Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ІНТЕГРОВАНА КОМП'ЮТЕРНА СИСТЕМА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА МІСЦЕВОСТІ

Виконав: студент 2 курсу, групи 2КІ-24м
спеціальності 123 — Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Бондаренко К.О.
(прізвище та ініціали)
Керівник: к.т.н., доц. доцент каф. ОТ
Крупельницький Л.В.
(прізвище та ініціали)
«12» 12 2025р.

Опонент: к.т.н., доц. каф. ПЗ
Войтко В.В
(прізвище та ініціали)
«12» 12 2025 р.

Допущено до захисту

Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.

«15» 12 2025 р.

ВНТУ 2025

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
_____
«25» вересня 2025 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИПЛОМНУ РОБОТУ

студентці Бондаренко Катерині Олександрівні

1 Тема роботи: «Інтегрована комп'ютерна система розпізнавання об'єктів на місцевості», керівник роботи Крупельницький Леонід Віталійович к.т.н., доцент кафедри ОТ, затверджені наказом вищого навчального закладу від «24» вересня 2025 року №313.

2 Строк подання студентом роботи 4.12.2025

3 Вихідні дані до роботи: цифрові зображення у форматах JPG, PNG та відеофайли у форматі MP4, анотовані дані з координатами об'єктів .

4 Зміст розрахунково-пояснювальної записки: вступ, огляд методів і систем розпізнавання об'єктів, проєктування інтегрованої системи розпізнавання об'єктів, реалізація та тестування системи розпізнавання об'єктів, тестування та оцінювання результатів роботи системи, висновки.

5 Перелік графічного матеріалу: класифікація алгоритмів виявлення об'єктів за кількістю проходів зображення через мережу, алгоритми видалення атмосферних спотворень з зображень, текстовий вивід процесу інференсу YOLOv8 для відеоданих.

6 Консультанти розділів роботи

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Крупельницький Леонід Віталійович к.т.н., доцент каф. ОТ		
5	Адлер Оксана Олександрівна к.т.н., доцент кафедри ЕПВМ		
Нормоконтроль	Швець Сергій Ілліч асистент кафедри ОТ		

7 Дата видачі завдання 25.09.2025 р.

8 Календарний план наведено в таблиці 2.

Таблиця 2 — Календарний план

№	Назва та зміст етапу	Строк виконання	Примітка
1	Постановка задачі	26.09.2025	виконано
2	Аналіз предметної області та огляд літератури	08.10.2025	виконано
3	Розробка методів та алгоритмів розпізнавання	17.10.2025	виконано
4	Проєктування архітектури системи	28.10.2025	виконано
5	Реалізація програмного забезпечення	01.11.2025	виконано
6	Експериментальні дослідження та тестування	05.11.2025	виконано
7	Оформлення пояснювальної записки	10.11.2025	виконано
8	Попередній захист	11.11.2025	виконано
9	Перевірка якості виконання роботи та усунення недоліків	01.12.2025	виконано

Студент Бондаренко К.О

Керівник роботи к.т.н.,доц. каф. ОТ Крупельницький Л.В.

АНОТАЦІЯ

УДК 004.932.2:004.8

Бондаренко К. О. Інтегрована комп’ютерна система розпізнавання об’єктів на місцевості. Магістерська кваліфікаційна робота зі спеціальності 123 — комп’ютерна інженерія, освітня програма комп’ютерна інженерія. Вінниця: ВНТУ, 2025. 120 с.

На укр. мові. Бібліогр.: 34 назви, рис. 19, табл. 11.

У магістерській кваліфікаційній роботі розглянуто та реалізовано інтегровану комп’ютерну систему автоматичного розпізнавання об’єктів на зображеннях і відеопотоках місцевості на основі методів комп’ютерного зору та глибокого навчання. Проведено аналіз сучасних підходів до задачі детекції об’єктів, досліджено архітектурні особливості згорткових нейронних мереж та обґрунтовано доцільність використання моделі YOLOv8 як базового ядра системи розпізнавання з огляду на її точність, швидкодію та придатність до роботи в режимі реального часу.

У роботі розроблено програмну реалізацію системи з модульною архітектурою, що забезпечує обробку зображень і відеоданих у режимі реального часу, виконано експериментальні дослідження продуктивності та точності на різних апаратних конфігураціях, а також проведено порівняльний аналіз з відомими моделями розпізнавання об’єктів.

Ключові слова: розпізнавання об’єктів, комп’ютерний зір, нейронні мережі, YOLOv8, глибоке навчання, обробка зображень, Python, OpenCV.

ABSTRACT

Bondarenko K. O. Integrated Computer System for Object Recognition on Terrain. Master's Qualification Thesis in specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2025, 120 p.

Language: Ukrainian. References: 34 sources, 19 figures, 11 tables.

The master's qualification thesis presents the design and implementation of an integrated computer system for automatic object recognition in terrain images and video streams based on computer vision and deep learning methods. A comprehensive analysis of modern approaches to the object detection problem is conducted, the architectural features of convolutional neural networks are examined, and the feasibility of using the YOLOv8 model as the core of the recognition system is substantiated, considering its accuracy, computational efficiency, and suitability for real-time operation.

The work develops a software implementation with a modular architecture that enables real-time processing of image and video data, conducts experimental studies of performance and accuracy across different hardware configurations, and performs a comparative analysis with well-known object recognition models.

Keywords: object recognition, computer vision, neural networks, YOLOv8, deep learning, image processing, Python, OpenCV.

ЗМІСТ

ВСТУП

Актуальність теми дослідження полягає в тому, що технології комп’ютерного зору та автоматичного розпізнавання об’єктів відіграють дедалі важливішу роль у наукових дослідженнях і практичних застосуваннях різних галузей. Системи розпізнавання об’єктів на місцевості активно впроваджуються у сфері безпеки, моніторингу навколишнього середовища, сільського господарства, урбаністики, автономного транспорту та інших напрямках, де необхідна точна автоматизована інтерпретація зображень і просторових даних.

Розвиток глибинного навчання та нейронних мереж відкрив нові можливості для створення високоточних систем розпізнавання. Сучасні архітектури, такі як YOLO, R-CNN, EfficientDet, демонструють вражаючі результати у виявленні та класифікації об'єктів у режимі реального часу. Водночас існує потреба в інтегрованих рішеннях, які б поєднували різні підходи до розпізнавання, забезпечували масштабованість, можливість обробки даних з різних джерел та інтеграцію з іншими інформаційними системами.

Дослідження виконується в рамках сучасних напрямків розвитку штучного інтелекту, комп'ютерного зору та систем обробки великих даних, що відповідає пріоритетним напрямкам розвитку науки і техніки в Україні.

Метою роботи є проєктування інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості з використанням сучасних методів машинного навчання та комп'ютерного зору.

Для досягнення поставленої мети необхідно вирішити такі завдання:

— провести аналіз існуючих методів та систем розпізнавання об'єктів

— обрати оптимальну архітектуру нейронної мережі для розв'язання поставленої задачі

— розробити базову архітектуру системи з можливістю подальшого розширення;

— перевірити роботу розробленої системи на тестових даних;

— провести порівняльний аналіз з існуючими аналогами.

У роботі використовуються методи глибинного навчання, згорткові нейронні мережі, методи комп'ютерного зору, алгоритми обробки зображень, об'єктно-орієнтоване проєктування програмних систем, методи тестування та оцінки якості програмного забезпечення.

Об’єктом дослідження є процеси автоматичного розпізнавання та класифікації об’єктів на зображеннях місцевості.

Предметом дослідження є методи, алгоритми та програмні засоби комп’ютерного зору, що забезпечують побудову інтегрованих систем

Новизна роботи полягає у розробленні та впровадженні інтегрованої архітектури системи розпізнавання об’єктів, що забезпечує підвищення точності, масштабованості та універсальності процесів автоматизованого аналізу зображень місцевості. Запропонований підхід поєднує переваги одноетапних і багаторівневих методів детекції, модульну організацію обробки даних, оптимізовані механізми виділення ознак та адаптивну взаємодію компонентів. Система підтримує гнучку інтеграцію з зовнішніми інформаційними платформами й орієнтована на роботу з багатоджерельними даними різної природи, що розширює її функціональні можливості та практичне застосування.

Практичне значення роботи полягає в можливості використання розробленої системи для автоматизації моніторингу територій, у безпекових та охоронних комплексах, при аналізі даних дистанційного зондування Землі, у компонентах систем типу «розумне місто», а також у низці прикладних задач, що потребують високоточної та швидкодійної обробки зображень у режимі реального часу. Реалізація системи створює передумови для впровадження сучасних інтелектуальних технологій у сфери геоінформаційних досліджень, інфраструктурного моніторингу та автоматизованих аналітичних сервісів.

Апробація результатів роботи здійснена у доповідях на міжнародних та всеукраїнських наукових конференціях. Матеріали дослідження були представлені на Міжнародній науковій інтернет-конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення», на XVII Всеукраїнській науковій конференції «Актуальні питання сучасної економіки»,

а також на Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи». Основні положення роботи опубліковані у відповідних збірниках матеріалів конференцій [1—3].

Бондаренко Н. В., Бондаренко К. О. Застосування інтегрованих систем штучного інтелекту для автоматизованого розпізнавання об’єктів на місцевості //Тези доповіді. Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення. Міжнародна наукова інтернет-конференція (13—14 листопада 2025 р., м. Тернопіль (Україна), м. Ополе (Польща)). — Вип. 104. — 2025. Режим доступу: http://www.konferenciaonline.org.ua/ua/article/id-2343/ [1]

Бондаренко К. О. Використання систем розпізнавання об’єктів у фінансовому секторі //Тези доповіді. Актуальні питання сучасної економіки. XVII Всеукраїнська наукова конференція (13 листопада 2025 р.). — Умань : УНУ, 2025. — С. 178—180 [2].

Бондаренко К. О., Гнідунець В. О., Крупельницький Л. В. Розпізнавання об’єктів на фото та відео //Тези доповіді. Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи» (2024). — Вінниця, 2024. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21343/17713 [3].

1 ОГЛЯД МЕТОДІВ І СИСТЕМ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ

1.1 Огляд сфери застосування напрямку розпізнавання образів

Розпізнавання образів є однією з базових технологій штучного інтелекту, що забезпечує можливість автоматичної інтерпретації та класифікації об'єктів у цифрових даних. Ця технологія має широкий спектр практичних застосувань — від систем комп'ютерного зору, медичної діагностики та безпекових рішень до автономного транспорту, біометричних ідентифікаторів та аналітики відеопотоків. Швидкий розвиток обчислювальних ресурсів та алгоритмів машинного навчання стимулює активне вдосконалення методів розпізнавання, що підвищує точність і швидкодію сучасних систем.

Поєднання нейронних і ненейронних методів дає змогу створювати гібридні підходи до аналізу зображень і відео, які поєднують переваги обох парадигм. Нейронні мережі добре працюють із великими обсягами даних і здатні самостійно навчатися виявляти складні ознаки, тоді як класичні алгоритми, наприклад методи статистичного аналізу чи опорних векторів, забезпечують стабільність і контрольованість рішень.

Така комбінація дозволяє досягти високої ефективності в задачах класифікації, сегментації та детекції об'єктів. Прогрес у цій сфері обіцяє значні покращення в точності, масштабованості та адаптивності автоматизованих систем, що робить розроблення нових рішень у галузі розпізнавання образів надзвичайно актуальним завданням [4].

У класичній постановці задачі розпізнавання об'єктів універсальна множина спостережень розділяється на підмножини, які називаються образами. Кожен образ характеризується унікальною комбінацією ознак або атрибутів, що описують об'єкт. Наприклад, у випадку розпізнавання тексту універсальна множина може включати всі можливі символи алфавіту, тоді як образ "А" охоплює всі варіації написання цього символу — різні шрифти, нахили, товщини ліній тощо. Завдання системи розпізнавання полягає у визначенні, до якого саме образу належить кожен фрагмент вхідних даних [4 — 6].

Процес віднесення елемента до певного образу реалізується за допомогою правила прийняття рішення, яке визначає критерії схожості або належності.

Важливою складовою цього процесу є метрика — спосіб обчислення відстані між елементами універсальної множини. Відстань визначає міру подібності між об’єктами: чим вона менша, тим більша схожість за набором ознак. У більшості випадків об’єкти представляються у вигляді векторів ознак, а метрика задається функцією, що оцінює відмінності між цими векторами (наприклад, евклідова або косинусна відстань).

Коректний вибір метрик оцінювання є визначальним чинником зменшення кількості помилок у задачах класифікації та підвищення надійності результатів розпізнавання. Однією з фундаментальних властивостей образів є здатність до узагальнення, яка полягає у розпізнаванні раніше невідомих представників певного класу на основі обмеженої навчальної вибірки.

Такі можливості забезпечуються здатністю алгоритмів автоматично виділяти інваріантні ознаки, спільні для об’єктів однієї категорії, навіть за наявності істотних відмінностей у зовнішньому вигляді. Саме ця властивість лежить в основі ефективного застосування сучасних методів машинного навчання у складних та різнорідних середовищах.

Ефективність програми розпізнавання значною мірою залежить від способу представлення образів, вибору алгоритму класифікації та реалізації метрики. Різні методи можуть демонструвати відмінну стійкість до шумів, змін освітлення чи масштабу, тому вибір оптимального підходу визначається специфікою поставленого завдання [5].

Отже, розпізнавання образів є складною багаторівневою задачею, що інтегрує методи математичного моделювання, машинного навчання та інтелектуальної обробки даних. Подальший розвиток цього напряму зумовлений необхідністю створення алгоритмів, здатних забезпечувати високу адаптивність, робастність і точність роботи в умовах різноманітності сцен, шумів та обмежених обчислювальних ресурсів, що визначає актуальність досліджень у сфері сучасного комп’ютерного зору.

1.2. Структура та сфери застосування систем розпізнавання образів

Система машинного зору є комплексним технічним засобом, призначеним для автоматичного отримання, обробки та аналізу зображень з метою прийняття рішень на основі візуальної інформації.

Вона складається з кількох ключових компонентів, серед яких виділяють підсистему формування зображень, обчислювальний елемент та блок обробки зображень. Підсистема формування зображень забезпечує отримання візуальних даних за допомогою камер, сенсорів або інших пристроїв збору інформації. Обчислювальний елемент виконує первинну обробку сигналів і передачу даних до модуля аналізу. Блок обробки зображень містить алгоритми детекції, сегментації, класифікації та інтерпретації об’єктів, які можуть бути реалізовані як програмними засобами, так і апаратними компонентами, наприклад FPGA або GPU. Узагальнену схему коп'ютерної системи розпізнавння образів подано на рисунку 1.1 [7].

Рисунок 1.1 — Загальна структура комп'ютерної системи розпізнавння образів.

FPGA (Field-Programmable Gate Array) — це програмована користувачем логічна інтегральна схема, яка дозволяє конфігурувати апаратну логіку під конкретне завдання. Використовується для високошвидкісної обробки даних і паралельних обчислень у реальному часі [3,4].

GPU (Graphics Processing Unit) — це графічний процесор, спочатку розроблений для рендерингу зображень, але нині широко застосовується для

виконання масових паралельних обчислень, зокрема у задачах машинного навчання та комп'ютерного зору [5].

Залежно від архітектури, ці елементи можуть функціонувати як окремі модулі або утворювати інтегровану систему, здатну забезпечувати повний цикл обробки — від збору до прийняття рішень у реальному часі [7,8].

Сфера розпізнавання образів охоплює широкий спектр технологій і методів, спрямованих на аналіз і інтерпретацію зображень або відеопотоків для автоматичного виявлення, класифікації та ідентифікації об'єктів, сцен чи подій. Залежно від поставлених завдань системи такого типу можуть функціонувати автономно або бути складовою більш комплексних інформаційних систем.

Застосування технологій розпізнавання образів охоплює численні галузі. Автоматичне виявлення, відстеження об'єктів, розпізнавання облич та аналіз поведінки підвищують ефективність систем контролю доступу та моніторингу територій. Аналіз медичних зображень (рентгенівських, МРТ, КТ) дає змогу автоматизувати процес діагностики, оцінювати динаміку захворювань і підтримувати клінічні рішення. Системи допомоги водієві (ADAS) та автономні транспортні засоби використовують алгоритми розпізнавання образів для виявлення перешкод, дорожніх знаків і пішоходів, що підвищує безпеку руху.

Автоматизований аналіз поведінки клієнтів, моніторинг полиць і облік товарів сприяють оптимізації логістичних процесів та підвищенню прибутковості. Системи розпізнавання документів і біометричної ідентифікації клієнтів використовуються для підвищення безпеки транзакцій і спрощення процесів верифікації.

Алгоритми комп'ютерного зору допомагають контролювати стан посівів, прогнозувати врожайність і виявляти хвороби рослин на ранніх стадіях розвитку [9].

Отже, системи розпізнавання образів є фундаментальним елементом сучасних інтелектуальних технологій. Їх розвиток сприяє автоматизації аналітичних процесів у різних сферах діяльності, забезпечуючи підвищення точності, оперативності та ефективності прийняття рішень.

1.3. Класифікація об’єктів на місцевості та постановка задачі розпізнавання

Зображення місцевості, отримані методами дистанційного зондування Землі (ДЗЗ), містять велику кількість різнорідних об’єктів, які відрізняються за своїм походженням, формою, розмірами, текстурою, спектральними характеристиками та контекстним оточенням. Для систематизації процесу їх аналізу й автоматичного розпізнавання першочерговим завданням є створення чіткої класифікаційної схеми, яка забезпечить логічну структуру даних і підвищить ефективність роботи алгоритмів комп’ютерного зору [10,12].

Всі об’єкти на місцевості можна умовно розділити на три основні групи: природні, антропогенні (штучні) та гібридні. Такий поділ відображає ступінь впливу людини на формування та зміну характеристик об’єктів.

Природні об’єкти — це елементи ландшафту, сформовані без втручання людини. До них належать:

— водні об’єкти (річки, озера, моря, болота, водосховища);

— рослинність (хвойні й листяні ліси, луки, степи, чагарники);

— форми рельєфу (гори, пагорби, яри, долини, дюни);

— відкритий ґрунт (скельні породи, піски, солончаки).

Антропогенні об’єкти — створені або суттєво змінені внаслідок людської діяльності. Серед них:

— будівлі та споруди (житлові, промислові, адміністративні, мости, вежі, греблі);

— транспортна інфраструктура (автомобільні дороги, залізниці, аеродроми, порти, трубопроводи);

— рухомі транспортні засоби (автомобілі, потяги, літаки, судна);

— лінії електропередач та комунікаційні мережі.

Гібридні об’єкти — поєднують природні характеристики з результатами діяльності людини. До цієї групи належать:

— сільськогосподарські угіддя (поля, сади, виноградники, пасовища);

— рекреаційні зони (парки, сквери, спортивні майданчики);

— кар’єри та відвали гірських порід.

Подібна класифікація дозволяє структурувати задачу розпізнавання та підбирати оптимальні методи аналізу для кожного типу об’єктів, враховуючи їхні спектральні та геометричні особливості [10, 11].

1.4 Постановка задачі розпізнавання об’єктів

Задача розпізнавання об’єктів у комп’ютерному зорі полягає у автоматичній ідентифікації семантичних об’єктів певного класу на цифровому зображенні або у відеопотоці. Залежно від прикладної мети та необхідного рівня деталізації, її можна декомпозувати на кілька підзадач різної складності.

Класифікація (Image Classification) — найпростіший варіант, коли система визначає категорію зображення загалом. На виході формується один ярлик для всього кадру (наприклад, «ліс», «міська забудова», «водна поверхня»).

Детекція об’єктів (Object Detection) — складніша задача, що передбачає одночасно класифікацію та локалізацію об’єктів на зображенні. Результатом є набір прямокутних рамок (bounding boxes), що обмежують кожен знайдений об’єкт, з вказанням відповідного класу.

Сегментація (Segmentation) — найдетальніша форма розпізнавання, спрямована на виділення контурів об’єктів на рівні окремих пікселів. Вона поділяється на два типи:

Семантична сегментація (Semantic Segmentation): кожен піксель класифікується як такий, що належить до певного класу (наприклад, усі пікселі, що відповідають дорогам, мають один колір, а будівлям — інший). При цьому окремі екземпляри не розрізняються [12].

Сегментація екземплярів (Instance Segmentation): поєднує семантичну сегментацію з детекцією — система визначає межі об’єктів і розрізняє окремі екземпляри одного класу (наприклад, кожен автомобіль на стоянці виділяється окремою маскою).

Усі алгоритми розпізнавання об’єктів зазвичай включають кілька основних етапів:

— попередня обробка зображення (нормалізація, фільтрація шумів, покращення контрасту);

— виділення ознак (геометричних, колірних, текстурних тощо);

— формування дескрипторів (компактне представлення інформації);

— класифікація (власне прийняття рішення про належність об’єкта до певного класу) [12, 13].

Приклад роботи алгоритму представлено на рисунку 1.2.

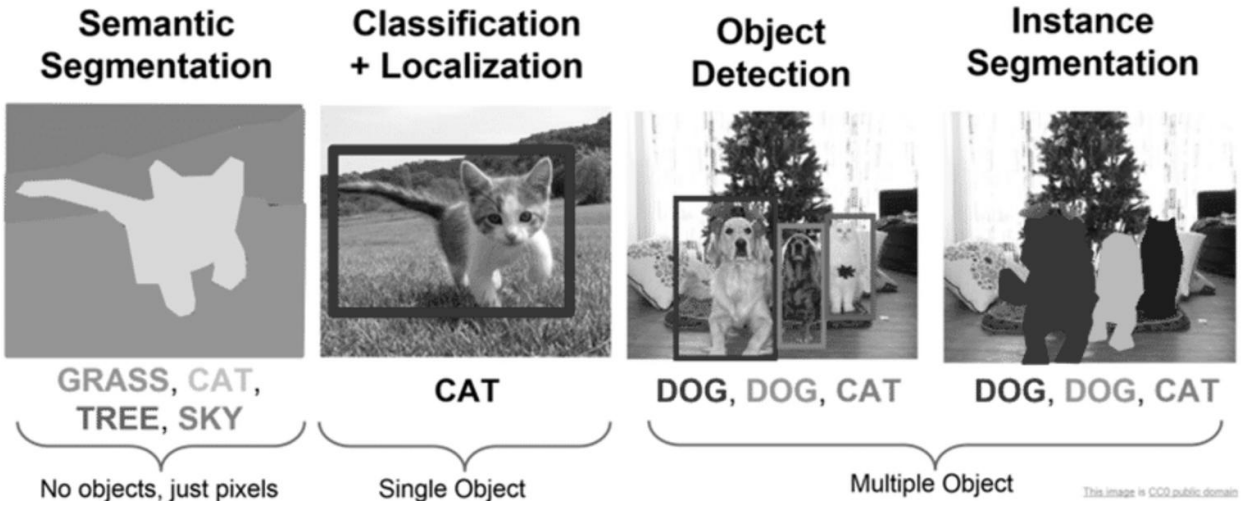


Рисунок 1.2 — Схема етапів сегментації та розпізнавання об’єктів

Отже, розпізнавання об’єктів на місцевості є багаторівневою та комплексною задачею, успішне розв’язання якої потребує вибору оптимальної комбінації методів — класифікації, детекції або сегментації — залежно від цілей дослідження, вимог до точності, швидкодії та масштабу обробки даних. Застосування таких технологій має вирішальне значення у сфері створення й оновлення цифрових карт, моніторингу змін землекористування, планування інфраструктури та забезпечення національної безпеки.

1.5. Огляд методів комп'ютерного зору для розпізнавання об'єктів

Розпізнавання об’єктів на зображеннях місцевості є однією з ключових задач комп’ютерного зору, яка має широке практичне застосування в картографії, дистанційному зондуванні Землі, моніторингу просторових змін

територій та автоматизованому аналізі аерофотознімків. Загальну схему алгоритмічного процесу розпізнавання образів наведено на рисунку 1.3.

На поданій схемі відображено основні етапи обробки даних, зокрема формування вхідного зображення, виконання попередньої обробки для покращення якості даних, виділення інформативних ознак та етап класифікації й локалізації об’єктів. Завершальною стадією алгоритму є формування результатів розпізнавання, які можуть бути використані для подальшого аналізу або інтеграції з прикладними інформаційними системами.

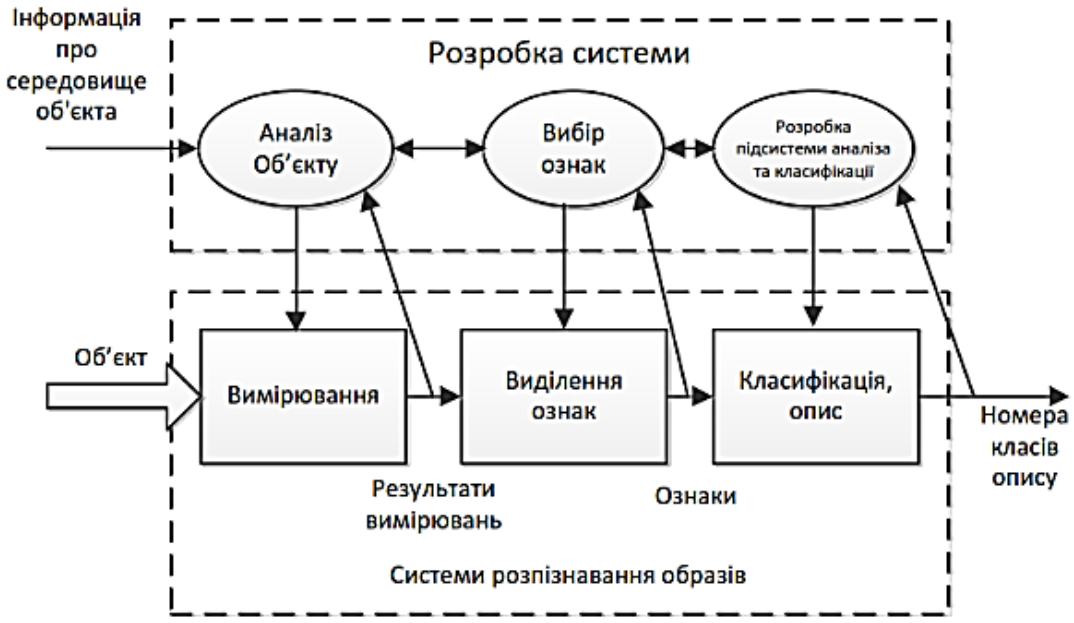


Рисунок 1.3 — Узагальнена схема процесу розпізнавання образів

Методи, що використовуються для цих задач, умовно поділяють на класичні алгоритми комп’ютерного зору та підходи на основі глибинного навчання, кожен із яких має власні переваги та обмеження [13, 14].

1.5.1. Класичні методи комп'ютерного зору

Перші підходи до автоматичного розпізнавання об’єктів ґрунтувалися на аналізі локальних ознак зображення, що дозволяло виявляти характерні точки та описувати їхні властивості.

Класичні методи розпізнавання об’єктів ґрунтуються на ручному виділенні

ознак зображення, тобто алгоритми аналізують геометричні, текстурні та інтенсивні характеристики пікселів для виявлення об’єктів. До найпопулярніших методів належать SIFT, SURF, HOG [15].

Одним із найвідоміших таких методів є SIFT (Scale-Invariant Feature Transform), який забезпечує стабільне виявлення ключових точок на зображенні та формування дескрипторів на основі локальних градієнтів яскравості.

Ця технологія відзначається інваріантністю до масштабу, обертання та часткової оклюзії, що дозволяє ефективно аналізувати складні сцени з великою кількістю об’єктів різної форми та текстури. Приклад обробка зображень із застосуванням алгортму SIFT пркдставлено на рисунку 1.4.

Рисунок 1.4 — Обробка зображень за алгортмом SIFT

Використання SIFT особливо актуальне при аналізі будівель, транспортних засобів і природних ландшафтів на аерофотознімках, де об’єкти можуть з’являтися у різних масштабах та ракурсах. Незважаючи на високу точність, метод SIFT є досить обчислювально затратним і чутливим до шумів, що може обмежувати його застосування для великих масивів зображень у реальному часі [15,17].

Інший метод — SURF (Speeded-Up Robust Features) — виник як вдосконалена альтернатива SIFT і дозволяє прискорити обчислення за рахунок використання наближеного детектора

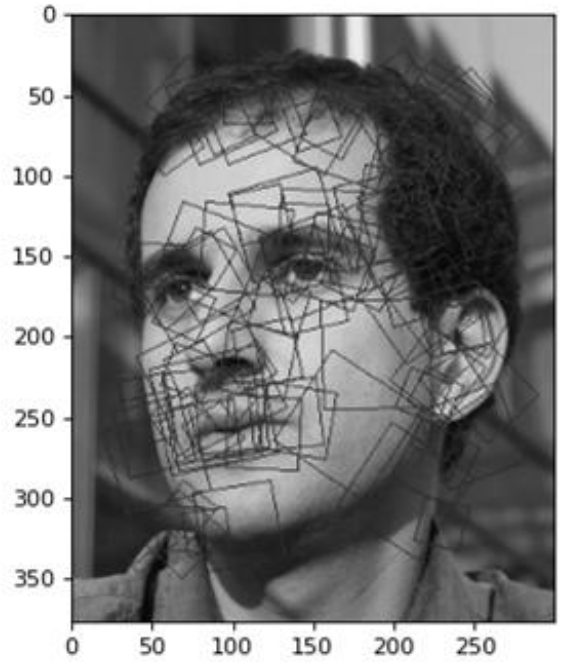


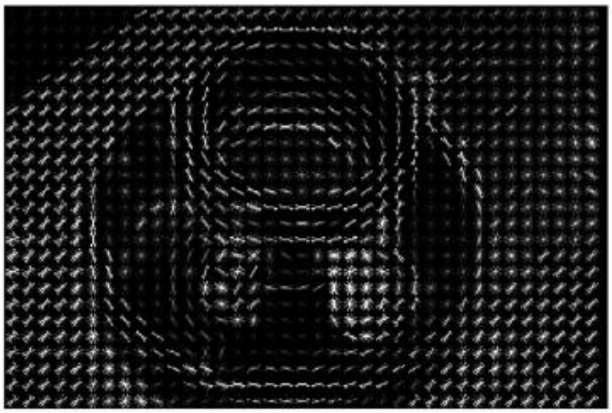
Рисунок 1.5 — Обробка зображень за алгортмом SURF

Гессіана та дескрипторів, сформованих на основі сум відгуків вейвлетів Хаара. SURF забезпечує порівняно високу точність при аналізі об’єктів на зображенні та зберігає інваріантність до масштабу та обертання, але менш стабільний при сильному шумі або значних змінах освітлення. Обробка зображень за алгортмом SURF представлена на рисунку 1.5. Його застосовують для автоматичного виявлення транспортних засобів, будівель та об’єктів інфраструктури, а також у задачах створення панорам і 3D-реконструкції місцевості [16].

Метод HOG (Histogram of Oriented Gradients) вирізняється принципово іншим підходом, орієнтованим на структуру та контури об’єктів. Зображення ділиться на локальні ділянки, для яких обчислюються гістограми орієнтованих градієнтів, що дозволяє формувати дескриптори, які відображають характерні контури об’єкта [17].

HOG показує особливу ефективність для детекції об’єктів з виразними формами, таких як пішоходи, транспортні засоби або будівлі, і забезпечує стабільність до часткових змін освітлення та обмеженої оклюзії. Обробка зображень за алгортмом HOG представлена на рисунку 1.6.

а) тестове зображення



б) гістогрма орієнтованих градієнтів

Рисунок 1.6 — Обробка зображень за алгортмом HOG

Разом із класичними класифікаторами, такими як метод опорних векторів, HOG часто застосовується у системах відеоспостереження та аерофотознімання.є ефективним для детекції об’єктів із характерною формою, зокрема пішоходів, транспортних засобів чи будівель [18].

Хоча класичні методи SIFT, SURF та HOG демонструють високу ефективність у багатьох завданнях, вони мають суттєві обмеження, зокрема потребують ручного проєктування ознак, слабо адаптуються до різноманітних типів об’єктів і часто втрачають точність у складних або зашумлених сценах. Ці недоліки стали однією з основних причин переходу до глибинного навчання, яке здатне автоматично виділяти ознаки та адаптуватися до різних умов зображень, забезпечуючи більш високу точність, гнучкість і стійкість розпізнавання об’єктів.

1.5.2. Методи глибокого навчання

З розвитком обчислювальних ресурсів та появою великих наборів даних традиційні методи розпізнавання образів поступово замінюються підходами на основі глибинного навчання (deep learning) [18].

Основною відмінністю таких методів є здатність автоматично виділяти ознаки зображення без потреби ручного проєктування дескрипторів. Глибинні

нейронні мережі, зокрема згорткові нейронні мережі (CNN, Convolutional Neural Networks), показали надзвичайно високу ефективність у задачах класифікації, детекції та сегментації об’єктів на різних типах зображень, включно з аерофотознімками та супутниковими даними.

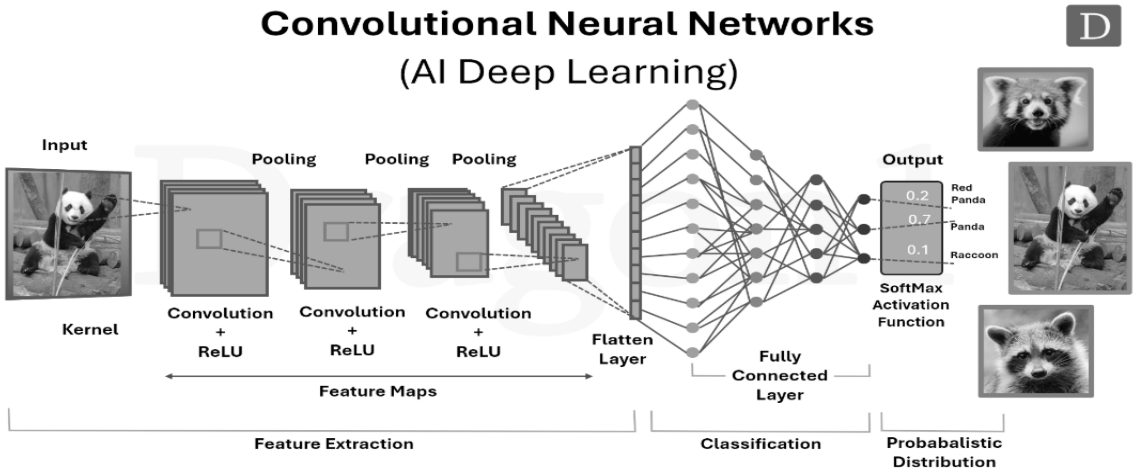


Рисунок 1.7 — Обробка зображень за алгортмом CNN

Методи глибинного навчання можна умовно класифікувати за характером задач, які вони вирішують. До першої групи належать мережі для класифікації зображень, які на виході видають категорію об’єкта для цілого кадру. Класичним прикладом є архітектури AlexNet, VGG, ResNet, які демонструють високу точність класифікації навіть на складних і різнорідних даних [18].

Наступна група включає методи детекції об’єктів, що поєднують класифікацію з локалізацією об’єктів у межах зображення. Серед відомих рішень варто зазначити R-CNN, Fast R-CNN, Faster R-CNN, а також сімейство алгоритмів YOLO (You Only Look Once), які характеризуються високою швидкістю роботи та точністю виявлення. Візуальне представлення архітектури YOLOv3 зображена на рисунку 1.8

Окремо виділяють методи сегментації, серед яких семантична сегментація здійснюється за допомогою мереж на кшталт FCN (Fully Convolutional Networks), а сегментація екземплярів реалізована в алгоритмах Mask R-CNN, де поєднується детекція та точне виділення контурів об’єктів на рівні пікселів. Серед сучасних тенденцій розвитку глибинного навчання у сфері розпізнавання

образів спостерігається низка напрямів, що визначають подальшу еволюцію алгоритмів комп'ютерного зору.

По-перше, активно розробляються легкі та високопродуктивні архітектури, орієнтовані на роботу в реальному часі в умовах обмежених обчислювальних ресурсів. Такі моделі оптимізовано для мобільних платформ, вбудованих мікросистем та edge-пристроїв, що дозволяє забезпечити швидке та енергоефективне розпізнавання об'єктів у польових умовах [19].

По-друге, дедалі ширше застосовуються моделі із самонавчанням та трансформерні архітектури, зокрема Vision Transformers (ViT) та їх похідні. На відміну від класичних згорткових мереж, ці моделі оперують глобальними контекстами сцени, що дозволяє більш точно виділяти просторові залежності та покращувати розпізнавання складних або слабко виражених об'єктів. Трансформери демонструють особливу ефективність у випадках, коли контекст має більшу інформативність, ніж локальні ознаки [20].

По-третє, значна увага приділяється методам підвищення точності локалізації, сегментації та багаторівневого аналізу об'єктів. Це критично важливо для обробки аерофотознімків та супутникових даних, де об'єкти можуть мати невеликий розмір, бути частково перекритими або втрачати контраст через атмосферні умови. У таких сценаріях використовуються мережі з багаторівневими пірамідами ознак, механізми адаптивної уваги та алгоритми об'єднання контекстів [20].

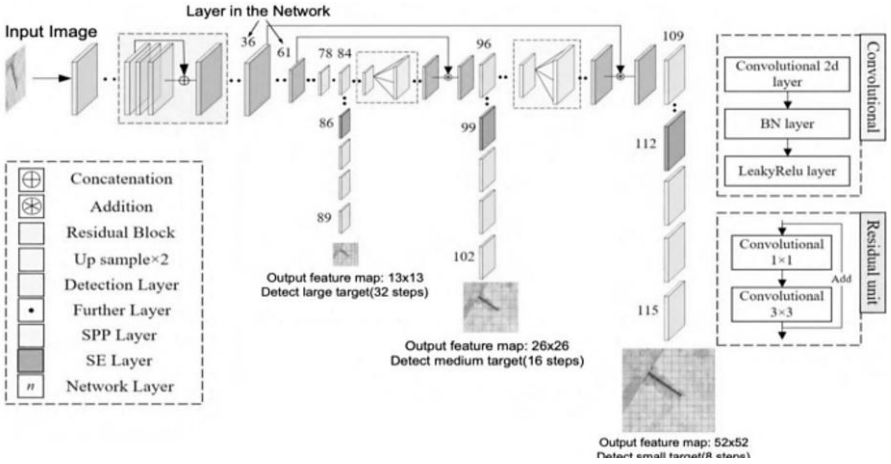


Рисунок 1.8 — Архітектура YOLOv3

Іншим важливим напрямом є інтеграція мультиспектральних та гіперспектральних даних. Поєднання інформації в різних спектральних діапазонах дозволяє підвищити точність класифікації природних та антропогенних об’єктів, а також визначати характеристики, недоступні у звичайних RGB-зображеннях. Це є ключовим для моніторингу екосистем, аналізу поверхні ґрунту та виявлення змін у природному середовищі.

З урахуванням зазначених тенденцій, у дослідженні використано вдосконалену архітектуру YOLOv8, яка поєднує високу швидкодію одноетапних детекторів із точністю, притаманною більш складним моделям. Алгоритм підтримує інтеграцію додаткових обчислювальних блоків, механізмів уваги та модулів для роботи з різнорідними джерелами даних, що дозволяє ефективно виявляти об’єкти різного масштабу на місцевості. Завдяки цьому YOLOv8 забезпечує збалансоване співвідношення між швидкістю обробки, точністю класифікації та стабільністю роботи в умовах змін зовнішнього середовища, що робить її оптимальною для задач картографування, моніторингу територій та автоматизованої аналітики аерофотознімків [21].

У межах сучасної класифікації (рис.В.1) алгоритми виявлення об’єктів поділяються на дві основні категорії залежно від того, скільки разів одне вхідне зображення проходить через нейронну мережу. Двоетапні детектори (two-stage detectors), представлені R-CNN, Fast R-CNN та Faster R-CNN, передбачають попереднє формування регіонів-кандидатів із подальшою їх класифікацією. Модифікації Fast R-CNN та Faster R-CNN частково усунули проблеми низької швидкодії завдяки спільному використанню згорткових шарів та впровадженню мережі пропозицій регіонів (Region Proposal Network), що значно прискорило обробку зображень і покращило точність локалізації об’єктів [22].

1.6. Проблеми та виклики

Розпізнавання об’єктів на зображеннях місцевості стикається з низкою складних проблем, які суттєво впливають на точність та робастність систем комп’ютерного зору. Однією з головних складностей є варіативність умов

освітлення, оскільки різниця між денним та нічним світлом може перевищувати шість порядків величини. Класичні CNN демонструють падіння точності на 30—40% при переході від денного до нічного освітлення, що ставить під загрозу надійність розпізнавання в умовах мінливого освітлення.

Частково проблему вирішують методи data augmentation, які симулюють різні умови освітлення під час навчання, проте цього недостатньо для повного забезпечення стійкості моделей. Більш ефективними є архітектури з адаптивною нормалізацією, такі як AdaIN, що дозволяють мережі коригувати внутрішні представлення під конкретні умови освітлення, а також підходи на кшталт domain adaptation, які переносять знання, отримані на денних зображеннях, на нічні сцени [23].

Фізичне розширення динамічного діапазону сцени за допомогою HDR-камер або adaptive exposure control також підвищує деталізацію зображення, хоча воно вимагає додаткових обчислювальних ресурсів і може призводити до артефактів руху. Внесок HDR-камер в підвищення чіткості зображень продемонстровано на рисунку 1.9.

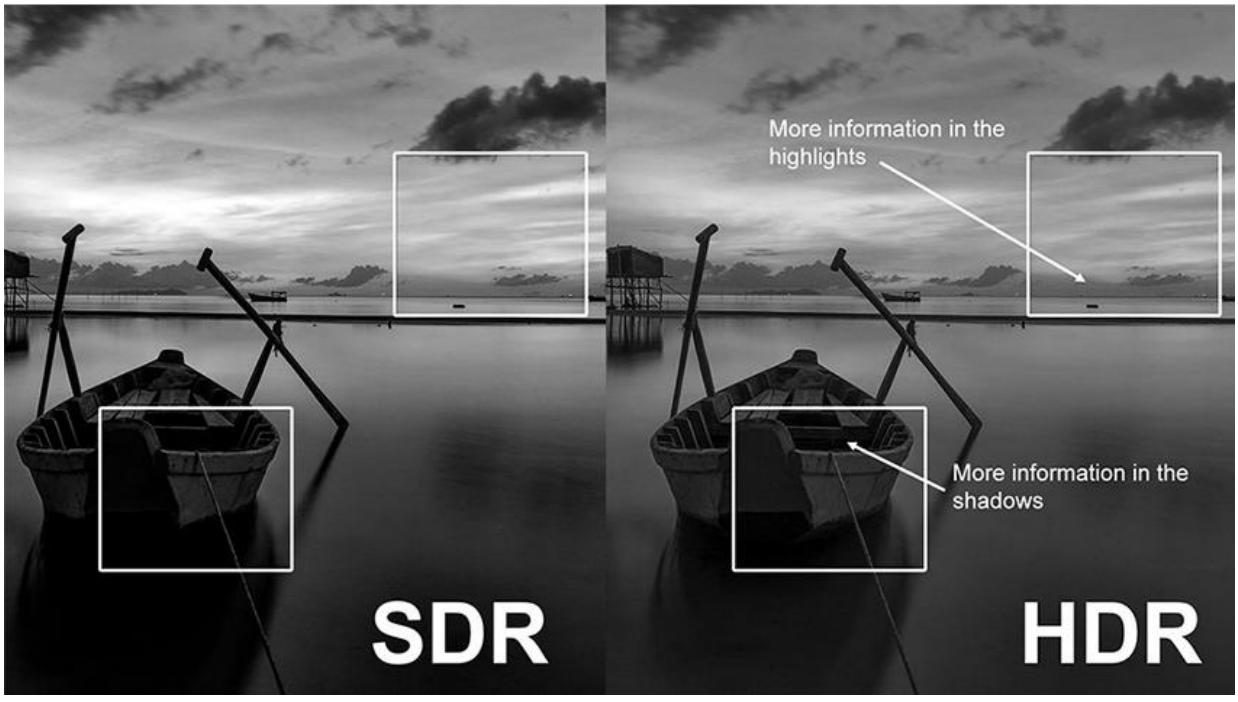


Рисунок 1.9 — Внесок HDR у деталізацію у світлих і темних ділянках

Сезонні зміни також змінюють візуальну структуру місцевості: сніговий покрив майже нівелює текстури поверхонь, а зміна кольору рослинності унеможливлює використання суто колірних ознак для детекції. Ще одним важливим викликом для систем автоматизованого розпізнавання об’єктів є вплив погодних умов та сезонних змін на якість вхідних зображень. Атмосферні явища — дощ, туман, сніг, пил чи підвищена вологість — спричиняють появу додаткових шумів і спотворень, що значно знижує інформативність сцени [23].

Дощові краплі можуть частково перекривати об’єктив, формуючи яскраві артефакти або напівпрозорі смуги, тоді як туман та аерозолі розсіюють світло, зменшуючи контраст, насиченість кольорів та різкість контурів. Для компенсації цих впливів застосовуються методи видалення туману (dehazing) та видалення дощу (deraining), які спрямовані на реконструкцію чистого зображення з мінімальною кількістю атмосферних артефактів. Видалення туману (dehazing) базується на фізичній атмосферній моделі зображення, що описує співвідношення між справжньою сценою та світлом, яке проходить через атмосферу [24]:

$$I(x) = J(x)\,t(x) + A\,(1 - t(x)) \tag{1.1}$$

де $I(x)$— спостережуване зображення,

$J(x)$— відновлене (справжнє) зображення сцени,

A— глобальна освітленість атмосфери,

$t(x)$— коефіцієнт пропускання (transmission), який характеризує ступінь розсіювання туману.

Процес відновлення полягає в оцінці параметрів Aта $t(x)$, після чого виконується реконструкція:

$$J(x) = \frac{I(x) - A}{t(x)} + A \tag{1.2}$$

Типовими методами оцінювання є аналіз фонової освітленості та локального контрасту, що продемонстровано в додатку Г (рис. Г.1), на цьому графіку червона лінія відповідає методу оцінки фонової (окружної) освітленості, синя лінія — методу оцінки передачі (transmission), а зелена лінія — отриманому результату [23,24].

Методи видалення дощу (deraining) спрямовані на усунення дощових смуг і крапель, які мають специфічну форму, високу яскравість та короткий час життя в кадрі. У загальному вигляді модель дощового зображення може бути описана як:

$$I = B + R \tag{1.3}$$

де I— зображення з дощем,

B— чиста сцена (background),

R— компонента, що відповідає за дощові артефакти.

Оскільки дощові смуги мають характерну спрямованість та високочастотну структуру, сучасні алгоритми використовують виділення градієнтів, покращення структури країв або нейронні мережі, які навчаються відокремлювати фонові та дощові компоненти. Це дозволяє забезпечити стабільність роботи алгоритмів розпізнавання в умовах активних опадів [23].

Таким чином, обидві категорії методів відіграють важливу роль у підвищенні точності детекції об’єктів під час несприятливих погодних умов, хоча вони можуть вводити додаткові артефакти та потребують обережного застосування. Особливо це актуально для систем реального часу, де наявність туману або дощу може суттєво вплинути на достовірність отриманих результатів і швидкість їх обробки.

Ефективним підходом є навчання детекторів безпосередньо на зображеннях зі складними погодними умовами або використання мультисенсорного злиття з термальними та радарними даними, що забезпечує високу надійність навіть в екстремальних умовах. Складність також створює

масштабування об’єктів та різні перспективи. Стандартні CNN мають обмежене рецептивне поле, що ускладнює одночасну детекцію малих і великих об’єктів [24].

Для вирішення цієї проблеми застосовуються ієрархічні архітектури, такі як Feature Pyramid Networks та їх модифікації (PANet, BiFPN), які дозволяють будувати багаторівневі представлення ознак і покращують multi-scale детекцію [25].

Виявлення дуже малих об’єктів на аерофотознімках та супутникових даних є одним із найскладніших завдань у сучасних системах комп’ютерного зору, оскільки такі об’єкти займають незначну кількість пікселів, часто зливаються з фоном і практично не містять текстурних ознак.

Для покращення якості їх детекції застосовуються методи попередньої обробки, зокрема суперроздільність (super-resolution), яка штучно підвищує деталізацію зображення. Найпоширенішою є модель підвищення роздільної здатності за допомогою згорткових нейронних мереж, яка формально описується як відображення низькодеталізованого зображення I_{LR} у високодеталізоване I_{SR}:

$$I_{SR} = F_\theta(I_{LR}), \tag{1.4}$$

де F_θ— нейронна мережа з параметрами θ, навчена на реконструкції дрібних структур.

У результаті модель розпізнавання отримує збільшені та чіткіші ознаки, що підвищує ймовірність коректного виділення малих об’єктів. Окрім того, важливу роль відіграють методи контекстуальної детекції (context-based detection), які дозволяють моделі враховувати просторове оточення об’єкта. Контекст часто містить більше інформації, ніж сам об’єкт, наприклад дорожня інфраструктура для виявлення автомобілів або характерні структури агроландшафтів для ідентифікації техніки. Математично контекст можна описати як функцію:

$$P(O \mid C) = f(C), \tag{1.5}$$

де O— імовірність наявності об’єкта,

C— контекстуальна область навколо нього,

f— модель, що враховує просторові залежності.

Ще одним ефективним механізмом є спрямована увага (guided attention), яка дозволяє нейронній мережі зосереджуватися на найбільш значущих регіонах, навіть якщо об’єкти майже непомітні. Механізми уваги формують матрицю ваг:

$$A = \text{softmax}(QK^T), \tag{1.6}$$

де Q та K— матриці запитів та ключів, що відповідають просторовим області зображення, а A визначає, які ділянки важливіші для детектора [25, 27].

Завдяки цьому модель «підсилює» ті частини зображення, де може знаходитись малий об’єкт. Особливу складність становить варіативність перспективи, характерна для зйомки з дронів та інших літальних апаратів, де об’єкти можуть спостерігатися під різними кутами, мати перспективні спотворення або масштабні деформації [26].

Для усунення цих ефектів використовують детектори, обізнані про 3D-геометрію сцени (3D-aware detectors). Такі моделі враховують параметри камери, кут зйомки та можливу 3D-орієнтацію об’єкта, що дозволяє коректніше реконструювати його проекцію на зображенні. Часто це доповнюється використанням синтетичних датасетів, де об’єкти рендеряться з різних висот і під різними кутами огляду:

$$D_{syn} = \{R(O_i, \alpha, \beta, h)\}, \tag{1.7}$$

де R— функція рендерингу об’єкта O_iпід кутами α, βта висотою h.

Таке розширення даних значно підвищує робастність моделі, забезпечуючи стабільність її роботи в реальних умовах польоту, де перспектива постійно змінюється [26].

Не менш суттєвим викликом для інтегрованих систем розпізнавання об’єктів є забезпечення обробки даних у реальному часі, оскільки широкий спектр практичних застосувань — зокрема системи відеоспостереження, навігація безпілотних літальних апаратів та моніторинг транспортних потоків — потребують стабільного аналізу відеопотоку з частотою не нижче 25 кадрів за секунду [26].

Двоетапні детектори (two-stage detectors), такі як R-CNN, Fast R-CNN та Faster R-CNN, виконують виявлення об’єктів у два послідовні кроки. На першому етапі формується набір регіонів-кандидатів (region proposals), зазвичай за допомогою мережі RPN (Region Proposal Network), яка генерує потенційні області, де можуть розташовуватися об’єкти. На другому етапі кожен із цих регіонів окремо подається до CNN-класифікатора, який визначає клас об’єкта та коригує межі його рамки. З математичного погляду загальна латентність двоетапної системи може бути описана як:

$$T_{\text{total}} = T_{\text{RPN}} + N \cdot T_{\text{ROI}}, \tag{1.8}$$

де T_{RPN}— час генерації пропозицій,

T_{ROI}— час обробки одного регіону,

N— кількість регіонів-кандидатів.

Оскільки Nможе сягати сотень або тисяч, двоетапні методи демонструють високу точність, але значну затримку обробки: зазвичай від 5 до 10 fps на сучасних GPU. Це робить їх малопридатними для систем реального часу, особливо у випадках високої динаміки сцени або великої кількості об’єктів [26].

Одноетапні моделі, навпаки, виконують проходження зображення через мережу лише один раз і усувають потребу в окремому етапі генерації регіонів-

кандидатів, що суттєво зменшує затримки обробки. Порівняльна класифікація одноетапних та двоетапних алгоритмів детекції представлена на рисунку Г.1, де наочно продемонстровано відмінності у кількості проходів через модель та вплив цього фактора на швидкодію [26].

Саме тому сучасні системи реального часу орієнтуються на використання моделей типу YOLOv8, CenterNet або SSD, які завдяки оптимізованим архітектурам, паралельній обробці та можливості розгортання на edge-платформах забезпечують необхідний баланс між точністю та пропускною здатністю [26].

Одноетапні детектори, такі як YOLO, SSD або EfficientDet, оптимізовані для швидкої обробки і досягають від 30 до 100 fps при прийнятній точності. Для обчислень на периферії (edge computing) та вбудованих систем застосовують полегшені нейромережеві архітектури, такі як MobileNet, ShuffleNet та EfficientNet, які базуються на глибинно-роздільних згортках (depthwise separable convolutions) та методах автоматизованого пошуку архітектур нейронних мереж (neural architecture search), що дозволяє суттєво зменшити обчислювальні витрати та споживання ресурсів без істотної втрати точності [26].

Апаратне прискорення на базі TPU, VPU та спеціалізованих AI-акселераторів забезпечує виконання інференсу в режимі реального часу навіть на ресурсно обмежених платформах, зокрема мобільних і вбудованих пристроях. Додатково застосування методів оптимізації моделей, зокрема квантування, проріджування параметрів та дистиляції знань, дозволяє істотно зменшити обчислювальну складність і обсяг пам’яті без суттєвого зниження точності. У сукупності це робить одноетапні детектори придатними для стабільної обробки відеопотоків у реальному часі та забезпечує їхню високу робастність до змінних умов обчислювального середовища.

Розвиток алгоритму YOLO від версій v2 і v3 до сучасних v8 і v9 поєднує високу швидкість обробки з точністю детекції, що робить його оптимальним для задач моніторингу та аналізу місцевості. Таким чином, сучасні методи глибинного навчання дозволяють об’єднати точність і швидкість, що є критично

важливим для практичного використання у задачах дистанційного зондування, автоматичного аналізу супутникових і аерофотознімків, а також у системах реального часу [26].

1.7 Порівняльний аналіз продуктивності методів розпізнавання

Сучасні методи розпізнавання об’єктів демонструють різні співвідношення між точністю, швидкістю обробки та вимогами до обчислювальних ресурсів. Класичні алгоритми, такі як SIFT, SURF та HOG, забезпечують відносно високу швидкість роботи та простоту реалізації, однак їх точність значно знижується у складних або зашумлених умовах, а також при варіативності масштабу та освітлення.

Традиційні алгоритми SIFT, SURF і HOG забезпечують високу швидкість обробки та низькі вимоги до апаратних ресурсів, проте їх точність обмежена через ручне проєктування ознак і низьку робастність до варіацій освітлення, масштабу чи ракурсів. Двоетапні моделі, такі як Faster R-CNN і Cascade R-CNN, навпаки, забезпечують найвищу точність (від 80% до 90% mAP) завдяки ефективній комбінації генератора регіонів (RPN) та класифікаційних мереж, однак їх швидкодія (від 5 до 10 FPS) є недостатньою для роботи в реальному часі та потребує високопродуктивних GPU. Одноетапні архітектури SSD та RetinaNet демонструють збалансованість між точністю та швидкістю, забезпечуючи 30—60 FPS при середніх ресурсних витратах, що робить їх придатними для відеопотоків та оперативних застосувань.

Двоетапні детектори, серед яких Faster R-CNN та Cascade R-CNN, відзначаються найвищою точністю завдяки багаторівневій обробці ознак і застосуванню складних регресійних механізмів для локалізації об’єктів, проте вони потребують потужних графічних процесорів і працюють із невеликою частотою кадрів, що робить їх малопридатними для систем реального часу [26].

Одноетапні моделі, зокрема YOLO, SSD та RetinaNet, демонструють більш збалансоване співвідношення швидкості та точності, що дозволяє ефективно використовувати їх у задачах моніторингу, аналізу відеопотоку та дистанційного

зондування. Трансформерні архітектури характеризуються високою здатністю до узагальнення та врахування довготривалих просторових залежностей, однак вони значно вимогливі до обчислювальних ресурсів і не завжди придатні для обробки відео в реальному часі [26].

Таблиця 1.1 — Порівняльна характеристика методів розпізнавання

Архітектура	Точність (mAP)	Швидкість (FPS)	Вимоги до ресурсів	Примітки та переваги
SIFT / SURF / HOG	Низька – Середня	Висока (від 50 до 100)	Низькі	Швидкі, прості, ручне проєктування ознак
Faster R-CNN / Cascade R-CNN	Висока (від 80 % до 90%)	Низька (від 5 до 10)	Високі GPU	Максимальна точність, повільна робота, не для реального часу
SSD / RetinaNet	Середня – Висока (від 70% до 85%)	Середня – Висока (від 30 до 60)	Середні	Баланс швидкості та точності, ефективні для відеопотоку
YOLOv8	Висока (80 % до 90%)	Висока (від 50 до 100)	Середні	Одноетапна детекція, швидка робота в реальному часі, підтримка multi-scale та edge computing
Vision Transformer (ViT)	Дуже висока (від 85 до 95%)	Низька (від 5 до 15)	Дуже високі	Висока здатність до узагальнення, ресурсоємні, не оптимальні для real-time

З огляду на зазначені характеристики, YOLOv8 обирається для реалізації системи розпізнавання об’єктів через поєднання високої швидкості обробки, значної точності та помірних вимог до обчислювальних ресурсів. Одноетапна архітектура YOLOv8 дозволяє ефективно вирішувати завдання детекції на зображеннях місцевості різного масштабу та варіативності, забезпечує роботу в реальному часі та інтегрується з edge-системами та мультиспектральними даними.

Крім того, YOLOv8 підтримує сучасні механізми оптимізації, включно з attention layers, multi-scale feature aggregation та покращеними loss-функціями, що

робить його оптимальним для практичних задач картографії, моніторингу змін на території та аерофотознімання [26].

YOLOv8 вирізняється оптимальним співвідношенням точності та продуктивності: модель досягає від 80% до 90% mAP та здатна працювати на швидкостях від 50 до 100 FPS, що забезпечує повноцінну обробку в реальному часі без суттєвих втрат у якості. Оптимізована архітектура (C2f, SPPF, anchor-free механізми) дозволяє досягати високої пропускної здатності при помірних вимогах до GPU. На противагу цьому, Vision Transformers (ViT) забезпечують найвищу точність завдяки глобальному механізму уваги, але мають низьку швидкодію (від 5 до 15 FPS) і дуже високі вимоги до обчислювальних ресурсів, що ускладнює їх застосування в реальних сценаріях моніторингу або потокової аналітики.

Вище наведено порівняльну характеристику основних методів розпізнавання об’єктів з огляду на їхню продуктивність, швидкодію та вимоги до обчислювальних ресурсів. Узагальнені дані подано у таблиці 1.1, що дозволяє наочно порівняти одноетапні та двоетапні підходи та визначити оптимальний метод для конкретних умов застосування [26].

Таким чином, наведене порівняння свідчить, що YOLOv8 є найбільш збалансованим рішенням, яке одночасно задовольняє вимоги до високої точності, стабільної роботи в реальному часі та помірної ресурсомісткості

2 ПРОЄКТУВАННЯ ІНТЕГРОВАНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ

2.1. Постановка задачі та вимоги до системи

Метою розробки є створення інтегрованої системи розпізнавання об'єктів, призначеної для автоматизованого аналізу аерофотознімків, супутникових даних та інших зображень місцевості. Така система повинна забезпечувати виявлення, класифікацію та візуалізацію об'єктів різних типів із високою точністю та швидкодією. Її використання є доцільним у галузях моніторингу територій, геоінформаційних системах, картографуванні, екологічному аналізі, а також у системах технічного або військового спостереження.

Основним функціональним призначенням системи є автоматичне розпізнавання об'єктів на зображеннях, що надходять із різних джерел, із подальшою інтеграцією результатів у зовнішні інформаційні або аналітичні сервіси. Програмний комплекс повинен забезпечувати точність розпізнавання не нижче заданого порогу, підтримувати обробку даних у режимі, близькому до реального часу, та працювати з різними форматами вхідних зображень, зокрема JPEG, PNG, TIFF і GeoTIFF [27].

Важливою вимогою до розробленої системи є забезпечення масштабованості при збільшенні обсягів вхідних даних. Це передбачає здатність системи зберігати стабільність і коректність функціонування за умов зростання інтенсивності потокових зображень, підключення додаткових джерел відеоінформації або розширення переліку об'єктів аналізу. Масштабованість системи досягається завдяки модульній архітектурі програмного забезпечення, використанню паралельної обробки даних та можливості адаптації до різних апаратних конфігурацій. Такий підхід дає змогу ефективно використовувати обчислювальні ресурси, мінімізувати затримки обробки та забезпечити перспективи подальшого розвитку системи без необхідності суттєвих змін її структурних компонентів.

Масштабованість передбачає можливість паралельної обробки даних на декількох обчислювальних вузлах, використання горизонтального масштабування серверів, оптимізованих черг повідомлень, а також підтримку контейнеризації (Docker, Kubernetes), що дозволяє гнучко розподіляти навантаження залежно від поточних потреб системи. Це особливо актуально для систем моніторингу, де кількість камер або дронів може динамічно зростати [26, 27].

Окрім цього, важливою є здатність системи інтегруватися з іншими інформаційними платформами, аналітичними модулями або зовнішніми сервісами. Для цього застосовуються стандартизовані протоколи обміну даними, серед яких найпоширенішим є REST API (Representational State Transfer Application Programming Interface). REST API — це архітектурний стиль взаємодії клієнта і сервера, який забезпечує простий, уніфікований та незалежний від конкретних технологій спосіб обміну даними. Взаємодія здійснюється за допомогою стандартних HTTP-методів (GET, POST, PUT, DELETE), а дані передаються у форматах JSON або XML, що гарантує сумісність із широким спектром систем [26, 28].

REST API дозволяє зовнішнім застосункам надсилати запити до модуля розпізнавання (наприклад, передавати зображення для обробки або отримувати результати детекції), отримувати структуровану відповідь та інтегрувати ці дані у власні аналітичні процеси, бази даних чи інформаційно-керуючі системи. Це робить систему відкритою та гнучкою, дозволяючи підключати нові сервіси без потреби змінювати внутрішню архітектуру. Завдяки REST API система може функціонувати як складова більших комплексних рішень, включно з геоінформаційними платформами, панелями моніторингу, мобільними застосунками та промисловими IoT-системами [27].

Серед нефункціональних вимог визначаються кросплатформність програмного забезпечення, оптимальне використання апаратних ресурсів та стабільна робота як на центральних процесорах, так і на графічних прискорювачах. Архітектура системи повинна мати модульну структуру, що

забезпечує можливість оновлення або заміни окремих компонентів без порушення загальної працездатності. Додатково система має гарантувати надійність, стійкість до збоїв під час обробки великих масивів даних і зручність використання завдяки наявності інтуїтивно зрозумілого інтерфейсу або вебдодатку для візуалізації результатів [27].

Таким чином, проєктована система покликана стати універсальним інструментом для розпізнавання об’єктів на зображеннях різного типу, який забезпечує високу точність, швидкодію, масштабованість і можливість інтеграції з іншими інформаційно-аналітичними рішеннями [28].

2.2. Архітектура системи розпізнавання об’єктів

Архітектура проєктованої системи розпізнавання об’єктів побудована за модульним принципом та має інтегрований характер, що забезпечує гнучкість, масштабованість і ефективну взаємодію компонентів. В основу покладено концепцію розділення функціональних блоків, що дозволяє ізолювати обробку даних, роботу нейронної мережі та візуалізацію результатів, а також забезпечує можливість модернізації окремих модулів без порушення загальної роботи системи.

Система розпізнавання об’єктів побудована за модульним принципом і складається з чотирьох основних підсистем: модуля попередньої обробки зображень (preprocessing), модуля інференсу нейронної мережі, модуля післяобробки результатів (postprocessing) та модуля візуалізації. Така структура забезпечує чіткий розподіл функцій між окремими компонентами, що спрощує супровід програмного забезпечення, підвищує його гнучкість і дозволяє здійснювати поетапну модернізацію або заміну окремих модулів без впливу на роботу системи загалом [28].

Модуль preprocessing відповідає за підготовку вхідних даних, включаючи очищення від шумів, корекцію освітлення, нормалізацію та масштабування зображень відповідно до вимог нейронної мережі. Цей модуль також забезпечує

конвертацію зображень у підтримувані формати (JPEG, PNG, TIFF, GeoTIFF) та підготовку пакетів даних для прискореного обчислення на GPU [28].

Модуль інференсу реалізує роботу обраної моделі глибокого навчання, YOLOv8, яка відповідає за детекцію та класифікацію об’єктів. Тут забезпечується висока швидкість обробки (до 100 кадрів на секунду) та можливість використання як на GPU, так і на edge-пристроях. Модуль обчислює координати об’єктів, ймовірності їх належності до певного класу та формує структуровані дані для подальшого аналізу [28].

Модуль postprocessing відповідає за оптимізацію результатів інференсу: фільтрацію помилкових спрацьовувань, об’єднання близько розташованих об’єктів, обчислення показників IoU (Intersection over Union) та підготовку даних для візуалізації. Крім того, цей модуль формує лог-файли та статистику для аналітичного контролю ефективності роботи системи [28].

Модуль візуалізації забезпечує відображення результатів роботи системи користувачу у вигляді графічного інтерфейсу або вебдодатку. Він дозволяє накладати розпізнані об’єкти на зображення, відображати інформацію про класи об’єктів, їхні координати та рівень впевненості моделі [28].

Для інтеграції з зовнішніми системами передбачено використання REST API, що дозволяє передавати дані у GIS-системи, аналітичні платформи або бази даних для подальшої обробки. Обмін даними між підсистемами здійснюється через внутрішні API та стандартизовані формати обміну, що забезпечує узгодженість, масштабованість і стійкість системи до великих обсягів даних [28].

Така архітектура, що зокрема представлена на рисунку 2.1, дозволяє ефективно комбінувати обчислювальні ресурси CPU та GPU, забезпечує роботу в реальному часі та дозволяє модернізувати окремі модулі без повного переписування системи. Архітектура YOLOv8 спирається на модифіковану версію CSPDarknet53, яка виконує роль основного каркасу нейронної мережі. Цей блок відповідає за ефективне витягування ознак з вхідного зображення, тобто виділення ключових деталей, таких як контури, текстури та форми об’єктів. Завдяки CSPDarknet53 мережа здатна на ранніх етапах обробки

отримувати достатньо інформації для точного розпізнавання об’єктів на різних рівнях складності.

У YOLOv8 використовується модуль C2f, який замінив попередній CSPLayer з YOLOv5. Новий модуль дозволяє краще організувати потік інформації всередині мережі, зменшуючи дублювання обчислень та покращуючи ефективність навчання. Завдяки C2f мережа здатна більш точно обробляти витягнуті ознаки, зберігаючи важливі деталі для подальшої класифікації та локалізації об’єктів.

Для прискорення обчислень та підвищення якості розпізнавання YOLOv8 застосовує шар SPPF (Spatial Pyramid Pooling Fast). Цей шар об’єднує ознаки з різних масштабів у єдину карту одного розміру, що дозволяє мережі одночасно враховувати як локальні, так і глобальні контексти об’єктів на зображенні. Крім того, стандартизація розмірів карт полегшує їх обробку наступними шарами мережі, що прискорює загальну роботу алгоритму [29].

Таким чином, поєднання компонентів CSPDarknet53, модуля C2f та шару SPPF формує оптимізовану архітектуру YOLOv8, яка забезпечує високу точність розпізнавання при раціональному використанні обчислювальних ресурсів. Базовий екстрактор ознак CSPDarknet53 виконує глибоке та багаторівневе виділення просторових і текстурних характеристик, модуль C2f підсилює ефективність обробки завдяки покращеному злиттю ознак та зменшенню надлишкових обчислень, а шар SPPF (Spatial Pyramid Pooling — Fast) агрегує інформацію на різних масштабах, що значно підвищує здатність мережі коректно детектувати об’єкти різних розмірів.

Узагальнена структура взаємодії зазначених компонентів подана на рисунку 2.1, де схематично відображено послідовність проходження вхідного зображення крізь основні блоки архітектури YOLOv8 — від первинного виділення ознак до формування багатомасштабних карт ознак, які використовуються для подальшої детекції об’єктів.

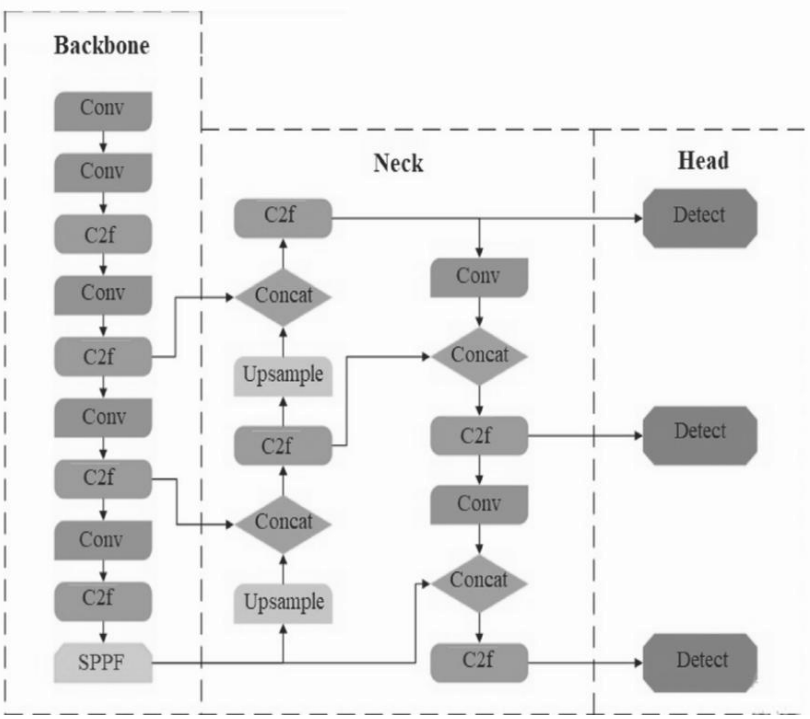


Рисунок 2.1 — Структура архітектури YOLOv8 з C2f та SPPF

2.3. Обґрунтування вибору моделі YOLOv8

Для реалізації системи розпізнавання об’єктів було обрано архітектуру YOLO (You Only Look Once), яка на сьогодні є однією з найбільш ефективних і популярних моделей згорткових нейронних мереж (CNN) для детекції та класифікації множини об’єктів на зображенні. Основною особливістю YOLO є те, що мережа застосовується до всього зображення одночасно, на відміну від інших підходів, таких як R-CNN або Fast R-CNN, де CNN обробляє окремі регіони зображення по черзі. Принцип роботи YOLO полягає у поділі вхідного зображення на сітку, кожна клітинка якої прогнозує координати bounding boxes та ймовірності належності об’єктів до певного класу [29].

Для реалізації системи розпізнавання об’єктів обрана архітектура YOLOv8 (You Only Look Once, версія 8). YOLO відноситься до архітектур нейронних мереж для детектування об’єктів на зображеннях, які працюють за принципом одноразового аналізу всього кадру, а не послідовного сканування регіонів. Це дозволяє забезпечити високу швидкість обробки (FPS, frames per second) при збереженні високої точності прогнозу (mAP — mean Average Precision).

YOLOv8 складається з трьох основних компонентів: Backbone, що відповідає за виділення ознак об’єктів зображення; Neck, який агрегує ознаки різних масштабів для підвищення точності детекції об’єктів різних розмірів; та Head, який генерує координати bounding boxes, класи об’єктів та confidence score для кожного прогнозу [29].

Принцип роботи YOLO полягає у розбитті зображення на сітку клітинок, кожна з яких прогнозує об’єкти, що потрапляють на її територію. Для кожної клітинки мережа передбачає координати центру та розмір bounding box, ймовірність належності об’єкта до певного класу та confidence score — загальну впевненість у наявності об’єкта. Формально прогноз bounding box для клітинки можна записати так:

$$B = (x, y, w, h, C) \tag{2.1}$$

де x, y— координати центру об’єкта у відносних одиницях сітки,

w, h— ширина та висота об’єкта,

C— confidence score.

Дане представлення дозволяє системі одночасно визначати положення об’єкта та його класифікацію. Для підвищення якості роботи моделі застосовуються додаткові методи попередньої обробки зображень. Зокрема, використовується Gaussian blur для зменшення високочастотного шуму, median filter для видалення імпульсного шуму, а також histogram equalization, що вирівнює яскравість зображення і покращує контраст об’єктів [29].

Після прогнозування проводиться постобробка, яка включає фільтрацію об’єктів за розміром та confidence score, що дозволяє відкинути помилкові або незначущі детекції. Вихідними даними системи є координати об’єктів, їх клас та confidence score, які можуть відображатися на зображенні у вигляді bounding boxes та підписів класів, забезпечуючи зручну інтерпретацію результатів.

Важливою складовою системи є інтеграція даних із зовнішніх джерел та

метаданих, таких як GPS-координати, час зйомки та умови освітлення, а також попередні дані про об’єкти [29].

Це дозволяє адаптувати роботу системи до різних сценаріїв, підвищуючи точність прогнозів і забезпечуючи контекстну фільтрацію. Наприклад, геолокаційні дані дозволяють відкидати об’єкти, що не відповідають очікуваному місцю появи, а часові та погодні параметри допомагають коригувати оцінку confidence score у різних умовах [29].

Таке поєднання нейронної мережі YOLOv8 із метаданими забезпечує більш точну детекцію та зручну інтеграцію результатів у прикладні програмні рішення, включаючи системи моніторингу, аналітики руху та картографічні сервіси.

Для наочності та оцінки ефективності роботи системи рекомендується використовувати таблиці з результатами тестування на різних наборах даних, де можна порівнювати FPS, mAP та використання пам’яті на різних апаратних платформах, таких як CPU та GPU. Це дозволяє здійснити обґрунтований вибір конфігурації системи для конкретного сценарію використання та забезпечує наукову повноту дослідження.

У таблиці 2.1 наведено FPS (кількість оброблених кадрів на секунду), mAP@0.5 (mean Average Precision з порогом IoU 0.5), використання оперативної пам’яті, а також позначку про врахування метаданих. Додання метаданих дозволяє підвищити точність детекції, оскільки система враховує контекстні дані, такі як GPS-координати, час та умови освітлення, що допомагає відкидати хибні детекції та підвищує впевненість прогнозів. Візуально дані представлено на рисунку 2.2.

Такий підхід дозволяє мережі враховувати контекст зображення повністю і уникати втрати інформації про взаємне розташування об’єктів, що позитивно впливає на точність розпізнавання. Використання YOLO забезпечує значно вищу швидкодію у порівнянні з іншими архітектурами детекції об’єктів. Так, YOLO обробляє кадри приблизно в 1000 разів швидше, ніж R-CNN, і близько в 100 разів швидше, ніж Fast R-CNN. Така продуктивність є критичною для задач реального

часу, особливо при запуску системи на мобільних або edge-пристроях для онлайн-обробки відеопотоку [29].

Таблиця 2.1 — Порівняльна характеристика основних методів розпізнавання

Набір даних	Платформа	FPS	mAP@0.5	Використання пам’яті	Використання метаданих	Коментарі
TestSet_1	CPU	12	0.72	2.1 GB	Ні	Базовий прогноз без контексту
TestSet_1	GPU	55	0.75	4.3 GB	Ні	Прискорена обробка, точність трохи вища
TestSet_1 + метадані	GPU	52	0.81	4.5 GB	Так	Враховано GPS, час зйомки та умови освітлення, точність підвищилась
TestSet_2	CPU	10	0.68	2.0 GB	Ні	Менш контрастні умови, базовий прогноз
TestSet_2 + метадані	GPU	50	0.79	4.4 GB	Так	Метадані допомагають коригувати confidence score, точність зросла

YOLO підтримує навчання на власних датасетах, має оптимізовану архітектуру (включно з модулями C2f та PAN, anchor-free підходами) і дозволяє ефективно використовувати обчислювальні ресурси GPU, що робить її найдоцільнішим вибором для інтегрованої системи розпізнавання об’єктів. Для реалізації інтегрованої системи розпізнавання об’єктів обрано архітектуру YOLOv8 (You Only Look Once, версія 8), яка є сучасною еволюцією сімейства YOLO та поєднує високу швидкість інференсу з точністю детекції. YOLOv8

належить до класу згорткових нейронних мереж (CNN) і призначена для одночасного розпізнавання множини об’єктів на зображенні [29].

Головна особливість цієї архітектури полягає в тому, що вона обробляє все зображення цілком за один прохід, на відміну від класичних підходів на кшталт R-CNN, Fast R-CNN чи SSD, де CNN застосовується до окремих регіонів або субзображень. YOLOv8 ділить зображення на сітку і для кожної клітинки прогнозує bounding boxes та ймовірності належності об’єктів до конкретних класів. Такий підхід дозволяє мережі враховувати контекст усієї сцени, що підвищує точність детекції та зменшує кількість помилкових спрацьовувань. Крім того, YOLOv8 має оптимізовану архітектуру з модулями C2f (Cross Stage Partial Fusion) та PAN (Path Aggregation Network), підтримує anchor-free детекцію і дозволяє адаптувати мережу під власні датасети без значного перепроектування моделі [29].

Порівняно з іншими підходами, YOLOv8 демонструє високу швидкодію та ефективність використання ресурсів. Наприклад, R-CNN обробляє кадри повільно через багаторазове застосування CNN до регіонів, Fast R-CNN прискорює процес, але все ще не підходить для реального часу, а SSD має нижчу точність на дрібних об’єктах. YOLOv8 дозволяє досягати швидкості інференсу понад 100 кадрів на секунду на сучасному GPU, що критично для онлайн-обробки відеопотоку навіть на мобільних пристроях або edge-пристроях.

Це робить модель YOLOv8 оптимальним вибором для інтегрованої системи, яка повинна працювати в реальному часі, обробляти зображення різних форматів і масштабуватись під збільшення обсягу даних. Нижче наведено порівняльну таблицю популярних алгоритмів детекції об’єктів із ключовими показниками продуктивності [29].

Окрім цього, архітектура YOLOv8 забезпечує гнучкість у використанні: мережу можна тренувати на власних зображеннях з різних джерел, вона ефективно використовує обчислювальні ресурси GPU та CPU, і підтримує інтеграцію з різними програмними платформами через стандартні API.

Таблиця 2.2 — порівняльна характеристика популярних алгоритмів детекції об’єктів із ключовими показниками продуктивності.

Модель	FPS (GPU)	Точність mAP	Ресурсоємність	Особливості
R-CNN	від 0.5до 1	від 70% до 75%	Високе	Повільний через обробку регіонів
Fast R-CNN	від 5до 7	від 75 до 78%	Високе	Прискорений, але все ще не real-time
SSD	від 22до 30	від 76% до 79%	Середнє	Швидший, точність нижча на дрібних об’єктах
YOLOv8	більше 100	від 80% до 90%	Оптимальне	Одноразовий прохід, підтримка edge/GPU, anchor-free

Таким чином, модель YOLOv8 обрана як базова модель для системи через високу швидкодію, можливість роботи на мобільних і edge-пристроях, підтримку навчання на власних датасетах, оптимізовану архітектуру та високу точність детекції об’єктів у реальному часі.

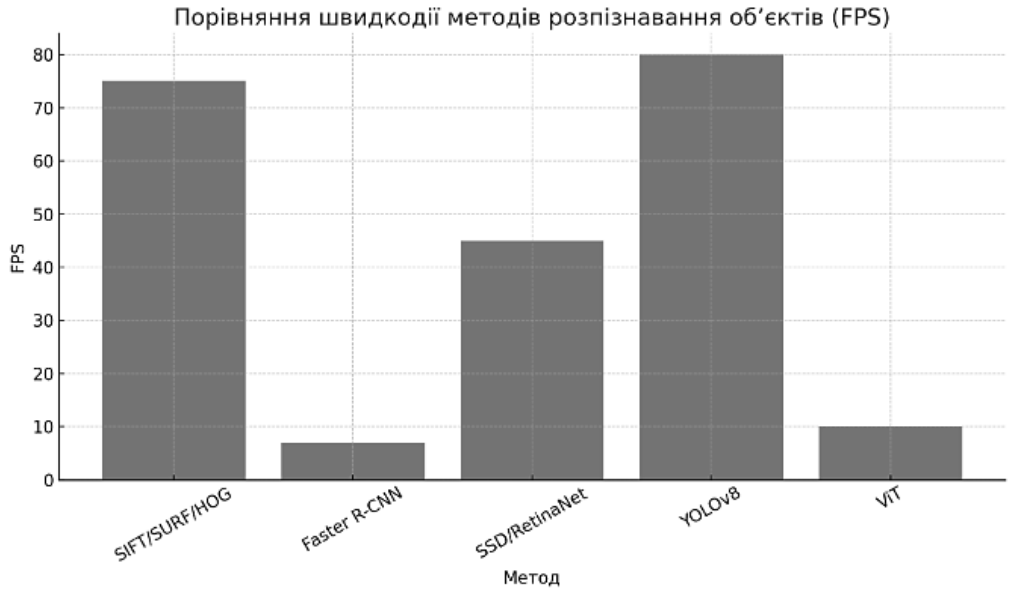


Рисунок 2.2 — Порівняльна діаграма швидкодії розпізнавання об’єктів (FPS/mAP)

2.4 Побудова та реалізація методів розпізнавання об’єктів

Розроблена система об’єктного детектування базується на сучасній одноетапній нейромережі YOLOv8 (You Only Look Once, версія 8), яка дозволяє

ефективно розпізнавати та локалізувати об’єкти на зображеннях у режимі реального часу. Архітектура моделі включає три основні компоненти: backbone, neck та head, що забезпечують послідовне виділення ознак, їх агрегацію та прогнозування координат і класів об’єктів [29].

Backbone реалізований на основі модифікованої версії CSPDarknet53 з використанням модуля C2f (Cross Stage Partial Fusion), що дозволяє зменшити обчислювальні витрати та покращити передачу ознак між шарами, що важливо для прискорення роботи системи на апаратних платформах з обмеженими ресурсами.

Neck представлений мережею PAN (Path Aggregation Network), яка забезпечує багаторівневу агрегацію ознак, поєднуючи локальні та глобальні характеристики об’єктів, що дозволяє моделі коректно розпізнавати як дрібні, так і великі об’єкти на зображенні [29].

Head реалізує anchor-free підхід до детекції, у якому кожен піксель карти ознак прогнозує координати об’єкта, його клас та confidence score, що спрощує навчання та підвищує точність прогнозування для об’єктів різних розмірів.Вхідне зображення попередньо обробляється шляхом масштабування до стандартного розміру, нормалізації пікселів та застосування методів data augmentation, таких як випадкове обертання, масштабування та зміна яскравості і контрасту для підвищення стійкості моделі. Нормалізація пікселів виконується за формулою:

$$I_{\text{norm}} = \frac{I-\mu}{\sigma} \tag{2.2}$$

де I— матриця пікселів зображення,

μ— середнє значення пікселів,

σ— стандартне відхилення.

Ця формула забезпечує приведення значень пікселів до єдиного діапазону, що прискорює та стабілізує процес навчання нейромережі.

Backbone обчислює карти ознак на кількох рівнях за допомогою згорток, які описуються формулою:

$$Fl+1=\sigma(W*Fl+b) \tag{2.3}$$

де F_l— вхідна карта ознак на шарі l,

W— ядро згортки, b— зсув,

$*$— операція згортки,

σ— функція активації (SiLU або Leaky ReLU).

Ця операція дозволяє моделі витягувати локальні ознаки зображення та передавати їх на наступні шари.

Агрегація ознак у neck здійснюється через поєднання top-down та bottom-up потоків за допомогою формули:

$$F_{\text{agg}} = \text{Conv}(\text{Concat}(F_{\text{top}}, F_{\text{bottom}})) \tag{2.4}$$

де F_{top}і F_{bottom}— карти ознак різних рівнів,

Concat — операція конкатенації,

Conv — згортковий шар.

Це дозволяє об’єднувати інформацію про об’єкти різного масштабу та покращує якість прогнозу [29].

Anchor-free head прогнозує координати об’єктів (x, y, w, h)та їх клас через softmax-функцію:

$$P(c_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}, \tag{2.5}$$

де z_i— логіт для класу i,

$P(c_i)$— ймовірність належності до класу.

Softmax перетворює необмежені значення логітів у ймовірності, сума яких дорівнює 1, що дозволяє коректно класифікувати об’єкти.

Для видалення накладених обмежувальних рамок застосовується алгоритм Non-Maximum Suppression (NMS), який базується на використанні коефіцієнта Intersection over Union (IoU):

$$IoU = \frac{|B_{pred} \cap B_{gt}|}{|B_{pred} \cup B_{gt}|}\ , \tag{2.6}$$

де B_{pred}— передбачена обмежувальна рамка,

B_{gt}— істинна (еталонна) обмежувальна рамка об’єкта,

$|\cdot|$— площа відповідної області.

Цей показник дозволяє оцінити перекриття між передбаченими та істинними об’єктами та залишати лише ті бокси, які мають максимальний confidence score для кожного класу.

Навчання моделі здійснюється із використанням комбінованої функції втрат:

Loss=λcoord∑(x−x^)2+λsize∑(w−w^)2+∑BCE(c,c^)+∑CE(class,class^), (2.7)

де λ_{coord}і λ_{size}— вагові коефіцієнти для координат та розмірів об’єкта,

BCE— бінарна крос-ентропія для confidence score,

CE— крос-ентропія для класифікації класів,

x, w, c,class— передбачені значення, $\hat{x}, \hat{w}, \hat{c}, \widehat{\text{class}}$— істинні значення.

Ця функція втрат одночасно мінімізує помилки координат, розмірів та невизначеність при класифікації, що забезпечує високу точність роботи системи. Для оптимізації застосовуються алгоритми Adam або SGD із learning rate scheduler та механізм early stopping для запобігання переобученню [29].

Додатково використовуються методи попередньої обробки зображень, зокрема Gaussian blur та median filter для зменшення шуму, histogram equalization для вирівнювання тонів та постобробка прогнозів із фільтрацією об’єктів за розміром та confidence score. Вихідними даними системи є координати об’єктів, їх клас та confidence score, які можуть відображатися на зображенні за допомогою bounding boxes та підписів класів, що забезпечує зручну інтерпретацію результатів та інтеграцію у прикладні програмні рішення [29].

У системі застосовуються методи попередньої обробки зображень, які значно підвищують якість детекції та стабільність роботи моделі. Зокрема, використовується Gaussian blur — розмиття зображення за допомогою гаусівського ядра. Цей метод дозволяє зменшити високочастотні шуми та дрібні деталі, що можуть заважати мережі коректно виділяти ознаки об’єктів. Розмиття виконується за формулою:

$$I_{\text{blur}}(x, y) = \sum_{i=-k}^{k} \quad \sum_{j=-k}^{k} \quad I(x + i, y + j) \cdot G(i, j), \tag{2.8}$$

де $I(x, y)$— значення пікселя у точці (x, y),

$G(i, j)$— значення гаусівського ядра,

k— радіус ядра.

Ця операція згладжує різкі переходи яскравості та зменшує шум, зберігаючи основні контури об’єктів. Ще одним методом є median filter (медіанний фільтр), який також використовується для шумоподавлення, але працює за іншим принципом — кожен піксель замінюється на медіану значень пікселів у його околі [30].

Це особливо ефективно для усунення імпульсного шуму типу salt-and-pepper noise, оскільки медіанна фільтрація зберігає чіткість контурів об’єктів і мінімізує втрату просторових деталей, що є важливим для подальшої коректної детекції. Для вирівнювання тонів та покращення контрасту застосовується histogram equalization (вирівнювання гістограми). Цей метод перерозподіляє

інтенсивності пікселів таким чином, щоб покращити видимість деталей у темних або світлих областях зображення. Гістограма яскравості перетворюється у рівномірний розподіл за формулою:

$$I_{\text{eq}} = \text{round}((L-1) \cdot \sum_{i=0}^{I} \frac{h(i)}{N}), \quad (2.9)$$

де L— кількість рівнів інтенсивності,

$h(i)$ — кількість пікселів з інтенсивністю i,

N— загальна кількість пікселів,

I— значення пікселя до перетворення.

Результатом є більш рівномірне освітлення та покращена видимість об’єктів на зображенні. Після прогнозу об’єктів використовується постобробка результатів, яка включає фільтрацію об’єктів за розміром та confidence score. Об’єкти, які не відповідають мінімальним розмірам або мають низьку ймовірність (confidence score), відкидаються. Це дозволяє уникнути помилкових детекцій та зменшити навантаження на подальші алгоритми аналізу [30].

Вихідними даними системи є координати об’єктів, їх клас та confidence score. Для зручності користувача та інтеграції у прикладні програмні рішення результати відображаються на зображенні за допомогою bounding boxes (рамок навколо об’єктів) та підписів класів, що забезпечує швидку та наочну інтерпретацію результатів роботи моделі.

2.5 Оцінювання точності детекції, класифікації та функції втрат у моделях розпізнавання об’єктів

Оцінювання точності детекції в сучасних нейронних мережах ґрунтується на комплексному підході, що поєднує оцінку наявності об’єкта, точність його локалізації та правильність класифікації. Ключовим показником є confidence score, який відображає рівень упевненості моделі в тому, що прогнозована рамка

дійсно містить об’єкт. Confidence score обчислюється як добуток ймовірності наявності об’єкта та показника перетину рамок Intersection over Union (IoU):

$$\text{Confidence} = P(\text{object}) \cdot \text{IoU}(B_{\text{pred}}, B_{\text{gt}}), \tag{2.10}$$

де $P(\text{object})$— прогнозована моделлю ймовірність наявності об’єкта в межах рамки;

$\text{IoU}(B_{\text{pred}}, B_{\text{gt}})$— показник перетину площ прогнозованої рамки B_{pred}та еталонної рамки B_{gt}.

Такий підхід дозволяє одночасно враховувати сам факт виявлення об’єкта і точність його просторової локалізації, що є критично важливим для аналізу складних сцен з неоднорідним фоном, змінним освітленням, різними ракурсами зйомки та частковим перекриттям об’єктів. Після встановлення факту наявності об’єкта модель виконує визначення його класу. Для цього формується вектор ймовірностей належності об’єкта до всіх можливих класів:

$$\text{P} = [p_1, p_2, \dots, p_K], \tag{2.11}$$

де p_i— ймовірність належності об’єкта до i-го класу;

K— загальна кількість класів у задачі класифікації.

У класичних алгоритмах розпізнавання образів ці ймовірності нормалізуються за допомогою softmax-функції:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{2.12}$$

де z_i— необроблене вихідне значення (логіт) нейронної мережі для i-го класу.

Використання функції softmax у класичних моделях класифікації забезпечує формування коректного і нормалізованого розподілу ймовірностей між можливими класами.

Водночас у моделі YOLOv8 застосовується підхід незалежного сигмоїдного оцінювання кожного класу, що дозволяє розглядати ймовірності належності до класів як незалежні величини. Такий підхід знижує взаємний вплив між класами, підвищує робастність детектора та забезпечує більш стабільну роботу в умовах неоднозначних сцен, часткових перекриттів об’єктів і складного фону.

Оскільки під час інференсу нейронна мережа формує значну кількість прогнозованих обмежувальних рамок для кожного кадру, виникає необхідність усунення дублювання та надлишкових детекцій, що досягається шляхом застосування процедур постобробки результатів.

Для цього застосовується механізм Non-Maximum Suppression (NMS), який базується на порівнянні значень IoU:

$$\mathrm{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{2.13}$$

де $|A \cap B|$ — площа перетину двох рамок Aта B;

$|A \cup B|$ — площа їх об’єднання.

Алгоритм NMS зберігає рамку з найбільшим значенням confidence score та відкидає всі інші рамки, які мають з нею значне перекриття, формуючи оптимальний набір детекцій без хибних накладень.

Висока точність детекції забезпечується також завдяки використанню спеціалізованих функцій втрат у процесі навчання моделі. У YOLOv8 застосовано комбінацію втрати локалізації, втрати класифікації та втрати наявності об’єкта. Основною функцією для оцінки точності локалізації є CIoU Loss:

$$L_{\mathrm{CIoU}} = 1 - \mathrm{CIoU}, \tag{2.14}$$

де

$$\mathrm{CIoU} = \mathrm{IoU} - \frac{\rho^2(b,b^{gt})}{c^2} - \alpha v, \tag{2.15}$$

де $\rho(b, b^{gt})$— евклідова відстань між центрами прогнозованої та еталонної рамок;

c— довжина діагоналі найменшої рамки, що охоплює обидві рамки;

v— показник різниці у співвідношенні сторін рамок;

α— ваговий коефіцієнт, який масштабує внесок параметра v.

Для оцінки коректності класифікації використовується функція бінарної крос-ентропії:

$$L_{\text{BCE}} = -[y\ln(p) + (1-y)\ln(1-p)], \tag{2.16}$$

де y— істинна мітка класу;

p— прогнозована ймовірність належності до цього класу.

Додатково застосовується функція втрат для оцінки факту наявності об’єкта:

$$L_{\text{obj}} = -[y_{\text{obj}}\ln(p_{\text{obj}}) + (1-y_{\text{obj}})\ln(1-p_{\text{obj}})], \tag{2.17}$$

де y_{obj}— істинна мітка наявності об’єкта;

p_{obj}— прогнозована ймовірність наявності об’єкта в рамці.

Сумарна функція втрат має вигляд:

$$L_{\text{total}} = \lambda_1 L_{\text{CIoU}} + \lambda_2 L_{\text{BCE-class}} + \lambda_3 L_{\text{obj}}, \tag{2.18}$$

де $\lambda_1, \lambda_2, \lambda_3$— коефіцієнти вагового внеску відповідних компонентів [30].

Такий підхід забезпечує збалансоване навчання моделі, у межах якого одночасно підвищується точність локалізації, правильність класифікації та здатність відрізняти об’єкти від фону. У сукупності це дозволяє системам на кшталт YOLOv8 ефективно працювати в умовах реального часу, зберігаючи високу точність навіть у складних сценах [30].

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ

3.1. Опис програмного середовища та використаних технологій

Для розробки системи розпізнавання об'єктів було обрано сучасне програмне середовище та набір спеціалізованих інструментів, які забезпечують високу ефективність обчислень, підтримку апаратного прискорення та гнучкість при проєктуванні й використанні нейронних мереж. Застосування таких засобів дозволяє реалізувати повний цикл обробки даних — від попередньої обробки зображень до інференсу та аналізу результатів — із мінімальними часовими витратами та можливістю масштабування системи під різні обчислювальні платформи.

Основною мовою програмування системи є Python — мова високого рівня, яка широко використовується у сфері машинного навчання та комп'ютерного зору завдяки наявності великої кількості спеціалізованих бібліотек, простоті синтаксису та активному співтовариству розробників. Для розробки та тестування моделі використовувалася версія Python 3.11, що забезпечує сумісність з останніми релізами фреймворків.

Як основний фреймворк для побудови та навчання нейронної мережі було обрано PyTorch. PyTorch забезпечує динамічне обчислювальне графове середовище, що дозволяє легко реалізовувати складні архітектури нейронних мереж та ефективно проводити їхнє навчання на графічних процесорах (GPU) [31].

Висока продуктивність PyTorch досягається завдяки оптимізації під CUDA, що дозволяє паралельно обробляти великі обсяги даних, а також використанню бібліотеки TorchVision для роботи з зображеннями та попередньої обробки даних. Для обробки зображень та їхньої підготовки до навчання застосовувалася бібліотека OpenCV, яка надає широкий спектр алгоритмів для фільтрації шуму, масштабування, трансформації, а також реалізації базових

методів комп’ютерного зору, таких як виявлення контурів, морфологічні операції та обчислення гістограм [31].

OpenCV дозволяє ефективно реалізувати етап попередньої обробки зображень, що підвищує точність моделі на етапі навчання. Для роботи з моделями серії YOLO було використано Ultralytics YOLOv8, який є сучасним і оптимізованим фреймворком для реалізації швидкого та точного розпізнавання об’єктів. Ultralytics забезпечує готові модулі для навчання, донавчання та тестування моделей, а також інтеграцію з популярними бібліотеками Python, що спрощує процес розробки та дозволяє отримувати результати у зручному для аналізу форматі [31].

Для роботи з даними та підготовки датасетів були використані бібліотеки NumPy та Pandas, які дозволяють ефективно оперувати масивами даних та табличними структурами, здійснювати нормалізацію, масштабування та агрегацію інформації. Для візуалізації результатів застосовувалися Matplotlib та Seaborn, що дозволяє будувати графіки точності, швидкості обробки та інших показників продуктивності системи [31].

Середовище розробки було організоване у вигляді інтегрованої системи на базі PyCharm, що дозволяє проводити відладку, тестування та документування коду в одному середовищі, забезпечуючи контроль версій через інтеграцію з Git. Для прискорення обчислень застосовувалася апаратна платформа з графічним процесором NVIDIA, що забезпечує роботу з великими обсягами даних та дозволяє оптимізувати процес навчання моделі [31].

3.2. Архітектура системи та структура коду

Розроблена система розпізнавання об’єктів побудована за модульним програмно-архітектурним принципом, що забезпечує її гнучкість, масштабованість та зручність супроводу на всіх етапах життєвого циклу. Такий підхід дозволяє ізолювати функціональні блоки, мінімізувати взаємні залежності між компонентами та здійснювати подальше розширення системи без суттєвої

перебудови її ядра. Архітектура системи реалізована у вигляді набору логічних і програмних підсистем, об’єднаних єдиним потоком обробки даних [32].

Підсистема введення даних відповідає за отримання та уніфікацію вхідної інформації. Вона підтримує роботу зі статичними зображеннями, відеофайлами та потоковим відео в режимі реального часу. На рівні програмної реалізації цей компонент виконує завантаження даних з файлової системи, відеокамер або зовнішніх потокових джерел, а також здійснює зчитування супутніх метаданих, зокрема часових позначок, географічних координат (GPS), параметрів камер та інших сенсорних характеристик. У коді ці функції інкапсульовані в окремому модулі завантаження даних, що забезпечує первинну валідацію вхідної інформації та її приведення до уніфікованого формату [31,32].

Модуль попередньої обробки зображень реалізує набір алгоритмів, спрямованих на покращення якості вхідних даних перед поданням їх до нейронної мережі. У рамках цього модуля застосовуються фільтрація шуму за допомогою Gaussian blur та median filter, корекція контрастності та освітлення із використанням histogram equalization, а також операції масштабування й нормалізації піксельних значень [31,32].

Програмно ці алгоритми реалізовані з використанням бібліотек обробки зображень та виконуються у вигляді послідовного конвеєра. Такий підхід дозволяє зменшити вплив шумів, компенсувати зміну умов освітлення та забезпечити стабільність прогнозів моделі незалежно від зовнішніх факторів.

Ядро системи становить нейронна мережа YOLOv8, інтегрована як окремий програмний компонент. Модель завантажується разом із попередньо навченими вагами або може бути додатково донавчена на спеціалізованому датасеті.

У процесі роботи модель отримує підготовлені зображення та виконує інференс, у результаті якого формується набір прогнозів, що включає координати обмежувальних рамок (bounding boxes), ідентифікатор класу об’єкта та значення confidence score, що характеризує ймовірність коректності детекції.

Реалізація інференсу оптимізована для використання GPU, що дозволяє забезпечити обробку кадрів у режимі, близькому до реального часу [32].

Результати роботи нейронної мережі передаються до підсистеми постобробки, яка виконує додаткову фільтрацію та структуризацію прогнозів. На цьому етапі реалізується відсікання детекцій за порогом confidence score, фільтрація за геометричними параметрами рамок та застосування алгоритму Non-Maximum Suppression для усунення дублювання. Програмно цей етап організований як окремий модуль, що дозволяє змінювати правила фільтрації без втручання в логіку роботи нейронної мережі. Вихідні результати формуються у структурованому вигляді та можуть бути представлені у форматі JSON, таблиць або безпосередньо накладені на зображення [32].

Інтерфейсна частина системи забезпечує взаємодію з користувачем і реалізована у вигляді окремого модуля. В залежності від сценарію використання вона може працювати у консольному режимі, у вигляді графічного інтерфейсу або через API. Користувач має можливість завантажувати дані, запускати процес детекції, налаштовувати порогові значення та отримувати візуалізовані результати у зручному форматі. Такий підхід забезпечує незалежність логіки обробки від способу представлення результатів [32].

Структура програмного коду організована у вигляді окремих модулів із чітким функціональним призначенням. Модуль завантаження даних відповідає за введення та попередню обробку, модуль моделі містить визначення архітектури YOLOv8 та процедури інференсу й навчання, модуль постобробки реалізує фільтрацію та підготовку результатів, а інтерфейсний модуль забезпечує взаємодію з користувачем. Така організація значно спрощує тестування окремих компонентів, дозволяє повторно використовувати код та полегшує масштабування системи для роботи з великими обсягами зображень або відеоданих.

Додатково система передбачає можливість інтеграції зовнішніх джерел інформації, включаючи дані від сенсорів або сторонніх баз даних. Це дозволяє поєднувати результати розпізнавання з часовими характеристиками та

просторовими координатами, аналізувати динаміку об’єктів у відеопотоках та виконувати кореляцію з іншими інформаційними системами. Завдяки цьому система набуває комплексного характеру та може використовуватися у прикладних задачах відеомоніторингу, транспортної аналітики, аналізу міської інфраструктури та автоматизованої обробки даних дистанційного зондування [32].

3.3 Підсистема введення та попередньої обробки даних

Підсистема введення та попередньої обробки даних є ключовим елементом розробленої системи розпізнавання об’єктів, оскільки саме вона забезпечує перетворення вхідної інформації у формат, придатний для подальшої обробки моделлю YOLOv8. Підсистема відповідає за інтеграцію різних джерел даних (статичні зображення, відеопотоки, набори кадрів), а також за використання додаткових метаданих (часові мітки, GPS-координати, параметри камер). Це дозволяє враховувати контекст зйомки та забезпечувати стабільність роботи системи за змінних умов освітлення й перспективи [32].

На першому етапі вхідні дані проходять попередню обробку, яка включає усунення шумів, нормалізацію та підвищення контрастності зображення. Для зменшення випадкових артефактів застосовуються згорткові фільтри, зокрема згладжування за гаусовим ядром (Gaussian blur) та медіанна фільтрація (median filter). Формально, операція згладжування зображення за гаусовим ядром описується як згортка:

$$I_{\text{blur}}(x, y) = \sum_{i=-k}^{k} \quad \sum_{j=-k}^{k} \quad I(x + i, y + j)\, G_{\sigma}(i, j), \tag{3.1}$$

де $I(x, y)$— інтенсивність вихідного зображення у точці (x, y);

$I_{\text{blur}}(x, y)$— інтенсивність згладженого зображення;

$G_{\sigma}(i, j)$— значення гаусової функції з параметром σу точці (i, j);

k— половина розміру ядра фільтра.

Застосування median filter передбачає заміну значення кожного пікселя медіаною значень у його локальному neighbourhood, що дозволяє ефективно пригнічувати імпульсний шум без помітного спотворення контурів об’єктів [32].

Для підвищення контрастності використовується метод вирівнювання гістограми (histogram equalization), який перерозподіляє яскравість зображення таким чином, щоб більш рівномірно заповнити доступний діапазон інтенсивностей. Перетворення інтенсивності пікселя $I(x, y)$може бути записане у вигляді:

$$I_{\text{eq}}(x, y) = \text{round}\left((L - 1) \cdot \text{CDF}(I(x, y))\right), \tag{3.2}$$

де $I_{\text{eq}}(x, y)$— нове значення інтенсивності;

L— кількість можливих рівнів яскравості;

CDF— кумулятивна функція розподілу яскравості для даного зображення;

round $(\cdot)$— операція округлення до найближчого цілого.

Реалізацію функції попередньої обробки на мові Python подано у лістингу 3.1.

Лістинг 3.1 — Функція попередньої обробки зображення

```
import cv2
import numpy as np
def preprocess_image(image: np.ndarray) -> np.ndarray:
    :param image: вхідне зображення у форматі BGR (np.ndarray).
    :return: оброблене зображення у форматі BGR (np.ndarray).
    if image is None or image.size == 0:
        raise ValueError("preprocess_image: порожнє або некоректне зображення")
    # Зменшення високочастотних шумів
    blur = cv2.GaussianBlur(image, (5, 5), 0)
    # Усунення імпульсного шуму
    median = cv2.medianBlur(blur, 5)
    img_yuv = cv2.cvtColor(median, cv2.COLOR_BGR2YUV)
    # Histogram equalization лише для Y-каналу (яскравість)
    img_yuv[:, :, 0] = cv2.equalizeHist(img_yuv[:, :, 0])
    processed = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
    return processed
```

Після попередньої обробки дані необхідно перетворити у формат, придатний для нейронної мережі, що включає масштабування до стандартного розміру, нормалізацію інтенсивностей у діапазоні [0; 1]та перетворення у тензор. У випадку YOLOv8 зазвичай використовується квадратне зображення (наприклад, 640×640), однак конкретний розмір може бути налаштований.

Перетворення зображення у тензор, сумісний із бібліотекою PyTorch, наведено у лістингу 3.2.

Лістинг 3.2 — Перетворення зображення у тензор для моделі YOLOv8

```
import torch
import numpy as np
def to_tensor(image: np.ndarray, device: str = "cpu") -> torch.Tensor:
    :param image: вхідне зображення у форматі BGR (np.ndarray).
    :param device: цільовий пристрій ("cpu" або "cuda").
    :return: тензор розмірності (1, C, H, W) на вказаному пристрої.
    if image is None or image.size == 0:
        raise ValueError("to_tensor: порожнє або некоректне зображення")
    # BGR -> RGB
    image_rgb = image[:, :, ::-1]
    tensor = torch.from_numpy(image_rgb).permute(2, 0, 1).float() / 255.0
    tensor = tensor.unsqueeze(0)  # додатковий вимір для батчу
    return tensor.to(device)
```

Таким чином, підсистема введення та попередньої обробки даних відіграє ключову роль у забезпеченні коректного функціонування всієї системи розпізнавання об’єктів. Вона формує стабільний, уніфікований та інформаційно насичений вхідний потік для моделі YOLOv8, що є необхідною умовою для досягнення високої точності детекції.

Застосування методів пригнічення шумів дозволяє зменшити вплив випадкових спотворень і артефактів, нормалізація контрасту забезпечує стійкість роботи моделі за різних умов освітлення, а уніфікація формату та розмірів зображень гарантує узгодженість вхідних даних з архітектурними вимогами нейронної мережі. У сукупності це підвищує надійність прогнозів, зменшує кількість хибних спрацьовувань і забезпечує стабільну роботу системи під час

обробки як статичних зображень, так і відеопотоків у реальних умовах експлуатації.

3.3 Підсистема інференсу та постобробки результатів

Підсистема інференсу та постобробки відповідає за безпосередній виклик моделі YOLOv8, обробку прогнозів та підготовку результатів до подальшого використання (візуалізація, збереження, передача через API тощо). На вході вона отримує попередньо оброблене зображення у вигляді тензора, на виході — структурований список детекцій, який містить координати межових прямокутників, класи об’єктів та значення confidence score.

У загальному випадку фільтрація детекцій за порогом довірчої міри може бути описана як:

$$s_i > \theta, \tag{3.3}$$

де s_i— значення confidence score для i-го об’єкта;

θ— заздалегідь встановлений поріг, нижче якого детекція вважається недостовірною.

Для забезпечення повного циклу обробки вхідних зображень у розробленій системі реалізовано функцію інференсу, яка відповідає за проходження даних через навчений екземпляр нейронної мережі YOLOv8, первинну фільтрацію результатів та підготовку детекцій до подальшої постобробки.

Дана функція є центральним елементом програмної реалізації, оскільки саме на цьому етапі формується набір прогнозованих об’єктів із відповідними координатами обмежувальних прямокутників, класами та значеннями впевненості.

Реалізація функції інференсу з використанням бібліотеки Ultralytics YOLOv8, що включає завантаження моделі, обробку тензорів та фільтрацію детекцій за порогом confidence score, наведена у лістингу 3.3.

Лістинг 3.3 — Функція інференсу та первинної постобробки детекцій

```
from typing import List, Dict, Any
import torch
def run_inference(
    model: torch.nn.Module,
    input_tensor: torch.Tensor,
    conf_threshold: float = 0.5
) -> List[Dict[str, Any]]:
    :param model
    :param input_tensor
    :param conf_threshold:
    :return
    model.eval()
    with torch.no_grad():
        raw_predictions = model(input_tensor)[0]
    detections: List[Dict[str, Any]] = []
    # [x1, y1, x2, y2, conf, class_id]
    for pred in raw_predictions:
        x1, y1, x2, y2, conf, class_id = pred.tolist()
        if conf < conf_threshold:
            continue
        detections.append(
            {
                "bbox": [x1, y1, x2, y2],
                "confidence": float(conf),
                "class_id": int(class_id),})
    return detections
```

У більш складних сценаріях додатково застосовується Non-Maximum Suppression (NMS) для усунення дублювальних рамок із високим ступенем перетину, але в більшості сучасних реалізацій YOLOv8 NMS інтегровано всередині самої моделі або у виклику високорівневого інтерфейсу.

Важливим етапом є інтеграція детекцій із метаданими, отриманими від зовнішніх сенсорів або інформаційних систем. Це можуть бути часові позначки, GPS-координати або службова інформація про джерело кадру. Така функція наведена у лістингу 3.4.

Лістинг 3.4 — Інтеграція детекцій із метаданими

```
from typing import List, Dict, Any, Optional
def integrate_metadata(
```

```
    detections: List[Dict[str, Any]],
    frame_metadata: Optional[Dict[str, Any]] = None
) -> List[Dict[str, Any]]:
    :param detections: список детекцій (результат run_inference).
    :param frame_metadata: словник метаданих кадру
                        (наприклад: {"timestamp": ..., "gps": ...}).
    :return: оновлений список детекцій із доданими метаданими
    if frame_metadata is None:
        return detections
    timestamp = frame_metadata.get("timestamp")
    gps = frame_metadata.get("gps")
    for det in detections:
        if timestamp is not None:
            det["timestamp"] = timestamp
        if gps is not None:
            det["gps"] = gps
    return detections
```

Такий підхід є значно реалістичнішим за варіант, де використовується індексований доступ metadata[i] до кожної детекції, оскільки у реальній системі метадані, як правило, стосуються всього кадру, а не кожної рамки окремо. При потребі може бути реалізована більш складна логіка прив’язки (наприклад, синхронізація кількох джерел даних за часом або просторовими координатами) [32].

З погляду загальної архітектури, результати підсистеми інференсу та постобробки можуть використовуватися як для візуалізації (накладання bounding boxes на зображення та відображення класів із confidence score), так і для подальшої аналітики, збереження в базі даних або передачі через REST API до інших модулів системи. Підсистема побудована таким чином, що її можна масштабувати для роботи з великими потоками даних, без необхідності рушити інші компоненти, що відповідають за введення, збереження чи інтерфейс із користувачем.

3.4. Ядро нейронної мережі та алгоритми розпізнавання

Ядро системи розпізнавання об’єктів реалізовано з використанням нейронної мережі YOLOv8, яка є сучасною розвитковою версією сімейства

алгоритмів You Only Look Once (YOLO). Основною особливістю цієї архітектури є одноетапний принцип детекції, за якого задача локалізації об’єктів і визначення їхньої класової належності вирішується за один прохід зображення через нейронну мережу. Такий підхід дозволяє досягати високої швидкодії при збереженні належного рівня точності, що є критичним для задач обробки відеопотоків у режимі реального часу [33].

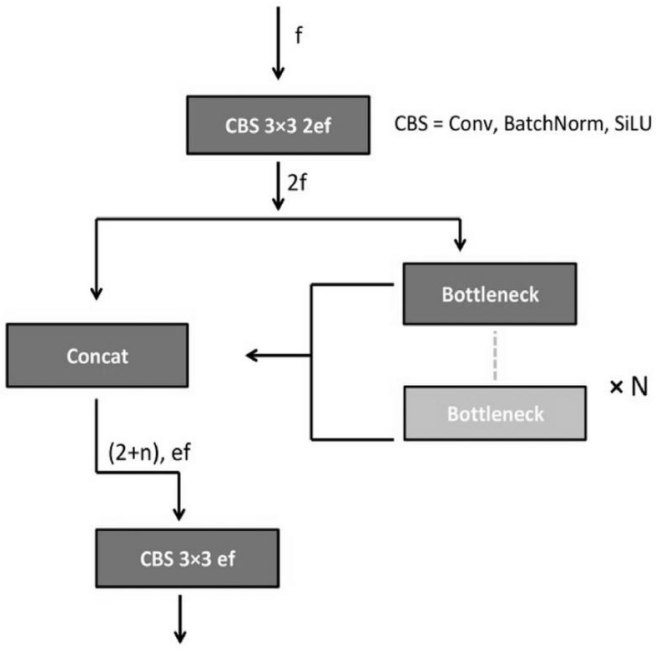


Рисунок 3.1 — Структура архітектури YOLOv8 з C2f та SPPF

Архітектура YOLOv8 базується на модифікованому згортковому каркасі CSPDarknet53, який доповнений блоками типу C2f. Використання цих блоків забезпечує ефективну передачу ознак між шарами мережі, зменшення інформаційних втрат та оптимізацію обчислювальних витрат. На початкових рівнях мережі відбувається виділення базових ознак зображення, таких як контури, текстури та локальні структури, тоді як на глибших рівнях формується узагальнене представлення об’єктів сцени [32, 33].

Для роботи з об’єктами різного масштабу YOLOv8 використовує багаторівневе агрегування ознак на основі Feature Pyramid Network (FPN) та Path Aggregation Network (PAN). Це дозволяє ефективно поєднувати інформацію з

різних рівнів глибини мережі та підвищувати якість детекції як великих, так і малих об’єктів. Загальна структура архітектури YOLOv8 із використанням блоків C2f та шару просторового пірамідального пулінгу SPPF наведена на рисунку 3.1.

Далі ознаки обробляються шляхом багаторівневого об’єднання в блоках Feature Pyramid Network (FPN) і Path Aggregation Network (PAN), що дозволяє моделі працювати з об’єктами різного масштабу. Модель YOLOv8 прогнозує положення об’єктів у форматі чотирьох координат: (x_center, y_center, width, height), де x_center і y_center визначають центр об’єкта, а width і height — його ширину та висоту.

Для оцінки точності детекції використовується confidence score, що обчислюється як ймовірність наявності об’єкта в межах прогнозованого прямокутника. Визначення класу об’єкта відбувається через багатокласову категоризацію, де для кожного прогнозованого прямокутника генерується вектор ймовірностей належності до класів.

Для інтеграції додаткових даних, таких як метадані камер або сенсорів, система використовує механізми умовного підключення: часові позначки, координати GPS та інші параметри можуть бути включені у процес прийняття рішення, наприклад для відсіювання об’єктів, які з’являються у позаочікуваних зонах або часі. Це дозволяє підвищити точність детекції та зменшити кількість хибних спрацьовувань у реальному середовищі.

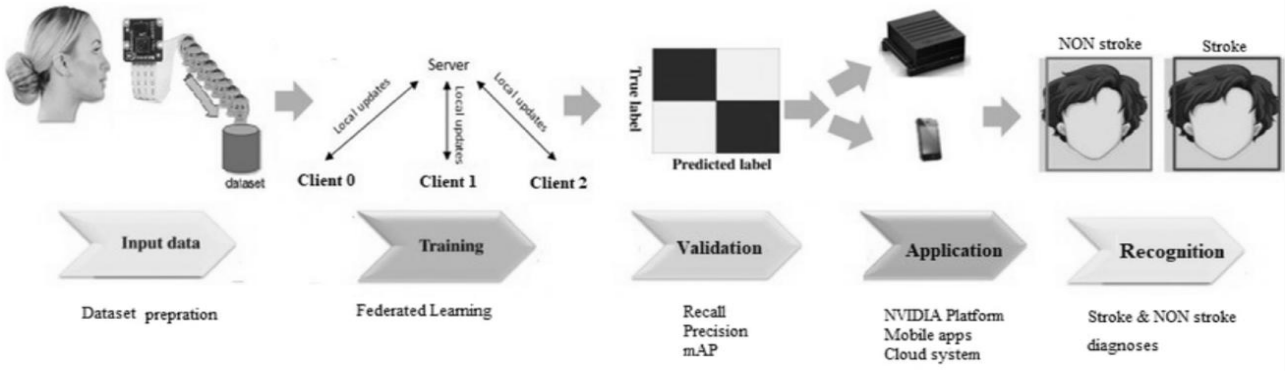


Рисунок 3.2 — Структура архітектури YOLOv8 з C2f та SPPF

Модель YOLOv8 відноситься до класу одноетапних детекторів (single-stage detectors), що дозволяє одночасно здійснювати локалізацію та класифікацію об’єктів на зображенні. Перевагою такого підходу є висока продуктивність та можливість обробки відеопотоків у режимі реального часу [33].

Вихід моделі представлений у вигляді координат bounding boxes, класу об’єкта та confidence score, що визначає ймовірність правильного розпізнавання. На етапі попередньої обробки вхідні дані проходять серію процедур, спрямованих на підвищення якості зображень та зменшення шумів.

Застосовуються алгоритми Gaussian blur, що розмиває зображення за гаусовим ядром для усунення високочастотних шумів, та median filter, який ефективно видаляє імпульсний шум. Для покращення контрастності зображень використовується метод histogram equalization, що вирівнює розподіл яскравості пікселів і забезпечує стабільність прогнозів моделі при різних умовах освітлення [33].

Після цих процедур дані нормалізуються, масштабуються до стандартного розміру та перетворюються у багатовимірні тензори, що дозволяє подавати їх на вхід Вихід нейронної мережі YOLOv8 формується у вигляді тензора розмірності $[N, 6]$, де Nвизначає кількість об’єктів, виявлених на зображенні. Кожен рядок цього тензора містить інформацію про одну детекцію та включає координати обмежувальної рамки у форматі $x_{\min}, y_{\min}, x_{\max}, y_{\max}$, а також показник достовірності s(confidence score) і код класу c, який відповідає ідентифікованому типу об’єкта.

Така структура вихідних даних забезпечує компактне та зручне подання результатів детекції, що спрощує подальші етапи постобробки, фільтрації та інтеграції результатів у прикладні системи аналізу.

Обчислення перекриття здійснюється за формулою Intersection over Union (IoU):

$$\text{IoU} = \frac{A \cap B}{A \cup B} \tag{3.4}$$

де A та B— площі двох прямокутників.

Якщо значення IoU перевищує заданий поріг, залишаються лише ті детекції, які мають більший confidence score, що забезпечує коректну обробку накладених об’єктів та зменшення помилок класифікації. Для підвищення точності та контекстності прогнозів система використовує додаткові джерела даних. До них належать часові позначки, географічні координати, параметри камер та інші метадані [33].

Інтеграція таких даних дозволяє здійснювати кореляцію прогнозів із зовнішніми джерелами інформації, а також проводити аналіз динаміки об’єктів у відеопотоках. Наприклад, на основі часових міток та координат можна розраховувати швидкість руху об’єктів за формулою:

$$v = \frac{\|P_t - P_{t-1}\|}{\Delta t}, \tag{3.5}$$

де P_{t-1}та P_t— координати об’єкта у попередньому та поточному кадрі відповідно

Δt— проміжок часу між кадрами.

Програмна реалізація системи організована за модульним принципом. Модуль data_loader відповідає за введення та попередню обробку даних, модуль model містить опис архітектури YOLOv8 і процедури інференсу, модуль post_processing реалізує алгоритми фільтрації та NMS, а модуль interface забезпечує взаємодію з користувачем і зовнішніми інформаційними системами.

Для оцінювання ефективності розробленої системи було проведено експериментальне тестування моделі YOLOv8 на стандартних наборах зображень та відеопотоків. Основними метриками продуктивності обрано середню швидкість обробки кадрів (FPS — frames per second), точність детекції, оцінену за показником $mAP@0.5$, а також обсяг використаної оперативної

пам’яті. Результати тестування на різних апаратних платформах наведено у таблиці 3.1.

Таблиця 3.1 — Результати тестування моделі YOLOv8 на різних апаратних конфігураціях

Параметр	CPU (Intel i7)	GPU (NVIDIA RTX 3060)	GPU (NVIDIA RTX 4090)
FPS (frames per second)	12	45	120
mAP@0.5 (%)	87.3	87.3	87.3
Використання пам’яті (MB)	3200	4100	5600
Розмір оброблюваного кадру	640×640	640×640	640×640
Час передобробки кадру (мс)	18	6	3

Аналіз отриманих результатів свідчить, що використання графічних прискорювачів суттєво підвищує продуктивність системи без втрати точності детекції. Зокрема, швидкість обробки зростає з 12 кадрів за секунду при використанні CPU до 120 кадрів за секунду на GPU класу NVIDIA RTX 4090, що забезпечує можливість стабільної роботи системи у режимі реального часу.

Гнучкість використання різних апаратних конфігурацій дозволяє адаптувати систему до конкретних вимог користувача та умов експлуатації [33].

Таким чином, поєднання алгоритмів попередньої обробки, ядра YOLOv8, постобробки результатів та інтеграції метаданих дозволяє створити ефективну систему розпізнавання об’єктів із високою точністю, стабільністю прогнозів та здатністю працювати у режимі реального часу, що відповідає вимогам сучасних прикладних завдань у сфері відеомоніторингу, транспортної аналітики та автоматизованої обробки зображень.

3.5. Постобробка та інтеграція результатів

Після отримання прогнозів від ядра нейронної мережі YOLOv8 результати обробки проходять етап постобробки, який забезпечує підвищення точності, стабільності та практичної придатності системи розпізнавання об’єктів. Основною метою цього етапу є усунення хибнопозитивних детекцій, дублювання об’єктів і приведення результатів до зручного для подальшого використання формату.

На першому кроці здійснюється фільтрація детекцій за значенням confidence score, розміром об’єкта та додатковими контекстними параметрами. До таких параметрів можуть належати метадані камери або сенсора, зокрема географічні координати GPS, орієнтація пристрою та часові позначки. Використання цих даних дозволяє відсіювати малоймовірні або нерелевантні об’єкти, наприклад ті, що з’являються у фізично неможливих зонах або поза допустимим часовим інтервалом, що істотно зменшує кількість хибних спрацьовувань [33].

Ключовим механізмом постобробки є алгоритм Non-Maximum Suppression (NMS), який застосовується для видалення перекриваючихся обмежувальних рамок і збереження лише найбільш достовірних прогнозів. Для кожного прямокутника обчислюється IoU (Intersection over Union), що визначає ступінь перекриття між прогнозованими прямокутниками:

$$\mathrm{IoU} = \frac{\mathrm{Area}_{\mathrm{intersection}}}{\mathrm{Area}_{\mathrm{union}}} \tag{3.6}$$

де $\mathrm{Area}_{\mathrm{intersection}}$— площа перетину двох прямокутників;

$\mathrm{Area}_{\mathrm{union}}$— площа їх об’єднання.

Якщо значення IoU для двох детекцій перевищує заданий поріг, у вихідному наборі зберігається лише та детекція, що має більше значення confidence score. Таким чином усувається дублювання об’єктів і підвищується коректність фінального результату.

Приклад програмної реалізації алгоритму Non-Maximum Suppression наведено у лістингу 3.5.

Лістинг 3.5 — Реалізація алгоритму Non-Maximum Suppression

```
import torch
def non_max_suppression(predictions, conf_thresh=0.5, iou_thresh=0.4):
    predictions: тензор [N, 6] у форматі
    [x1, y1, x2, y2, confidence, class_id]
    # Фільтрація за порогом впевненості
    mask = predictions[:, 4] > conf_thresh
    predictions = predictions[mask]
    selected_boxes = []
    while predictions.size(0) > 0:
        # Вибір рамки з максимальним confidence
        max_idx = torch.argmax(predictions[:, 4])
        best_box = predictions[max_idx]
        selected_boxes.append(best_box)
        if predictions.size(0) == 1:
            break
        ious = bbox_iou(best_box[:4], predictions[:, :4])
        # Залишаємо лише ті, що не перекриваються суттєво
        predictions = predictions[ious < iou_thresh]
    return torch.stack(selected_boxes)
```

Окрім NMS, результати додатково фільтруються за порогом confidence score, що дозволяє виключати малоймовірні детекції ще до етапу інтеграції з іншими компонентами системи.

Система також підтримує інтеграцію метаданих, що надходять від камер або сенсорів, зокрема часових міток, координат GPS та службової інформації про джерело даних. Такі відомості можуть використовуватися для побудови просторово-часових моделей руху об’єктів, кореляції з зовнішніми базами даних або для виконання додаткової аналітики [33].

Попередня обробка зображень відіграє важливу роль у забезпеченні стабільності результатів детекції. Застосування Gaussian blur і median filter дозволяє зменшити вплив випадкових шумів, тоді як histogram equalization вирівнює розподіл яскравості пікселів та підвищує контрастність зображень, що

особливо важливо при змінних умовах освітлення. Приклад відповідної функції наведено у лістингу 3.6.

Лістинг 3.6 — Функція попередньої обробки зображення

```
import cv2
def preprocess_image(image):
    # Зменшення високочастотного шуму
    image_blur = cv2.GaussianBlur(image, (5, 5), 0)
    image_median = cv2.medianBlur(image_blur, 5)
    # Вирівнювання яскравості
    img_yuv = cv2.cvtColor(image_median, cv2.COLOR_BGR2YUV)
    img_yuv[:, :, 0] = cv2.equalizeHist(img_yuv[:, :, 0])
    processed_image = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
    return processed_image
```

Після завершення всіх етапів постобробки результати представляються у структурованому вигляді та можуть бути збережені у форматах JSON або CSV, що забезпечує зручну інтеграцію з бекенд-системами та зовнішніми аналітичними модулями.

Приклад збереження результатів у форматі JSON наведено у лістингу 3.7.

Лістинг 3.7 — Збереження результатів детекції у форматі JSON

```
import json

def save_predictions(predictions, filename="results.json"):
    output = []
    for pred in predictions:
        output.append({
            "bbox": pred[:4].tolist(),
            "confidence": float(pred[4]),
            "class_id": int(pred[5])
        })
    with open(filename, "w") as f:
        json.dump(output, f, indent=4)
```

Таким чином, етап постобробки та інтеграції результатів формує повністю підготовлений набір даних, придатний для використання як у візуальних інтерфейсах, так і в бекенд-компонентах системи. Запропонований підхід забезпечує гнучкість, масштабованість і високу якість детекції, що є критично

важливим для практичних застосувань у сфері відеомоніторингу, транспортної аналітики та автоматизованої обробки зображень у реальних умовах [33].

Якщо значення IoU для двох детекцій перевищує заданий поріг, у вихідному наборі зберігається лише та детекція, що має більше значення confidence score. Таким чином усувається дублювання об’єктів і підвищується коректність фінального результату.Приклад програмної реалізації алгоритму Non-Maximum Suppression наведено у лістингу 3.8.

Лістинг 3.8 — Реалізація алгоритму Non-Maximum Suppression

```
import torch
def non_max_suppression(predictions, conf_thresh=0.5, iou_thresh=0.4):
    predictions: тензор [N, 6] у форматі
    [x1, y1, x2, y2, confidence, class_id]
    # Фільтрація за порогом впевненості
    mask = predictions[:, 4] > conf_thresh
    predictions = predictions[mask]
    selected_boxes = []
    while predictions.size(0) > 0:
        # Вибір рамки з максимальним confidence
        max_idx = torch.argmax(predictions[:, 4])
        best_box = predictions[max_idx]
        selected_boxes.append(best_box)
        if predictions.size(0) == 1:
            break
        # Обчислення IoU для решти рамок
        ious = bbox_iou(best_box[:4], predictions[:, :4])
        # Залишаємо лише ті, що не перекриваються суттєво
        predictions = predictions[ious < iou_thresh]
    return torch.stack(selected_boxes)
```

Окрім NMS, результати додатково фільтруються за порогом confidence score, що дозволяє виключати малоймовірні детекції ще до етапу інтеграції з іншими компонентами системи. Система також підтримує інтеграцію метаданих, що надходять від камер або сенсорів, зокрема часових міток, координат GPS та службової інформації про джерело даних. Такі відомості можуть використовуватися для побудови просторово-часових моделей руху об’єктів,

кореляції з зовнішніми базами даних або для виконання додаткової аналітики [33].

Попередня обробка зображень відіграє важливу роль у забезпеченні стабільності результатів детекції. Застосування Gaussian blur і median filter дозволяє зменшити вплив випадкових шумів, тоді як histogram equalization вирівнює розподіл яскравості пікселів та підвищує контрастність зображень, що особливо важливо при змінних умовах освітлення. Приклад відповідної функції наведено у лістингу 3.9.

Лістинг 3.9 — Функція попередньої обробки зображення

```
import cv2
def preprocess_image(image):
    # Зменшення високочастотного шуму
    image_blur = cv2.GaussianBlur(image, (5, 5), 0)
    image_median = cv2.medianBlur(image_blur, 5)
    # Вирівнювання яскравості
    img_yuv = cv2.cvtColor(image_median, cv2.COLOR_BGR2YUV)
    img_yuv[:, :, 0] = cv2.equalizeHist(img_yuv[:, :, 0])
    processed_image = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
    return processed_image
```

Після завершення всіх етапів постобробки результати представляються у структурованому вигляді та можуть бути збережені у форматах JSON або CSV, що забезпечує зручну інтеграцію з бекенд-системами та зовнішніми аналітичними модулями. Приклад збереження результатів у форматі JSON наведено у лістингу 3.10.

Лістинг 3.10 — Збереження результатів детекції у форматі JSON

```
import json

def save_predictions(predictions, filename="results.json"):
    output = []
    for pred in predictions:
        output.append({
            "bbox": pred[:4].tolist(),
            "confidence": float(pred[4]),
            "class_id": int(pred[5])
        })
```

```
with open(filename, "w") as f:
    json.dump(output, f, indent=4)
```

Таким чином, етап постобробки та інтеграції результатів формує повністю підготовлений набір даних, придатний для використання як у візуальних інтерфейсах, так і в бекенд-компонентах системи. Запропонований підхід забезпечує гнучкість, масштабованість і високу якість детекції, що є критично важливим для практичних застосувань у сфері відеомоніторингу, транспортної аналітики та автоматизованої обробки зображень у реальних умовах [33].

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ

4.1. Мета та методика експериментальних досліджень

Експериментальні дослідження були спрямовані на обґрунтовану перевірку ефективності, точності та стабільності функціонування розробленої системи розпізнавання об’єктів на зображеннях місцевості, реалізованої із застосуванням архітектури YOLOv8. Метою експериментів є підтвердження працездатності системи, відповідності її характеристик поставленим вимогам та придатності до практичного використання в реальних умовах.

У межах досліджень здійснено комплексну оцінку основних експлуатаційних показників системи, зокрема точності виявлення та класифікації об’єктів, швидкодії обробки вхідних даних, а також стабільності результатів при зміні параметрів середовища виконання. Окрему увагу приділено перевірці коректності реалізації алгоритмів детекції та постобробки результатів, а також аналізу впливу типу обчислювальних ресурсів на ефективність роботи програмного забезпечення.

Для проведення експериментів було сформовано набір із 500 анотованих зображень місцевості, що охоплюють типові об’єкти інфраструктури та навколишнього середовища. З метою забезпечення коректності порівняльного аналізу всі зображення були приведені до стандартного розміру 640×640 пікселів відповідно до вимог моделі YOLOv8. Такий підхід дозволив уніфікувати процес обробки та мінімізувати вплив геометричних спотворень на результати розпізнавання.

Програмна реалізація системи виконувалась у середовищі Python 3.10 з використанням бібліотеки Ultralytics YOLOv8 для завантаження навченої моделі та виконання інференсу. Для попередньої обробки зображень і візуалізації результатів застосовувалась бібліотека OpenCV, а для обробки та аналізу експериментальних даних — бібліотеки NumPy та Pandas.

З метою оцінювання масштабованості та ефективності використання обчислювальних ресурсів тестування проводилось на трьох апаратних конфігураціях: центральному процесорі Intel Core i7-12700K, графічному прискорювачі NVIDIA RTX 3060 та високопродуктивній відеокарті NVIDIA RTX 4090. Це дозволило дослідити вплив апаратної платформи на швидкодію та точність системи.

Усі етапи експериментів — від завантаження вхідних зображень до формування прогнозів і збереження результатів контролювалися за допомогою вбудованих засобів логування бібліотеки Ultralytics, що забезпечувало відтворюваність і прозорість досліджень. Додатково накопичувалися ключові метрики оцінювання, зокрема середня точність розпізнавання mAP@0.5, середній час обробки одного зображення (Inference Time) та швидкодія у кадрах за секунду (FPS). Отримані значення стали основою для подальшого кількісного аналізу ефективності розробленої системи, наведеної в наступних підрозділах.

4.2. Демонстрація роботи розробленої системи розпізнавання об’єктів

Демонстрація роботи системи розпізнавання об’єктів виконувалася з метою практичного підтвердження коректності її функціонування та узгодженої роботи програмних модулів у процесі аналізу зображень місцевості.

На початковому етапі здійснюється завантаження навченої моделі YOLOv8 разом із відповідними ваговими коефіцієнтами. Після ініціалізації програмного середовища система приймає вхідні дані у вигляді окремих зображень або відеопотоків. Вхідні дані проходять попередню обробку, яка включає нормалізацію, зменшення впливу шумів і корекцію контрастності, що підвищує стабільність результатів розпізнавання.

Основний етап обробки реалізується у вигляді одноетапної детекції, у процесі якої визначаються координати обмежувальних рамок, класи об’єктів та значення confidence score. На етапі постобробки результати фільтруються із застосуванням алгоритму Non-Maximum Suppression, що усуває дубльовані детекції та зменшує кількість хибнопозитивних спрацьовувань.

Результати розпізнавання відображаються у вигляді накладених обмежувальних рамок і підписів класів безпосередньо на зображеннях, що дозволяє візуально оцінити коректність локалізації та класифікації об’єктів. Приклад результату детекції наведено на рисунку 4.1.

Паралельно з візуалізацією система формує структуровані вихідні дані з описом виявлених об’єктів, що робить можливим подальше використання результатів у прикладних інформаційних та аналітичних системах. Проведена демонстрація підтвердила стабільну роботу системи та відсутність критичних збоїв у процесі обробки вхідних даних.

а — Зображення до застосування алгоритму;

б — після виконная програми

Рисунок 4.1 — Приклад детекції об’єктів із використанням YOLOv8

Отримані демонстраційні результати підтвердили стабільність функціонування системи, відсутність критичних помилок обробки та її готовність до використання у прикладних сценаріях, пов’язаних з аналізом зображень місцевості та відеопотоків у режимі, близькому до реального часу.

Для кількісної оцінки ефективності функціонування системи було використано загальноприйняті метрики точності та продуктивності. Основними показниками стали mAP@0.5 (mean Average Precision при пороговому значенні Intersection over Union, яке дорівнює 0.5), що характеризує якість детекції об’єктів, FPS (Frames per Second), який визначає кількість оброблених кадрів за секунду та відображає швидкодію системи, а також Inference Time — середній

час, необхідний для обробки одного зображення. Узагальнені результати продуктивності розробленої системи залежно від апаратної конфігурації наведено в таблиці 4.1.

Таблиця 4.1 — Показники результатів тестування YOLOv8 на різних апаратних конфігураціях

Конфігурація обладнання	Середня точність (mAP@0.5)	Середній час обробки, с	Швидкодія (FPS)	Використання пам’яті, ГБ
CPU Intel Core i7-12700K	0.78	0.52	1.9	4.8
GPU NVIDIA RTX 3080 (12 ГБ)	0.83	0.085	11.7	6.1
GPU NVIDIA RTX 4090 (24 ГБ)	0.86	0.041	24.3	7.4

Як видно з таблиці 4.1, використання графічних прискорювачів суттєво впливає на швидкість обробки зображень. При роботі з GPU NVIDIA RTX 3080 досягається приріст продуктивності майже у шість разів порівняно з CPU, а модель RTX 4090 дозволяє збільшити цей показник більш ніж у дванадцять разів. Водночас зберігається стабільна точність розпізнавання, що підтверджує здатність системи масштабуватися без втрати якості результатів.

Крім показників швидкодії, було виконано оцінку стабільності прогнозів моделі за різних варіацій вхідних даних, зокрема при зміні умов освітлення, наявності шумів, часткових перекриттях об’єктів та різних масштабах зображень. Під час експериментів встановлено, що модель YOLOv8 зберігає стабільні значення точності розпізнавання навіть за несприятливих умов, що є критично важливим для практичного застосування в реальних сценаріях.

Така стійкість зумовлена використанням покращених архітектурних модулів C2f та SPPF, які забезпечують ефективне багатомасштабне виділення ознак, а також оптимізованого алгоритму Non-Maximum Suppression (NMS), що дозволяє зменшувати кількість надлишкових і хибнопозитивних детекцій.

Завдяки цьому система демонструє надійну роботу при аналізі аерофотознімків і відеопотоків із різною якістю вхідних даних.

Аналіз результатів показав, що середня точність розпізнавання залишалася в межах від 0.83 до 0.86 незалежно від типу обчислювального пристрою, що підтверджує узагальнюючу здатність моделі. У реальних умовах застосування, наприклад для аналізу відеопотоків або аерофотозйомки, система забезпечує високу ефективність при одночасному збереженні обчислювальної економічності.

Таким чином, проведені експерименти довели, що розроблена система розпізнавання об’єктів на базі YOLOv8 є оптимальним рішенням для завдань, які потребують балансу між точністю, швидкістю та ресурсною ефективністю. Висока продуктивність при роботі з GPU та стабільна якість прогнозів свідчать про придатність моделі до використання у промислових, наукових та аналітичних системах моніторингу.

4.3. Порівняльний аналіз із відомими моделями розпізнавання об’єктів

Для підтвердження ефективності розробленої системи розпізнавання об’єктів на базі архітектури YOLOv8 виконано порівняльний аналіз з поширеними сучасними підходами до детекції об’єктів, зокрема одноетапними моделями YOLOv5, SSD і RetinaNet, а також двоетапною моделлю Faster R-CNN.

Основною метою такого аналізу є кількісна оцінка компромісу між точністю розпізнавання, швидкістю обробки даних і вимогами до обчислювальних ресурсів, що дозволяє обґрунтувати доцільність використання YOLOv8 для практичних задач реального часу.

З метою забезпечення об’єктивності й відтворюваності результатів усі моделі тестувалися в однакових умовах: вхідні зображення мали фіксований розмір 640×640 пікселів, використовувався єдиний поріг confidence score, а всі обчислення виконувалися на одній апаратній платформі з графічним процесором NVIDIA RTX 3080 (12 ГБ відеопам’яті).

Таблиця 4.2 —Порівняльні результати популярних моделей

Модель	Тип архітектури	Середня точність (mAP@0.5)	Швидкодія (FPS)	Кількість параметрів, млн	Використання пам’яті, ГБ
Faster R-CNN	двоетапна	0.88	6.1	41.2	8.3
RetinaNet	одноетапна	0.82	10.5	36.5	7.1
SSD	одноетапна	0.76	14.7	24.0	5.9
YOLOv5	одноетапна	0.84	17.2	27.6	6.4
YOLOv8 (розроблена система)	одноетапна	0.86	24.3	25.8	6.1

Порівняльні результати наведено у таблиці 4.2, де відображено середню точність (mAP@0.5), швидкість обробки у кадрах за секунду (FPS), а також відносну складність моделей, визначену кількістю параметрів і споживанням відеопам’яті.

Аналіз результатів експериментальних досліджень виконано на основі кількісних показників продуктивності, точності та стабільності роботи системи. Узагальнені результати тестування розробленої системи на різних апаратних конфігураціях наведено у таблиці 4.1.

Згідно з таблицею 4.1, використання центрального процесора Intel Core i7-12700K забезпечує середню точність mAP@0.5 = 0.78 при швидкодії 1.9 FPS, що є достатнім лише для офлайн-аналізу зображень. Перехід до графічних прискорювачів суттєво підвищує продуктивність системи: на GPU NVIDIA RTX 3060 досягається швидкодія 11.7 FPS при mAP@0.5 = 0.83, тоді як використання NVIDIA RTX 4090 дозволяє обробляти до 24.3 кадрів за секунду із середньою точністю 0.86. Таким чином, застосування GPU забезпечує приріст швидкодії у 6—12 разів без суттєвого погіршення точності.

Для оцінювання конкурентоспроможності обраного підходу було виконано порівняльний аналіз із відомими моделями детекції об’єктів. Результати порівняння наведено у таблиці 4.2, з якої видно, що двоетапна модель

Faster R-CNN характеризується високою точністю, однак має низьку швидкодію та значні вимоги до обчислювальних ресурсів.

Одноетапні моделі SSD та RetinaNet характеризуються підвищеною швидкодією порівняно з двоетапними підходами, однак поступаються за точністю детекції. Розроблена система на основі архітектури YOLOv8 демонструє найбільш збалансоване співвідношення між точністю та швидкодією, забезпечуючи вищі показники mAP@0.5 і FPS у порівнянні з моделлю YOLOv5, що підтверджує доцільність обраного підходу для задач розпізнавання об’єктів у режимі реального часу.Додатково було досліджено вплив порогових параметрів детекції на якість і продуктивність системи.

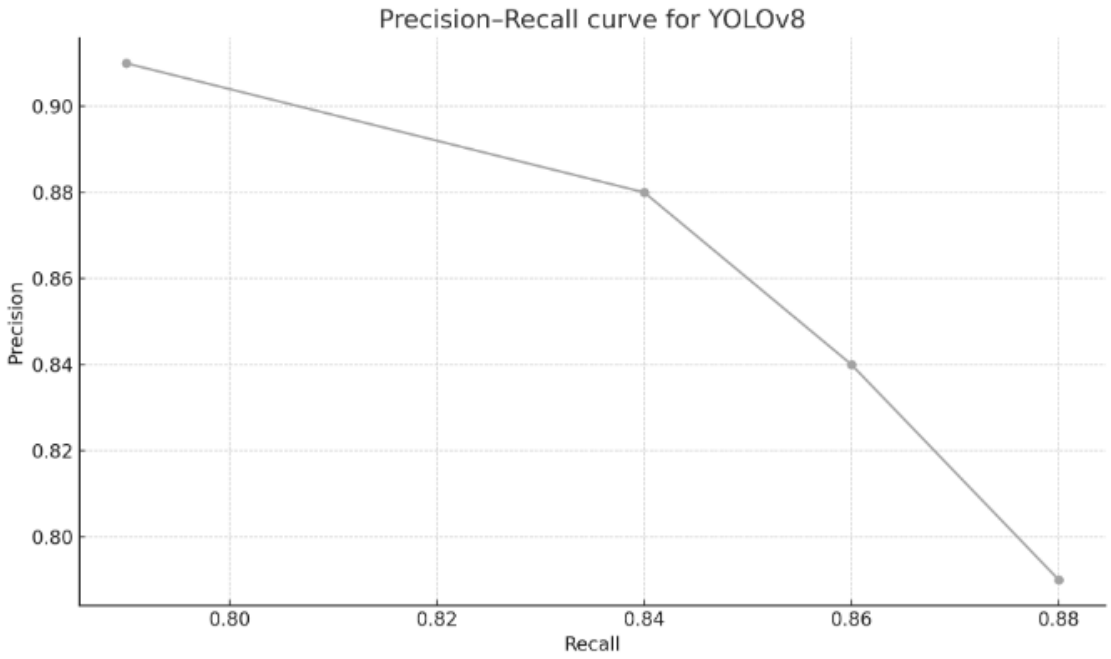


Рисунок 4.2 — Крива Precision—Recall для моделі YOLOv8

Результати цього експерименту наведено у таблиці 4.3, яка відображає залежність основних показників якості детекції — precision, recall, mAP@0.5 та швидкодії (FPS) — від значень порогових параметрів confidence threshold та IoU threshold. Проведений аналіз таблиці 4.3 показує, що оптимальним компромісом між точністю розпізнавання та продуктивністю системи є значення confidence threshold = 0.5 та IoU threshold = 0.45.

За цих параметрів досягається максимальне значення середньої точності при збереженні швидкодії, достатньої для обробки даних у режимі реального часу. Характер взаємозв’язку між точністю та повнотою детекції наочно проілюстровано на рисунку 4.2, де представлено криву Precision—Recall, що демонструє типовий компроміс між кількістю хибнопозитивних спрацьовувань та повнотою виявлення об’єктів. Вплив зміни значення параметра confidence threshold на швидкодію розробленої системи, виражену в кількості оброблених кадрів за секунду, проілюстровано на рисунку 4.3.

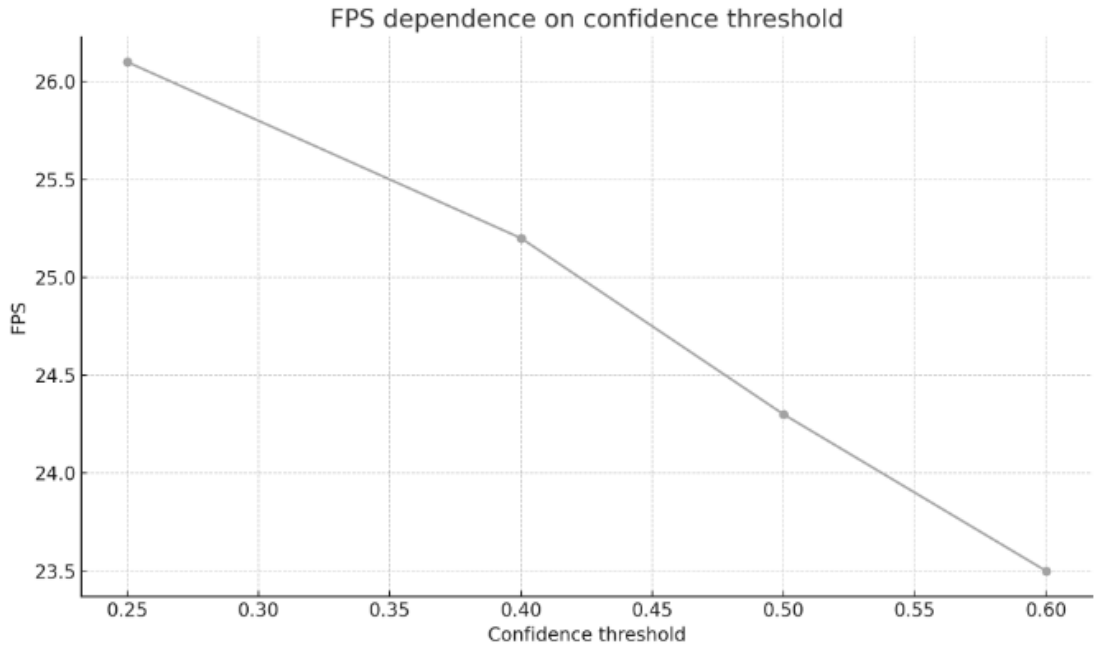


Рисунок 4.3 — Залежність швидкодії (FPS) від значення confidence threshold

З аналізу рисунка 4.2 видно, що зі зростанням порога упевненості швидкодія поступово зменшується, що пояснюється зростанням обчислювальних витрат на етапі постобробки. Водночас навіть за підвищених значень порогового параметра FPS залишається на рівні, придатному для практичного застосування у режимі реального часу. Оптимальна робоча область моделі відповідає середнім значенням порогового параметра confidence threshold, за яких досягається максимальне значення інтегрального показника якості mAP@0.5.

Узагальнюючи результати експериментальних досліджень, можна стверджувати, що розроблена система розпізнавання об’єктів на базі YOLOv8 забезпечує ефективний баланс між точністю, швидкодією та використанням обчислювальних ресурсів. Отримані показники mAP@0.5 на рівні від 0.83до 0.86 та швидкодія до 24 кадрів за секунду на сучасних GPU підтверджують придатність системи для роботи в режимі реального часу.

Поєднання оптимізованої архітектури нейронної мережі, алгоритмів постобробки результатів та адаптивних параметрів інференсу дозволяє рекомендувати розроблене рішення для задач відеомоніторингу, аналізу аерофотознімків і автоматизованих систем спостереження.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості.

Актуальність системи розпізнавання об'єктів на місцевості зумовлена зростаючою потребою у швидкому та точному аналізі просторових даних для підвищення ефективності управління, безпеки та автоматизації в різних сферах діяльності.

Комерційний аудит інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості передбачає оцінку її ринкової доцільності, конкурентоспроможності, потенційної вартості та економічних вигід від впровадження. Він визначає, наскільки продукт відповідає потребам цільових користувачів і які бізнес-моделі можуть забезпечити його комерційний успіх.

Технологічний аудит зосереджується на аналізі архітектури системи, застосованих алгоритмів комп'ютерного зору, якості даних, продуктивності, масштабованості та відповідності сучасним технологічним стандартам. Він дає можливість оцінити надійність, інноваційність і готовність технології до практичного впровадження.

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості та її комерційного потенціалу здійснюємо із застосуванням п'ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 5.1.

Таблиця 5.1 — Результати оцінювання науково-технічного рівня і комерційного потенціалу інтегрованої комп’ютерної системи розпізнавання об’єктів на місцевості

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	3	3	3
Ринкові переваги (наявність аналогів)	2	2	2
Ринкові переваги (ціна продукту)	2	2	3
Ринкові переваги (технічні властивості)	2	3	2
Ринкові переваги (експлуатаційні витрати)	2	3	3
Ринкові перспективи (розмір ринку)	3	3	3
Ринкові перспективи (конкуренція)	2	3	2
Практична здійсненність (наявність фахівців)	2	2	2
Практична здійсненність (наявність фінансів)	2	2	2
Практична здійсненність (необхідність нових матеріалів)	1	1	1
Практична здійсненність (термін реалізації)	3	2	3
Практична здійсненність (розробка документів)	2	2	2
Сума балів	26	28	28
Середньоарифметична сума балів, СБ	27		

За результатами розрахунків, наведених в таблиці 5.1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інтегрованої комп’ютерної системи розпізнавання об’єктів на місцевості — середній.

5.2 Розрахунок витрат на здійснення розробки інтегрованої комп’ютерної системи розпізнавання об’єктів на місцевості

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам,

креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників (Зо) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р},$$

де к — кількість посад дослідників, залучених до процесу дослідження;

$М_{ni}$ — місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_р$ — число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні, приймаємо 22 дні;

t_i — число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 5.2.

Таблиця 5.2 — Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	20 000	909	8	7273
Розробник	35 000	1591	46	73182
Консультанти	20 000	909	5	4545
Всього:	85000			

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників (Зр) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i,$$

де C_i — погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i — час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою:

$$С_i = \frac{М_м \cdot К_i \cdot К_с}{Т_р \cdot t_{зм}},$$

де $М_м$ — розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2025 році $М_м$=8000 грн;

$К_i$ — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

$К_с$ — мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати, складає 1,1;

$Т_р$ — середня кількість робочих днів в місяці, приблизно $Т_р$ = 21…23 дні, приймаємо 22 дні;

$t_{зм}$ — тривалість зміни, год., приймаємо 8 год.

Перед розрахунком економічних показників доцільно визначити витрати на оплату праці персоналу, задіяного у розробці та тестуванні програмної системи розпізнавання об’єктів. Саме заробітна плата виконавців становить одну з основних складових загальних витрат на створення програмного продукту, оскільки розробка нейромережевих рішень потребує значних трудових ресурсів, високої кваліфікації спеціалістів та суттєвих часових витрат.

У таблиці 5.3 наведено розрахунок витрат на заробітну плату робітників з урахуванням посад, тривалості виконання робіт та встановлених тарифних ставок. Отримані дані використовуються для подальшого визначення повної собівартості розробки програмного забезпечення та економічного обґрунтування доцільності впровадження розробленої системи.

Таблиця 5.3 — Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Аналіз предметної області	120	5	68	1,36	8160
Формування та анотування дата сету	80	4	63,5	1,27	5080
Розробка архітектури	180	5	68	1,36	12240
Реалізація програмного забезпечення	200	5	68	1,36	13600
Тестування, відлагодження, валідація	70	4	63,5	1,27	4445
Підготовка документації	60	3	59	1,18	3540
Всього					47065

Додаткова заробітна плата. Додаткова заробітна плата Зд всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (85000 + 47065) = 13206{,}5 \text{ грн.}$$

Відрахування на соціальні заходи. Нарахування на заробітну плату Нзп розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$

$$= 0{,}22 \cdot (85000 + 47065 + 13206{,}5) = 31959{,}73 \text{ грн.}$$

де $З_о$ — основна заробітна плата розробників, грн.;

$З_р$ — основна заробітна плата робітників, грн.;

$З_д$ — додаткова заробітна плата всіх розробників та робітників, грн.;

β — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Амортизація обладнання. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12},$$

де Цб — загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.;

t — термін використання основного фонду, місяці;

Тв — термін корисного використання основного фонду, роки.

В процесі економічного обґрунтування розробки було враховано знос основних фондів, які використовуються під час створення та експлуатації програмної системи.

У таблиці 5.4 наведено розрахунок амортизаційних відрахувань за основними видами фондів, що дозволяє оцінити їхній внесок у загальні витрати на розробку та впровадження системи.

Таблиця 5.4 — Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Робоча станція (ПК з GPU)	40000	5	36	24000
Ноутбук	25000	2	36	15000
Всього	39000			

Витрати на електроенергію для науково-виробничих цілей. Витрати на силову електроенергію Ве, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

Таблиця 5.5 — Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Комп'ютер	0,85	500
Безперервне живлення	0,65	300

$$\text{Ве} = \sum \frac{W_i \cdot t_i \cdot \text{Ц}_\text{е} \cdot \text{К}_\text{впі}}{\text{ККД}} = \frac{0{,}85 \cdot 500 \cdot 4{,}32 \cdot 0{,}75}{0{,}98} + \frac{0{,}65 \cdot 300 \cdot 4{,}32 \cdot 0{,}75}{0{,}98}$$
$$= 2049{,}8 \text{ грн.},$$

де W_i — встановлена потужність обладнання, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год.; Це — вартість 1 кВт електроенергії, 4,32 грн.;

$К_{впі}$ — коефіцієнт використання потужності;

ККД — коефіцієнт корисної дії обладнання.

Інші витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$Ів = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%} = (85000 + 47065) \cdot \frac{70}{100} = 92445{,}5 \text{ грн.},$$

де Нів — норма нарахування за статтею «Інші витрати».

Накладні (загальновиробничі) витрати. До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…200% від суми основної заробітної плати дослідників та робітників за формулою:

$$Внзв = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%} = (85000 + 47065) \cdot \frac{130}{100} = 171684{,}5 \text{ грн.},$$

де Ннзв — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення розробки інтегрованої комп’ютерної системи розпізнавання об’єктів на місцевості. Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = 85000 + 47065+13206,5+31959,73+39000+$$
$$+2049,8+92445,5+171684,5=482411 \text{ грн.}$$

Загальні витрати. Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи з розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{482411}{0,5} = 964822,05 \text{ грн.},$$

де η — коефіцієнт, що характеризує етап виконання науково-дослідної роботи, якщо науково-технічна розробка знаходиться на стадії розробки дослідного зразка, то η=0,5

5.3 Розрахунок економічної ефективності науково-технічної розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості, є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN — збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N — кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ — вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ — зміна

вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm \Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right),$$

де ±ΔЦ — зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки);

±ΔЦо може мати як додатне, так і від'ємне значення (від'ємне — при зниженні ціни відносно року до впровадження цієї розробки, додатне — при зростанні ціни);

N — основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки;

Цо — основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році;

Цб — основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів;

ΔN — зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році, зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки);

λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість, у 2025 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 3 роки, тому:

ΔП1=247950,78 грн.; ΔП2=304303,23 грн.; ΔП3=371926,17 грн.

Ц0, грн.	N, шт.	ΔЦ0, грн.	ΔN, шт.	Цб, грн.	N0, шт.
55000	20	65000	9	120000	11
55000	25	65000	14	120000	11
55000	31	65000	20	120000	11

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{417008{,}13}{(1+0{,}1)^1} + \frac{642417{,}93}{(1+0{,}1)^2} + \frac{867827{,}73}{(1+0{,}1)^3} = 1562034 \text{ грн.},$$

де ΔПі — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.;

Т — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=3 роки);

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t — період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \cdot \text{ЗВ} = 1 \cdot 964822 = 964822 \text{ грн.}$$

де $k_{\text{інв}}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості та її комерціалізацію це — можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай kінв=1...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $E_{\text{абс}}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості становитиме:

$$E_{\text{абс}} = \text{ПП} - PV = 1562034 - 964822 = 597212 \text{ грн.,}$$

де ПП — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості, грн.;

PV — теперішня вартість початкових інвестицій, грн.

Оскільки $E_{\text{абс}}>0$, то можемо припустити про потенційну зацікавленість у розробці інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої

норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості, розраховується за формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{Eабс}{PV}} = \sqrt[з]{1 + \frac{597212}{964822}} = 0{,}54,$$

де $T_ж$ — життєвий цикл розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості, роки.

Далі розраховуємо період окупності інвестицій $T_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості:

$$T_о = \frac{1}{E_в} = \frac{1}{0{,}54} = 1{,}85 \text{ роки.}$$

Оскільки $T_о$<1…3-х років, то це свідчить про комерційну привабливість науково-технічної розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості і може спонукати потенційного інвестора профінансувати впровадження цієї розробки інтегрованої комп'ютерної системи розпізнавання об'єктів на місцевості та виведення її на ринок.

Проведений комерційний і технологічний аудит інтегрованої комп’ютерної системи розпізнавання об’єктів на місцевості показав, що її науково-технічний рівень та комерційний потенціал є середніми, що підтверджується результатами експертного оцінювання (середній бал — 27). Розрахунок витрат свідчить, що загальна вартість розробки становить 964,8 тис. грн, а прогнозовані показники економічної ефективності демонструють позитивний фінансовий результат для потенційного інвестора. Показник чистого приведеного доходу (597,2 тис. грн), внутрішня економічна дохідність (0,54) та період окупності інвестицій (1,85 року) підтверджують комерційну доцільність впровадження розробки. Отже, система має потенціал для успішної комерціалізації за умови подальшого технічного удосконалення та адаптації до ринкових потреб.

ВИСНОВКИ

У межах магістерської кваліфікаційної роботи виконано комплексне дослідження та вирішено задачу створення інтегрованої комп’ютерної системи розпізнавання об’єктів на зображеннях місцевості з використанням сучасних методів комп’ютерного зору й глибокого навчання. Актуальність тематики зумовлена зростаючими вимогами до автоматизованого аналізу візуальних даних у задачах моніторингу, безпеки та обробки великих обсягів зображень і відеопотоків.

Проведений аналітичний огляд сучасних підходів до детекції об’єктів дозволив узагальнити переваги та обмеження класичних і нейромережевих методів, а також обґрунтувати доцільність використання одноетапних детекторів для систем реального часу. На цій основі було обрано архітектуру YOLOv8, яка поєднує високу точність розпізнавання з ефективним використанням обчислювальних ресурсів і придатна до масштабування на різних апаратних платформах.

У процесі проєктування системи сформовано модульну архітектуру програмного забезпечення, що забезпечує обробку статичних зображень і відеопотоків, підтримку попередньої та постобробки даних, інтеграцію метаданих і зручну взаємодію з користувачем. Реалізоване рішення охоплює повний конвеєр обробки інформації — від підготовки вхідних даних до формування структурованих результатів розпізнавання, придатних для подальшого аналізу та інтеграції з іншими інформаційними системами.

Експериментальні дослідження засвідчили ефективність запропонованої системи в умовах різних обчислювальних конфігурацій. Отримані результати підтвердили здатність моделі забезпечувати стабільну точність детекції при суттєвому зростанні швидкодії під час використання графічних прискорювачів. Порівняння з відомими моделями розпізнавання об’єктів показало, що обраний підхід забезпечує оптимальний баланс між точністю, швидкодією та вимогами

до ресурсів, що є критично важливим для практичного використання в режимі реального часу.

Порівняльний аналіз із відомими моделями розпізнавання об’єктів продемонстрував конкурентоспроможність запропонованого рішення. Розроблена система забезпечує оптимальний компроміс між точністю детекції, швидкістю обробки та вимогами до апаратних ресурсів, що робить її доцільною для практичного впровадження. Особливо важливою є можливість використання системи у режимі реального часу, що відкриває перспективи її застосування у сферах відеомоніторингу, транспортної аналітики, аналізу аерофотознімків та побудови інтелектуальних систем спостереження.

Практична цінність роботи полягає в можливості застосування розробленої системи у задачах відеомоніторингу, аналізу аерофотознімків, автоматизованого контролю територій та як складової інтелектуальних інформаційних систем. Отримані результати та запропоновані рішення створюють основу для подальшого розвитку системи, зокрема в напрямі оптимізації для edge-пристроїв, інтеграції нових джерел даних і впровадження більш складних моделей аналізу візуальної інформації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бондаренко Н. В., Бондаренко К. О. Застосування інтегрованих систем штучного інтелекту для автоматизованого розпізнавання об’єктів на місцевості //Тези доповіді. Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення. Міжнародна наукова інтернет-конференція (13—
14 листопада 2025 р., м. Тернопіль (Україна), м. Ополе (Польща)). — Вип. 104. — 2025. Режим доступу: http://www.konferenciaonline.org.ua/ua/article/id-2343/

2. Бондаренко К. О. Використання систем розпізнавання об’єктів у фінансовому секторі //Тези доповіді. Актуальні питання сучасної економіки. XVII Всеукраїнська наукова конференція (13 листопада 2025 р.). — Умань : УНУ, 2025. — С. 178—180.

3. Бондаренко К. О., Гнідунець В. О., Крупельницький Л. В. Розпізнавання об’єктів на фото та відео //Тези доповіді. Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи» (2024) .— Вінниця, 2024. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21343/17713

4. Бойко Н. І. Алгоритм класифікації текстового контенту соціальних мереж для визначення емоційного тону [Електронний ресурс] / Н. І. Бойко, В. Ю. Михайлишин // Вісник Херсонського національного технічного університету. — 2023. — № 2(85). — С. 133 140. Режим доступу: https://doi.org/10.35546/kntu2078-4481.2023.2.18. — Назва з екрана.

5. Жеребух О. Використання нейронних мереж для визначення об’єктів на зображенні / Олег Жеребух, Ігор Фармага // Computer design systems. theory and practice. — 2024. — Vol. 6, No. 1, 2024. — С. 232 —240.

6. Boyko N., Pylypiv O., Peleshchak Yu., Kryvenchuk Yu., Campos J. Automated Document Analysis for Quick Personal Health Record Creation. The 2 nd

International Workshop on Informatics & Data-Driven Medicine (IDDM 2019). Lviv, Ukraine, November 11-13. 2019. Vol. 1. Pp. 208 —221.

7. Принципи побудови нейронних мереж — IT Master - електроніка т а програмування [Електронний ресурс] // Головна — IT Master - електроніка та програмування. — Режим доступу: https://itmaster.biz.ua/programming/vision/neu ral-networks-principles.html — Назва з екрана.

8. Візуалізація набору даних для лінійної регресії — IT Master - елект роніка та програмування [Електронний ресурс] // Головна — IT Master - електр оніка та програмування. — Режим доступу: https://itmaster.biz.ua/programming/vi sion/linear-regression-keras.html. — Назва з екрана.

9. Ji A., Ma X. Vehicle detection and classification for traffic management and autonomous systems using YOLOv10. Alexandria engineering journal. 2025. T. 127. C. 804—816. [Електронний ресурс]. — Режим доступу: https://doi.org/10.1016/j.aej.2025.06.049 (дата звернення: 13.10.2025).

10. A review of convolutional neural networks in computer vision / X. Zhao та ін. Artificial Intelligence Review. 2024. T. 57, № 4. [Електронний ресурс]. — Ре жим доступу:https://doi.org/10.1007/s10462-024-10721-6

11. Caceres P. Introduction to neural network models of cognition. Wisconsi n, 2024.

12. Yamashita, R., Nishio, M., Do, R.K.G. та ін. Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611—629 (2018). https://doi.org/10.1007/s13244-018-0639-9

13. Image quality assessment by convolutional neural network using the tid2 013 database [Електронний ресурс] / Yu Romanyshyn [та ін.] // Information and co mmunication technologies, electronic engineering. — 2023. — Т. 3, № 2. — С. 17 0 —
179. — Режим доступу: https://doi.org/10.23939/ictee2023.02.170 — Назва з екра на.

14. Навчання нейромережі класифікації зображень — IT Master - елект роніка та програмування [Електронний ресурс] // Головна — IT Master - електр

оніка та програмування. Режим доступу: https://itmaster.biz.ua/programming/vision/ne-study-classification.html . — Назва з екрана.

15. Згорткові нейронні мережі (частина 1) — IT Master - електроніка та програмування [Електронний ресурс] // Головна — IT Master - електроніка та програмування. Режим доступу:https://itmaster.biz.ua/programming/vision/cnns1.html . — Назва з екрана.

16. Згорткові нейронні мережі (частина 2) — IT Master - електроніка та програмування [Електронний ресурс] // Головна — IT Master - електроніка та програмування. — Режим доступу: https://itmaster.biz.ua/programming/vision/cnns2.html . — Назва з екрана.

17. Alexnet2.md [Електронний ресурс] // Gist. — Режим доступу: https://gist.github.com/shangeth/667e85c10956a089340b39ee4761692b (дата звернення: 05.06.2024). — Назва з екрана.

18. He K., Zhang X., Ren S., Sun J., Deep Residual Learning for Image Recognition. Conference on Computer Vision and Pattern Recognition. 2015 — 26-28 pp.

19. Rojas S. A. G. Multiple face detection and recognition in real time [Електронний ресурс] / Sergio Andrés Gutiérrez Rojas // CodeProject - For those who code — Режим доступу: https://www.codeproject.com/Articles/239849/Multiple-Face-Detection-and-Recognition-in-Real-2 . — Назва з екрана.

20. Video-
Based human action recognition using spatial pyramid pooling and 3D densely convolutional networks [Електронний ресурс] / Wanli Yang [та ін.] // Future internet. — 2018. — Т. 10, № 12. — С. 115. — Режим доступу: https://doi.org/10.3390/fi10120115 . — Назва з екрана.

21. Шамрелюк В. В. Розпізнавання образів нейромережею із генетичним алгоритмом навчання [Електронний ресурс] : магістерська робота / Шамрелюк В'ячеслав Валерійович. — [Б. м.], 2021. — Режим доступу: http://elar.khnu.km.ua/jspui/handle/123456789/10987 . — Назва з екрана.

22. Функції активації: важливість у контексті штучного інтелекту [Електронний ресурс] // clicknoob.fun. — Режим доступу: https://clicknoob.fun/technologies/funkcziyi-aktyvacziyi-vazhlyvist-u-konteksti-shtuchnogo-intelektu/. — Назва з екрана.

23. Liang X. Biased ReLU neural networks [Електронний ресурс] / XingLong Liang, Jun Xu // Neurocomputing. — 2021. — Т. 423. — С. 71 — 79. — Режим доступу: https://doi.org/10.1016/j.neucom.2020.09.050 . — Назва з екрана.

24. Поркуян О. В. Вплив функції активації лінійної нейронної мережі на апроксимацію даних основних каналів керування реактру синтезу оцтової кислоти [Електронний ресурс] / О. В. Поркуян, Ж. Г. Самойлова // Вісник Східноукраїнського національного університету імені Володимира Даля. — 2024. — № 1 (281). — С. 91 —97. — Режим доступу: https://doi.org/10.33216/1998-7927-2024-281-1-91-97 . — Назва з екрана.

25. Використання алгоритмів розпізнавання образів в системах відеоспостереження [Електронний ресурс] : thesis / О Длужевський А. — [Б. м.], 2015. — Режим доступу: http://er.nau.edu.ua/handle/NAU/19295 . — Назва з екрана.

26. Face Images Classification using VGG-CNN [Електронний ресурс] / I. Nyoman Gede Arya Astawa [та ін.] // Knowledge engineering and data science. — 2021. — Т. 4, № 1. С. 49. — Режим доступу: https://doi.org/10.17977/um018v4i12021p49-54 . — Назва з екрана.

27. Де використовується python і чому вам потрібно знати цю мову - genius.space [Електронний ресурс] // Genius.Space. — Режим доступу: https://genius.space/lab/de-vikoristovuyetsya-python-i-chomu-vam-potribno-znati-tsyu-movu/ . — Назва з екрана.

28. Програмування числових методів мовою Python : підруч. / А. В. Анісімов, С. Д. Погорілий, А. Ю. Дорошенко, Я. Ю. Дорогий ; за ред. А. В. Анісімова. — К. : Видавничо-поліграфічний центр "Київський університет", 2014. — 640 с.

29. Як зробити нейронну мережу на python [Електронний ресурс] // Bezpeka.zapisi.cx.ua. — Режим доступу: https://bezpeka.zapisi.cx.ua/ukraincyam/yak-zrobiti-neyronnu-merezhu-na-python.html . — Назва з екрана.

30. Analysis of the Best Optimizer Used by Convolutional Neural Network Algorithms in Detecting Masked Faces. 2023 Eighth International Conference on Informatics and Computing (ICIC). 2023.URL:https://doi.org/10.1109/ICIC60109.2023.10381927.

31. Nurhopipah A., Larasati N. A. CNN Hyperparameter Optimization using Random Grid Coarse-to-
fine Search for Face Classification. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control. 2021. P. 19—
26. URL: https://doi.org/10.22219/kinetik.v6i1.1185

32. Аношкін О. М. Програмне забезпечення системи розпізнавання графічних образів [Електронний ресурс] : thesis / Аношкін О. М., Конопліцька О. К. — [Б. м.], 2012. — Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4411 . — Назва з екрана.

33. freeCodeCamp. How to Detect Objects in Images Using the YOLOv8 Neural Network. freeCodeCamp.org. URL: https://www.freecodecamp.org/news/how-to-detect-objects-in-images-using-yolov8/ .

34. Методичні вказівки до виконання магістерських кваліфікаційних робіт студентами спеціальності 123 «Комп'ютерна інженерія». / Укладачі О. Д. Азаров, О. В. Дудник, С. І. Швець − Вінниця : ВНТУ, 2023. − 57 с.

ДОДАТОК А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ проф.,
д.т.н.. Азаров О.Д.

«__03__» __жовтня__ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи
«Інтегрована комп'ютерна система розпізнавання об'єктів на місцевості»

Науковий керівник: д.т.н., доц. каф. ОТ
_____Крупельницький Л.В.

Студент групи 2КІ-24м
_____Бондаренко К.О.

ВНТУ 2025

1 Підставою для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність магістерської кваліфікаційної роботи обумовлена необхідністю проєктування інтегрованих комп'ютерних систем розпізнавання об'єктів на місцевості з використанням сучасних методів комп'ютерного зору. Зростаючі обсяги візуальної інформації та вимоги до оперативності її обробки зумовлюють потребу у застосуванні автоматизованих методів аналізу зображень і відеопотоків для задач моніторингу та аналізу територій.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — розробка та дослідження інтегрованої комп'ютерної системи автоматичного розпізнавання об'єктів на зображеннях і відеопотоках місцевості з використанням методів глибокого навчання на основі архітектури YOLOv8.

2.2 Призначення розробки — інтегрована комп'ютерна система розпізнавання об'єктів призначена для автоматизованого виявлення, класифікації та аналізу об'єктів на місцевості з фіксацією результатів обробки.

3 Вихідні дані для виконання МКР

3.1 Цифрові зображення місцевості у форматах JPG та PNG.

3.2 Відеофайли у форматі MP4, що містять відеопотоки місцевості.

3.3 Анотовані набори даних з координатами об'єктів, призначені для навчання та тестування системи розпізнавання

3.4 Застосування бібліотеки Ultralytics.

3.5 Середовище розробки Visual Studio Code.

4 Вимоги до виконання МКР

4.1 Провести аналіз сучасних методів і систем розпізнавання об'єктів.

4.2 Розробити алгоритм розпізнавання об'єктів.

4.3 Реалізувати програмну систему розпізнавання об’єктів.

4.4 Провести тестування працездатності, точності та ефективності системи.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз методів та систем розпізнавання об’єктів	26.09.2025	07.10.2025	Розділ 1
2	Розробка моделі та алгоритму системи розпізнавання	08.10.2025	16.10.2025	Розділ 2
3	Програмна реалізація системи розпізнавання об’єктів	17.10.2025	27.10.2025	Розділ 3
4	Експериментальні дослідження та тестування системи	28.10.2025	31.10.2025	Розділ 4
5	Розрахунок економічної доцільності розробки	01.11.2025	04.11.2025	Розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	05.11.2025	09.11.2025	Розроблена програма, графічний матеріал
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	10.11.2025	10.11.2025	Оформлені документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія» (освітня програма «Комп’ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2023;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Інтегрована комп'ютерна система розпізнавання об'єктів на місцевості
Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)
Підрозділ кафедра Обчислювальної техніки, ФІТКІ, група 2КІ-24м
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 2 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Азаров О.Д. завідувач кафедри ОТ _____
(прізвище, ініціали, посада) (підпис)
Мартинюк Т.Б. гарант освітньої програми _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Крупельницький Л.В. доцент кафедри ОТ
(підпис) (прізвище, ініціали, посада)
Здобувач _____ Бондаренко К.О.
(підпис) (прізвище, ініціали)

ДОДАТОК В

Класифікація алгоритмів виявлення об’єктів за кількістю проходів зображення через мережу

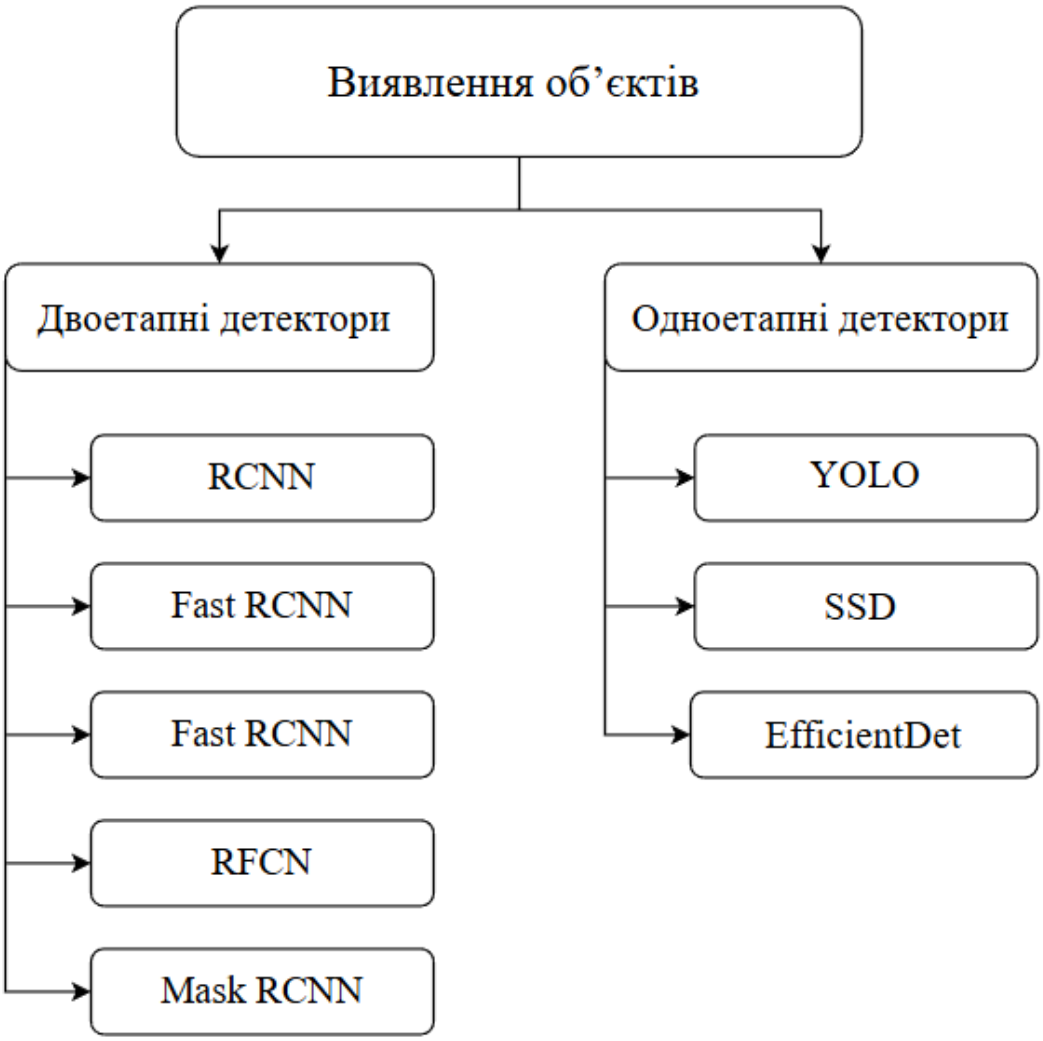


Рисунок В.1 — Класифікація алгоритмів виявлення об’єктів за кількістю проходів зображення через мережу

ДОДАТОК Г

Алгоритми видалення атмосферних спотворень з зображень

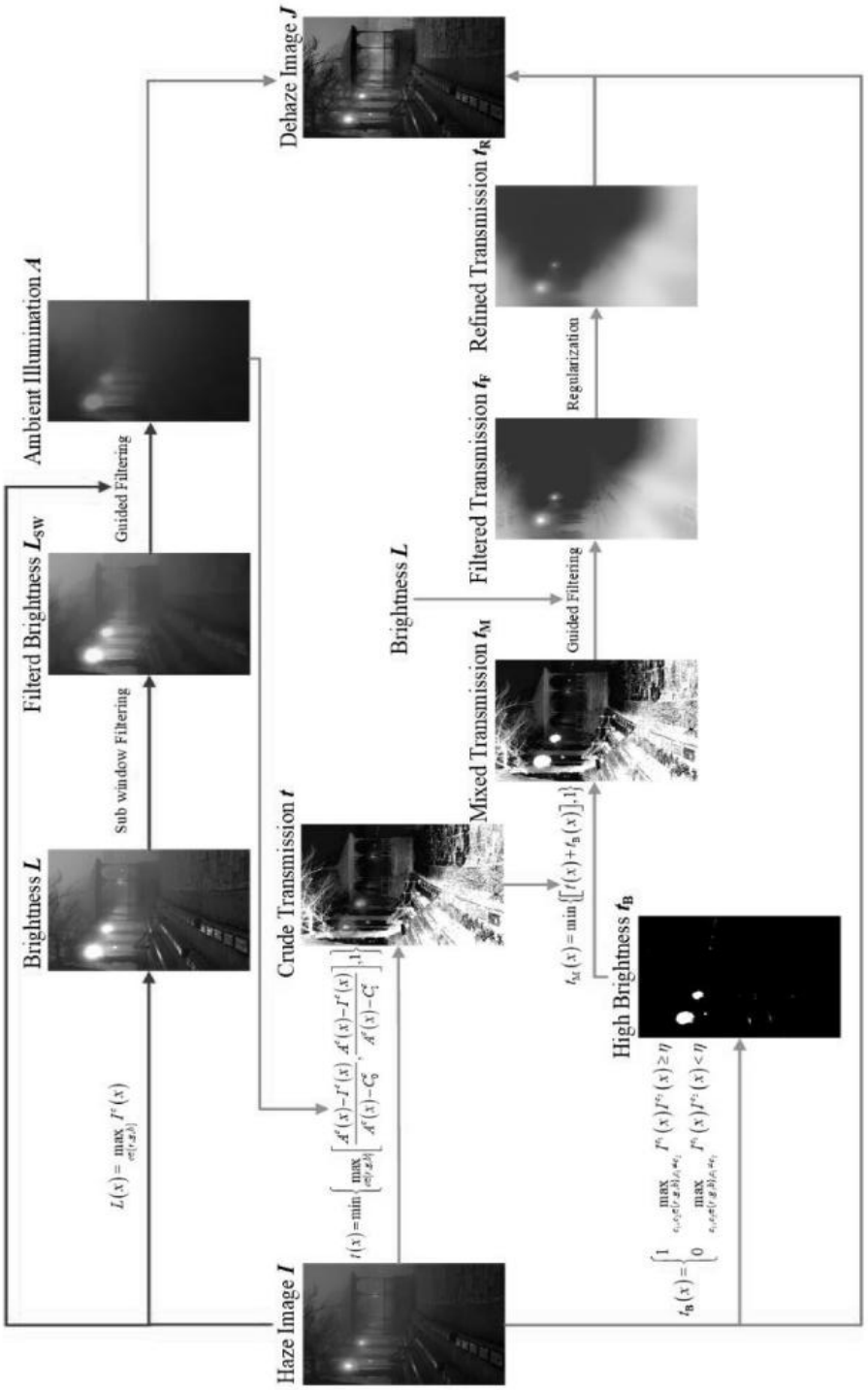


Рисунок Г.1 — Схема алгоритму видалення туману (dehazing) на основі оцінювання яскравості, передачі сцени та керованої фільтрації

ДОДАТОК Д

Текстовий вивід процесу інференсу YOLOv8 для відеоданих

```
video 1/1 (1225/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 bus, 1 truck, 63.0ms
video 1/1 (1226/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 bus, 1 truck, 63.5ms
video 1/1 (1227/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 truck, 67.5ms
video 1/1 (1228/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 1 truck, 74.0ms
video 1/1 (1229/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 1 truck, 64.5ms
video 1/1 (1230/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 62.0ms
video 1/1 (1231/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 1 bus, 61.0ms
video 1/1 (1232/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 62.0ms
video 1/1 (1233/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 bus, 64.5ms
video 1/1 (1234/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 60.0ms
video 1/1 (1235/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 70.5ms
video 1/1 (1236/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 1 truck, 62.0ms
video 1/1 (1237/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 1 bus, 2 trucks, 64.2ms
video 1/1 (1238/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 bus, 2 trucks, 67.5ms
video 1/1 (1239/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 bus, 60.5ms
video 1/1 (1240/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 2 buss, 62.0ms
video 1/1 (1241/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 4 cars, 2 buss, 67.1ms
video 1/1 (1242/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 59.5ms
video 1/1 (1243/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 1 truck, 61.0ms
video 1/1 (1244/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 2 buss, 1 truck, 62.5ms
video 1/1 (1245/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 64.5ms
video 1/1 (1246/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 8 cars, 1 bus, 61.5ms
video 1/1 (1247/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 1 train, 63.5ms
video 1/1 (1248/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 1 train, 64.0ms
video 1/1 (1249/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 8 cars, 1 bus, 2 trucks, 57.1ms
video 1/1 (1250/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 1 train, 2 trucks, 60.6ms
video 1/1 (1251/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 bus, 2 trucks, 62.5ms
video 1/1 (1252/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 bus, 2 trucks, 62.5ms
video 1/1 (1253/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 2 buss, 1 truck, 63.0ms
video 1/1 (1254/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 2 buss, 2 trucks, 67.0ms
video 1/1 (1255/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 2 buss, 1 truck, 71.5ms
video 1/1 (1256/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 64.3ms
video 1/1 (1257/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 1 bus, 1 truck, 62.0ms
video 1/1 (1258/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 8 cars, 1 bus, 67.1ms
video 1/1 (1259/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 8 cars, 1 bus, 62.5ms
video 1/1 (1260/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 8 cars, 1 bus, 69.5ms
video 1/1 (1261/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 8 cars, 1 bus, 62.5ms
video 1/1 (1262/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 9 cars, 1 bus, 71.0ms
video 1/1 (1263/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 2 buss, 1 truck, 64.0ms
video 1/1 (1264/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 2 buss, 1 truck, 66.5ms
video 1/1 (1265/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 1 truck, 60.2ms
video 1/1 (1266/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 1 bus, 1 truck, 62.5ms
video 1/1 (1267/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 1 truck, 65.0ms
video 1/1 (1268/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 1 truck, 63.0ms
video 1/1 (1269/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 8 cars, 1 truck, 62.0ms
video 1/1 (1270/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 2 trucks, 70.5ms
video 1/1 (1271/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 1 truck, 63.0ms
video 1/1 (1272/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 truck, 62.0ms
video 1/1 (1273/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 truck, 65.0ms
video 1/1 (1274/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 66.0ms
video 1/1 (1275/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 8 cars, 62.5ms
video 1/1 (1276/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 9 cars, 1 truck, 67.5ms
video 1/1 (1277/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 61.0ms
video 1/1 (1278/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 64.5ms
video 1/1 (1279/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 64.7ms
video 1/1 (1280/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 84.5ms
video 1/1 (1281/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 79.4ms
video 1/1 (1282/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 61.5ms
video 1/1 (1283/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 62.5ms
video 1/1 (1284/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 4 cars, 66.5ms
video 1/1 (1285/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 9 cars, 1 bus, 62.0ms
video 1/1 (1286/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 9 cars, 1 bus, 61.5ms
video 1/1 (1287/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 1 bus, 62.5ms
video 1/1 (1288/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 bus, 63.0ms
video 1/1 (1289/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 bus, 60.8ms
video 1/1 (1290/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 6 cars, 1 bus, 66.1ms
video 1/1 (1291/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 5 cars, 1 bus, 64.9ms
video 1/1 (1292/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 1 truck, 73.5ms
video 1/1 (1293/1314) D:\GitHub\YOLOv8\Implementation\demo.mp4: 384x640 7 cars, 1 bus, 2 trucks, 62.0ms
```

Рисунок Д.1 — Консольний вивід при детекції об’єктів на відео за допомогою YOLOv8

ДОДАТОК Е

Лістинг словника об’єктів, які може розпізнавати модель

```
print(result.names)
{0: 'person',
 1: 'bicycle',
 2: 'car',
 3: 'motorcycle',
 4: 'airplane',
 5: 'bus',
 6: 'train',
 7: 'truck',
 8: 'boat',
 9: 'traffic light',
 10: 'fire hydrant',
 11: 'stop sign',
 12: 'parking meter',
 13: 'bench',
 14: 'bird',
 15: 'cat',
 16: 'dog',
 17: 'horse',
 18: 'sheep',
 19: 'cow',
 20: 'elephant',
 21: 'bear',
 22: 'zebra',
 23: 'giraffe',
 24: 'backpack',
 25: 'umbrella',
 26: 'handbag',
 27: 'tie',
 28: 'suitcase',
 29: 'frisbee',
 30: 'skis',
 31: 'snowboard',
 32: 'sports ball',
 33: 'kite',
 34: 'baseball bat',
 35: 'baseball glove',
 36: 'skateboard',
 37: 'surfboard',
 38: 'tennis racket',
 39: 'bottle',
```

```
40: 'wine glass',
41: 'cup',
42: 'fork',
43: 'knife',
44: 'spoon',
45: 'bowl',
46: 'banana',
47: 'apple',
48: 'sandwich',
49: 'orange',
50: 'broccoli',
51: 'carrot',
52: 'hot dog',
53: 'pizza',
54: 'donut',
55: 'cake',
56: 'chair',
57: 'couch',
58: 'potted plant',
59: 'bed',
60: 'dining table',
61: 'toilet',
62: 'tv',
63: 'laptop',
64: 'mouse',
65: 'remote',
66: 'keyboard',
67: 'cell phone',
68: 'microwave',
69: 'oven',
70: 'toaster',
71: 'sink',
72: 'refrigerator',
73: 'book',
74: 'clock',
75: 'vase',
76: 'scissors',
77: 'teddy bear',
78: 'hair drier',
79: 'toothbrush'}
```

ДОДАТОК Ж

Лістинг веб-інтерфейсу для завантаження зображення та візуалізації результатів детекції об’єктів

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>YOLOv8 Object Detection</title>
    <style>
        canvas {
            display:block;
            border: 1px solid black;
            margin-top:10px;
        }
    </style>
</head>
<body>
    <input id="uploadInput" type="file"/>
    <canvas></canvas>
    <script>
        /**
        * "Upload" button onClick handler: uploads selected
        * image file to backend, receives an array of
        * detected objects and draws them on top of image
        */
        const input = document.getElementById("uploadInput");
        input.addEventListener("change",async(event) => {
            const file = event.target.files[0];
            const data = new FormData();
            data.append("image_file",file,"image_file");
            const response = await fetch("/detect",{
                method:"post",
                body:data
            });
            const boxes = await response.json();
            draw_image_and_boxes(file,boxes);
        })
        /**
        * Function draws the image from provided file
        * and bounding boxes of detected objects on
        * top of the image
        * @param file Uploaded file object
        * @param boxes Array of bounding boxes in format
```

```
        [[x1,y1,x2,y2,object_type,probability],...]
      */
      function draw_image_and_boxes(file,boxes) {
        const img = new Image()
        img.src = URL.createObjectURL(file);
        img.onload = () => {
          const canvas = document.querySelector("canvas");
          canvas.width = img.width;
          canvas.height = img.height;
          const ctx = canvas.getContext("2d");
          ctx.drawImage(img,0,0);
          ctx.strokeStyle = "#00FF00";
          ctx.lineWidth = 3;
          ctx.font = "18px serif";
          boxes.forEach(([x1,y1,x2,y2,label]) => {
            ctx.strokeRect(x1,y1,x2-x1,y2-y1);
            ctx.fillStyle = "#00ff00";
            const width = ctx.measureText(label).width;
            ctx.fillRect(x1,y1,width+10,25);
            ctx.fillStyle = "#000000";
            ctx.fillText(label,x1,y1+18);
          });
        }}
  </script>
</body>
</html>
```

ДОДАТОК И

Лістинг серверної частини веб-системи детекції об’єктів на основі YOLOv8

```
from ultralytics import YOLO
from flask import request, Response, Flask
from waitress import serve
from PIL import Image
import json

app = Flask(__name__)

@app.route("/")
def root():
    """
    Site main page handler function.
    :return: Content of index.html file
    """
    with open("index.html") as file:
        return file.read()

@app.route("/detect", methods=["POST"])
def detect():
    """
        Handler of /detect POST endpoint
        Receives uploaded file with a name "image_file",
        passes it through YOLOv8 object detection
        network and returns an array of bounding boxes.
        :return: a JSON array of objects bounding
        boxes in format
        [[x1,y1,x2,y2,object_type,probability],..]
    """
    buf = request.files["image_file"]
    boxes = detect_objects_on_image(Image.open(buf.stream))
    return Response(
      json.dumps(boxes),
      mimetype='application/json'
    )
def detect_objects_on_image(buf):
    """
    Function receives an image,
    passes it through YOLOv8 neural network
    and returns an array of detected objects
    and their bounding boxes
    :param buf: Input image file stream
```

```
    :return: Array of bounding boxes in format
    [[x1,y1,x2,y2,object_type,probability],..]
    """
    model = YOLO("best.pt")
    results = model.predict(buf)
    result = results[0]
    output = []
    for box in result.boxes:
        x1, y1, x2, y2 = [
          round(x) for x in box.xyxy[0].tolist()
        ]
        class_id = box.cls[0].item()
        prob = round(box.conf[0].item(), 2)
        output.append([
          x1, y1, x2, y2, result.names[class_id], prob
        ])
    return output
serve(app, host='0.0.0.0', port=8080)
```