класів транспортних засобів. Однак цей метод має значні недоліки, а саме: висока обчислювальна складність і низька швидкість, що робить його менш придатним для обробки відеопотоків без використання потужних GPU. Крім того, потреба в окремому етапі генерування пропозицій додає накладні витрати, що обмежує його використання в задачах із високою динамікою, таких як моніторинг дорожнього руху.

Одноступеневі методи, такі як SSD (Single Shot MultiBox Detector) та YOLO, пропонують альтернативу двоступеневим підходам, забезпечуючи одночасне передбачення координат рамок і класифікацію об’єктів за один прохід мережі. SSD використовує багатошарове виявлення об’єктів на різних масштабах, що дозволяє

ефективно виявляти як великі, так і малі об’єкти. Ця архітектура досягає швидкості до 20-30 FPS на сучасних GPU, що є значним прогресом порівняно з Faster R-CNN.

YOLO стало одним із найпопулярніших рішень завдяки своїй швидкості та гнучкості. YOLOv3, наприклад, використовує багатошарову детекцію для покращення точності, а YOLOv4 додає модулі, такі як CSPDarknet, для підвищення продуктивності. Новіша версія, YOLOv8, досягає високих швидкостей обробки у стандартних умов, що робить її ідеальним вибором для задач реального часу.

Переваги одноступеневих методів включають високу швидкість, здатність працювати з відеопотоками та простоту інтеграції з апаратним забезпеченням. Однак, вони можуть поступатися двоступеневим методам у точності для складних сцен із перекриваннями об’єктів, що вимагає додаткової оптимізації та донавчання на специфічних датасетах.

Одним із прикладів є робота [7], де оглянуто сучасні методи виявлення та класифікації транспортних засобів на основі глибинного навчання, охоплюючи архітектури, функції активації, втрат та алгоритми оптимізації. Систематизовано два основні підходи до обробки зображень: двоетапні алгоритми (R-CNN, Fast R-CNN, Faster R-CNN, R-FCN), що демонстрували високу точність, але потребували значних обчислювальних ресурсів, та одноетапні алгоритми (YOLO, SSD, RetinaNet), що працювали швидше двоетапних. Обмеженнями є вразливість до змін умов середовища, різні ракурси, недостатня точність на малих об’єктах та імбаланс класів, де більшу частину набору даних займали легкові авто, що є проблемою для розпізнавання інших класів.

У роботі [8] представлено систему для виявлення, підрахунку та класифікації транспортних засобів на основі відео. Система використовує Gaussian Mixture Model (GMM) для віднімання фону, що дозволяє ефективно виявляти рухомі об’єкти, та включає два методи класифікації: Contour Comparison (CC) і Bag of Features (BoF) з Support Vector Machine (SVM). Метод CC базується на порівнянні площі контурів із заздалегідь заданими значеннями, тоді як BoF із SVM використовує SIFT-функції для навчання класифікатора. Результати тестування на п’яти відео показали високу точність підрахунку та класифікації (помилки CC варіювалися від 1,3% до 13%, BoF із SVM — від 0,7% до 22,76%), причому метод

CC виявився більш стабільним порівняно з BoF і SVM. Обмеженнями системи є чутливість до затемнень транспортних засобів і необхідність ручного визначення регіону інтересу.

У дослідженні [9] запропоновано метод розпізнавання марки та моделі транспортних засобів у реальному часі на основі модифікованої архітектури SqueezeNet із залишковими з'єднаннями (Residual SqueezeNet). Система використовує фронтальні зображення транспортних засобів, оброблені через глибоку нейронну мережу, яка включає Fire-модулі з обхідними з'єднаннями для підвищення точності. Для підготовки великого набору даних (291 602 зображення, 766 моделей) застосовано метод кластеризації з використанням K-means та PCA, що значно прискорило маркування. Результати показали 96,3% точності на рівні rank-1 із часом розпізнавання 108,8 мс на транспортний засіб, а розмір моделі скомпресовано до менш ніж 5 МБ, що робить її придатною для реального часу. Обмеженням є орієнтація лише на фронтальні зображення та потреба у подальшому вдосконаленні для нелінійних сценаріїв.

У дослідженні [10] представлено систему розумного управління трафіком із використанням обробки зображень. Система використовує веб-камеру для захоплення зображень доріг, де аналіз трафіку виконується через методи виявлення об'єктів (blob detection) і віднімання фону. На основі оцінки густоти трафіку (кількості транспортних засобів) система адаптивно регулює час роботи світлофорів, враховуючи пріоритет для аварійних транспортних засобів (наприклад, швидких). Додатково реалізовано розпізнавання номерних знаків за допомогою OCR для виявлення порушень сигналів і відображення попереджень про перешкоди. Після декількох тестів, результат показав хорошу точність і швидкість. Обмеженнями є залежність від статичної камери та чутливість до змін умов освітлення.

У роботі [11] представлено просту систему підрахунку транспортних засобів на основі глибокого навчання з використанням попередньо навченої моделі YOLOv3. Система аналізує відео у роздільній здатності 1080p із чотирма сценаріями запису (фронтальна/задня сторона, 1x/2x зум) для підрахунку автомобілів, мотоциклів, автобусів і вантажівок без відстеження їхнього руху.

Підрахунок базується на відстані центроїдів транспортних засобів від умовної межі, із додатковою корекцією для уникнення подвійного підрахунку. Тестування на чотирьох відео показало точність до 97,72% для фронтального запису з 1x зумом при 30 fps, але виникали помилки через подвійне виявлення (наприклад, автомобілі як вантажівки). Обмеженнями є чутливість до умов зйомки та низька точність для вантажівок через їхню схожість із іншими класами.

Для даної роботи основними вимогами є висока швидкість і достатня точність для розпізнавання класів транспортних засобів на відео. Традиційні методи відкинуто через їхню низьку точність, двоступеневі методи, такі як Faster R-CNN – не відповідають вимогам за швидкістю, а одноступеневі методи, зокрема YOLOv8 – найкраще відповідають поставленим цілям завдяки своїй швидкості та можливості адаптації.

Недоліки існуючих рішень, такі як потреба в потужних GPU для максимальної продуктивності або втрата точності при квантуванні, були враховані при виборі YOLOv8 і подальшій оптимізації. Таким чином, YOLOv8 обрано як базову модель, що забезпечує баланс між швидкістю, точністю та можливостями інтеграції з апаратним забезпеченням.

1.3 Теоретичні основи розв’язання задачі розпізнавання транспортних засобів

Розпізнавання транспортних засобів у відеопотоці є складною задачею комп’ютерного зору, яка потребує інтеграції теоретичних концепцій і методів машинного навчання, що є основою для розробки ефективних алгоритмів, таких як YOLOv8.

Комп’ютерний зір є галуззю штучного інтелекту, що займається інтерпретацією візуальних даних, таких як зображення та відео [3]. Основою для розпізнавання транспортних засобів є обробка зображень, яка включає кілька етапів: попередню обробку, сегментацію, виявлення ознак та класифікацію. Попередня обробка включає нормалізацію яскравості та масштабування усіх зображень до єдиного розміру. Сегментація дозволяє виділити регіони інтересу, де можуть перебувати транспортні засоби. Виявлення ознак, таких як краї або

текстури, є важливим для визначення контурів об’єктів, хоча сучасні підходи дедалі більше покладаються на автоматичне навчання ознак через нейронні мережі.

Згорткові нейронні мережі є важливим інструментом для розпізнавання об’єктів на відео [6]. Їхня архітектура базується на згорткових шарах, які застосовують фільтри для виділення локальних ознак, таких як краї, кути чи текстури. Глибинні CNN, такі як VGG, ResNet чи EfficientNet, слугують основою для сучасних моделей детекції, включаючи YOLO.

У контексті розпізнавання транспортних засобів, CNN дозволяють автоматично навчатися на великих наборах даних та власних датасетах і адаптуватися до мінливих умов. Перевагою CNN є їхня здатність узагальнювати ознаки, що робить їх придатними для задач із високою варіативністю, характерною для транспортних потоків.

YOLO є одним із найпопулярніших підходів для виявлення об’єктів. На відміну від двоступеневих методів, таких як R-CNN, які спочатку пропонують регіони інтересу, а потім класифікують їх, YOLO використовує єдину нейронну мережу для одночасного передбачення координат рамок і ймовірностей класів. Це забезпечує високу швидкість обробки відеопотоків.

Основною проблемою, що вирішується в даній роботі є низька точність та швидкість розпізнавання транспортних засобів. Традиційні методи мають значні обмеження в точності та ефективності. Метою є розробка модуля на основі комп’ютерного зору, що забезпечить високу точність та швидкість розпізнавання транспортних засобів.

Математично задачу розпізнавання транспортних засобів можна сформулювати так:

1. Функція розпізнавання об’єктів:

$$f(I, O) = \{C, B\}, \tag{1.1}$$

де I – вхідне зображення (кадр відео);

O – параметри нейронної мережі;

C – клас транспортного засобу (наприклад, легковий автомобіль, вантажівка);

B – координати обмежувальної рамки (у форматі (x, y, w, h), де x, y – координати центра, а w, h – ширина та висота рамки).

Функція $f(I, O)$ реалізується за допомогою нейронної мережі YOLOv8, яка одночасно передбачає класи та координати об'єктів у кадрі.

2. Функція втрат для оптимізації моделі:

$$L = L_{cls} + L_{loc}, \tag{1.2}$$

де L_{cls} – втрати класифікації, що вимірює точність визначення класу транспортного засобу;

L_{loc} – втрати локалізації (наприклад, середньоквадратична помилка), що оцінює точність передбачення координат рамки.

Мета полягає в мінімізації L шляхом налаштування параметрів O під час навчання, що дозволяє моделі досягти високої точності як у класифікації, так і в локалізації транспортних засобів.

3. Обмеження продуктивності:

$$T_{proc} \leq \frac{1}{FPS_{min}}, \tag{1.3}$$

де T_{proc} – час обробки одного кадру;

FPS_{min} – мінімально необхідна частота кадрів за секунду.

Ця умова забезпечує, що система здатна обробляти відеопотік із достатньою швидкістю, щоб уникнути затримок, які можуть зробити її непридатною для практичного використання, наприклад, у задачах моніторингу дорожнього руху.

Ці формулювання відображають основні аспекти задачі:

- точність розпізнавання;
- локалізація об'єктів;
- швидкість обробки.

Дані формули інтегруються у модуль комп'ютерного зору, де модель, на основі нейронної мережі, використовується для реалізації функції $f(I, O)$, а процес

навчання спрямований на оптимізацію L із врахуванням обмежень продуктивності, забезпечуючи ефективне розпізнавання транспортних засобів у відеопотоці.

1.4 Постановка задачі

Розпізнавання транспортних засобів на відео є складним завданням, яке потребує високої точності та швидкості обробки для забезпечення ефективного функціонування в реальних умовах. Основною проблемою в цій сфері є недостатня ефективність існуючих систем у ситуаціях із мінливими зовнішніми факторами, такими як різне освітлення, погодні умови, висока динаміка об’єктів і наявність перешкод у кадрі. Традиційні методи та навіть деякі сучасні рішення на основі нейронних мереж не завжди здатні забезпечити стабільну роботу в таких сценаріях, що призводить до помилок у класифікації, пропусків об’єктів або низької продуктивності.

Дослідження, проведені раніше, показують, що системи, попри певні переваги, мають свої недоліки:

- проблеми подвійного виявлення об’єктів або помилкової класифікації, наприклад, коли автомобілі визначаються як вантажівки;
- обмеження в потребі отримання тільки фронтальних зображень об’єктів;
- висока чутливість до змін умов освітлення.

У зв’язку з цим виникає потреба у розробці нової системи, яка б подолала ці обмеження і забезпечила стабільну роботу в реальних умовах.

Актуальність теми зумовлена зростаючим попитом на автоматизовані системи моніторингу транспортних потоків, які здатні швидко й точно ідентифікувати об’єкти на відео. У сучасному світі, де обсяги транспортного руху невпинно зростають, ефективне розпізнавання транспортних засобів на відео є важливим для забезпечення безпеки, оптимізації дорожнього руху та розвитку інтелектуальних транспортних систем. Розробка програмних рішень у цій сфері спрямована на подолання технічних обмежень традиційних методів і створення інструментів, що відповідають сучасним вимогам до швидкості, точності та гнучкості.

Метою даного дослідження є створення програмного модуля розпізнавання транспортних засобів на відео за допомогою нейронних мереж, який забезпечить високу точність, швидкість і адаптивність до реальних умов експлуатації, таких як мінливе освітлення, погодні фактори та динаміка руху.

Завдання дослідження включають:

- проаналізувати існуючі методи розпізнавання транспортних засобів, визначити їх переваги й обмеження в реальних умовах;

- сформувати концепцію побудови програмного модуля, яка дозволяє автоматизувати процес розпізнавання транспортних засобів шляхом аналізу відеоданих;

- розробити архітектуру програмного модуля розпізнавання транспортних засобів на відео, яка має поєднувати високу продуктивність, адаптивність до різноманітних умов і можливість роботи на стандартному обладнанні для практичного застосування;

- реалізувати нейромережеву модель виявлення транспортних засобів, на основі YOLO, з урахуванням забезпечення стабільної роботи за умов мінливого освітлення, погодних факторів та високої динаміки руху;

- провести експериментальні дослідження ефективності запропонованої моделі.

Виконання цих завдань дозволить створити ефективний інструмент для автоматизованого моніторингу транспортних потоків, що відповідає сучасним вимогам.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ РОЗПІЗНАВАННЯ ТРАНСПОРТНИХ ЗАСОБІВ

2.1 Загальний підхід до розробки програмного модуля розпізнавання

Розробка програмного модуля для розпізнавання транспортних засобів на відео спрямована на вирішення проблем традиційних і частково автоматизованих систем, таких як низька точність у змінних умовах і недостатня швидкість обробки. Пропоноване рішення базується на використанні глибинних нейронних мереж, зокрема моделі YOLO [2], яка забезпечує швидке й точне виявлення об’єктів у відеопотоці завдяки своїй одноступеневій архітектурі. Архітектура модуля повинна бути розроблена так, щоб поєднувати високу продуктивність, адаптивність до різноманітних умов і можливість роботи на стандартному обладнанні для практичного застосування в задачах моніторингу дорожнього руху.

Архітектура програмного модуля побудована за модульним принципом для легкого розширення, модифікації та інтеграції з іншими системами. Такий підхід забезпечує гнучкість і можливість додавання нових функцій без значних змін у загальній структурі. Основні компоненти даного модуля включають:

- модуль захоплення відеопотоку – відповідає за отримання даних з файлу через OpenCV, яка забезпечує зчитування кадрів із високою ефективністю та підтримку різних форматів відео;
- модуль обробки кадрів – готує кадри (зміна розміру, нормалізація) для нейронної мережі;
- модуль розпізнавання об’єктів – застосовує YOLOv8 для виявлення транспортних засобів;
- модуль постобробки – використовує Non-Maximum Suppression для підвищення точності;
- модуль візуалізації – відображає результати з обмежувальними рамками і класами об’єктів.

Структура програмного модуля подана на рисунку 2.1. Компоненти об’єднано для обробки відеопотоку: захоплення кадрів, їх обробка, розпізнавання об’єктів, постобробка та візуалізація результатів.

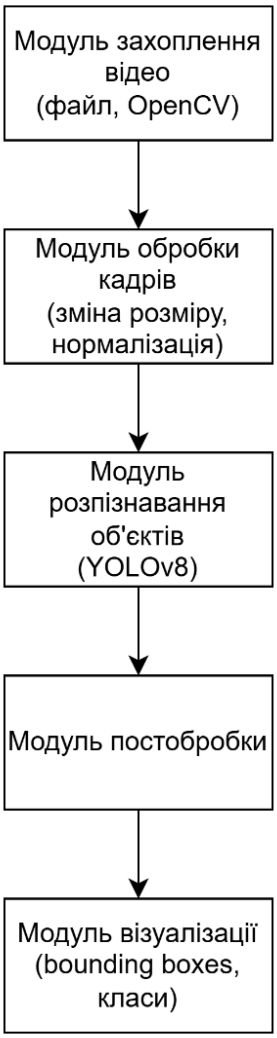

Рисунок 2.1 – Структура програмного модуля

Основна концепція рішення полягає в автоматизації процесу розпізнавання транспортних засобів шляхом аналізу відеоданих. Використання YOLO [2] як основи дозволить одночасно визначати координати транспортних засобів і їхні класи за один прохід мережі, що значно зменшує час обробки порівняно з двоступеневими моделями. Навчання моделі на спеціалізованому наборі даних забезпечує її здатність адаптуватися до специфічних умов, таких як різні типи транспортних засобів і зовнішні фактори.

Архітектура рішення передбачає модульність і гнучкість, що дозволить розширювати функціональність, наприклад, додаванням можливостей для аналізу додаткових ознак чи інтеграції з іншими системами. Розробка включатиме кілька етапів: вибір та підготовку даних для обробки, створення програмного забезпечення на базі Python [12], навчання нейронної мережі та оптимізацію

продуктивності. У результаті модуль забезпечуватиме не лише точне розпізнавання транспортних засобів, але й створить основу для подальшого масштабування й адаптації до потреб конкретних застосувань, таких як моніторинг дорожнього руху чи логістичні задачі.

2.2 Вибір та підготовка даних

Ефективність моделі машинного навчання значною мірою залежить від якості та різноманітності даних, на яких вона тренується. Для розпізнавання транспортних засобів з високою точністю, незважаючи на умови середовища, потрібно створити власний набір даних із зображеннями транспортних засобів у різних умовах. Цей набір також має охоплювати широкий спектр транспортних засобів різних типів із різними кутами огляду та відстанями.

Загальний обсяг набору даних складатиме тисячі кадрів, із яких відібрано вибірку для анотування. Процес анотування здійснюється за допомогою напівавтоматичного інструмента, такого як LabelImg [13], із зазначенням координат обмежувальних рамок (bounding boxes) і класів транспортних засобів, як зображено на рисунку 2.2.

Рисунок 2.2 – Процес анотації зображення

Додатково, для підвищення точності розпізнавання текстових ознак (наприклад, номерних знаків), частина зображень містила анотації символів, що можуть бути використані для подальшої роботи з EasyOCR [14]. Попередня обробка даних включає кілька етапів.

Першим етапом є нормалізація розміру зображення [15], що є важливим етапом попередньої обробки, який передбачає зміну розмірів зображень до єдиного стандарту перед подачею їх у нейронну мережу. Ця вимога зумовлена архітектурою мережі, де розмір вхідного шару є визначеним на етапі проектування. Приведення всіх зображень до однакового розміру забезпечує однорідність даних, що полегшує моделі вивчення закономірностей і зменшує обчислювальну складність. Коли подається зображення, розмір змінюється відповідно до цього фіксованого вхідного розміру перед обробкою. Це гарантує, що дані матимуть послідовну структуру, з якою мережа розрахована працювати. Усі кадри приведено до стандартного розміру 640x640 пікселів, щоб відповідати вхідним вимогам YOLO [2].

Другим етапом є корекція якості зображень за допомогою бібліотеки OpenCV, що широко використовується в комп'ютерному зорі [3]. Яскравість відноситься до загальної освітленості зображення, її регулювання підвищує або знижує значення всіх тонів у зображенні. Контраст відноситься до різниці в інтенсивності між світлими та темними ділянками зображення, його регулювання усуває середні тони, роблячи темні ділянки темнішими, а світлі – світлішими. Правильне налаштування яскравості та контрасту може значно покращити видимість об'єктів на зображеннях, особливо в складних умовах освітлення.

Останнім етапом є техніка штучного збільшення розміру та різноманітності навчального набору даних шляхом створення модифікованих копій існуючих даних. Це передбачає застосування різних перетворень до оригінальних даних для створення нових, правдоподібних варіантів. У комп'ютерному зорі ці перетворення можуть включати обертання, масштабування, зсуви, віддзеркалення, зміни освітлення та додавання шуму. Аугментація даних у машинному навчанні є важливою складовою, коли обсяг доступних навчальних даних є обмеженим [16]. Завдяки збільшенню різноманітності навчальних даних, аугментація допомагає моделям краще узагальнювати результати на невідомих даних, зменшуючи проблему перенавчання. Перенавчання відбувається, коли модель занадто добре вивчає навчальні дані та погано працює на нових даних. Аугментація даних допомагає моделі навчитися розпізнавати об'єкти за різних умов і з різних кутів. Це важливо для завдань, таких як виявлення об'єктів, де об'єкти можуть з'являтися в

різних розмірах, орієнтаціях та умовах освітлення. У таблиці 2.1 представлено техніки аугментації, що допомагають моделі краще працювати з невідомими даними.

Таблиця 2.1 – Техніки аугментації даних

Техніка	Опис
Обертання	Обертання зображення на певний кут для реальних сценаріїв, де об’єкти можуть з’являтися під різними кутами. Навчання моделі на обернених зображеннях робить її більш стійкою до змін.
Масштабування	Збільшення або зменшення розміру зображення для розпізнавання об’єктів різних розмірів та на різних відстанях.
Зміна освітлення	Регулювання яскравості, контрасту, відтінку та насиченості зображення для імітації денних та нічних умов освітлення.
Додавання штучного шуму	Додавання штучного шуму для підвищення стійкості моделі, що дає зосередитись на важливих ознаках.

Підготовлений набір даних розділено на три частини: 80% - для навчання (використовується для тренування моделі YOLOv8, дозволяючи їй адаптуватися до наявних класів транспортних засобів), 10% - для валідації (для відстеження процесу навчання, дозволяючи коригувати гіперпараметри) та 10% - для тестування (для оцінки точності). Це забезпечило можливість оцінки продуктивності моделі та уникнення перенавчання. Використання власного набору даних дозволить адаптувати модель до конкретних умов і підвищити її точність у реальних сценаріях.

2.3 Розробка моделі машинного навчання

Розробка моделі машинного навчання є головним етапом створення програмного модуля для розпізнавання транспортних засобів на відео. Цей етап визначає головні характеристики системи, такі як точність, швидкість і адаптивність до реальних умов експлуатації, що є важливим для ефективного моніторингу дорожнього руху. Модель має забезпечувати високу точність і швидкість виявлення об’єктів, а також адаптивність до різноманітних умов експлуатації, таких як зміна освітлення та погода. Для цього можна використати нейронну мережу YOLO [2], реалізовану через бібліотеку Ultralytics [4], яка дозволяє навчати модель на спеціалізованих наборах даних і оптимізувати її продуктивність. Процес розробки включає вибір і підготовку даних, а також створення й налаштування моделі для досягнення поставлених цілей, зокрема забезпечення реального часу (не менше 30 FPS) і розпізнавання класів транспортних засобів. Крім того, розробка моделі враховує потребу в її подальшому вдосконаленні, що може включати інтеграцію з додатковими технологіями, такими як системи відстеження чи хмарні обчислення.

Для реалізації обрано YOLOv8 – одну з версій моделі від Ultralytics [4], яка вирізняється покращеною точністю, швидкістю та гнучкістю порівняно з попередніми версіями (наприклад, YOLOv5 [17]). YOLOv8 підтримує кілька варіантів розміру (Nano, Small, Medium, Large, Extra Large), що дозволяє адаптувати її до різних апаратних можливостей і вимог продуктивності. Менші моделі (Nano, Small) мають меншу кількість параметрів та шарів, що призводить до швидшого часу виведення, але потенційно нижчої точності, особливо для складних сцен із перекриваннями об’єктів. Більші моделі (Large, Extra Large) мають більшу кількість параметрів та шарів, що дозволяє їм захоплювати складніші ознаки та досягати вищої точності, але з повільнішим часом виведення. У цьому випадку використано варіант Medium (YOLOv8m), який забезпечує оптимальне співвідношення між швидкістю обробки і точністю розпізнавання, що відповідає вимогам реального часу та специфіці датасету. Для кращого розуміння компромісів між різними

розмірами моделей YOLOv8, можна розглянути таблицю 2.2, яка узагальнює основні характеристики [18].

Таблиця 2.2 – Порівняння основних характеристик моделей YOLO

Модель	Точність (mAP@0.5)	Швидкість (FPS, GTX 1080)	Кількість параметрів	Примітка
YOLOv5s	~0.85	60-80	~7m	Швидка, але менш точна
YOLOv8n	~0.87	70-90	~3m	Легка, для слабкого GPU
YOLOv8m	~0.92	45-55	~25m	Оптимальна для модуля
YOLOv8l	~0.94	25-35	~44m	Висока точність, повільна
YOLOv8x	~0.95	15-25	~68m	Висока точність, надто повільна

Ця таблиця демонструє загальну тенденцію: зі збільшенням розміру моделі, зростає точність, але також збільшується кількість параметрів, що потенційно призводить до зниження швидкості виведення та збільшення вимог до обчислювальних ресурсів.

Процес навчання моделі включає:

- ініціалізація: тренування моделі на великому наборі даних, що дає базові знання про розпізнавання об’єктів;

- налаштування гіперпараметрів: кількість епох варіюється залежно від розміру набору даних (наприклад, 50 – 100 епох), із застосуванням ранньої зупинки для уникнення перенавчання;

- аугментація: під час навчання застосовуються техніки аугментації (обертання, зміна яскравості, шум), вбудовані в Ultralytics [4, 16], для підвищення здатності моделі розпізнавати об’єкти.

Оптимізація моделі спрямована на досягнення високої точності та продуктивності. Після навчання модель тестується на відкладеній тестовій вибірці, що дозволить оцінити її здатність розпізнавати транспортні засоби в різних умовах. Додатково застосовуватиметься техніка NMS (Non-Maximum Suppression) для

видалення дублюючих рамок, щоб підвищити якість результатів [19]. Використання квантування також розглядатиметься для подальшого прискорення інференсу без значної втрати точності.

2.4 Метрики оцінки точності та оптимізації моделі

Для оцінки продуктивності розробленого модулю розпізнавання та підрахунку транспортних засобів використовуватимуться кілька основних метрик, які дозволять кількісно та якісно оцінити її ефективність. Розглянемо метрики оцінки продуктивності та методи оптимізації, що будуть застосовані для вдосконалення роботи моделі [20].

Однією з основних метрик для оцінки здатності моделі правильно ідентифікувати типи транспортних засобів (наприклад, автомобіль, мотоцикл, автобус, вантажівка) є точність класифікації. Ця метрика визначається як відношення кількості правильно класифікованих об’єктів до загальної кількості об’єктів у наборі даних:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \quad (2.1)$$

де TP — істинно позитивні результати (правильно визначені об’єкти);

TN — істинно негативні результати (правильно відхилені об’єкти);

FP — хибно позитивні результати (помилково класифіковані об’єкти);

FN — хибно негативні результати (пропущені об’єкти).

Ця метрика є стандартним інструментом для оцінки загальної якості класифікації моделі.

Також для системи важливим параметром є швидкість обробки відеоданих. Час обробки одного кадру вимірюється в мілісекундах і має відповідати вимогам частоти кадрів (FPS). Наприклад, для забезпечення роботи при 30 fps:

$$Processing\ Time \leq \frac{1000}{FPS} ms, \quad (2.2)$$

При FPS = 30, час обробки не повинен перевищувати 33,3 мс. Ця метрика дозволяє оцінити, чи здатна система чудово працювати.

Для підвищення продуктивності моделі, особливо в умовах обмежених обчислювальних ресурсів, планується застосовувати наступні методи оптимізації:

- використання апаратного прискорення з CUDA [21]. Для прискорення обробки відеоданих модель буде оптимізована для роботи на графічних процесорах (GPU) із застосуванням технології CUDA. Це дозволить виконувати паралельні обчислення, що значно скорочує час аналізу кожного кадру;

- зменшення розміру моделі за допомогою квантування [22]. Щоб знизити обчислювальну складність, планується застосувати квантування ваг моделі. Цей метод зменшує точність представлення ваг, що дозволяє економити пам'ять і прискорювати обчислення.

Формула квантування:

$$\omega_{\mathrm{q}} = \mathrm{round}\left(\frac{\omega}{\Delta}\right) \times \Delta, \tag{2.3}$$

де ω – оригінальна вага;

ω_{q} – квантова вага;

Δ – крок квантування.

Цей підхід забезпечує баланс між швидкістю і точністю.

- методи постобробки для підвищення точності. У ситуаціях із щільним трафіком або коли об'єкти на відео перекривають один одного, для покращення результатів буде застосовано метод NMS [19]. Цей алгоритм допомагає усунути зайві рамки, які модель створює навколо об'єктів, використовуючи показник співвідношення перетину до об'єднання (Intersection over Union, IoU) для оцінки їхньої схожості:

$$IoU = \frac{\text{Площа перетину}}{\text{Площа об'єднання}}, \tag{2.4}$$

де площа перетину – це область, у якій два прямокутники, що позначають об’єкти, перекриваються;

площа об’єднання – це загальна область, яку займають обидва прямокутники разом, враховуючи їхнє перекриття лише один раз.

Якщо показник співвідношення перетину до об’єднання між двома прямокутниками перевищує заданий поріг, прямокутник із меншою впевненістю видаляється. Це допомагає зменшити кількість помилкових виявлень.

Описані метрики, такі як точність класифікації та час обробки, дозволять комплексно оцінити продуктивність системи. Методи оптимізації, включаючи апаратне прискорення, квантування та постобробку, забезпечать ефективну роботу моделі.

3 ПРОГРАМНО-ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ ПРОГРАМНОГО МОДУЛЯ РОЗПІЗНАВАННЯ ТРАНСПОРТНИХ ЗАСОБІВ

3.1 Реалізація програмного забезпечення

Розробка програмного модуля для розпізнавання транспортних засобів на відео автоматично аналізує відеопотоки та ідентифікує об’єкти. Для реалізації використано сучасні технології комп’ютерного зору та машинного навчання, зокрема модель YOLOv8, що забезпечує високу точність і швидкість роботи завдяки своїй одноступеневій архітектурі та оптимізованому інференсу. Процес розробки включає вибір інструментів, проектування архітектури, підготовку даних, навчання моделі та інтеграцію компонентів, що дозволяє створити гнучку та масштабовану систему для моніторингу дорожнього руху. Крім того, розробка враховує можливість подальшого розширення функціоналу, наприклад, інтеграції з системами відстеження чи хмарними платформами для обробки великих обсягів даних.

Для розробки обрано мову програмування Python через її підтримку бібліотек для машинного навчання та обробки зображень. Основні використані бібліотеки:

- OpenCV – для захоплення відеопотоків і базової обробки зображень;
- Ultralytics YOLOv8 – для розпізнавання об’єктів на кадрах відео;
- PySide6 – для створення графічного інтерфейсу користувача.

Додатковий функціонал реалізовано через бібліотеку EasyOCR [14], яка, в даному випадку, використовується для розпізнавання номерних знаків транспортних засобів. Результати ідентифікації структуруються за допомогою модуля json і зберігаються у файлах для можливого подальшого аналізу. Для обробки текстових даних із OCR застосовуються бібліотека re та defaultdict із collections, що допомагає групувати й аналізувати отриману інформацію.

Обробка відеопотоків виконується у фоновому потоці з використанням класу VideoProcessor, що забезпечує паралельну роботу та оновлення інтерфейсу. Фрагменти реалізації цього механізму наведено в додатку А.

Для підготовки даних використано набір даних із анотованих кадрів транспортних засобів у різних умовах. Приклад анотованих кадрів зображено на

рисунку 3.1. Додатково були застосовані техніки аугментації даних (повороти, масштабування, зміна яскравості), що сприяє підвищенню стійкості моделі до варіацій у вхідних даних.

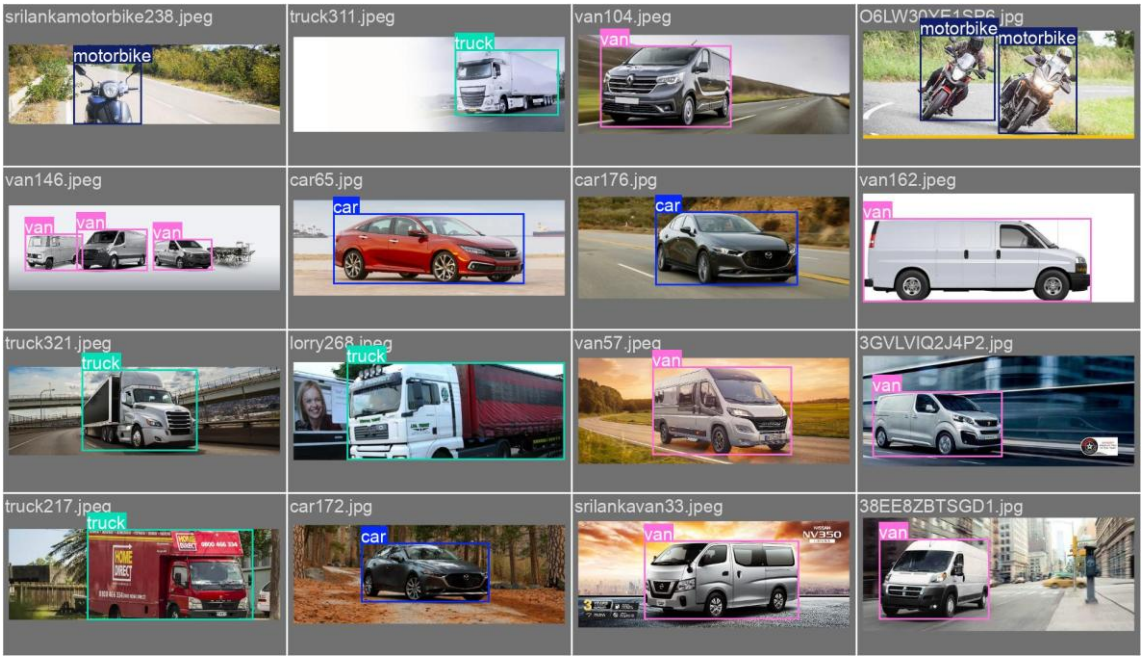

Рисунок 3.1 – Приклад анотованих кадрів з набору даних для навчання

Навчання нейронної мережі відбувалося протягом 100 епох з розміром батча 16. Було застосовано ранню зупинку – механізм, який запобігає перенавчанню, припиняючи навчання, якщо продуктивність моделі на валідаційній вибірці перестає покращуватися. На рисунку 3.2 представлено код для запуску процесу навчання нейронної мережі.

```python
from ultralytics import YOLO

# Завантажуємо модель
model = YOLO("yolov8m.pt")

# Навчання
results = model.train(
    data="vehicle.yaml",
    imgsz=640,
    epochs=100,
    batch=16,
    name="yolov8m_own"
)
```

Рисунок 3.2 – Код запуску навчання нейронної мережі

Після завершення навчання модель збережено у форматі .pt. На рисунку 3.3 зображено ряд графіків, що показують, як змінювалися різні метрики та функції втрат під час навчання моделі.

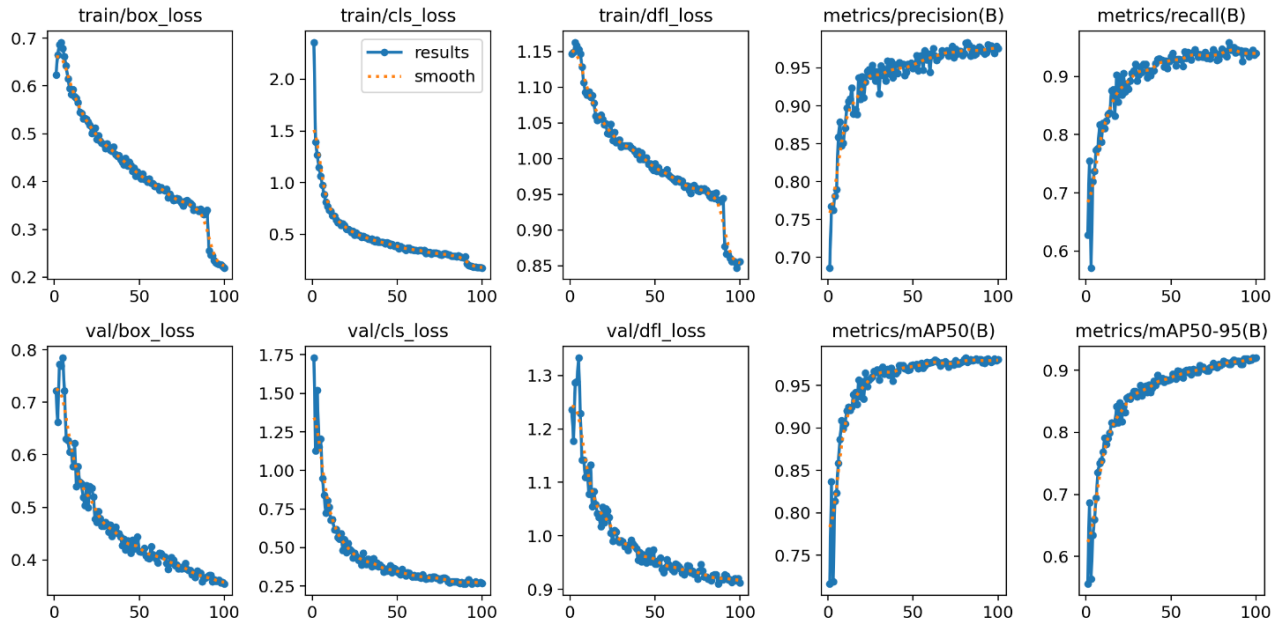

Рисунок 3.3 – Графіки процесу навчання моделі

Розберемо дані графіки більш детально для розуміння, чи був успішний процес навчання моделі:

- train/box_loss, train/cls_loss, train/dfl_loss (верхній ряд зображення). Це графіки значень різних компонентів функції втрат на навчальній вибірці. Box_loss – втрати, які пов’язані з точністю визначення обмежувальних рамок, тобто їх позиції та розміру. Cls_loss – втрати, які пов’язані з точністю класифікації об'єктів усередині рамок. Dfl_loss – втрати, які пов’язані з підвищенням точності виявлення об’єктів, що зосереджується на складних випадках для полегшення навчання й досягнення вищої точності з часом. Ці графіки показують зменшення значень втрат з кожною епохою, що свідчить про те, що модель успішно навчається та покращує свої передбачення на даних, які вона бачила. Жовта лінія «smooth» показує усереднене значення, що допомагає згладити коливання;

- val/box_loss, val/cls_loss, val/dfl_loss (нижній ряд зображення). Аналогічні графіки, але для валідаційної вибірки. Ці графіки потрібні для виявлення перенавчання. Якщо втрати на навчальній вибірці продовжують падати, а на

валідаційній починають зростати, це вказує на перенавчання. У моєму випадку, втрати на валідаційній вибірці також стабільно зменшуються, що є хорошим показником. Це означає, що модель добре працює на даних, які вона не бачила під час навчання;

- metrics/precision(B), metrics/recall(B), metrics/mAP50(B), metrics/mAP50-95(B). Графіки метрик продуктивності моделі на валідаційній вибірці. Precision – точність правильних позитивних передбачень серед усіх позитивних передбачень. Recall – повнота правильних передбачень серед усіх фактично позитивних об’єктів. mAP50 – метрика для виявлення об’єктів, що вимірює точність розпізнавання об'єктів з перетином рамок не менше 50%. mAP50-95 – більш сувора метрика, що враховує точність при різних рівнях перекриття рамок.

Ці графіки показують зростання значень метрик з кожною епохою, що свідчить про постійне покращення продуктивності моделі. Досягнення високих значень (близько 0.9-0.95 для mAP50) є хорошим результатом. Також, графіки підтверджують, що модель не тільки навчається, але й ефективно узагальнює, уникаючи перенавчання та демонструє покращення за метриками.

Для детальнішого аналізу продуктивності класифікації моделі для кожного класу маємо нормалізовану матрицю плутанини, що зображено на рисунку 3.4.

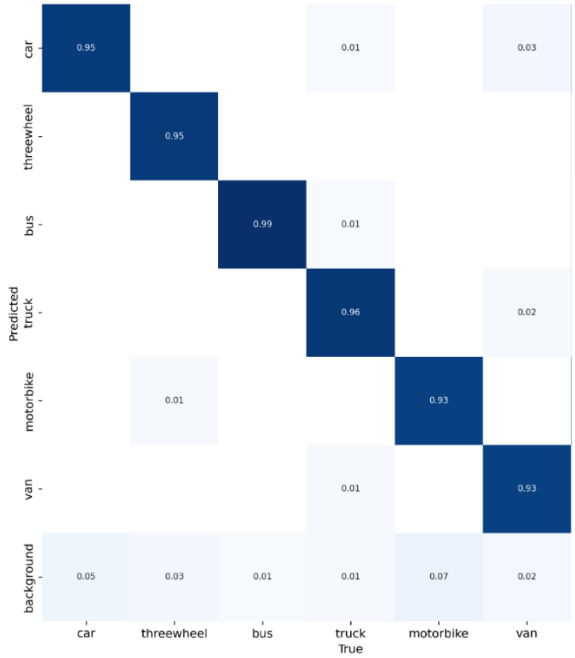

Рисунок 3.4 – Нормалізована матриця плутанини результатів розпізнавання транспортних засобів

Нормалізована матриця плутанини потрібна для глибокого аналізу продуктивності класифікації моделі виявлення об'єктів. Ця матриця візуалізує відносну кількість правильних та неправильних передбачень для кожного класу, нормовану за загальною кількістю фактичних об'єктів даного класу, де:

- вісь «True» представляє істинні класи об’єктів, які були присутні на зображеннях;
- вісь «Predicted» представляє класи, які були передбачені моделлю;
- цифри у клітинках – частка об’єктів істинного класу «i», які були передбачені моделлю, як клас «j». Сума значень у кожному рядку дорівнює 1.0 (100%), оскільки вони є частками від загальної кількості об’єктів цього істинного класу;
- інтенсивність кольору – відображає величину значення в клітинці, чим темніший синій, тим вища частка.

Аналіз нормалізованої матриці плутанини дозволяє детально оцінити точність та повноту розпізнавання для кожного класу. Значення на головній діагоналі (зверху зліва до низу справа) представляють частку об'єктів, які були правильно класифіковані у межах свого істинного класу. Ці значення також відповідають показнику повноти для кожного класу.

- «car»: 0.95 – 95% фактичних автомобілів були правильно розпізнані;
- «threewheel»: 0.95 – 95% триколісних транспортних засобів були правильно розпізнані;
- «bus»: 0.99 – 99% автобусів були правильно розпізнані. Це свідчить про майже ідеальну повноту розпізнавання автобусів;
- «truck»: 0.96 – 96% вантажівок були правильно розпізнані;
- «motorbike»: 0.93 – 93% мотоциклів були правильно розпізнані;
- «van»: 0.93 – 93% фургонів були правильно розпізнані.

Модель демонструє високу точність класифікації між розпізнаними об'єктами, що свідчить про її здатність добре розрізняти різні типи транспортних засобів.

Хоча нормалізована матриця плутанини дозволяє оцінити показники повноти для кожного класу та виявити основні джерела помилок, вона не відображає, як ці

метрики змінюються залежно від порогу впевненості моделі. Більш повне розуміння здатності моделі до виявлення та класифікації об'єктів надає аналіз Precision-Recall кривих, зображених на рисунку 3.5.

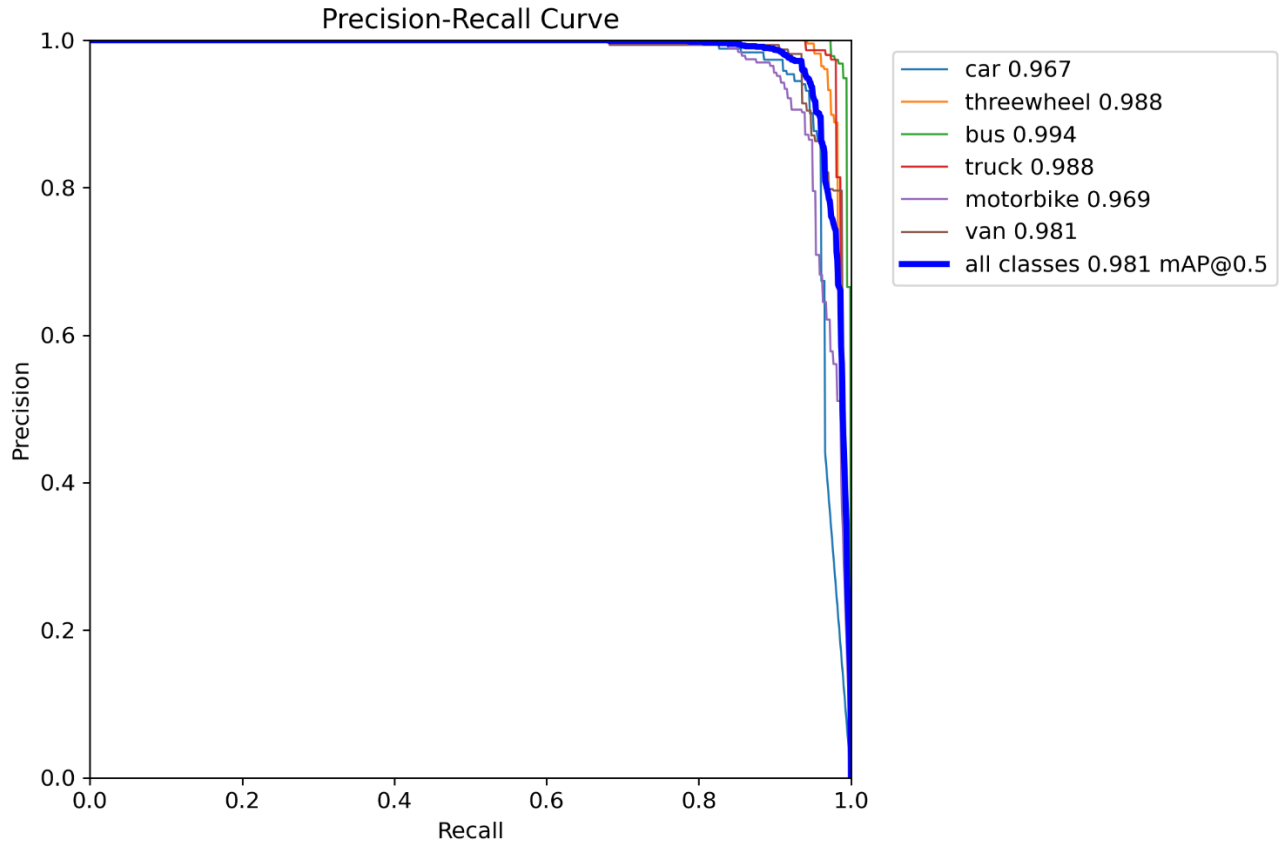

Рисунок 3.5 – Криві передбачень та повноти для кожного класу та усереднена mAP@0.5

Крива Precision-Recall це графік, що показує залежність між точністю та повнотою при різних порогах впевненості моделі. Кожна крива представляє окремий клас об'єктів. Товста синя лінія - це середня точність-повнота (mAP) для всіх класів, де:

- precision – частка правильно ідентифікованих об'єктів серед усіх, які модель вважала об'єктами $\frac{TP}{TP+FP}$;

- recall – частка правильно ідентифікованих об'єктів серед усіх фактично існуючих об'єктів $\frac{TP}{TP+FN}$;

- велика площина під кривою – вказує на високу продуктивність моделі. mAP

– усереднене значення площі під цими кривими;

- значення середньої точності класів car = 0.967, threewheel = 0.988, bus = 0.994, truck = 0.988, motorbike = 0.969 та van = 0.981 є високими, що свідчить про чудову продуктивність моделі.

3.2 Інтерфейс користувача

Інтерфейс користувача забезпечує зручну взаємодію з програмним модулем та візуалізацію результатів. UI розроблено так, щоб бути інтуїтивно зрозумілим, інформативним і мінімалістичним, дозволяючи користувачу легко керувати модулем та аналізувати дані. Для роботи з відеофайлами застосовується FFmpeg через модулі subprocess і ffmpeg-python, що дозволяє ефективно обробляти різні формати відео для його правильного відображення у панелі результату [23].

При розробці інтерфейсу враховано такі принципи:

- зручність – простота використання, що дозволяє освоїти інтерфейс без спеціальної підготовки;
- інформативність – чітке відображення всіх даних, таких як відеопотік, розпізнані об’єкти та статистика;
- мінімалізм – виключення зайвих елементів для зосередження уваги на основних функціях;
- адаптивність – інтерфейс підтримує зміну розміру вікна та коректно відображається.

Інтерфейс складається з одного основного вікна, яке включає:

- панель результату – показ результату завантаженого відеофайлу вже з обмежувальними рамками і мітками розпізнаних транспортних засобів;
- панель керування – кнопки для запуску аналізу, вибору джерела відео, збереження результату відео і налаштування параметрів (можливість обрати тільки певні типи транспортних засобів для розпізнавання, розпізнавання номерних знаків);
- панель статистики – відображення кількості розпізнаних транспортних засобів за типами.

На рисунках 3.6 та 3.7 зображено інтерфейс програмного модуля.

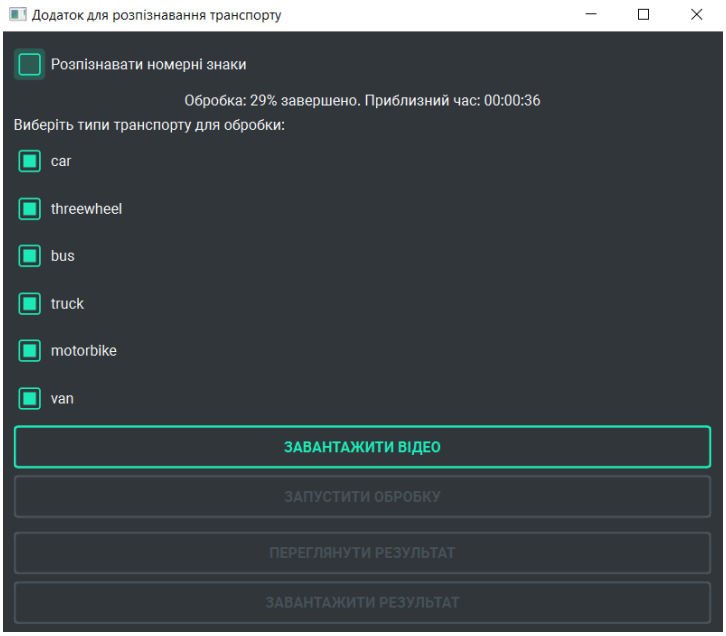

Рисунок 3.6 – Інтерфейс програмного модуля з розпочатим процесом обробки відео

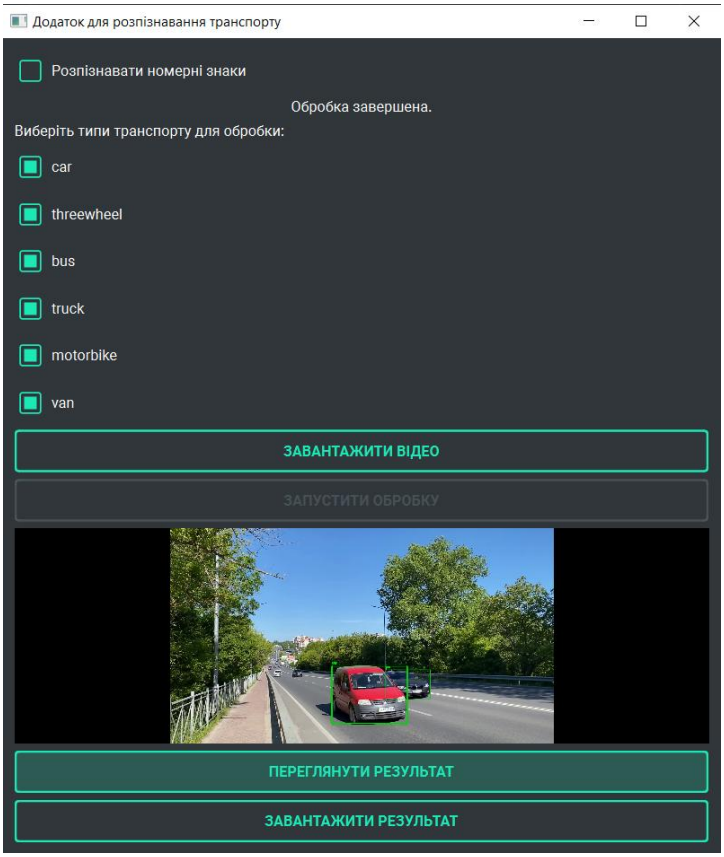

Рисунок 3.7 – Інтерфейс програмного модуля з готовим результатом обробки відео

Графічний інтерфейс користувача створено з використанням PySide6 [24], що

включає віджети для відтворення відео (QVideoWidget), вибір файлів (QFileDialog), кнопки керування й прогрес-бар для відслідковування приблизного часу завершення обробки (QProgressBar).

3.3 Тестування програмного модуля

Тестування програмного модуля для розпізнавання транспортних засобів дозволяє оцінити його продуктивність, точність і здатність працювати в різних умовах.

Для оцінки модуля на точність та ефективність, використано наступні метрики:

- точність класифікації (Accuracy) – відсоток правильно розпізнаних об’єктів від загальної кількості;
- середня точність (mAP) – середнє значення точності для всіх класів, що відображає загальну якість розпізнавання;
- частота кадрів за секунду (FPS) – показник швидкості обробки.

Тестування проводилося на апаратному забезпеченні з графічним процесором NVIDIA GeForce GTX 1080. Було обрано три сценарії:

- денне світло з низькою щільністю трафіку;
- ніч із середньою щільністю трафіку;
- дощ із високою щільністю трафіку.

Результати тестування наведено у таблиці 3.1.

Таблиця 3.1 – Результати тестування модуля

Сценарій	Точність класифікації	mAP	FPS
Денне світло, низький трафік	96%	0.98	58
Ніч, середній трафік	92%	0.93	55
Дощ, високий трафік	88%	0.89	51

Програмний модуль продемонстрував високу стійкість, зберігаючи точність у межах 89-96% у різних умовах. Найкращі показники досягнуто за денного світла,

тоді як уночі та під час дощу точність дещо знижувалася через погіршення видимості.

3.4 Оптимізація продуктивності програмного модуля

Оптимізація продуктивності модуля розпізнавання транспортних засобів спрямована на підвищення швидкості обробки відео, зменшення обчислювального навантаження та збереження або покращення точності розпізнавання.

Спочатку модуль демонстрував продуктивність у межах 45-55 FPS на апаратному забезпеченні з графічним процесором NVIDIA GeForce GTX 1080, що є достатнім, але потребує покращення для роботи на менш потужних пристроях. Окрім того, висока інтенсивність обчислень може призводити до перевантаження в умовах високої щільності трафіку або складних погодних умов. Оптимізація спрямована на:

- прискорення обробки кадрів без значної втрати точності;
- зменшення розміру моделі для її розгортання на пристроях із обмеженими ресурсами;
- підвищення енергоефективності.

Для досягнення кращої продуктивності, було застосовано такі методи оптимізації:

1. Використання апаратного прискорення (CUDA). Для прискорення обчислень використано CUDA 12.1, яка інтегрована з PyTorch та YOLOv8 [21]. Завдяки цьому, перенесено обчислення нейронної мережі на GPU, що дозволило паралельно обробляти кілька шарів моделі. В результаті, вийшло пришвидшення обробки кадрів приблизно на 30% у порівнянні з CPU.

2. Квантування моделі. Метою даного експерименту було підвищення швидкості обробки шляхом квантування моделі у формати .pth та .onnx [25, 26].

Формат .pth є стандартним форматом для збереження моделей у бібліотеці PyTorch, яка призначена для розробки та тренування нейронних мереж. Квантування моделі у форматі .pth виконується за допомогою інструменту torch.quantization, який дозволяє конвертувати ваги та операції з 32-бітного формату

з плаваючою комою (FP32) у 8-бітний цілочисельний формат (INT8), зменшуючи обсяг обчислень і розмір моделі, тим самим покращивши швидкість обробки відео.

Формат .onnx (Open Neural Network Exchange) є форматом для представлення моделей машинного навчання, розробленим для забезпечення сумісності між різними фреймворками та апаратними платформами. ONNX підтримує широкий спектр операцій і типів даних, включаючи квантовані моделі (INT8) для оптимізації продуктивності. Експорт моделі у формат .onnx із PyTorch зазвичай виконується за допомогою функції torch.onnx.export, після чого модель може бути додатково оптимізована за допомогою інструменту ONNX Runtime. Завдяки універсальності та підтримці апаратного прискорення .onnx використовується для розгортання моделей у реальних системах, де важливі швидкість і ефективність.

Для оптимізації моделі YOLOv8 було виконано квантування, яке передбачає зменшення точності представлення ваг і активацій із FP32 до INT8. Це дозволяє знизити обчислювальну складність і розмір моделі, що призводить до значного приросту швидкості інференсу. Квантована модель була збережена у форматі .pth для використання в екосистемі PyTorch, а також експортована у формат .onnx для подальшого інференсу за допомогою ONNX Runtime. Обидві моделі тестувалися на обробці одних і тих самих відео.

Експерименти з квантуванням у форматі .pth не виправдали очікувань. Незважаючи на використання INT8 для зменшення обчислювальної складності, середній час обробки кадру квантовою моделлю залишався ідентичним до часу обробки оригінальною моделлю у форматі FP32. Приріст продуктивності був відсутній, що може бути пояснено кількома факторами. По-перше, PyTorch може мати обмежену підтримку апаратного прискорення для INT8-операцій на певних GPU, якщо вони не оптимізовані для таких обчислень. По-друге, накладні витрати на квантування та деквантування під час інференсу могли нівелювати потенційні переваги. Тобто, використання квантування у форматі .pth не забезпечило очікуваного підвищення швидкості обробки.

На відміну від .pth, експерименти з квантуванням у форматі .onnx продемонстрували певний прогрес. Завдяки ONNX Runtime, який оптимізовано для роботи з GPU через CUDAExecutionProvider, квантована модель у форматі .onnx

показала приріст швидкості приблизно на 10% порівняно з оригінальною моделлю.

Тестування було ще раз проведене з використанням тих самих умов освітлення та трафіку. Результати порівняння до та після оптимізації зображено в таблиці 3.2.

Таблиця 3.2 – Порівняння продуктивності до та після оптимізації

Сценарій	mAP (до)	mAP (після)	FPS (до)	FPS (після)
Денне світло, низький трафік	0.98	0.98	58	65
Ніч, середній трафік	0.93	0.92	55	61
Дощ, високий трафік	0.89	0.88	51	58

Оптимізований модуль демонструє підвищення швидкості обробки до 65 FPS, зниження навантаження на процесор і відеокарту, що робить його придатним для використання на слабших та автономних пристроях.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи:

1. Проведено аналіз існуючих методів розпізнавання транспортних засобів, визначено їхні переваги та обмеження в реальних умовах, що дозволило обґрунтувати вибір подальшого підходу.

2. Сформовано концепцію побудови програмного модуля, яка дозволяє автоматизувати процес розпізнавання транспортних засобів шляхом аналізу відеоданих.

3. Розроблено архітектуру програмного модуля розпізнавання транспортних засобів на відео, яка поєднує високу продуктивність, адаптивність до різноманітних умов і можливість роботи на стандартному обладнанні для практичного застосування.

4. Реалізовано нейромережеву модель виявлення транспортних засобів на основі YOLO, забезпечено стабільну роботу за умов мінливого освітлення, погодних факторів та високої динаміки руху.

5. Проведено експериментальні дослідження ефективності запропонованої моделі, що підтвердило її придатність для практичного використання.

Порівняно з традиційними методами, такими як ручна обробка чи базові алгоритми комп’ютерного зору, запропонований модуль демонструє значні переваги: хорошу швидкість, високу точність і гнучкість, що дозволило подолати обмеження існуючих рішень, зокрема чутливість до зовнішніх умов і труднощі з розпізнаванням у щільних сценах.

Дослідження заклало основу для створення ефективного модуля розпізнавання транспортних засобів, а отримані результати можуть бути використані для подальшого вдосконалення технологій у цій сфері.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bishop C. M., Bishop H. Deep Learning: Foundations and Concepts. Cham: Springer, 2023. 656 p.

2. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. P.779-788.

3. Howse J., Minichino J. Learning OpenCV 4 Computer Vision with Python 3, Third Edition. Packt Publishing, 2020. 372 p.

4. Jocher G., Chaurasia A., Qiu J. YOLO by Ultralytics. GitHub Repository, 2023. URL: https://github.com/ultralytics/ultralytics (дата звернення: 10.01.2025).

5. Madrigal R. Blob-Detection in Image Processing: Comparing the Blob Detection Algorithms. 2022. URL: https://rafael-madrigal.medium.com/blob-detection-in-image-processing-8111e1967eb7 (дата звернення: 17.01.2025).

6. Ren J., Wang Y. Overview of Object Detection Algorithms Using Convolutional Neural Networks. Journal of Computer and Communications. 2022. Vol. 10. No.10. P.104-126.

7. Berwo M. A., Khan A., Fang Y., Fahim H., Javaid S., Mahmood J., Abideen Z. U., Syam M. S. Deep Learning Techniques for Vehicle Detection and Classification from Images/Videos: A surve. Sensors. 2023. Vol. 23. No.10. P.4832.

8. Memon S., Bhatti S., Thebo L. A., Talpur M. M. B., Memon M. A. A Video-Based Vehicle Detection, Counting and Classification System. International Journal of Image, Graphics and Signal Processing (IJIGSP). 2018. Vol. 10. No. 9. P.34-41.

9. Lee H. J., Ullah I., Wan W., Gao Y., Fang Z. Real-Time Vehicle Make and Model Recognition with the Residual SqueezeNet Architecture. Sensors. 2019. Vol. 19. No. 5. P.982.

10. Jadhav P., Kelkar P., Patil K., Thorat S. Smart Traffic Control System Using Image Processing. International Research Journal of Engineering and Technology (IRJET). 2016. P.1207-1211.

11. Fachrie M. A Simple Vehicle Counting System Using Deep Learning with YOLOv3 Model. RESTI Journal (System Engineering and Information Technology). 2020. Vol. 4. No. 3. P.462-468.

12. What Is Deep Learning with Python? Flatiron School. URL: https://flatironschool.com/blog/what-is-deep-learning-with-python/ (дата звернення: 28.01.2025).

13. Karki S. Installing and Using LabelImg to Annotate Images. Medium. 2024. URL: https://medium.com/@samn.krki/installing-and-using-labelimg-to-annotate-images-92f754e910a5 (дата звернення: 05.02.2025).

14. Jaided AI. EasyOCR: Ready-to-Use OCR with 40+ Languages Supported. GitHub Repository. 2023. URL: https://github.com/JaidedAI/EasyOCR (дата звернення: 11.02.2025).

15. Normalization. Ultralytics. URL: https://www.ultralytics.com/glossary/normalization (дата звернення: 13.02.2025).

16. YOLOv8 Data Augmentation. YOLOv8. URL: https://yolov8.org/yolov8-data-augmentation/ (дата звернення: 19.02.2025).

17. Khanam R., Hussain M. What is YOLOv5: A Deep Look into the Internal Features of the Popular Object Detector. 2024. URL: https://arxiv.org/html/2407.20892v1.

18. YOLOv8 Introduction. RevComm. 2023. URL: https://tech.revcomm.co.jp/yolov8-introduction-en (дата звернення: 26.02.2025).

19. Singh A. Selecting the Right Bounding Box Using Non-Max Suppression (with Implementation). Analytics Vidhya. 2024. URL: https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/ (дата звернення: 08.03.2025).

20. YOLO: Real-Time Object Detection Explained. URL: https://www.youtube.com/watch?v=9w4HJ0VUy2g&ab_channel=TechEngineerSchool (дата звернення: 12.03.2025).

21. NVIDIA Corporation. CUDA Toolkit Documentation. URL: https://docs.nvidia.com/cuda/ (дата звернення: 20.03.2025).

22. Deploying quantized Ultralytics YOLOv8 models on edge devices with DeGirum. URL: https://www.ultralytics.com/blog/deploying-quantized-yolov8-models-on-edge-devices-with-degirum (дата звернення: 27.03.2025).

23. FFmpeg Developers. FFmpeg: A Complete, Cross-Platform Solution to Record, Convert and Stream Audio and Video. URL: https://ffmpeg.org/ (дата звернення: 05.04.2025).

24. Qt for Python 6 Documentation. The Qt Company. URL: https://doc.qt.io/qtforpython-6/index.html (дата звернення: 09.04.2025).

25. Quantization. PyTorch. URL: https://docs.pytorch.org/docs/stable/quantization.html (дата звернення: 15.04.2025).

26. Poorna H. D. PyTorch to Quantized ONNX Model. Medium. 2023. URL: https://medium.com/@hdpoorna/pytorch-to-quantized-onnx-model-18cf2384ec27 (дата звернення: 23.04.2025).

27. Букевич І. І., Биковий П. Є. Попередня обробка даних для розпізнавання транспортних засобів. *Інтелектуальні інформаційні технології в прикладних дослідженнях (IITAR-2025)*: збірник тез доповідей студентської науково-практичної конференції м. Тернопіль, 27–29 травня 2025 р. Тернопіль: ЗУНУ, 2025. С. 28-31.

28. Комар М. П., Саченко А. О., Васильків Н. М., Гладій Г. М., Коваль В. С., Ліп'яніна-Гончаренко Х. В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Фрагменти коду розпізнавання транспортних засобів

```
import cv2
import time
from PySide6.QtCore import QThread, Signal
from ultralytics import YOLO
import json
import easyocr
import re
from collections import defaultdict, Counter


# === Фоновий потік для обробки відео ===
class VideoProcessor(QThread):
    progress = Signal(int)  # Сигнал для оновлення прогрес-бару
    progress_message = Signal(str)  # Сигнал для оновлення статусу
    finished = Signal(str)  # Сигнал, коли обробка завершена

    def __init__(self, input_path, output_path, model_function):
        super().__init__()
        self.input_path = input_path
        self.output_path = output_path
        self.model_function = model_function

    def run(self):
        """Обробка відео в фоновому потоці з оновленням прогресу."""
        self.progress_message.emit("Обробка відео розпочата...")
        cap = cv2.VideoCapture(self.input_path)
        if not cap.isOpened():
            self.progress_message.emit("Помилка: не вдалося відкрити відео.")
            return

        fourcc = cv2.VideoWriter_fourcc(*"mp4v")
        fps = int(cap.get(cv2.CAP_PROP_FPS))
        frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
        out = cv2.VideoWriter(
            self.output_path, fourcc, fps, (frame_width, frame_height)
        )

        frame_count = 0
        all_recognized_plates = []
        processing_times = []
        counted_ids = set()
        object_tracks = defaultdict(list)
        counts = defaultdict(int)
        temp_id_counter = 0
        previous_positions = defaultdict(int)
        LINE_Y = int(frame_height * 0.85)  # Лінія підрахунку (75% від верху)
        COUNTING_RANGE = int(frame_height * 0.05)  # Діапазон підрахунку

        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break
```

```
            frame_start_time = time.time()
            processed_frame, recognized_plates = self.model_function(
                frame,
                object_tracks,
                counts,
                counted_ids,
                previous_positions,
                LINE_Y,
                COUNTING_RANGE,
                temp_id_counter,
            )
            out.write(processed_frame)
            all_recognized_plates.extend(recognized_plates)
            frame_end_time = time.time()

            frame_processing_time = frame_end_time - frame_start_time
            processing_times.append(frame_processing_time)
            avg_time_per_frame = sum(processing_times) / len(processing_times)
            frames_left = total_frames - frame_count
            estimated_time_left = avg_time_per_frame * frames_left

            frame_count += 1
            progress = int((frame_count / total_frames) * 100)
            time_left_str = time.strftime("%H:%M:%S", time.gmtime(estimated_time_left))
            self.progress.emit(progress)
            self.progress_message.emit(
                f"Обробка: {progress}% завершено. Час: {time_left_str}"
            )

        cap.release()
        out.release()

        # Збереження результатів
        filtered_data = process_recognized_plates(all_recognized_plates)
        save_to_json(filtered_data, "filtered_plates.json")
        with open("traffic_stats.json", "w") as f:
            json.dump({model.names[k]: v for k, v in counts.items()}, f, indent=2)
        self.progress_message.emit("Обробка завершена.")
        self.finished.emit(self.output_path)


# Завантаження моделі
model = YOLO("100epoch.pt")
text_recognizer = easyocr.Reader(["en"], gpu=True)


def process_frame_with_yolo(
    frame,
    object_tracks,
    counts,
    counted_ids,
    previous_positions,
```

```
    LINE_Y,
    COUNTING_RANGE,
    temp_id_counter,
):
    """Обробка кадру: детекція, трекінг, розпізнавання номерів і візуалізація."""
    results = model.track(frame, persist=True, conf=0.5, verbose=False)
    selected_classes = [0, 1, 2, 3, 4, 5]  # Усі класи за замовчуванням
    if not selected_classes:
        return frame, []

    processed_frame = frame.copy()
    recognized_plates = []

    if results[0].boxes is None or results[0].boxes.data is None:
        return processed_frame, recognized_plates

    boxes = results[0].boxes
    track_ids = boxes.id.int().cpu().tolist() if boxes.id is not None else None
    current_track_ids = set()

    for i, det in enumerate(boxes.data):
        det = det.cpu().tolist()
        try:
            if len(det) == 7:  # x1, y1, x2, y2, track_id, conf, cls
                x1, y1, x2, y2, track_id, conf, cls = det
            elif len(det) == 6:  # x1, y1, x2, y2, conf, cls
                x1, y1, x2, y2, conf, cls = det
                track_id = (
                    track_ids[i]
                    if track_ids and i < len(track_ids)
                    else temp_id_counter
                )
                temp_id_counter += 1
            else:
                continue
        except Exception:
            continue

        track_id = int(track_id)
        cls = int(cls)
        if cls not in selected_classes:
            continue
        x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
        current_track_ids.add(track_id)

        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2
        object_tracks[track_id].append(cls)

        # Підрахунок об'єктів, що перетинають лінію вниз
        prev_cy = previous_positions.get(track_id, cy)
        if (
            track_id not in counted_ids
            and LINE_Y - COUNTING_RANGE < cy < LINE_Y + COUNTING_RANGE
```

```
            and cy > prev_cy
        ):
            if len(object_tracks[track_id]) >= 3:
                final_cls = Counter(object_tracks[track_id]).most_common(1)[0][0]
                counts[final_cls] += 1
                counted_ids.add(track_id)

        previous_positions[track_id] = cy
        cv2.rectangle(processed_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        label = model.names[cls]
        plate_region = frame[y1:y2, x1:x2]

        # Розпізнавання номерних знаків
        try:
            plate_text = text_recognizer.readtext(plate_region, detail=0)
            if plate_text:
                plate_text = "".join(plate_text).replace(" ", "").upper()
                if re.match(r"^[A-Z]{2}[0-9]{4}[A-Z]{2}$", plate_text):
                    recognized_plates.append({"plate": plate_text, "type": label})
        except Exception:
            pass

        # Візуалізація класу
        try:
            label = label.encode("utf-8").decode("latin1")
        except Exception:
            label = "Unknown"
        cv2.putText(
            processed_frame,
            label,
            (x1, y1 - 10),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.5,
            (0, 255, 0),
            2,
        )

    # Очищення старих треків
    for track_id in list(object_tracks.keys()):
        if track_id not in current_track_ids:
            del object_tracks[track_id]
            del previous_positions[track_id]

    # Малювання лінії підрахунку та статистики
    cv2.line(processed_frame, (0, LINE_Y), (frame_width, LINE_Y), (0, 0, 255), 2)
    y_offset = 30
    for cls_id, count in counts.items():
        name = model.names[cls_id]
        cv2.putText(
            processed_frame,
            f"{name}: {count}",
            (10, y_offset),
            cv2.FONT_HERSHEY_SIMPLEX,
```

```
            0.6,
            (255, 255, 0),
            2,
        )
        y_offset += 30

    return processed_frame, recognized_plates


# === Функції для обробки та збереження даних ===
def process_recognized_plates(recognized_data):
    """Фільтрує унікальні номерні знаки з одним типом транспорту."""
    unique_plates = {}
    for entry in recognized_data:
        plate = entry["plate"]
        transport_type = entry["type"]
        if plate not in unique_plates or transport_type not in unique_plates[plate]:
            unique_plates.setdefault(plate, set()).add(transport_type)
    return [
        {"plate": plate, "type": transport_type}
        for plate, types in unique_plates.items()
        for transport_type in types
    ]


def save_to_json(data, file_name):
    """Зберігає дані у JSON-файл."""
    with open(file_name, "w") as f:
        json.dump(data, f, indent=4)
    print(f"Результати збережено у файл: {file_name}")
```

Додаток Б

Копії опублікованих результатів

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(IITAR-2025)

27-29 травня 2025 року

Тернопіль

2025

Зміст

Букевич Ілля
студент групи КН-41
devillived2004@gmail.com
Биковий Павло
к.т.н., доцент
pb@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

ПОПЕРЕДНЯ ОБРОБКА ДАНИХ ДЛЯ РОЗПІЗНАВАННЯ ТРАНСПОРТНИХ ЗАСОБІВ

Розпізнавання транспортних засобів на відео знаходить застосування в таких галузях, як управління дорожнім рухом, логістика, безпека та автономні транспортні системи. Ця технологія дозволяє не лише ідентифікувати транспортні засоби, але й отримувати додаткову інформацію, наприклад, тип транспортного засобу чи його ідентифікаційні ознаки, що сприяє підвищенню ефективності моніторингу та аналізу транспортних потоків.

Традиційні методи розпізнавання транспортних засобів, такі як використання алгоритмів виділення контурів чи шаблонної відповідності [1], мали суттєві обмеження, зокрема низьку точність у змінних умовах і високу обчислювальну складність. Поява згорткових нейронних мереж [2] відкрила нові можливості для обробки зображень і відео, забезпечивши високу точність і швидкість розпізнавання. Серед сучасних підходів особливу увагу привертають архітектури, такі як YOLO [3], які оптимізовано для швидкої обробки відеопотоків із мінімальними втратами точності.

Ефективність моделі машинного навчання значною мірою залежить від якості та різноманітності даних, на яких вона тренується. Для розпізнавання транспортних засобів було створено власний набір даних, що включає зображення транспортних засобів у різних умовах. На рисунку 1 зображено зразки транспортних засобів у цьому наборі даних.

Рисунок 1 – Зразки транспортних засобів у наборі даних

Процес анотування даних виконувався за допомогою LabelImg [4], із зазначенням координат обмежувальних рамок і класів транспортних засобів, як зображено на рисунку 2. Під час навчання модель читає ці файли з мітками разом із відповідним зображенням і вчиться визначати де знаходяться об'єкти, якого вони класу та як їх визначити за обмежувальними рамками.

Рисунок 2 – Процес анотації зображення

Якість маркованого набору даних значною мірою визначає якість моделі. Щоб навчити модель виявляти транспортні засоби в реальних сценаріях дорожнього руху, потрібно позначати усі транспортні засоби, які з'являються на зображенні, навіть якщо це тільки частина кузова.

На основі статистичної інформації про маркування було проведено аналіз графіків, зображених на рисунку 3.

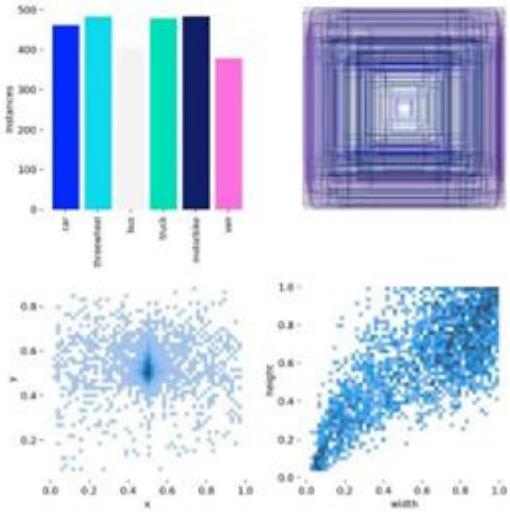

Рисунок 3 – Статистика розмітки датасету

На рисунку 3 наведено статистичну візуалізацію розміченого датасету для розпізнавання транспортних засобів у форматі YOLO. У верхньому лівому куті показано кількість об'єктів за класами, де видно відносно збалансований розподіл. У верхньому правому куті зображено всі накладені обмежувальні рамки, що демонструє концентрацію об'єктів у центральній частині зображення. Нижній лівий графік – це 2D гістограма центрів об'єктів, яка також вказує на те, що транспортні засоби найчастіше трапляються ближче до центру кадру, особливо по вертикалі. Нижній правий графік відображає розподіл ширини і висоти обмежувальних рамок, з чого видно, що більшість об'єктів є відносно невеликими, що типово для зображень із широким полем зору або зйомки на відстані.

Для підготовки якісного і придатного для навчання машинного алгоритму датасету, необхідна попередня обробка даних, яка включала кілька етапів:

1. Нормалізація розміру зображень. Усі кадри приведено до стандартного розміру 640×640 пікселів, щоб відповідати вхідним вимогам моделі YOLO [3].
2. Корекція якості. Використовувалася бібліотека OpenCV для регулювання яскравості та контрасту, що покращило видимість об’єктів у складних умовах освітлення [5].
3. Аугментація даних. Застосовувалися техніки для штучного збільшення різноманітності даних, що допомагало моделі краще узагальнювати результати [6].

Підготовлений набір даних було розділено на три частини:

- 80% для навчання;
- 10% для валідації;
- 10% для тестування.

Такий розподіл дозволив оцінити продуктивність моделі та уникнути перенавчання, забезпечивши її здатність працювати в реальних сценаріях.

Тепер, коли набір даних ретельно підготовлений і сегментований, він готовий до використання для навчання нейронної мережі розпізнавання транспортних засобів на основі YOLO. Різноманітні зображення, процес анотації та методи аугментації сприятимуть ефективності моделі в реальних програмах.

Список використаних джерел

1. Madrigal R. Blob-Detection in Image Processing: Comparing the Blob Detection Algorithms. 2022. URL: https://rafael-madrigal.medium.com/blob-detection-in-image-processing-8111e1967eb7.
2. Ren J., Wang Y. Overview of Object Detection Algorithms Using Convolutional Neural Networks. Journal of Computer and Communications. 2022. Vol. 10. No.10. P.104-126. URL: https://www.scirp.org/journal/paperinformation?paperid=115011.

3. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. P.779-788. URL: 10.1109/CVPR.2016.91.
4. Karki S. Installing and Using LabelImg to Annotate Images. Medium, 2024. URL: https://medium.com/@samn.krki/installing-and-using-labelimg-to-annotate-images-92f754e910a5.
5. Howse J., Minichino J. Learning OpenCV 4 Computer Vision with Python 3, Third Edition. Packt Publishing, 2020.
6. YOLOv8 Data Augmentation. YOLOv8. URL: https://yolov8.org/yolov8-data-augmentation/.