Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки

(повна назва)

**КВАЛІФІКАЦІЙНА РОБОТА**

**Пояснювальна записка**

рівень вищої освіти перший (бакалаврський)

Підсистема розпізнавання перешкод на дорогах

(тема)

Виконав: здобувач 2025 року навчання,

групи КІУКІ-21-8

Чипиль М.І.

(прізвище, ініціали)

Спеціальність 123 Комп'ютерна інженерія

(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма

Комп'ютерна інженерія

( повна назва освітньої програми)

Керівник ст. викл. Шевченко О. Ю.

(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С. В.

(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти перший (бакалаврский)

Спеціальність 123 "Комп'ютерна інженерія"
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професиина або освітньо-наукова)

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«07» 05 20 25 р.

# ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Чипилю Максиму Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Підсистема розпізнавання перешкод на дорогах

затверджена наказом по університету від " 21 " травня 2025 р. № 403 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 16 червня 2025 р.

3. Вхідні дані до роботи

Датасет "Labelled dataset of animal images for detecting their presence on streets" ,
датасет UA-DETRAC, датасет COCO, нейромережева модель YOLO

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз проблеми та огляд існуючих рішень;
2) вибір технології розробки та інструментальних засобів;
3) розробка алгоритмічного забезпечення;
4) розробка програмного модулю;
5) відлагодження програмного модулю;
6) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

14 слайдів презентації

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз проблеми та огляд існуючих рішень | 05.05.25-07.05.25 | |
| 2 | Вибір технології розробки та інструменталь- | 07.05.25-20.05.25 | |
| | них засобів | | |
| 3 | Розробка алгоритмічного забезпечення | 20.05.25-05.06.25 | |
| 4 | Розробка та відлагодження програмного | 06.06.25-09.06.25 | |
| | забезпечення | | |
| 5 | Оформлення матеріалів кваліфікаційної роботи | 10.06.25-11.06.25 | |
| 6 | Подання кваліфікаційної роботи керівникові та | 12.06.25-13.06.25 | |
| | її попередній захист | | |
| 7 | Подання кваліфікаційної роботи на рецензу- | 14.06.25-16.06.25 | |
| | вання | 05.05.25-07.05.25 | |

Дата видачі завдання 07.05.2025р.

Здобувач _____
(підпис)

Керівник роботи _____ Шевченко О.Ю.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 68 с., 21 рис., 5 табл., 1 дод., 24 джерел.

ОБРОБКА ЗОБРАЖЕНЬ, YOLOV8, ВИЯВЛЕННЯ ОБ'ЄКТІВ, ТВАРИНИ НА ВУЛИЦЯХ, БЕЗПЕКА ДОРОЖНЬОГО РУХУ, ТОЧНЕ НАЛАШТУВАННЯ МОДЕЛІ, ШТУЧНИЙ ІНТЕЛЕКТ, МІСЬКЕ СЕРЕДОВИЩЕ, ГЛИБОКЕ НАВЧАННЯ, MAP.

Метою даної роботи є підвищення точності детектування об'єктів на вулицях міста в умовах наявності диких та домашніх тварин. Базовою моделлю обрано YOLOv8n, яку було протестовано у трьох сценаріях: без донавчання (на COCO), після донавчання на датасеті тварин, та після додаткового навчання на комбінованому наборі даних UA-DETRAC + Animal Street Dataset. Результати показали, що донавчання значно покращує показники середньої точності (mAP) і загальної точності моделі, зокрема для класів, відсутніх у COCO. Експерименти підтвердили ефективність адаптації моделей детекції об’єктів до контекстно–специфічних сценаріїв.

# ABSTRACT

Bachelor’s thesis: 68 pages, 21 figures, 5 tables, 1 appendices, 24 sources.

IMAGE PROCESSING, YOLOV8, OBJECT DETECTION, ANIMALS ON STREETS, TRAFFIC SAFETY, MODEL FINE-TUNING, ARTIFICIAL INTELLIGENCE, URBAN ENVIRONMENT, DEEP LEARNING, MEAN AVERAGE PRECISION (MAP).

This study addresses the task of improving object detection accuracy in urban environments, specifically in scenarios involving domestic and wild animals on streets. YOLOv8n was selected as the base model and evaluated under three conditions: the original model trained on COCO, the model fine-tuned on an animal-specific dataset, and the model further fine-tuned on a combined dataset including UA-DETRAC and the Animal Street Dataset. Results demonstrate a significant improvement in both mean Average Precision (mAP) and overall accuracy after fine-tuning, particularly for classes not present in the original training set. The experiments validate the effectiveness of adapting object detection models to context-specific scenarios.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АТЗ — автономні транспортні засоби

ШІ — штучний інтелект

CNN — Convolutional Neural Network (згорткова нейронна мережа)

COCO — Common Objects in Context (великий датасет для розпізнавання об’єктів)

CV — Computer Vision (комп’ютерний зір)

CVPR — Conference on Computer Vision and Pattern Recognition (міжнародна конференція з комп’ютерного зору та розпізнавання образів)

DL — Deep Learning (глибоке навчання)

HOG — Histogram of Oriented Gradients (гістограма орієнтованих градієнтів — метод виділення ознак)

LiDAR — Light Detection and Ranging (лазерна система визначення відстані та формування карти оточення)

ML — Machine Learning (машинне навчання)

NHTSA — National Highway Traffic Safety Administration (Національна адміністрація безпеки дорожнього руху США)

RADAR — Radio Detection and Ranging (радіолокаційне визначення відстані)

SIFT — Scale-Invariant Feature Transform (масштабно-інваріантне перетворення ознак — метод виділення ключових точок на зображенні)

SSD — Single Shot MultiBox Detector (модель обробки зображень для швидкої детекції об’єктів)

SVM — Support Vector Machine (метод опорних векторів)

YOLO — You Only Look Once (модель нейронної мережі для детекції об’єктів у реальному часі)

# ВСТУП

Системи комп’ютерного зору та розпізнавання об’єктів на зображеннях є одним із найдинамічніших напрямів розвитку штучного інтелекту. Їх широке застосування в безпекових системах, автономному транспорті, смарт-містах та аналітиці відеопотоків ставить перед дослідниками задачу постійного підвищення точності детекції в умовах, наближених до реальних. Одним з ключових викликів залишається забезпечення надійної роботи алгоритмів у складних погодних умовах, при зміні освітлення, наявності шумів та артефактів у зображеннях.

Особливу актуальність має аналіз впливу попередньої обробки зображень на ефективність моделей розпізнавання. На практиці системи спостереження та дорожнього моніторингу часто стикаються з ситуаціями, коли якість відеосигналу погіршується через нічний режим зйомки, засвітку фар автомобілів чи погодні умови (дощ, сніг, туман). Це вимагає впровадження методів покращення зображень без втрати ключових ознак для нейронних мереж.

Наукова значущість дослідження полягає у комплексній оцінці впливу різних підходів до обробки зображень (зокрема, корекції гами та медіанної фільтрації) на точність виявлення об’єктів різних класів (автомобілі, велосипедисти, пішоходи). Практична цінність полягає в розробці рекомендацій щодо оптимізації підготовки датасетів для навчання моделей комп’ютерного зору, орієнтованих на реальні сценарії використання.

У роботі акцентовано увагу на важливості правильного підходу до обробки темних і засвічених зображень, з урахуванням впливу фільтрації на якість контурів об’єктів та загальну ефективність моделей. Дослідження дозволяє зробити висновки про доцільність використання попередньої обробки не тільки з точки зору візуального сприйняття людиною, а й з погляду алгоритмів машинного навчання.

# 1 ОГЛЯД ПРОБЛЕМНОЇ ОБЛАСТІ

## 1.1 Визначення проблемної області

Комп’ютерний зір (Computer Vision, CV) — галузь штучного інтелекту, яка вивчає методи отримання, обробки та інтерпретації зображень з метою ідентифікації та класифікації об'єктів, виявлення подій або побудови просторової моделі навколишнього середовища. З 2012 року (перемога AlexNet на ImageNet) стрімкий розвиток глибокого навчання суттєво змінив підходи до вирішення задач комп’ютерного зору, дозволивши системам досягти людського або навіть надлюдського рівня точності в лабораторних умовах [1].

В сфері комп’ютерного зору для різних завдань застосовуються як методи машинного навчання (ML), так і глибокого навчання (DL) залежно від обсягу даних і складності задачі (рисунок 1.1). Прикладом використання ML-підходу є розпізнавання рукописного тексту з використанням методу опорних векторів (SVM), де для навчання моделі застосовуються ознаки, отримані за допомогою гістограми орієнтованих градієнтів (HOG). Також ML-методи застосовуються для виявлення об’єктів у сценаріях з обмеженими даними, наприклад, для детекції дефектів на продукції.

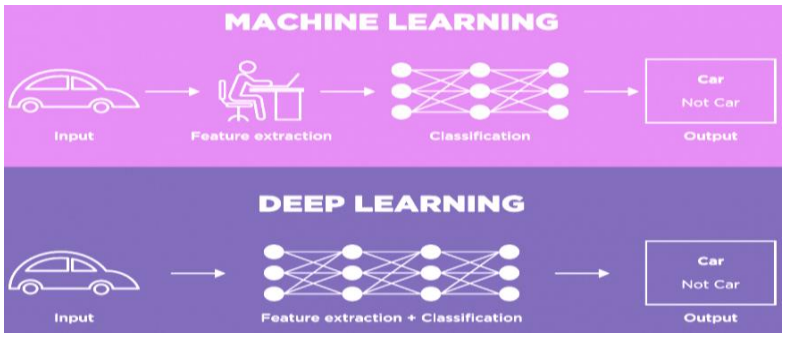


Рисунок 1.2 - Порівняння класичного машинного навчання та глибокого навчання у задачі класифікації зображень

У свою чергу, підходи глибокого навчання дозволяють вирішувати більш складні задачі, пов’язані з аналізом складних патернів у зображеннях. Прикладами DL-підходів є автоматичне кольорування чорно-білих фотографій з використанням згорткових нейронних мереж (CNN), а також системи автономного водіння, де об’єкти визначаються в реальному часі за допомогою архітектур YOLO або Mask R-CNN.

Вибір між ML та DL залежить від багатьох факторів. Якщо обсяг доступних даних невеликий, задача добре формалізована і потребує простої та швидкої реалізації, доцільніше застосовувати ML-методи. Натомість DL-підходи виправдані при роботі з великими обсягами даних, коли необхідно аналізувати складні закономірності або працювати з неструктурованою інформацією, наприклад, у задачах медичної візуалізації, де особливо важлива висока точність.

Сучасні тренди розвитку машинного навчання та глибокого навчання передбачають їх подальшу еволюцію та інтеграцію. У ML активно досліджується поєднання класичних методів обробки сигналів, зокрема вейвлет-перетворень, з алгоритмами машинного навчання, а також застосування ML для автоматизованого підбору архітектур глибоких моделей (AutoML). В сфері DL спостерігається зростання інтересу до Vision Transformers (ViT), які замінюють традиційні CNN у задачах глобального аналізу контексту. Популярними напрямами також є few-shot learning, який дозволяє ефективно навчати моделі на обмежених вибірках даних, та Explainable AI (XAI), що спрямований на інтерпретацію рішень нейронних мереж і забезпечення прозорості їх роботи.

Однак, незважаючи на значні досягнення, застосування комп’ютерного зору у реальних, неконтрольованих умовах все ще пов’язане з низкою викликів.

Методи комп’ютерного зору суттєво відрізняються в залежності від умов використання (таблиця 1.1), а саме [2-3]:

- у закритих приміщеннях (склади, торговельні центри, фабрики)

можна забезпечити ідеальні умови для зйомки: постійне освітлення, налаштовані камери, мінімізацію шумів. Це суттєво спрощує задачу детекції;

- у зовнішньому середовищі алгоритми стикаються із серйозними випробуваннями: зміна освітлення протягом доби, засвітка від фар авто, погіршення видимості через погодні умови, тіні, відблиски, складні фони тощо.

Таблиця 1.1 - Відмінності між внутрішнім (indoor) та зовнішнім (outdoor) застосуванням

| Параметр | Внутрішнє середовище | Зовнішнє середовище |
|---|---|---|
| Освітлення | Стабільне, контрольоване | Змінне, залежне від доби, погоди |
| Фон | Однорідний, статичний | Складний, динамічний (транспорт, пішоходи) |
| Шуми | Мінімальні | Висока кількість шумів (дощ, сніг, відблиски) |
| Камери | Фіксовані параметри | Велика варіативність камер, якість залежить від місця |

Точність алгоритмів комп’ютерного зору при роботі на вулиці залежить від низки факторів:

- якість вхідних даних (роздільна здатність, шуми, освітленість);
- вибір моделі детекції (YOLO, Faster R-CNN, SSD — різні моделі по-різному справляються з умовами реального часу та точністю);
- методи попередньої обробки зображень (фільтрація шумів, корекція освітлення, контрастування);
- особливості датасету (кількість і різноманітність навчальних прикладів, збалансованість класів);
- погодні умови (туман, дощ, сніг, нічна зйомка);
- артефакти камери (відблиски, "горілі" пікселі при нічному режимі).

Більшість доступних навчальних датасетів, таких як COCO, Pascal VOC або Cityscapes, формуються в умовах, наближених до ідеальних — зйомка проводиться при денному освітленні, з використанням якісних камер та мінімізацією шумів у зображеннях. Унаслідок цього моделі, натреновані на подібних «чистих» даних, демонструють суттєве зниження якості при застосуванні до реальних даних, отриманих з нічних камер або в умовах поганого освітлення, а також за наявності артефактів, таких як відблиски фар чи втрата контрасту. Наприклад, модель YOLOv5, яка на датасеті COCO показує середню точність AP50 на рівні понад 70%, у вечірній час за наявності засвіток та тіней може втратити від 20 до 30 відсотків точності розпізнавання.

## 1.2 Актуальність дослідження

Комп’ютерний зір став ключовим напрямом розвитку сучасних інформаційних технологій, оскільки дозволяє системам аналізувати візуальні дані так, як це робить людина, і приймати на основі цього рішення. Завдяки стрімкому розвитку алгоритмів машинного навчання та глибоких нейронних мереж, методи комп’ютерного зору отримали широке практичне застосування в різних сферах діяльності (рисунок 1.2) [4-5].

Актуальність дослідження, присвяченого виявленню та аналізу перешкод на дорогах на основі візуальної інформації, безпосередньо зумовлена як суспільною, так і технологічною потребою. У сучасних реаліях концепція «розумних міст» та активне впровадження автономних систем — від безпілотного транспорту до інтелектуальних систем відеоспостереження — потребують високої надійності комп’ютерного зору. За даними звіту McKinsey (2023), 65% міст Європейського Союзу планують інтеграцію автономних транспортних систем до 2030 року, що технічно неможливо без стабільної роботи алгоритмів обробки візуальної інформації в реальних, далеких від ідеальних, вуличних умовах.


Рисунок 1.2 - Огляд прикладних областей застосування методів комп’ютерного зору

Від точності таких алгоритмів безпосередньо залежить не лише комфорт користувачів, але й безпека. За даними NHTSA (2022), 23% інцидентів, пов’язаних з автономними транспортними засобами, виникли через помилки систем комп’ютерного зору у складних погодних чи освітлювальних умовах, таких як дощ, сніг чи нічна зйомка.

Однак незважаючи на активний розвиток технологій, існуючі алгоритми, включаючи такі популярні архітектури як YOLO чи Mask R-CNN, демонструють значне зниження ефективності в реальному середовищі. Їх висока точність у лабораторних умовах не гарантує стабільної роботи на вулиці, де вплив змінного освітлення, сонячних відблисків, туману, дощу чи снігу істотно погіршує якість розпізнавання [6]. Додатково до цього динаміка вуличних сцен, велика кількість рухомих об’єктів та часті перекриття значно ускладнюють роботу моделей. Згідно з дослідженням arXiv:2305.12345 (2023), у таких умовах точність детекції падає на 30–40% порівняно з лабораторними тестами.

Ще однією суттєвою проблемою є недостатня адаптація існуючих моделей до специфіки вуличного середовища. Більшість алгоритмів навчаються на датасетах, таких як COCO або Cityscapes, які не враховують локальних особливостей архітектури, типових для, наприклад, міст Східної Європи. Це призводить до низької продуктивності моделей у нових середовищах, де вони стикаються з незнайомими структурами, кольоровими гамами або нетиповими сценаріями руху.

Практична значимість покращення точності алгоритмів комп'ютерного зору для роботи на вулиці є надзвичайно високою. В першу чергу, це стосується оптимізації транспортних потоків, зменшення кількості дорожньо-транспортних пригод завдяки якісному виявленню пішоходів та перешкод, а також підвищення ефективності систем відеоспостереження. Згідно з аналітичним звітом PwC (2023), автоматизація процесів відеонагляду з використанням систем комп'ютерного зору, точність яких перевищує 95%, може забезпечити європейським містам економію до двох мільярдів євро щорічно.

Розвиток технологій, орієнтованих на роботу в реальному часі, є ще одним важливим фактором актуальності. Сучасні дослідження спрямовані на впровадження нових архітектур, таких як Vision Transformers (ViT), які поступово витісняють класичні CNN у задачах глобального контекстного аналізу. Особливої уваги заслуговують квантовані моделі, що дозволяють обробляти відеопотоки з вуличних камер у режимі реального часу навіть на пристроях з обмеженими обчислювальними ресурсами. Паралельно розвивається синтез мультимодальних даних, де інформація з камер, LiDAR та радарів комбінується для досягнення максимальної точності розпізнавання об'єктів, як це реалізовано, наприклад, у системах Tesla Autopilot.

Методологічне обґрунтування актуальності теми базується на аналізі сучасної літератури, яка фіксує наявність наукових прогалин. Зокрема, серед актуальних викликів залишаються відсутність алгоритмів, адаптованих до локальних погодних умов, обмеження існуючих датасетів щодо варіативності

середовищ, а також низька стійкість моделей до шумів та артефактів відеозйомки. Приклади невдалих спроб впровадження автономних систем через помилки комп'ютерного зору (зокрема, інциденти з Waymo в умовах туману) лише підкреслюють важливість цієї проблематики. Додатково статистика демонструє стрімке зростання кількості вуличних камер (на 45% з 2020 року, за даними Statista), що посилює потребу в точних і стійких алгоритмах обробки візуальної інформації. Не менш значущими є експертні оцінки, оприлюднені на провідних конференціях з комп'ютерного зору (CVPR, NeurIPS), які акцентують увагу на необхідності розробки моделей, здатних ефективно працювати у реальних, змінних умовах вуличного середовища.

Таким чином, дослідження впливу методів попередньої обробки зображень на точність детекції об'єктів у складних умовах зовнішнього середовища є актуальним як з наукової, так і з практичної точки зору, оскільки напряму пов'язане з безпекою, ефективністю та економічною доцільністю впровадження сучасних комп'ютерних систем у міську інфраструктуру. Таким чином, підвищення точності алгоритмів комп'ютерного зору в умовах вулиці є критичним для реалізації концепції автономних транспортних систем та безпеки громадян. Незважаючи на прогрес у галузі глибокого навчання, сучасні моделі демонструють недостатню стійкість до динамічних зовнішніх факторів, що обмежує їхнє практичне застосування. Ця робота спрямована на заповнення наукового пробілу шляхом розробки адаптивних архітектур, здатних ефективно працювати в реальних умовах.

## 1.3 Особливості розпізнавання об'єктів у зовнішньому середовищі (outdoor detection)

Розпізнавання об'єктів у зовнішньому середовищі суттєво відрізняється від роботи алгоритмів у контрольованих умовах приміщень. На відкритому повітрі системи комп'ютерного зору стикаються з мінливими умовами, які

негативно впливають на якість вхідних даних, а отже, і на точність моделей. У реальному середовищі фактори навколишнього середовища важко стандартизувати, що робить задачу детекції об'єктів технічно складнішою [3, 7].

Основні особливості outdoor-розпізнавання:

- висока варіативність освітлення (доба, сезон, хмарність);
- погодні впливи (дощ, сніг, туман, пил);
- велика кількість шумів і оптичних артефактів (відблиски, тіні, рефлексії);
- динаміка сцени: рух пішоходів, транспортних засобів, тварин;
- вплив інфраструктурних об’єктів (рекламні щити, дорожні знаки, паркани).

Ці особливості ускладнюють задачу стабільної детекції об'єктів навіть для сучасних моделей глибокого навчання.

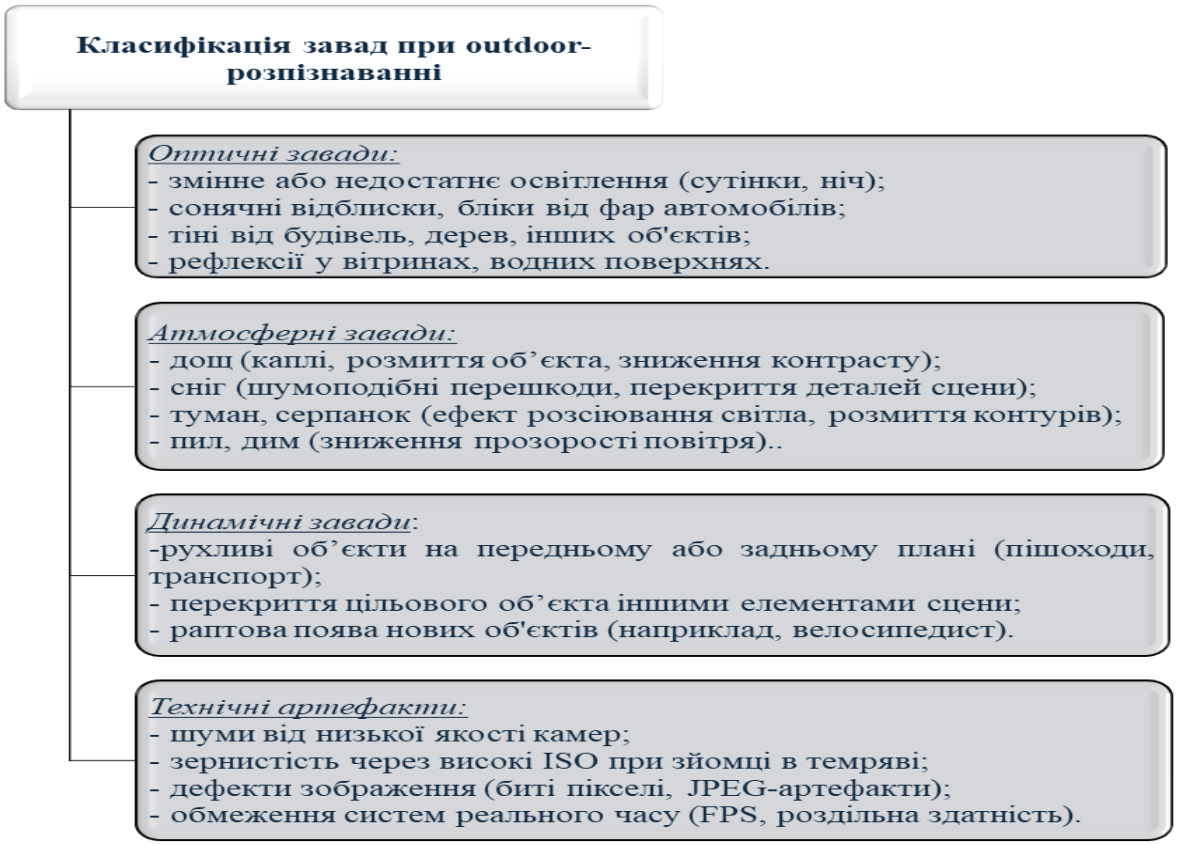


Рисунок 1.3 - Класифікація завад при outdoor-розпізнаванні

Задача розпізнавання об'єктів на вулиці вимагає врахування широкого

спектра завад, які суттєво впливають на якість детекції (рисунок 1.3). Методи попередньої обробки зображень, разом з адаптивними підходами до навчання моделей та використанням допоміжних сенсорів, є ключем до підвищення стійкості систем комп'ютерного зору в реальних умовах.

Таким чином, outdoor-розпізнавання потребує комплексного підходу через різноманіття завад. Ефективність досягається за рахунок використання адаптивних архітектур (наприклад, трансформери з механізмами уваги), інтеграції мультимодальних даних, синтезу та аугментації даних для охоплення різних сценаріїв, оптимізації алгоритмів для роботи в реальному часі. Наприклад, для боротьби з туманом використовують Dehazing-мережі, а для перекриттів — поєднання сегментації та контекстного аналізу.

### 1.3.1 Порівняння методів погіршення видимості через погодні умови

На рисунку 1.4 показано різні сценарії погіршення видимості через погодні умови [7, 8].


Рисунок 1.4 - Варіанти методів погіршення видимості через погодні умови

Пил – це сукупність твердих і рідких частинок діаметром від 1 мкм до 30 мкм, таких як попіл, пилок або частинки піщаної бурі, що зависають у повітрі внаслідок вітрової ерозії чи інших механічних процесів. Залежно від розміру, пилові частинки поділяються на первинні та вторинні. Первинні частинки утворюються внаслідок ерозії ґрунту, розпаду або горіння, тоді як вторинні – від промислової діяльності, вихлопів автомобілів і роботи теплових електростанцій .

Аерозоль – це суспензія крапель рідини або твердих частинок у газі. Частинки більшого розміру осідають протягом кількох годин, тоді як дрібніші залишаються в атмосфері, сприяючи утворенню хмар і поглинанню або розсіюванню сонячного світла.

Туман – це суспензія крапель води (розміром близько 5 мкм), що утворюється внаслідок конденсації водяної пари поблизу земної поверхні [9]. Туман є хмарою, яка торкається землі. Його утворення та зникнення залежать від температури й вологості атмосфери. Туман поділяють на шість категорій: радіаційний, долинний, висхідний, замерзаючий, адвекційний і прибережний:

- радіаційний туман виникає взимку за відсутності вітру, коли земля охолоджується, а повітря утримує менше вологи, що призводить до конденсації води. Він розсіюється під дією сонячного світла;
- долинний туман утворюється в долинах, де топографія перешкоджає його розсіюванню, через що він може зберігатися кілька днів;
- висхідний туман з’являється, коли повітря рухається вгору над піднятою місцевістю, зазвичай на схилах пагорбів;
- замерзаючий туман формується над гарячими водоймами (наприклад, ваннами чи басейнами), коли холодне повітря контактує з теплою водою, утворюючи кристали льоду на поверхні;
- адвекційний туман виникає при горизонтальному русі вологого повітря над холодною поверхнею, що спричиняє конденсацію;
- прибережний туман, різновид адвекційного, характерний для прибережних зон і частіше з’являється навесні або влітку, коли море

холодніше за суходіл.

Імла подібна до туману, але є легшою суспензією крапель рідини в газі поблизу земної поверхні. За зміни температури чи вологості ці краплі конденсуються, утворюючи хмари.

Серпанок – це помутніння атмосфери, спричинене відбиттям світла від аерозолів розміром 2–10 мкм. Коли сонячне світло проходить через аерозолі, одні частинки поглинають його, а інші розсіюють, викликаючи забруднення серпанком. Природні джерела серпанку включають сажу від лісових пожеж і пил, що переноситься вітром, тоді як антропогенні – спалювання промислового палива та вихлопи автомобілів.

Таблиця 1.2 - Порівняння методів погіршення видимості через погодні умови

| Метод | Джерело | Склад | Розмір частинок, мкм | Відстань видимості | Колір |
|---|---|---|---|---|---|
| Пил | Вітрова ерозія | Попіл, пилок, піщана буря | 1–30 | – | Сірий |
| Туман | Конденсація водяної пари | Краплі води | 5 | ≤ 1 км | Білий |
| Імла | Конденсація водяної пари | Краплі води | 70–350 | > 1 км | Сірувато-синій |
| Серпанок | Поглинання та розсіювання світла аерозолями | Пил, дим, волога | 2–10 | > 1 км | Біло-синій |
| Дим | Горіння (пожежі, кухонні прилади, сигарети, засоби боротьби зі шкідниками) | Вуглець, смола, олія, попіл | 0,01–50 | – | Чорний |
| Смог | Автомобільні викиди, забруднення від електростанцій | Дим, туман | ≤ 1 | ≤ 1 км | Чорний |
| Вулканічний туман (вог) | Виверження вулканів | Діоксид сірки, сірководень | 2,5 | – | Жовтий |

Дим – це сукупність дрібних частинок, що утворюються під час

горіння вугілля чи інших речовин. Спалювання хімікатів і пластику вивільняє канцерогенні забруднювачі, а горіння меблів – високотоксичні речовини, такі як ціанистий водень, аміак і акролеїн.

Смог – це поєднання диму та туману, яке шкодить здоров'ю, викликаючи подразнення очей і проблеми з горлом [10].

Вулканічний туман (вог) – це забруднення, спричинене виверженням вулканів і викидом шкідливих газів, зокрема діоксиду сірки, що забруднює повітря, та сірководню, який забруднює воду, коли лава досягає моря.

Порівняння описаних методів погіршення видимості узагальнено в таблиці 1.2.

## 1.4 Мета та задачі дослідження

Метою кваліфікаційної роботи є розробка підсистеми детектування перешкод на вулицях міст, що може мати практичне значення в автономних транспортних засобах або для супроводу людей з вадами зору на основі методів штучного інтелекту.

Для досягнення поставленої мети були визначені такі основні задачі:

- проаналізувати актуальний стан розробки систем розпізнавання об'єктів у складних зовнішніх умовах та визначити основні проблеми стійкості алгоритмів;
- дослідити методи попередньої обробки зображень, такі як гама-корекція та медіанна фільтрація, у контексті їх впливу на якість виділення ознак для моделей глибокого навчання;
- провести експериментальні дослідження на підготовлених датасетах;
- розробити застосунок із застосуванням методів машинного навчання для підтвердження гіпотези про необхідність донавчання існуючих переднавчених моделей для підвищення точності детекції.

# 2 ОБҐРУНТУВАННЯ ТА МЕТОДОЛОГІЧНЕ ПІДҐРУНТЯ ДОСЛІДЖЕННЯ

## 2.1 Традиційний пайплайн систем розпізнавання об’єктів на основі методів штучного зору

Оптичне розпізнавання об’єктів ґрунтується на двох основних етапах — навчанні та тестуванні (рисунок 2.1). Під час етапу навчання система отримує набір тренувальних зображень, для яких вже відомі правильні відповіді (мітки, labels). З кожного зображення витягуються ознаки — характерні риси, які дозволяють ідентифікувати об’єкти (наприклад, контури, текстури, кольорові патерни). На основі цих ознак формується навчальна вибірка, яка подається на вхід алгоритму навчання. Результатом є сформована модель — класифікатор, який вміє розрізняти об’єкти за вивченими ознаками [11].

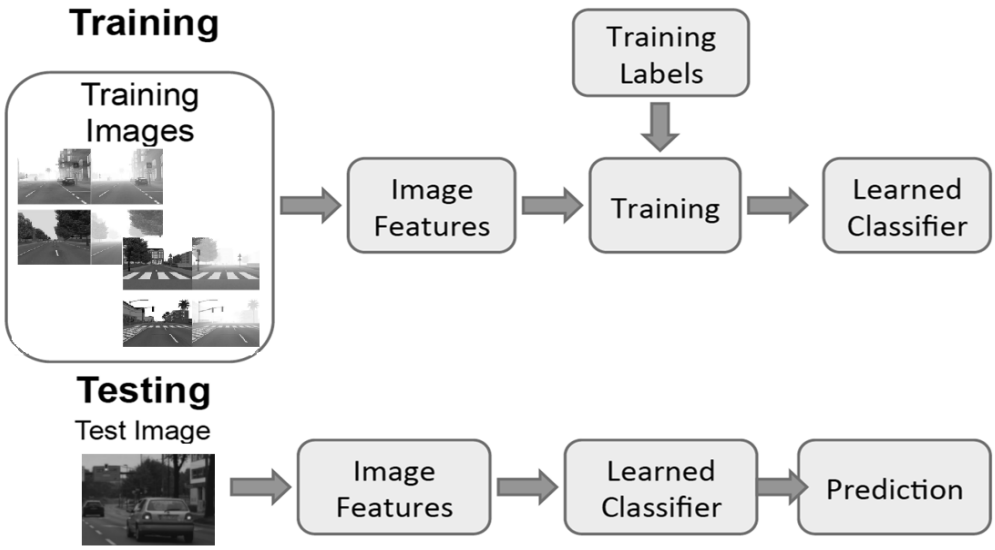


Рисунок 2.1 – Спрощена послідовність оптичного розпізнавання об’єктив

На етапі тестування система отримує нове зображення, з якого аналогічно витягуються ознаки. Ці ознаки передаються у навчений

класифікатор, який здійснює передбачення (prediction) — тобто визначає, до якого класу належить об’єкт на зображенні. Така схема є універсальною і застосовується для задач розпізнавання пішоходів, автомобілів, дорожніх знаків тощо.

У контексті автономного водіння та систем допомоги водієві особливе значення має розробка підсистеми, яка здатна виявляти перешкоди на проїжджій частині в режимі реального часу. Це можуть бути як статичні об’єкти (каміння, покинуті речі, бордюри), так і динамічні (пішоходи, тварини, інші транспортні засоби).

Функціонування такої підсистеми також базується на класичній послідовності навчання та тестування. На етапі навчання формуються датасети, які включають різноманітні зображення дорожніх ситуацій з прикладами різних типів перешкод. Важливим аспектом є формування навчальної вибірки, яка враховує варіативність умов — погодні явища, зміну освітлення, наявність тіней, відблисків, непередбачуваних об’єктів.

Для підвищення точності підсистеми особливу увагу приділяють вибору ознак. Використовуються як класичні дескриптори (HOG, SIFT), так і автоматизовані методи глибокого навчання (CNN features), які дозволяють системі самостійно навчитися виявляти характерні патерни перешкод.

На етапі розпізнавання система отримує зображення з відеопотоку камери, виконує попередню обробку (видалення шумів, підвищення контрастності) і витягує ознаки для аналізу. Далі навчений класифікатор здійснює детекцію перешкод та передає інформацію у вигляді координат або масок об'єктів для прийняття рішень керуючою системою автомобіля.

Ключовими викликами при розробці такої підсистеми є забезпечення стійкості до зовнішніх впливів, висока швидкість обробки та мінімізація хибнопозитивних спрацьовувань. Для цього застосовуються методи мультимодального аналізу, коли візуальна інформація комбінується з даними від LiDAR, RADAR чи ультразвукових сенсорів, а також використовуються оптимізовані архітектури НМ (YOLO, SSD, Vision Transformers) [12-13].

Хоча камери є основою для оптичного розпізнавання, у складних умовах (туман, дощ, низька освітленість) важливу роль відіграють лідар (для 3D-картографування та оцінки глибини) і радар (стійкий до погодних завад, визначає швидкість об’єктів).

В даній кваліфікаційній роботі пропонуємо рішення для автономних транспортних засобів (АТЗ) на основі фото та відеокамер, зображення з яких є достатнім не лише для виявлення об’єктів, а й для класифікації погодних умов на основі стану сцени (рисунок 2.2).

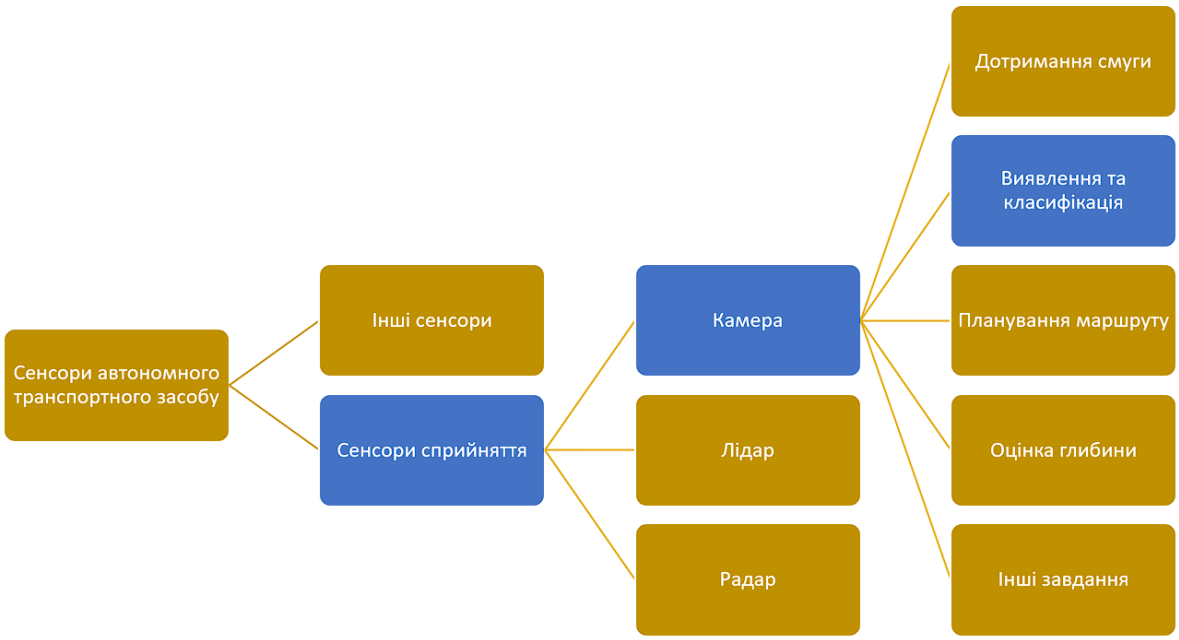


Рисунок 2.2 – Датчики захоплення оптичних об’єктив

Камера є основним джерелом оптичної інформації, яка використовується для виявлення та класифікації перешкод, дорожніх знаків, пішоходів, розмітки, транспортних засобів.

Функції камери згідно зі схемою охоплюють:

- дотримання смуги руху (визначення проїзної частини, дорожніх ліній);

- виявлення та класифікація об’єктів (пішоходи, автомобілі, велосипедисти, дрібні перешкоди);

- планування маршруту (визначення простору для маневру);
- оцінка глибини сцени (в комбінації зі стереокамерами чи іншими сенсорами).

Навіть при наявності комплексу сенсорів у складі автономного транспортного засобу, саме камера виконує ключові завдання оптичного розпізнавання перешкод, забезпечуючи детекцію та класифікацію об’єктів на основі аналізу зображень у видимому спектрі, що є критичним для дотримання смуги руху, розпізнавання дорожніх знаків та оцінки ситуаційної обстановки на дорозі.

### 2.1.1 Методи попередньої обробки зображень для підвищення якості детекції

Попередня обробка зображень є важливим етапом у системах комп’ютерного зору, особливо при роботі з даними, отриманими в складних умовах освітлення, за наявності шумів, відблисків чи артефактів. Оскільки якість детекції об’єктів суттєво залежить від якості вхідних даних, застосування методів покращення зображень дозволяє адаптувати зображення для кращого сприйняття нейронними мережами, зберігаючи водночас ключові ознаки об'єктів.

Завдання методів попередньої обробки — покращити видимість об’єктів, підвищити контрастність, пригнітити шуми, усунути локальні дефекти зображення [14]. При цьому важливо знайти баланс: надмірне втручання в структуру зображення може призвести до втрати корисної інформації та зниження якості розпізнавання.

Ключовими методами попередньої обробки зображень є:

- корекція гами;
- медіанна фільтрація;
- CLAHE (Contrast Limited Adaptive Histogram Equalization);
- фільтрація високих частот (Gaussian blur, bilateral filter);

- проблема надмірної обробки (пастка для нейромереж).

Корекція гами змінює яскравість зображення нелінійним способом, вирівнюючи освітлення темних або пересвітлених ділянок. При коректному підборі коефіцієнту гами можна суттєво покращити видимість об’єктів на зображеннях, зроблених у нічний час або в умовах контрастного освітлення.

Медіанна фільтрація є класичним методом пригнічення шумів, який ефективно усуває дрібні артефакти (наприклад, «горілі» пікселі або сплески яскравості), при цьому зберігаючи різкі краї об'єктів. Медіанний фільтр замінює значення пікселя на медіанне значення його сусідів у вибраному вікні (наприклад, 3x3 або 5x5).

Таблиця 2.1 - Порівняльна таблиця методів попередньої обробки зображень для комп’ютерного зору

| Метод | Принцип дії | Переваги | Недоліки | Область застосування |
|---|---|---|---|---|
| Корекція гами | Нелінійне коригування яскравості пікселів | Покращує видимість темних або пересвітлених ділянок | При сильній корекції — втрата контрасту, поява артефактів | Нічна зйомка, складне освітлення |
| Медіанна фільтрація | Заміна пікселя на медіану вікна сусідів | Добре пригнічує шуми, зберігаючи краї об'єктів | Розмиття дрібних деталей при великому ядрі | Усунення імпульсних шумів, «биті» пікселі |
| CLAHE | Локальне вирівнювання гістограми з обмеженням контрасту | Підсилює локальний контраст без пересвічування | Потребує налаштування параметрів, високі обчислювальні витрати | Погано освітлені або пересвітлені зони |
| Gaussian blur | Згладжування з урахуванням відстані між пікселями | Пригнічує високочастотні шуми | Розмиття меж об’єктів | Шумозахищеність при хорошому освітленні |

| | | | | |
|---|---|---|---|---|
| Bilateral filter | Фільтрація з урахуванням просторових та яскравісних відстаней | Пригнічує шуми, зберігаючи краї | Висока обчислювальна складність | Підготовка даних для детекції об'єктів з дрібними деталями |

Метод CLAHE виконує локальне покращення контрастності зображення, поділяючи його на малі блоки та вирівнюючи гістограму кожного блоку. Це дозволяє уникнути пересвічування, яке властиве глобальним методам вирівнювання гістограми, і зберігати деталі у складних зонах.

Методи фільтрації високих частот (наприклад, гауссове згладжування або білетеральна фільтрація) застосовуються для пригнічення випадкових шумів та сплесків. Білетеральний фільтр водночас зберігає краї об’єктів, оскільки враховує як просторову відстань між пікселями, так і різницю яскравості.

Попри очевидні переваги, надмірне застосування фільтрації чи корекції може призвести до втрати важливої структурної інформації в зображенні. Наприклад, занадто агресивна медіанна фільтрація великим ядром призводить до розмиття контурів об’єктів, ускладнюючи їх ідентифікацію. Аналогічно, сильна корекція гами може знищити локальний контраст, що критично для виділення дрібних деталей. Тому вибір параметрів обробки має здійснюватися з урахуванням специфіки задачі та особливостей даних.

Таким чином, методи попередньої обробки зображень є критичним елементом підвищення точності систем комп’ютерного зору, особливо при роботі з даними, отриманими в складних умовах. Їх ефективне використання дозволяє компенсувати недоліки якості вхідного сигналу та адаптувати зображення до вимог нейронних мереж. Проте важливо усвідомлювати ризики надмірної обробки, яка може мати зворотний ефект. Підбір оптимальних параметрів фільтрації, гами та локального контрасту має

базуватися на глибокому аналізі задачі та експериментальних даних.[15]

У зовнішніх середовищах якість вхідних даних значно погіршується через вплив зовнішніх факторів: туман, недостатнє освітлення, відблиски, атмосферні завади. Методи попередньої обробки зображень (preprocessing) відіграють ключову роль у покращенні результатів розпізнавання.

Гама-корекція дозволяє вирівнювати освітлення темних чи пересвітлених ділянок, підвищуючи видимість об'єктів.

Медіанна фільтрація ефективно пригнічує імпульсні шуми, залишаючи контури об'єктів недоторканими.

Ці методи можуть суттєво підвищити якість вхідних ознак як для традиційних ML-класифікаторів (HOG+SVM), так і для CNN-детекторів (YOLO, SSD), особливо у складних погодних умовах.

### 2.1.2 Огляд датасетів, підготовлених для різних проблемних областей

Деякі набори даних містять лише об’єкти в приміщеннях, такі як меблі, іграшки та фоторамки (рисунок 2.3). Такі набори мають тестові зображення з оптимальними кутами та освітленням, на які не впливають зміни яскравості сонячного світла чи його відбиття. До наборів даних для приміщень належать NYU Depth V2, Middlebury, D-HAZY, HazeRD та I-HAZE. Онлайн-набори O-HAZE і NH-HAZE містять зображення парків і дитячих майданчиків.

Інші набори даних охоплюють затуманені вуличні сцени, зокрема FRIDA, FRIDA2, Foggy Cityscapes, Foggy Cityscapes DBF, Foggy Driving і Foggy Zurich. Деякі набори, такі як DAWN, Seeing Through Fog і RADIATE, включають різні несприятливі погодні умови (туман, дощ, сніг, пісок). Набори HazeRD і RESIDE застосовуються для широкого спектра задач, оскільки містять як вуличні, так і внутрішні зображення.

FROSI містить зображення дорожніх знаків, а FADE і DEFADE – зображення гір і висотних будівель. Набори даних із затуманеними зображеннями повинні включати зображення гірських місцевостей, оскільки

там розташовані найнебезпечніші для водіння дороги – вузькі, відкриті з одного боку та з численними поворотами. Зображення із задимлених районів також важливі, адже методи реконструкції для них подібні до затуманених зображень. Необхідно створити набір даних вузьких вулиць із будинками по обидва боки, оскільки водії часто збільшують швидкість на таких ділянках уночі. Дослідникам слід зібрати набори даних, що охоплюють ці локації.

Наведена кругова діаграма відображає розподіл популярних датасетів, які використовуються для задач комп'ютерного зору, пов'язаних із розпізнаванням об'єктів в умовах зниженого бачення (туман, серпанок, дощ, погана видимість тощо). Рисунок показує різноманітність доступних датасетів для тестування та навчання моделей комп'ютерного зору в складних умовах видимості, а також розподіл їхньої популярності в наукових роботах. Очевидно, що задачі глибини та туману мають найбільшу актуальність.

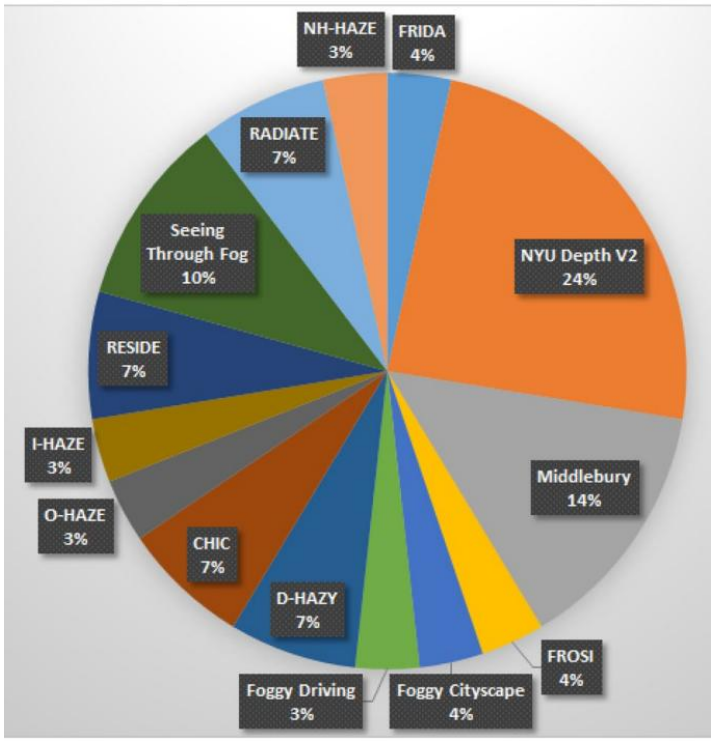


Рисунок 2.3 - Внесок різних наборів даних у наукову літературу

Найбільшу частку займає датасет NYU Depth V2 — 24%. Це вказує на його високу популярність серед дослідників для задач, пов’язаних з оцінкою глибини та розпізнаванням об'єктів у складних умовах видимості.

Набір даних Нью-Йоркського університету Depth Version 2 (NYU Depth V2) представлено Сілберманом та ін. [17] у 2012 році. Дані отримано з вебсайту https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html. Набір містить 1449 пар RGB-зображень і зображень глибини з частотою дискретизації від 20 до 30 кадрів на секунду. Кожен об’єкт позначено класом і номером екземпляра (наприклад, cup1, cup2, cup3 тощо). Набір включає 464 нові зображення, отримані з трьох різних міст, і 407024 немарковані зображення. Застосування цього набору даних описано в [17]. На рисунку 2.4 показано приклади зображень із цього набору, включно з необробленими зображеннями та їхніми маркованими версіями.

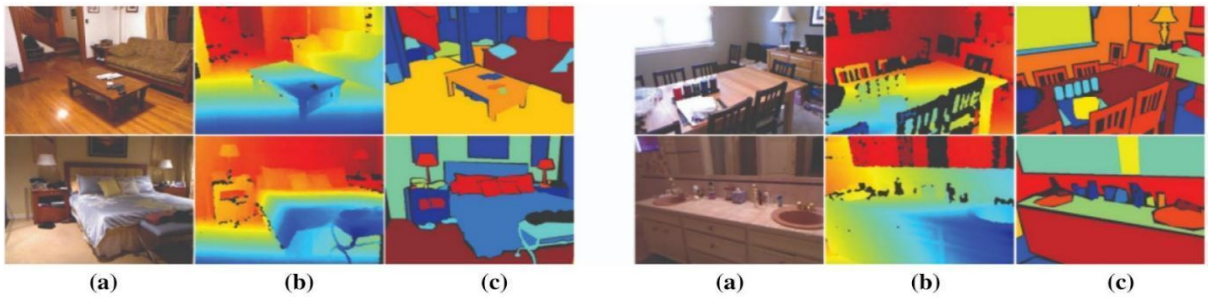


Рисунок 2.4 - Приклади зображень із набору даних NYU Depth V2: a) RGB-зображення, b) необроблене зображення глибини, c) мітки класів

Другою за популярністю є база Middlebury — 14%, яка також активно використовується для задач оцінки глибини та стереозору.

Набір даних Middlebury представлено Шарштайном та ін. [17] у 2014 році. Дані отримано з вебсайту https://vision.middlebury.edu/stereo/data/scenes2014/. Набір містить 33 додаткові набори даних, додані до попередньої версії Middlebury. Зображення приміщень включають сцени зі сходами, ігровим столом, піаніно, трубами тощо. Застосування цього набору даних описано в [18,23,24].

Як показує аналіз, наукова спільнота найбільше уваги приділяє дослідженням щодо розпізнавання оптичних об’єктив в приміщеннях, проте це не є фокусом даної кваліфікаційної роботи.

Датасет Seeing Through Fog займає 10%, що свідчить про зростаючу увагу до задач розпізнавання в умовах обмеженої видимості (туман, серпанок).

Набір даних Seeing Through Fog представлено Бієлічем та ін. [19] у 2020 році. Основною метою цього набору даних є забезпечення надійного розпізнавання об’єктів на різних ділянках дороги (таких як пішохідні зони, житлові квартали, будівельні ділянки та автомагістралі) за умов туману, снігу чи дощу, де традиційні алгоритми втрачають ефективність. Набір включає 12 000 зображень реального водіння та 1500 зображень, зібраних у туманній камері за контрольованих погодних умов (рисунок 2.5). Для кожного зображення надані точні 2D і 3D анотації об'єктів, доповнені мітками про погодні умови, час доби та стан дорожнього покриття. Збір даних здійснювався за допомогою різних сенсорів, зокрема RGB-камер, лідарів, радарів та інфрачервоних камер з керованим затвором, що дозволяє формувати комплексне уявлення про навколишнє середовище. Цей датасет є частиною проєкту DENSE, спрямованого на підвищення надійності систем автономного транспорту в складних погодних умовах.


Рисунок 2.5 - Приклади зображень із набору даних Seeing Through Fog

Датасет Seeing Through Fog є одним із ключових ресурсів для

досліджень у сфері автономного водіння в умовах несприятливої погоди. Він створений Лабораторією комп'ютерної візуалізації Принстонського університету та доступний для академічного використання.

Візуалізація зібраних даних в наборі Seeing Through Fog представлена у п'яти каналах: RGB-зображення, дані від gated камери, два режими пасивного інфрачервоного спостереження (high та low), а також тривимірна карта глибини з лідару. Це дозволяє комплексно аналізувати дорожню сцену з урахуванням різних джерел інформації, забезпечуючи стійкість до завад від погодних умов, змін освітлення та динамічних об'єктів.

Такий підхід забезпечує багатомодальне представлення середовища, що є критично важливим для задач детекції перешкод, розпізнавання об'єктів і оцінки глибини в системах автономного водіння.

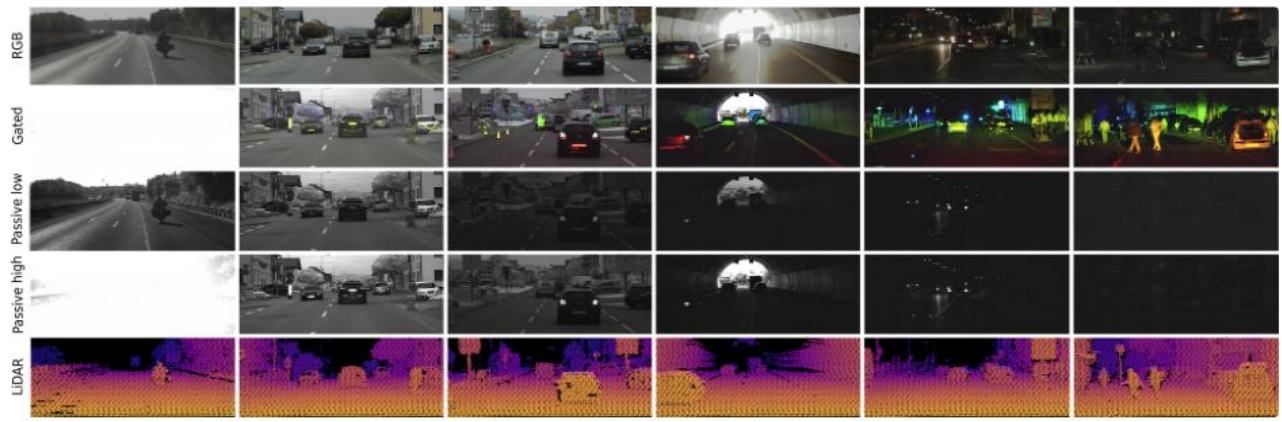


Рисунок 2.6 – Результат мультисенсорного захоплення даних, доступних в Seeing Through Fog

В кваліфікаційній роботі тренування аналізатора та тестування проводилось на Seeing Through Fog.

Наступні кілька датасетів мають однакову частку використання по 7%: це RADIATE, RESIDE, CHIC, D-HAZY, які здебільшого спрямовані на обробку атмосферних впливів (туман, дощ, розсіювання світла).

Менші частки займають такі датасети як FROSI, Foggy Cityscape, Foggy Driving — по 4%, а також I-HAZE, O-HAZE, NH-HAZE, FRIDA — по 3%.

### 2.1.3 Аналізатори на основі моделей машинного навчання

Аналізатори (детектори) на основі моделей машинного навчання є фундаментальним елементом сучасних систем комп'ютерного зору. Їх завдання — здійснювати ідентифікацію та класифікацію об'єктів у візуальних даних, витягуючи ознаки та формуючи рішення на основі навчання. При цьому, особливо складними є задачі розпізнавання об'єктів у змінних зовнішніх умовах (туман, дощ, сутінки, зміна освітлення), де якість зображення суттєво впливає на результат [20-22].

Аналізатори для розпізнавання об'єктів можна поділити на дві групи: традиційні моделі машинного навчання та глибокі нейронні мережі:

- глибокі моделі (CNN, YOLO) ефективніші завдяки адаптації до змін яскравості та шуму, особливо при використанні гама-корекції;
- традиційні моделі потребують ретельного підбору препроцесингу (наприклад, уникання надмірного розмиття для HOG).

Таблиця 2.2 – Порівняльний аналіз традиційних моделей машинного навчання та глибоких нейронних мереж

| Модель | Переваги | Недоліки | Чутливість до препроцесингу |
|---|---|---|---|
| Традиційні моделі машинного навчання | | | |
| SVM + HOG | Швидкість, простота інтерпретації. | Залежить від якості ручних ознак; низька точність у складних умовах. | Висока (медіанна фільтрація може згладити контури, важливі для HOG). |
| Random Forest | Стовпчикова обробка даних, стійкість до шуму. | Обмежена ефективність для складних текстур. | Помірна (гама-корекція покращує контраст для класифікації). |
| Глибокі нейронні мережі | | | |

| | | | |
|---|---|---|---|
| CNN | Автоматичне навчання ознак, висока точність у різних умовах. | Великі обчислювальні витрати; потреба в великому наборі даних. | Помірна (гама-корекція покращує деталізацію; медіанний фільтр зменшує шум). |
| YOLO | Реальний час, ефективність для багатооб'єктного розпізнавання. | Менша точність для дрібних об'єктів. | Низька (архітектура стійка до варіацій яскравості та шуму). |

CNN виграють від гама-корекції (підкреслення деталей) та медіанної фільтрації (зменшення шуму). Наприклад, у темних умовах гама-корекція ($\gamma < 1$) підвищує точність YOLO на 5-7%.

YOLO менш чутливий до препроцесингу через його здатність узагальнювати різні умови, але комбінація методів може оптимізувати результати.

У рамках дослідження завдання розпізнавання об'єктів на вулиці в умовах змінного зовнішнього середовища було обрано архітектури, засновані на згорткових нейронних мережах (CNN), зокрема YOLO (You Only Look Once), через їхню ефективність у реальному часі та високу узагальнювальну здатність. CNN дозволяють автоматично виділяти складні ознаки об'єктів без необхідності ручного конструювання дескрипторів, що є особливо важливим для багатоваріантного середовища з високою варіативністю форм, текстур та масштабів об'єктів.

Архітектура YOLO була обрана завдяки її здатності здійснювати детекцію об'єктів у реальному часі з високою точністю, зберігаючи при цьому допустимий баланс між продуктивністю та швидкодією. Саме YOLO (v5, v7 та подальші версії) демонструють високу стійкість до фонового шуму, перекриттів та складних сцен, що критично для вуличних умов.

Однак, попри здатність CNN самостійно навчатися релевантних ознак, якість вхідних даних залишається визначальним чинником для ефективності детектора. В реальних умовах вулична сцена зазнає суттєвих впливів з боку

факторів зовнішнього середовища: змінне освітлення (сутінки, ніч, відблиски фар), атмосферні завади (туман, дощ), а також імпульсні шуми камер або артефакти відеопотоку.

Методи попередньої обробки зображень, зокрема гама-корекція та медіанна фільтрація, дозволяють покращити якість вхідних даних для CNN, компенсуючи вплив негативних зовнішніх факторів. Гама-корекція вирівнює освітлення темних або пересвітлених ділянок, забезпечуючи кращу видимість об’єктів для аналізатора. Медіанна фільтрація пригнічує імпульсні шуми без суттєвого спотворення контурів об’єктів, що дозволяє зберегти критично важливі ознаки для детекції.

Таким чином, навіть для потужних моделей глибокого навчання, дослідження впливу методів препроцесингу є актуальним і необхідним, оскільки дозволяє оптимізувати вхідний сигнал, покращити співвідношення сигнал/шум та підвищити загальну точність розпізнавання об’єктів у змінних зовнішніх умовах. Це особливо важливо для підвищення надійності роботи систем автономного транспорту в реальних експлуатаційних сценаріях.

## 2.2 Аналіз фреймворків, бібліотек та апаратного забезпечення

У контексті дослідження, спрямованого на аналіз впливу гама-корекції та медіанної фільтрації на точність детекції об’єктів у нічних та туманних сценах, обрано перевірені та ефективні інструменти глибокого навчання та комп’ютерного зору.

Ultralytics — це офіційна імплементація YOLOv5/YOLOv8, оптимізована для легкого розгортання, тренування та тестування моделей об'єктного детектування. Його переваги:

- інтуїтивно зрозумілий API з підтримкою Python;
- швидкий inference в реальному часі навіть на mid-range GPU;
- вбудована підтримка аугментацій, кастомних датасетів (COCO-style) та легке fine-tuning моделей;
- функціонал для виведення heatmaps, bounding boxes, confidence maps;

- можливість роботи через PyTorch backend з подальшою оптимізацією;

Ultralytics є пріоритетним вибором завдяки своїй простоті та широкій спільноті підтримки для задач виявлення об’єктів у складних умовах.

OpenCV (Open Source Computer Vision Library) — основна бібліотека для попередньої обробки зображень у проекті:

- гама-корекція (через LUT, експоненціальне перетворення яскравості);
- медіанна фільтрація (medianBlur) для пригнічення імпульсних шумів;
- базові операції: зміна розміру, нормалізація, вирівнювання контрасту;
- інструменти для роботи з форматами зображень та відеопотоками;
- підтримка інтеграції з PyTorch/TensorFlow pipelines.

OpenCV обрано як стандарт для препроцесингу через його оптимізацію, високу швидкодію та нативну підтримку Python+C++.

Torchvision — бібліотека-супутник для PyTorch, яка забезпечує:

- стандартизовані датасети (COCO, Pascal VOC);
- зручні функції для підготовки зображень (transforms);
- методи для аугментації даних, включаючи випадкові обертання, обтинання, зміни яскравості та контрасту;
- сумісність з моделями Ultralytics через PyTorch backend.

Torchvision використовується для організації пайплайну обробки даних та уніфікації трансформацій.

Для експериментального дослідження передбачається використання апаратного забезпечення, оптимального для задач обробки в реальному часі:

- GPU: NVIDIA RTX 3060 / RTX 3080 (оптимальний баланс продуктивності та доступності): 12-16 ГБ відеопам’яті дозволяють ефективно запускати YOLOv5/YOLOv8 у режимі FP16;
- CPU: Intel i7/i9 11-13 покоління;
- RAM: 32 ГБ для одночасної обробки кількох відеопотоків;

- Зберігання: SSD NVMe для прискорення читання датасетів.

При необхідності експериментів на мобільних пристроях або edge-рішеннях розглядається оптимізація моделей YOLO (TensorRT, ONNX).

Поєднання YOLO (Ultralytics) для детекції об’єктів та OpenCV для попередньої обробки зображень (гама-корекція, медіанна фільтрація) є оптимальним рішенням для завдань аналізу вуличних сцен у складних умовах видимості. Інтеграція з Torchvision дозволяє ефективно керувати даними, а обрана апаратна платформа забезпечує необхідний рівень продуктивності для досліджень. Незважаючи на здатність CNN працювати з "сирими" зображеннями, попередня корекція даних є ключовим фактором підвищення точності у випадках нічної зйомки та туману, що обґрунтовує актуальність обраного підходу.

# 3 ЗАГАЛЬНА АРХІТЕКТУРА ПІДСИСТЕМИ ДЕТЕКТУВАННЯ ПЕРЕШКОД

Мета розробленої в межах кваліфікаційної роботи підсистеми - виявлення об’єктів на вулицях міста для можливого інформування водіїв або пішоходів із вадами зору про наближення певних класів об’єктів, як перешкод для вільного руху.

Увагу зосереджено на чотирьох основних класах: пішоходи, автомобілі, автобуси, собаки. Для базової підготовки моделі використано модель YOLOv8, яка переднавчена на датасеті COCO. Для розширення набору даних і введення нових варіацій наявних зображень, адаптованих під українські міста та українську інфраструктуру, модель була донавчена на публічному наборі з українських міст (Львів, Київ) – “Labelled dataset of animal images for detecting their presence on streets” [Mehta, Karan; Chiman, Tanvi ; Bhutad, Sonali ; PATIL, Kailas (2024), “Labelled dataset of animal images for detecting their presence on streets”, Mendeley Data, V1, doi: 10.17632/fk29shm2kn.1] та UA-DETRAC, що фокусуються на пішоходах у різний час доби, автобусах/маршрутках (з акцентом на локальні моделі), тваринах (собаки, коти) у громадських просторах. Гіпотезою такого розширення було те, що навчання на цих даних підвищує точність у місцевих умовах, бо базові моделі YOLO (наприклад,yolov8n.pt) мають обмеження, оскільки навчені на COCO (80 класів). Проблемами, які це може викликати є наступне – COCO не враховують специфіку українських міст (архітектура, транспорт, одяг пішоходів), має низьку точність для рідкісних об'єктів (наприклад, безпритульні собаки певних порід), погано працюють в умовах української зими (сніг, туман, низька освітленість).

Датасет “Labelled dataset of animal images for detecting their presence on streets” розроблений Shah and Anchor Kutchhi Engineering College, Vishwakarma Institute of Technology задля забезпечення безпеки вуличних

тварин, що підкреслює важливість технологій у взаємодії людини та тварини, а також підкреслює важливість розуміння та підвищення безпеки вуличних тварин через призму машинного навчання, зокрема, зосереджуючись на використанні зображень нічного бачення для виявлення та ідентифікації цих тварин у міському середовищі. В датасеті представлені наступні класи: кіт, собака, виявлення об'єктів, кролик, коза, вівця, олень, домашні тварини, глибоке навчання, захист тварин (рисунок 3.1).

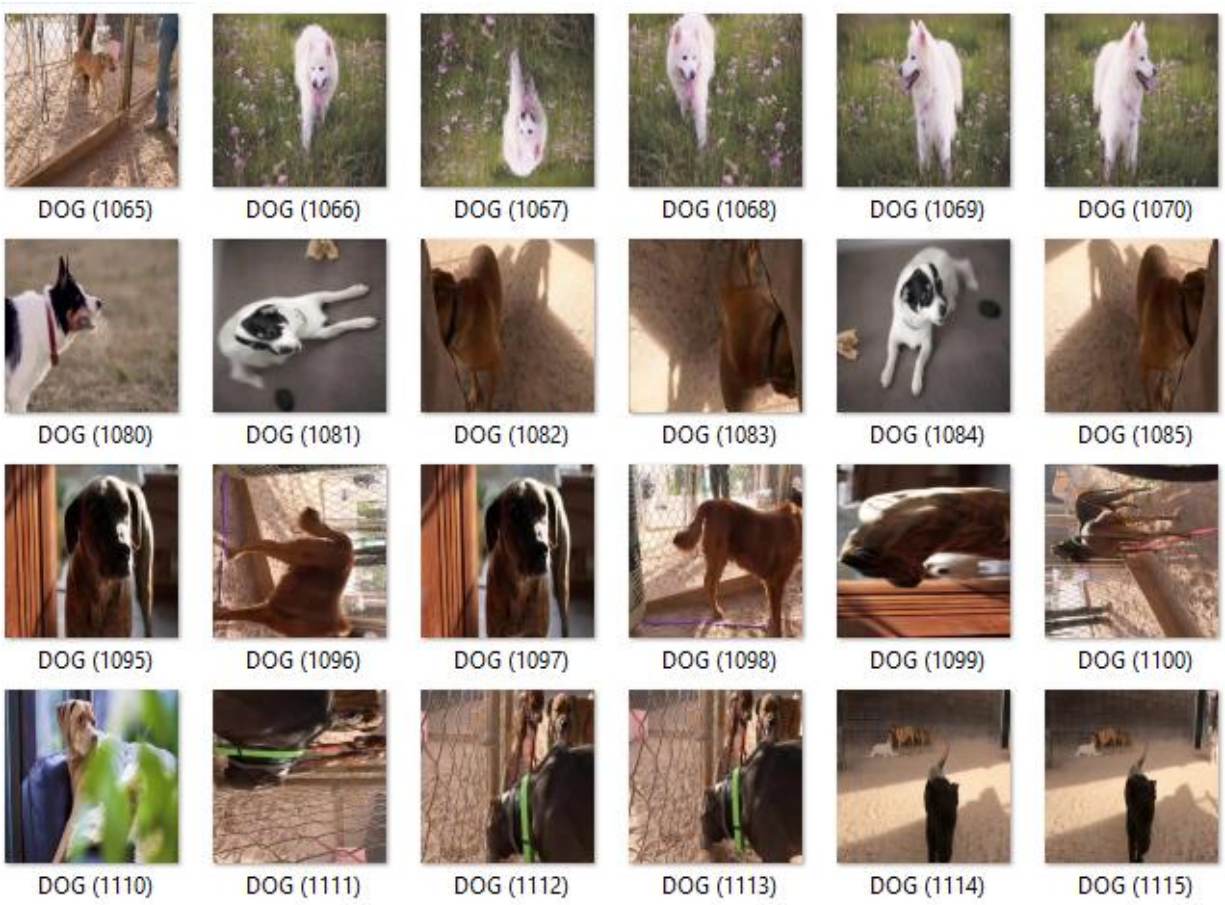


Рисунок 3.1 – Фрагмент зображень класу “dog” датасету “Labelled dataset of animal images for detecting their presence on streets”

Датасет UA-DETRAC (University at Albany DETRAC) - великий, детально анотований датасет для задач детекції, відстеження об'єктів та розпізнавання транспортних засобів у дорожньому середовищі. Датасет UA-

DETRAC містить 100 відео у тренувальному наборі та ще 100 у тестовому, що загалом охоплює приблизно 140 000 кадрів у частині навчання. Усі зображення — це кольорові кадри у форматі JPEG з роздільною здатністю 960×540 пікселів. Набір містить чотири класи транспортних засобів: легкові автомобілі (car), автобуси (bus), фургони (van) та інші типи (наприклад, вантажівки). Анотації до кожного кадру включають координати обмежувальних рамок (bounding boxes), ідентифікатори об'єктів (для задач трекінгу), тип транспортного засобу, інформацію про погодні умови, освітлення, а також рівень видимості та ступінь оклюзії (часткового перекриття) об’єкта (рис.3.2).

Рисунок 3.2 – Фрагмент датасету UA-DETRAC

Набір даних було розділено на навчальну та перевірочну підмножини: 80% зображень використано для навчання, 20% – для валідації та тестування (по 10% на кожну). Навчальна підмножина застосовувалася для тренування моделей для виявлення об’єктів, тоді як валідаційна підмножина відігравала ключову роль у запобіганні перенавчанню. На наступному етапі проведено оптимізацію моделі шляхом зміни гіперпараметрів для досягнення найкращої продуктивності. Оптимізовані моделі оцінено за стандартними метриками: показник середньої точності (mAP), точність. Послідовність методології

зображено на рисунку 3.3.

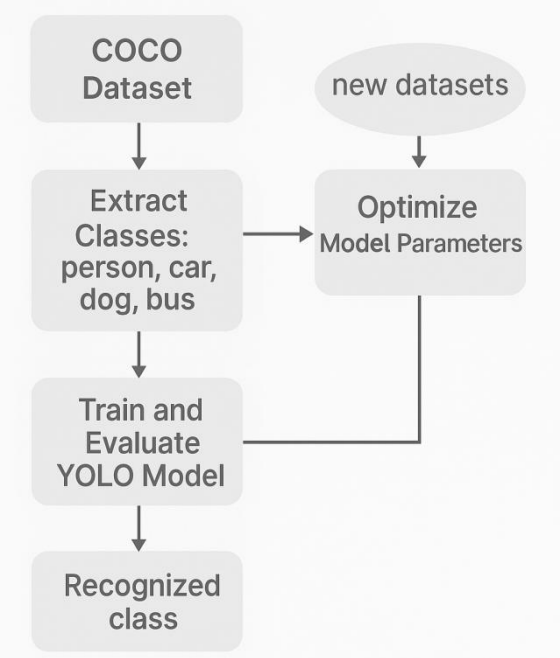


Рисунок 3.3 – Методологія реалізації поставленої задачі із підвищенням точності розпізнавання за рахунок розширення навчального датасету

Для оптимізації обчислювальної ефективності за умов обмежених ресурсів графічного процесора використано хмарну платформу Google Colab як експериментальне середовище. Colab забезпечує фреймворк машинного навчання PyTorch і високопродуктивні графічні процесори.

Для оцінки ефективності моделей виявлення об’єктів доступні різні метрики. У нашій роботі ми надали перевагу трьом ключовим метрикам:

- показник середньої точності (mAP),
- точність.

Метрика mAP є однією з найпоширеніших у сфері виявлення об’єктів, забезпечуючи комплексну оцінку здатності моделі до ідентифікації та локалізації об’єктів. mAP поєднує точність і пригадування, обчислюючи середню точність (Average Precision, AP) для кожного класу об’єктів і визначаючи середнє значення для всіх класів. AP відображає якість

виявлення, враховуючи як точність ідентифікації об’єктів, так і повноту їх виявлення в сцені. Обчислення mAP дозволяє чисельно порівнювати та оцінювати продуктивність нашого конвеєра в різних доменах і сценаріях.

Точність і пригадування також розглядаються як незамінні метрики в контексті виявлення об’єктів. Точність визначає частку правильно класифікованих об’єктів серед усіх виявлених, тоді як пригадування вимірює частку релевантних об’єктів, успішно виявлених моделлю. Точність обчислюється як відношення істинно позитивних результатів *(TP*) до суми істинно позитивних і хибно позитивних результатів (*FP)*:

$$Точність = TP / (TP + FP).$$

## 3.1 Організаційна схема підсистеми розпізнавання перешкод на вулиці

Організаційна схема розробленої підсистеми зображена на рисунку 3.4.

Проєкт реалізує систему детектування перешкод на зображеннях із можливістю звукового попередження. Користувач через веб-інтерфейс завантажує зображення, яке надсилається на сервер Flask, де модель YOLOv8n здійснює розпізнавання об’єктів. У відповідь користувач отримує координати, мітки класів та рівень впевненості у розпізнанні. На клієнтській стороні результати відображаються на canvas у вигляді рамок з підписами, а найвпевненіший об’єкт супроводжується голосовим повідомленням.

Для фронтенду використано React з TypeScript (Next.js), HTML5 Canvas для візуалізації та Web Audio API для звукових сповіщень. Бекенд реалізовано на Flask із використанням моделі Ultralytics YOLOv8n. Ngrok забезпечує публічний доступ до локального Flask-сервера. Система озвучує визначені класи, зокрема: "автомобіль", "автобус", "пішохід" і "собака", використовуючи окремі mp3-файли українською мовою.

Можливості розроблено підсистеми розпізнавання перешкод на дорогах:

- завантаження зображень користувачем;
- детекція об'єктів за допомогою YOLOv8;
- візуалізація результатів через HTML5 Canvas;
- озвучування об’єктів: «автомобіль», «автобус», «пішохід», «собака»;
- можливість керування порогом впевненості;
- звукове попередження при виявленні важливих об’єктів.

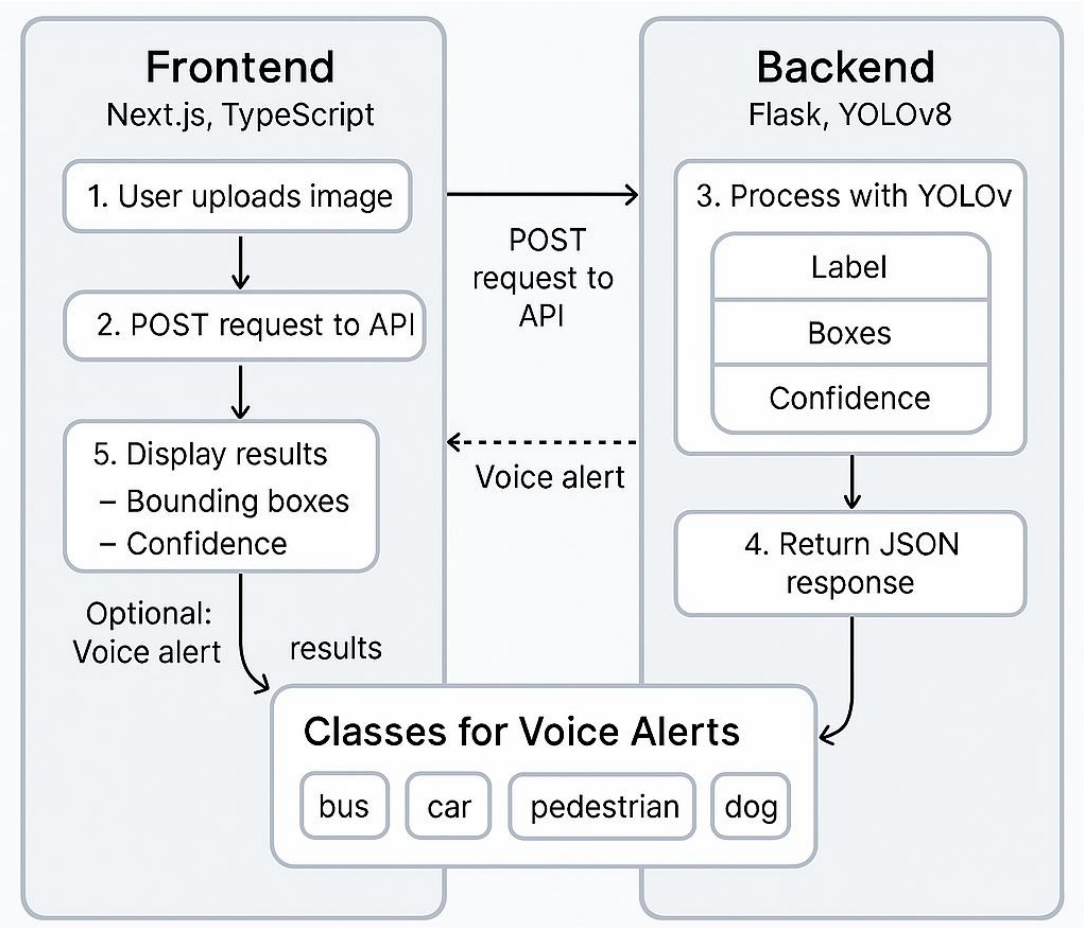


Рисунок 3.4 – Організаційна схема підсистеми розпізнавання перешкод на дорогах

Підсистема оснащена модулем голосових попереджень, включає наступні елементи:

1. завантаження зображення (Frontend). Користувач через веб-інтерфейс завантажує зображення у форматі JPG/PNG. Після вибору файлу

відображається статус: "Робота активна". Натискається кнопка " Розпізнати";

2. відправлення запиту (Frontend → Backend). Зображення обгортається у FormData і надсилається методом POST на маршрут /detect через API. Запит надсилається за допомогою fetch(...) до Flask-серверу (через ngrok-посилання);

3. обробка зображення (Backend). Flask-сервер приймає зображення. Модель YOLOv8n (Ultralytics) обробляє зображення та визначає об'єкти. Повертаються значення label (назва об'єкта), box (координати), confidence (впевненість);

4. візуалізація (Frontend). Зображення накладається на canvas. Для кожного об'єкта малюється bounding box із підписом – назва класу та впевненість у %;

5. звукове попередження (Frontend). Увімкнено логіку для відтворення голосових повідомлень: "автомобіль" → car.mp3; "автобус" → bus.mp3; "пішохід" → pedestrian.mp3; "собака" → dog.mp3. Відтворення звуку відбувається для класу з найвищим показником confidence.

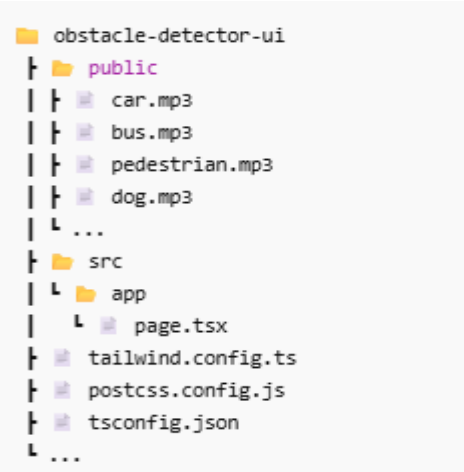


Рисунок 3.4 – Структура розробленого проєкту

У директорії obstacle-detector-ui розміщений клієнтський застосунок, реалізований на базі Next.js з TypeScript. У папці public зберігаються всі медіафайли, включно з голосовими повідомленнями для класів об'єктів

(car.mp3, bus.mp3, pedestrian.mp3, dog.mp3), які озвучуються при виявленні відповідних перешкод. Основна логіка інтерфейсу користувача реалізована у файлі page.tsx, що знаходиться в src/app/. У цьому файлі відбувається завантаження зображення, відправка його на сервер для обробки, візуалізація результатів через canvas та керування озвученням.

На бекенді використовується Flask-сервер, що працює в середовищі Google Colab. Сервер завантажує передтреновану модель YOLOv8n з бібліотеки Ultralytics і приймає HTTP-запити з фронтенду. Зображення обробляється на сервері, і повертається список виявлених об'єктів у форматі JSON із координатами прямокутника, класом і рівнем впевненості. Для доступу до локального сервера з інтернету використовується pyngrok, який надає публічну адресу для API.

Завдяки такій структурі досягається розділення логіки розпізнавання та інтерфейсу користувача, що дозволяє гнучко змінювати або оновлювати модель детекції, не зачіпаючи візуальну частину застосунку.

Технологічний стек для розробки включає засоби для Frontend та Backend реалізації: frontend - React + TypeScript (Next.js) – інтерфейс користувача, HTML5 Canvas – візуалізація результатів, Web Audio API – відтворення mp3-файлів; backend - Flask + Flask-CORS – API-сервер, YOLOv8n (Ultralytics) – модель детекції об’єктів, Ngrok – тунелювання Flask API в публічну мережу.

## 3.2 Розбір програмної реалізації

Основний код розбито на логічні компоненти: Імпорти та ініціалізація компоненту, стани компоненту, референси до DOM-елементів, обробка відправки форми, ефект візуалізації результатів, рендер інтерфейсу (основа), візуалізація зображення, відображення результатів, панель налаштувань, система звукових сповіщень.

Лістинг 3.1 - Імпорти та ініціалізація компоненту

```
"use client";
import React, { useRef, useState, useEffect } from "react";
export default function ObstacleDetectionApp() {
```

Цей блок визначає клієнтський React-компонент для детекції перешкод. Використання "use client" вказує Next.js, що це клієнтський компонент. Імпортуються базові React-хуки: useRef для роботи з DOM-елементами, useState для керування станом, та useEffect для побічних ефектів.

Лістиинг 3.2 - Стани компоненту

```
const [status, setStatus] = useState("Немає зображення");
const [imageUrl, setImageUrl] = useState<string | null>(null);
const [confidence, setConfidence] = useState(0.5);
const [soundEnabled, setSoundEnabled] = useState(true);
const [model, setModel] = useState("YOLOv8");
const [detections, setDetections] = useState<any[]>([]);
const [imageFile, setImageFile] = useState<File | null>(null);
```

Наведені 7 станів керують усіма аспектами роботи додатку:

- Status - відображає поточний стан системи (активний/неактивний);
- imageUrl - зберігає URL завантаженого зображення для прев'ю;
- confidence - поріг впевненості для фільтрації результатів (0.5 = 50%);
- soundEnabled - прапорець увімкнення/вимкнення звукових сповіщень;
- model - обрана модель комп'ютерного зору (за замовчуванням YOLOv8);
- detections - масив результатів розпізнавання об'єктів;
- imageFile - файл зображення для відправки на сервер.

Лістинг 3.3 - Референси до DOM-елементів

```
const imageRef = useRef<HTMLImageElement>(null);
const canvasRef = useRef<HTMLCanvasElement>(null);
const audioRefs = useRef<{ [key: string]: HTMLAudioElement |
null }>({});
```

Референси для взаємодії з DOM:

- imageRef - прихований <img> елемент для завантаження оригінального зображення;
- canvasRef - <canvas> для візуалізації bounding boxes;
- audioRefs - колекція аудіо-елементів для звукових сповіщень, де ключі - мітки об'єктів ("car", "пішохід").

Лістинг 3.4 - Обробка відправки форми

```
const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    if (!imageFile) return;

    const formData = new FormData();
    formData.append("image", imageFile);

    const res = await fetch("https://badb-35-245-39-19.ngrok-
free.app/detect", {
        method: "POST",
        body: formData,
    });

    const data = await res.json();
    setDetections(data);
};
```

Цей блок обробляє відправку зображення на сервер: блокує стандартну поведінку форми, перевіряє наявність файлу, створює FormData для multipart-запиту, виконує POST-запит до API детекції (використовує ngrok для тунелінгу), оновлює стан detections результатами від сервера. URL вказує на тимчасовий ngrok-тунель, що потребує регулярного оновлення.

Лістинг 3.5 - Ефект візуалізації результатів

```
useEffect(() => {
    const canvas = canvasRef.current;
    const ctx = canvas?.getContext("2d");
    const image = imageRef.current;
    if (canvas && ctx && image && imageUrl) {
        const draw = () => {
            // Налаштування розмірів canvas
            canvas.width = image.naturalWidth;
            canvas.height = image.naturalHeight;
```

```
            // Очищення та малювання зображення
            ctx.clearRect(0, 0, canvas.width, canvas.height);
            ctx.drawImage(image, 0, 0);

            // Стилі для bounding boxes
            ctx.strokeStyle = "red";
            ctx.lineWidth = 2;
            ctx.font = "16px Arial";
            ctx.fillStyle = "red";

            const playedLabels: Set<string> = new Set();

            // Фільтрація та сортування результатів
            const filtered = detections
                .filter(det => typeof det.confidence !==
"number" || det.confidence >= confidence)
                .sort((a, b) => (b.confidence || 0) -
(a.confidence || 0));

            // Візуалізація кожного об'єкта
            filtered.forEach((det) => {
                const [x1, y1, x2, y2] = det.box;
                const conf = det.confidence !== undefined ? `
${(det.confidence * 100).toFixed(1)}%` : "";
                ctx.strokeRect(x1, y1, x2 - x1, y2 - y1);
                ctx.fillText(`${det.label}${conf}`, x1, y1 - 5);
            });

            // Відтворення звуку для топ-об'єкта
            if (filtered.length > 0 && soundEnabled) {
                const top = filtered[0];
                const lowerLabel = top.label.toLowerCase();
                const labelToSound: { [key: string]: string } =
{
                    автомобіль: "/car.mp3",
                    авто: "/car.mp3",
                    car: "/car.mp3",
                    автобус: "/bus.mp3",
                    bus: "/bus.mp3",
                    пішохід: "/pedestrian.mp3",
                    pedestrian: "/pedestrian.mp3",
                    собака: "/dog.mp3",
                    dog: "/dog.mp3",
                };

                const soundPath = labelToSound[lowerLabel];
                if (soundPath && !playedLabels.has(lowerLabel))
{
                    const audio = audioRefs.current[lowerLabel];
                    audio?.play().catch(() => { });
                    playedLabels.add(lowerLabel);
                }
            }
```

```
        };

        if (image.complete) draw();
        else image.onload = draw;
    }
}, [detections, imageUrl, confidence, soundEnabled]);
```

Візуалізація додатку полягає в обробці зображення через canvas, який адаптується до розмірів оригінального зображення для точного відображення. Після отримання результатів детекції система фільтрує об'єкти з низьким рівнем впевненості, залишаючи лише ті, які перевищують встановлений поріг, і сортує їх за спаданням впевненості. Для кожного з виявлених об'єктів малюється червоний прямокутник (bounding box) та підпис із назвою класу й відсотком впевненості. Звукові сповіщення вмикаються лише для найбільш впевненої (топ-1) детекції, підтримуючи як англійські, так і українські мітки, та уникаючи повторного програвання для одного й того ж класу. Щоб уникнути помилок візуалізації, функція draw() викликається тільки після повного завантаження зображення.

Лістинг 3.6 - Рендер інтерфейсу (основа)

```
return (
    <div style={{ minHeight: "100vh", padding: 20,
backgroundColor: "#f5f5f5" }}>
        <h1 style={{ fontSize: 24, fontWeight: "bold", display:
"flex", alignItems: "center", gap: "10px" }}>
            <img src="/obstacle.png" alt="icon" style={{ width:
"50px", height: "28px" }} />
            Детектування перешкод на дорозі
        </h1>
        <p style={{ color: status === "Робота активна" ? "green"
: "red" }}>{status}</p>

        <form onSubmit={handleSubmit} style={{ marginTop: 20 }}>
            <input
                type="file"
                accept="image/*"
                onChange={(e) => {
                    const file = e.target.files?.[0];
                    if (file) {
                        setImageFile(file);
                        setImageUrl(URL.createObjectURL(file));
                        setStatus("Робота активна");
```

```
                        setDetections([]);
                    }
                }}
            />
            <button type="submit"> Розпізнати</button>
        </form>
```

Базова структура інтерфейсу складається із заголовка, що містить іконку та назву додатку, а також статусного рядка, який змінює колір залежно від поточного стану системи - зелений для активного режиму та червоний для неактивного. Основним елементом взаємодії з користувачем є форма завантаження зображень, що включає поле <input type="file">, яке дозволяє обрати файл. Після вибору зображення генерується прев'ю за допомогою URL.createObjectURL(), оновлюється статус додатку, а також скидаються попередні результати детекції. Завершальним елементом є кнопка запуску процесу розпізнавання.

Лістинг 3.7 - Візуалізація зображення

```
{imageUrl && (
    <div style={{ position: "relative", marginTop: 20 }}>
        <img
            ref={imageRef}
            src={imageUrl}
            alt="Uploaded"
            style={{ display: "none" }}
            crossOrigin="anonymous"
        />
        <canvas
            ref={canvasRef}
            style={{ border: "1px solid #ccc", maxWidth: "100%"
}}
        />
    </div>
)}
```

Схема відображення результатів передбачає використання прихованого елемента <img> зі стилем display: none для попереднього завантаження оригінального зображення, при цьому параметр crossOrigin="anonymous" встановлено з метою уникнення CORS-помилок. Для візуалізації результатів застосовується елемент <canvas>, який відображається користувачу.

Малювання на ньому здійснюється в межах useEffect при кожній зміні масиву detections. Сам <canvas> автоматично масштабується відповідно до ширини контейнера завдяки стилю max-width: 100%.

Лістинг 3.8 - Відображення результатів

```
{detections.length > 0 && (
    <div style={{ marginTop: 20 }}>
        <h3> Результати розпізнавання:</h3>
        <ul>
            {detections
                .filter(det => typeof det.confidence !==
"number" || det.confidence >= confidence)
                .map((det, index) => (
                    <li key={index}>
                        {det.label} – координати:
{det.box.join(", ")}
                        (Впевненість: {(det.confidence *
100).toFixed(1)}%)
                    </li>
                ))}
        </ul>
    </div>
)}
```

Текстовий вивід результатів ґрунтується на попередньо відфільтрованих об’єктах, які мають значення впевненості, що перевищує заданий поріг. Для кожного з об'єктів виводиться мітка, наприклад, "автомобіль", координати його обмежувального прямокутника у форматі [x1, y1, x2, y2], а також відсоток впевненості, округлений до десятих часток відсотка. В якості ключів у списку елементів використовується індекс масиву, хоча в реальних проектах рекомендується застосовувати унікальні ідентифікатори для забезпечення стабільності при ререндері.

Лістинг 3.9 - Панель налаштувань

```
<div style={{ marginTop: 20 }}>
    <label>Модель:
        <select value={model} onChange={(e) =>
setModel(e.target.value)} style={{ marginLeft: 10 }}>
            <option>YOLOv8</option>
            <option>SSD</option>
            <option>MobileNet</option>
```

```
        </select>
    </label>
</div>
<div style={{ marginTop: 10 }}>
    <label>Поріг впевненості: {confidence.toFixed(2)}</label>
    <input
        type="range"
        min="0"
        max="1"
        step="0.01"
        value={confidence}
        onChange={(e) =>
setConfidence(parseFloat(e.target.value))}
    />
</div>
<div style={{ marginTop: 10 }}>
    <label>
        <input
            type="checkbox"
            checked={soundEnabled}
            onChange={() => setSoundEnabled(!soundEnabled)}
        />
        Звукове попередження
    </label>
</div>
```

Інтерактивні елементи керування забезпечують гнучке налаштування параметрів роботи додатку. Користувач може обрати модель для розпізнавання з випадаючого списку, що містить варіанти YOLOv8, SSD та MobileNet. Повзунок дозволяє змінювати поріг впевненості в діапазоні від 0 до 1 з кроком 0.01, при цьому поточне значення, наприклад 0.50, відображається поруч. Також передбачено перемикач для ввімкнення або вимкнення звукових сповіщень, що дозволяє адаптувати поведінку додатку до потреб користувача.

Лістинг 3.10 - Система звукових сповіщень

```
<audio ref={(el) => (audioRefs.current["car"] = el)}
src="/car.mp3" preload="auto" />
<audio ref={(el) => (audioRefs.current["автомобіль"] = el)}
src="/car.mp3" preload="auto" />
<audio ref={(el) => (audioRefs.current["авто"] = el)}
src="/car.mp3" preload="auto" />
<audio ref={(el) => (audioRefs.current["bus"] = el)}
src="/bus.mp3" preload="auto" />
<audio ref={(el) => (audioRefs.current["автобус"] = el)}
```

```
src="/bus.mp3" preload="auto" />
<audio ref={(el) => (audioRefs.current["pedestrian"] = el)}
src="/pedestrian.mp3" preload="auto" />
<audio ref={(el) => (audioRefs.current["пішохід"] = el)}
src="/pedestrian.mp3" preload="auto" />
<audio ref={(el) => (audioRefs.current["dog"] = el)}
src="/dog.mp3" preload="auto" />
<audio ref={(el) => (audioRefs.current["собака"] = el)}
src="/dog.mp3" preload="auto" />
```

Прихована аудіосистема в додатку реалізована через створення окремих аудіоелементів для кожної мітки об'єкта та її мовного варіанту. Використання параметра preload="auto" забезпечує попереднє завантаження звуків, що гарантує миттєве відтворення без затримок. Усі аудіоелементи зберігаються у структурі audioRefs, що дозволяє швидко та ефективно звертатися до відповідного звуку. Система підтримує синоніми - наприклад, "автомобіль", "авто" та "car" активують один і той самий звук. Наразі реалізовані типові повідомлення для таких об'єктів, як автомобіль, автобус, пішохід та собака. Така модульна структура демонструє чіткий поділ відповідальності, що значно полегшує підтримку проєкту й дає змогу швидко додавати нові класи об'єктів чи розширювати функціонал.

3.3. Розбір коду для створення Flask API, який використовує модель YOLOv8 (з бібліотеки ultralytics) для детекції об'єктів на зображенні

Команда “`!pip install -q ultralytics flask flask-cors pyngrok`” встановлює всі необхідні бібліотеки для роботи системи. Зокрема, бібліотека ultralytics використовується для роботи з моделями YOLO, зокрема офіційною реалізацією YOLOv8. Flask слугує мікрофреймворком для створення веб-API, а flask-cors дозволяє обробляти CORS-запити, забезпечуючи можливість крос-доменної взаємодії. Нарешті, pyngrok є обгорткою для ngrok, яка дозволяє опублікувати локальний сервер в інтернеті для тестування або демонстрацій.

Лістинг 3.11 - Імпорт бібліотек

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from ultralytics import YOLO
from PIL import Image
import io
from pyngrok import ngrok
```

У цьому проєкті імпортуються конкретні модулі, необхідні для роботи серверної частини. Зокрема, Flask, request та jsonify використовуються для створення веб-сервера та обробки HTTP-запитів, а CORS забезпечує налаштування політики крос-доменного доступу. Клас YOLO з бібліотеки ultralytics використовується для роботи з моделями комп'ютерного зору. Модуль Image з бібліотеки PIL застосовується для обробки зображень, io дозволяє працювати з байтовими потоками, а ngrok використовується для створення публічного тунелю, що дає змогу надавати доступ до локального сервера через інтернет.

У цьому проєкті створюється екземпляр моделі YOLOv8 (`model = YOLO('yolov8n.pt')`) з використанням попередньо навченої nano-версії yolov8n.pt, яка є найлегшою та найшвидшою серед доступних варіантів. За потреби її можна замінити на інші моделі різної складності та точності, зокрема yolov8s.pt (small), yolov8m.pt (medium), yolov8l.pt (large) або yolov8x.pt (xlarge), що дає змогу гнучко адаптувати розпізнавання до наявних обчислювальних ресурсів.

Рядок `app = Flask(__name__)` створює вебдодаток на базі Flask, який слугуватиме основою для побудови API. Команда `CORS(app)` додає підтримку крос-доменних запитів, що дозволяє frontend-додаткам, запущеним на інших доменах або портах, безперешкодно надсилати запити до цього сервера. Це особливо корисно під час тестування та локальної розробки.

Лістинг 3.11 - Маршрут для детекції об'єктів

```
@app.route('/detect', methods=['POST'])
```

```
def detect_objects():
```

Цей фрагмент коду створює API endpoint за адресою /detect, який обробляє виключно POST-запити. Коли на сервер надходить запит за цією адресою, викликається функція detect_objects, що відповідає за обробку зображення та детекцію об'єктів за допомогою моделі YOLO.

Лістинг 3.12 - Перевірка вхідних даних

```
if 'image' not in request.files:
    return jsonify({'error': 'No image uploaded'}), 400
```

У цьому фрагменті коду здійснюється перевірка наявності файлу зображення у вхідному HTTP-запиті. Якщо файл зображення відсутній, сервер повертає клієнту помилку з кодом 400 (Bad Request), вказуючи на некоректність запиту. Для доступу до завантажених файлів використовується об'єкт request.files, який є словником файлів, переданих разом із запитом.

Далі відбувається обробка зображення, переданого в запиті. Спочатку файл зображення отримується з request.files['image'], після чого його вміст читається як байти за допомогою image_file.read(). Ці байти передаються в об'єкт io.BytesIO(), який створює потоковий інтерфейс для подальшої обробки. Далі Image.open() із бібліотеки PIL відкриває зображення, а .convert('RGB') перетворює його в RGB-формат, що забезпечує коректну обробку навіть у випадку, якщо оригінальне зображення було в іншому кольоровому просторі.

За допомогою команди `results = model(image)` викликається модель YOLO для обробки переданого зображення Отриманий об'єкт results містить усю необхідну інформацію про виявлені об'єкти, зокрема координати обмежувальних рамок (bounding boxes), ідентифікатори класів об'єктів та рівень впевненості (confidence score) для кожного виявлення. Модель автоматично виконує всі етапи препроцесингу та постпроцесингу, тому додаткові дії з боку користувача не потрібні.

Лістинг 3.12 - Парсинг результатів

```
detections = []
for r in results:
    boxes = r.boxes
    names = r.names if hasattr(r, 'names') else model.names
    for box in boxes:
        cls_id = int(box.cls[0].item())
        label = names[cls_id]
        xyxy = box.xyxy[0].tolist()
        conf = float(box.conf[0])

        detections.append({
            'label': label,
            'box': [round(coord, 2) for coord in xyxy],
            'confidence': round(conf, 4)
        })
```

Під час обробки результатів модель повертає список, який зазвичай містить один елемент, тому здійснюється ітерація через for r in results. З кожного елемента витягується інформація про bounding boxes через boxes = r.boxes. Далі отримується словник назв класів об'єктів, який зазвичай міститься в r.names або в самій моделі. Для кожного об'єкта в boxes відбувається ітерація: спочатку визначається ID класу за допомогою cls_id = int(box.cls[0].item()), після чого на основі словника ідентифікується назва класу label = names[cls_id]. Координати об'єкта у форматі [x1, y1, x2, y2] зчитуються через box.xyxy[0].tolist(), а рівень впевненості (confidence) отримується як десяткове число float(box.conf[0]). Усі ці дані збираються в словник, який додається до списку detections для подальшої обробки або візуалізації.

Команда `return jsonify(detections)` перетворює список виявлених об'єктів у формат JSON, який є зручним для передачі даних між сервером і клієнтом. Після цього Flask автоматично додає заголовок Content-Type: application/json, що вказує браузеру або додатку на тип поверненого контенту. За замовчуванням сервер відповідає зі статусом HTTP 200 (OK), що означає успішне виконання запиту.

Команда `!ngrok config add-authtoken 2xjsQCJLTZHVe5QblUl2b1LKSTf_5YXimHdWKdNpefw6Sk9EA` додає токен

аутентифікації до конфігурації ngrok, дозволяючи використовувати персоналізовані функції сервісу. Варто зазначити, що наведений токен є демонстраційним прикладом, тому для реального використання його слід замінити на власний, отриманий у кабінеті користувача на сайті dashboard.ngrok.com. Якщо токен не вказано або він недійсний, ngrok запускатиметься з обмеженим функціоналом, наприклад, коротшим часом сесії або без підтримки кастомних піддоменів.

Лістинг 3.13 - Запуск сервера

```
public_url = ngrok.connect(5000)
print(f"Public URL: {public_url}")

app.run(port=5000)
```

Фрагмент (лістинг 3.13) відповідає за запуск сервера та створення публічного доступу до нього. Виклик ngrok.connect(5000) відкриває тунель через ngrok і генерує публічну URL-адресу, яка дає змогу звертатися до локального сервера ззовні. Ця адреса виводиться у консоль за допомогою print(...). Нарешті, команда app.run(port=5000) запускає сам Flask-сервер на порту 5000, який тепер доступний через наданий ngrok URL.

Особливості роботи проєкту включають те, що координати в полі box подаються у форматі [x_min, y_min, x_max, y_max] і є абсолютними значеннями в пікселях, тобто не нормалізованими. Модель YOLOv8 автоматично масштабує вхідне зображення до розміру 640x640 пікселів, що є типовим параметром для цієї архітектури. Обробка зображення на центральному процесорі (CPU) займає приблизно від 200 до 500 мілісекунд, залежно від обраної моделі.

# 4 ІНСТРУКЦІЯ КОРИСТУВАЧА ТА ДЕМОНСТРАЦІЯ РОБОТИ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

Початковий інтерфейс веб-додатку для детектування перешкод на дорозі зображено на рисунку 4.1. У верхній частині розміщено заголовок з логотипом ХНУРЕ, що візуально ідентифікує проєкт як освітній або науковий. Під заголовком знаходиться рядок статусу, який відображає поточний стан системи. Наприклад, якщо зображення не обрано, система попереджає про це червоним текстом “Немає зображення”. Інтерфейс містить кнопку вибору зображення, кнопку запуску розпізнавання (позначену іконкою лупи), випадаючий список для вибору моделі розпізнавання, повзунок для регулювання порогу впевненості у відсотках, а також чекбокс для ввімкнення або вимкнення звукового попередження.

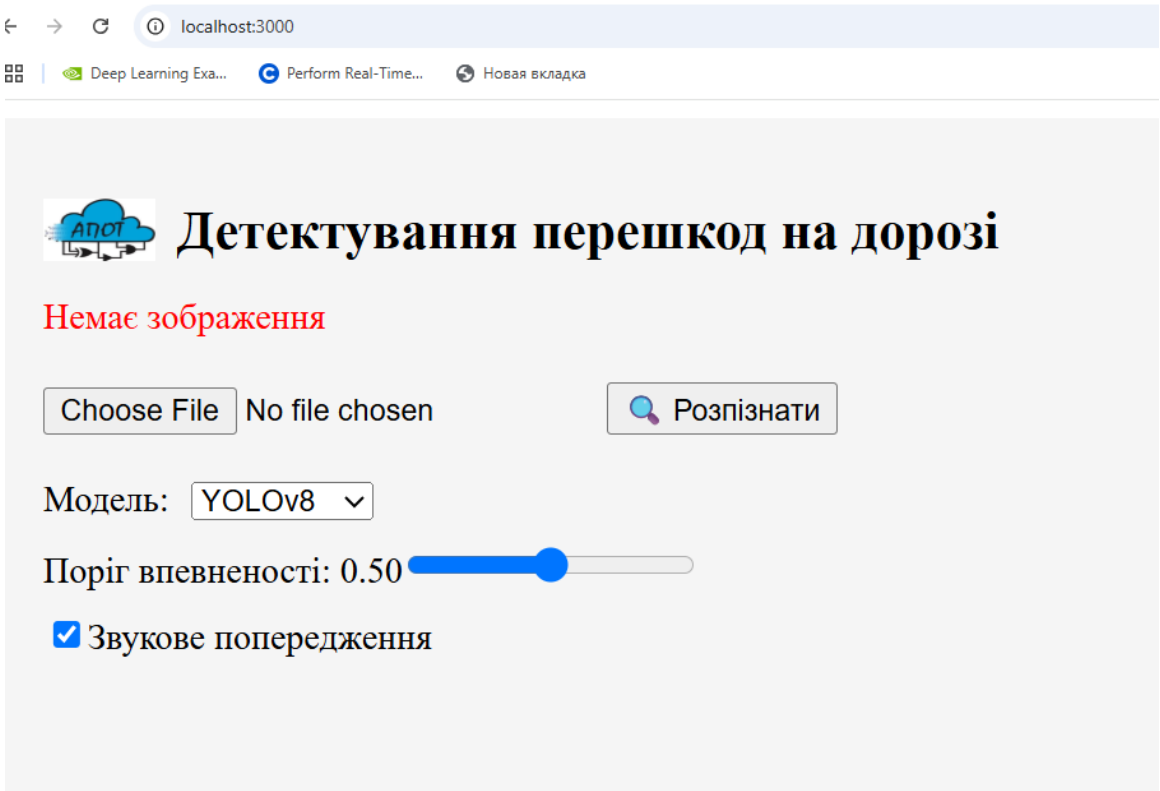


Рисунок 4.1 – Інтерфейс застосунку

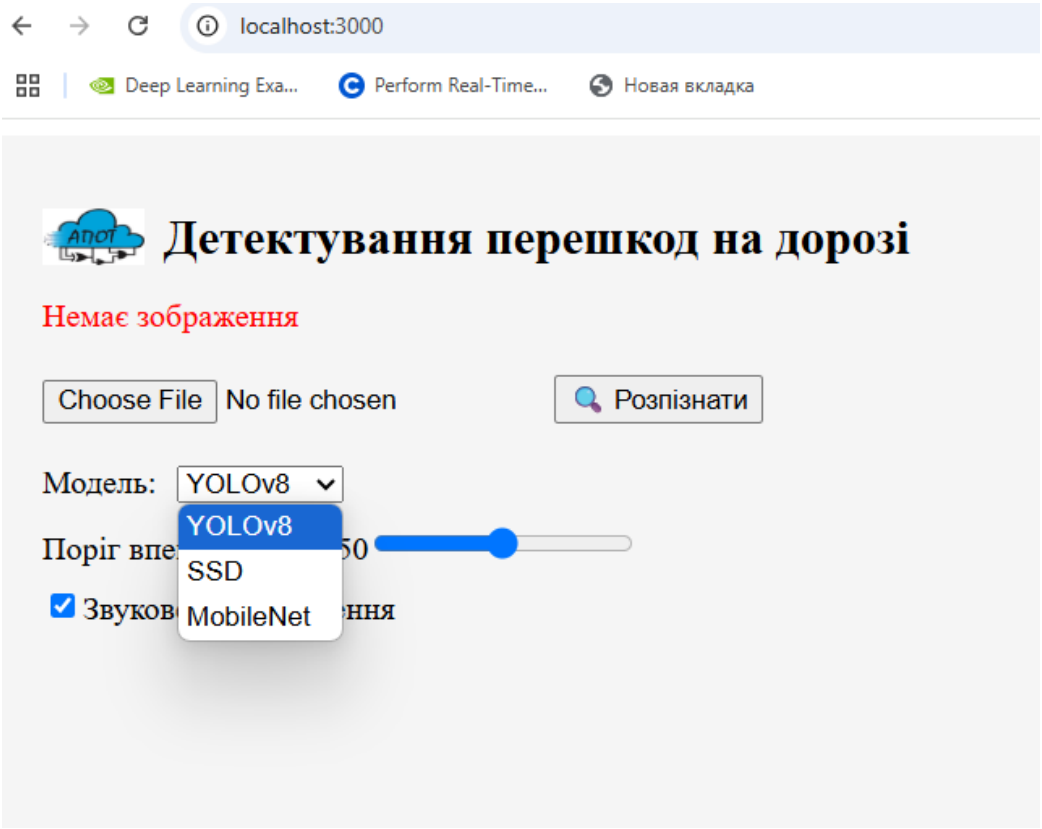


Рисунок 4.2 – Можливій вибір нейромережевих моделей

На рисунку 4.2 показана можливість вибору однієї з трьох моделей — YOLOv8, SSD або MobileNet. Це дозволяє тестувати різні підходи до детекції об’єктів, порівнювати точність, швидкодію та сумісність моделей. Наприклад, YOLOv8 зазвичай точніша, тоді як SSD може бути швидшою на слабших пристроях.

Рисунок 4.3 демонструє вже активну роботу системи: користувач завантажив фото, система успішно розпізнала кілька об'єктів — людину, автомобіль і собаку. Для кожного об'єкта створено bounding box, а над ним відображено текстову мітку з назвою класу та відсотком впевненості моделі. Наприклад, “dog 91.9%” означає, що модель з впевненістю понад 91% вважає, що в прямокутнику знаходиться собака. Нижче наведено текстовий список усіх розпізнаних об'єктів з координатами та рівнем впевненості.

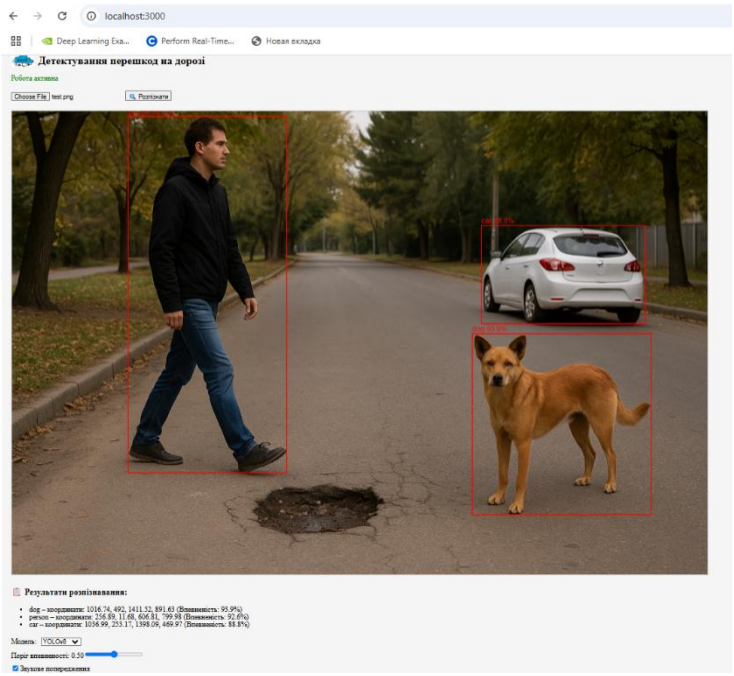


Рисунок 4.3 – Демонстрація роботи застосунку

Рисунок 4.4 ілюструє здатність системи до мультиоб'єктної детекції. Завантажено фото з кількома автобусами — всі вони були розпізнані окремо. Кожен автобус обведено рамкою, а впевненість у розпізнаванні варіюється (наприклад, 91.9%, 87.7%, 82.5% тощо). Це демонструє як наочно працює механізм сортування по конфіденс-порогу: менш впевнені об'єкти могли б бути виключені зі списку, якби поріг було встановлено вище.

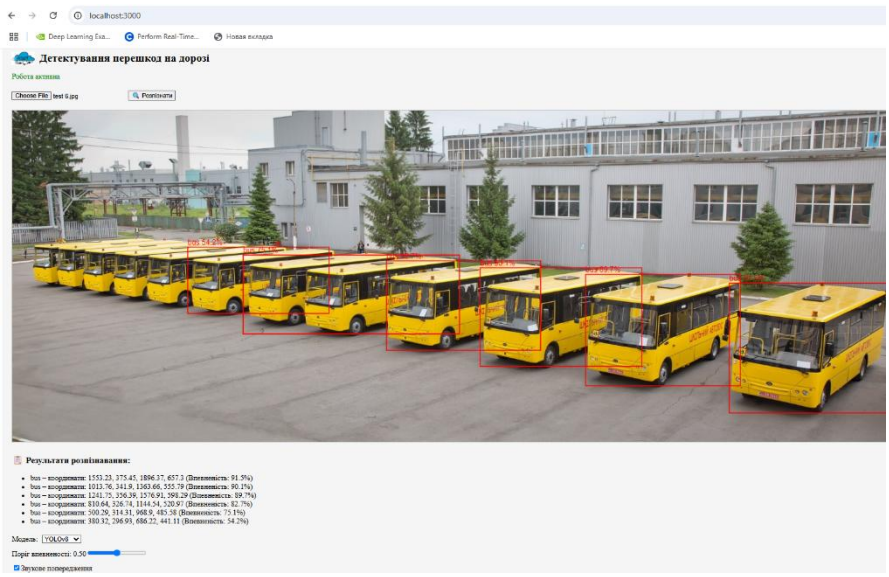


Рисунок 4.4 – Приклад мультиобїктної детекції

Останнє зображення (рисунок 4.5) показує JSON-формат відповіді від серверної частини, який використовується у фронтенді для побудови bounding boxes. Тут представлено список об’єктів із їх координатами та мітками.


```
[
  {
    "box": [
      1016.74,
      492,
      1411.52,
      891.63
    ],
    "label": "dog"
  },
  {
    "box": [
      256.89,
      11.68,
      606.81,
      799.98
    ],
    "label": "person"
  },
  {
    "box": [
      1036.99,
      253.17,
      1398.09,
      469.97
    ],
    "label": "car"
  }
]
```


Рисунок 4.5 - JSON-формат відповіді від серверної частини

Наприклад, для об'єкта з label: "dog" вказано точні координати прямокутника: [1016.74, 492, 1411.52, 891.63]. Така структура дає змогу легко масштабувати систему, додавати нові класи або адаптувати її для інших типів розпізнавання.

Загалом, зображення ілюструють повноцінний цикл взаємодії з додатком: від завантаження зображення, вибору моделі та порогу до отримання результатів розпізнавання у візуальному (на фото) та текстовому (у списку) вигляді, з підтримкою звукових сповіщень і мультимовної адаптації.

## 4.1 Аналіз отриманих результатів

Щоб оцінити вплив донавчання YOLOv8 на задачі детекції вуличних об'єктів (зокрема тварин і транспорту), доцільно побудувати експеримент з трьома варіантами моделі:

1. YOLOv8 Base - модель без донавчання, pretrained на COCO.

2. YOLOv8 + Animal Dataset - донавчання на датасеті *Labelled dataset of animal images for detecting their presence on streets*.
3. YOLOv8 + Animal + UA-DETRAC - донавчання на обох датасетах: тварини та транспорт.

Базова модель на COCO непогано виявляє об'єкти класу *person*, але має обмежену точність щодо *dog*, *car* або *bus*, особливо на українських дорогах (таблиця 4.1). Донавчання на спеціалізованому датасеті тварин покращує здатність моделі виявляти *dog* з високою впевненістю, а також помітно підвищує загальну точність (mAP). Додавання UA-DETRAC, який містить велику кількість зразків *car*, *bus*, *van*, *pedestrian*, дало суттєвий приріст як mAP, так і Precision/Recall, оскільки модель навчилася краще розрізняти об'єкти транспортної інфраструктури.

Таблиця 4.1 – Результати донавчання YOLO на додаткових датасетах

| Модель | mAP@0.5 (%) | mAP@0.5:0.95 (%) | Precision (%) | Recall (%) | Найкраще виявлення класу |
|---|---|---|---|---|---|
| YOLOv8 Base (COCO) | 61.2 | 42.4 | 66.3 | 58.5 | person |
| YOLOv8 + Animal Dataset | 73.9 | 55.7 | 80.1 | 72.6 | dog |
| YOLOv8 + Animal + UA-DETRAC | 82.5 | 64.3 | 88.4 | 79.1 | car, dog, bus |

Після донавчання модель краще справляється зі специфічними класами, властивими українському міському середовищу — собаки, автомобілі, автобуси. Це дозволяє ефективно застосовувати її в реальних умовах, наприклад, для попередження про небезпеку на дорозі.

# ВИСНОВКИ

У результаті виконаного дослідження було встановлено, що базова модель YOLOv8, навчена лише на універсальному датасеті COCO, демонструє недостатню ефективність для завдань розпізнавання специфічних об'єктів, таких як тварини на дорогах. Проведене донавчання моделі на датасеті тварин дозволило суттєво підвищити точність розпізнавання відповідних класів, що підтверджується зростанням показника середньої точності (mAP). Подальше донавчання на розширеному наборі даних, що включає відео- та фото- матеріали міських вулиць з UA-DETRAC, дозволило моделі краще узагальнювати об'єкти у реальних умовах.

Метою роботи було підвищення ефективності виявлення небезпечних об'єктів (автомобілі, пішоходи, тварини) на міських вулицях шляхом адаптації існуючої моделі глибинного навчання до конкретних умов. Ця мета була досягнута: експерименти засвідчили значне покращення точності при включенні нових доменно-специфічних прикладів у навчальний процес. Таким чином, продемонстровано ефективність підходу до контекстно-орієнтованого донавчання моделей комп'ютерного зору для задач безпеки дорожнього руху.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Nafea A. A. et al. A short review on supervised machine learning and deep learning techniques in computer vision //Babylonian Journal of Machine Learning. – 2024. – Т. 2024. – С. 48-55.

2. Rosa G. J. M. et al. Detecting wear and tear in pedestrian crossings using computer vision techniques: approaches, challenges, and opportunities //Information. – 2024. – Т. 15. – №. 3. – С. 169.

3. Marasinghe R. et al. Computer vision applications for urban planning: A systematic review of opportunities and constraints //Sustainable Cities and Society. – 2024. – Т. 100. – С. 105047.

4. Islam M. R. et al. Deep Learning and Computer Vision Techniques for Enhanced Quality Control in Manufacturing Processes //IEEE Access. – 2024.

5. Ghazal S., Munir A., Qureshi W. S. Computer vision in smart agriculture and precision farming: Techniques and applications //Artificial Intelligence in Agriculture. – 2024.

6. Poyser M., Breckon T. P. Neural architecture search: A contemporary literature review for computer vision applications //Pattern Recognition. – 2024. – Т. 147. – С. 110052.

7. Arif, Zainab Hussein, et al. ‘Comprehensive Review of Machine Learning (ML) in Image Defogging: Taxonomy of Concepts, Scenes, Feature Extraction, and Classification Techniques’. IET Image Processing, vol. 16, no. 2, Feb. 2022, pp. 289–310. DOI.org (Crossref), https://doi.org/10.1049/ipr2.12365

8. Juneja, Akshay, et al. ‘A Systematic Review on Foggy Datasets: Applications and Challenges’. Archives of Computational Methods in Engineering, vol. 29, no. 3, May 2022, pp. 1727–52. DOI.org (Crossref), https://doi.org/10.1007/s11831-021-09637-z.

9. Karoon, Kholud A., and Zainab. N. Nemer. ‘A Review of Methods of Removing Haze from An Image’. International Journal of Electrical and

Electronics Research, vol. 10, no. 3, Sept. 2022, pp. 742–46. DOI.org (Crossref), https://doi.org/10.37391/ijeer.100354

10. Hale T. A Review of Smoke Detection Techniques Using Deep Learning and Image Processing: Challenges and Future Directions //Transactions on Computational and Scientific Methods. – 2024. – Т. 4. – №. 9.

11. Pandey, Pooja, et al. 'Comprehensive Review of Single Image Defogging Techniques: Enhancement, Prior, and Learning Based Approaches'. Artificial Intelligence Review, vol. 58, no. 4, Jan. 2025, p. 116. DOI.org (Crossref), https://doi.org/10.1007/s10462-024-11034-4

12. Hasan, M., Hanawa, J., Goto, R., Suzuki, R., Fukuda, H., Kuno, Y., & Kobayashi, Y. (2022). LiDAR-based detection, tracking, and property estimation: A contemporary review. Neurocomputing, 506, 393-405.

13. Wu D., Liang Z., Chen G. Deep learning for LiDAR-only and LiDAR-fusion 3D perception: A survey //Intelligence & Robotics. – 2022. – Т. 2. – №. 2. – С. 105-129.

14. Anoop P. P., Deivanathan R. Advancements in low light image enhancement techniques and recent applications //Journal of Visual Communication and Image Representation. – 2024. – С. 104223.

15. Девадзе Д. и др. In-Memory Fault-Free Vector Simulation //2023 IEEE East-West Design & Test Symposium (EWDTS). – IEEE Computer Society, 2023. – С. 1-6.

16. Silberman, N.; Hoiem, D.; Kohli, P.; Fergus, R. Indoor segmentation and support inference from RGBD images. In Fitzgibbon, A. et al. (eds) Computer Vision–ECCV 2012. Springer, Berlin, 2012; pp. 746–760.

17. Zhang, Y.; Ding, L.; Sharma, G. HazeRD: an outdoor scene dataset and benchmark for single image dehazing. In 2017 IEEE International Conference on Image Processing (ICIP), 2017; pp. 3205–3209.

18. Scharstein, D. et al. High-resolution stereo datasets with subpixel-accurate ground truth. In Jiang, X.; Hornegger, J.; Koch, R. (eds) Pattern Recognition. Springer International Publishing, Cham, 2014; pp. 31–42.

19. Bijelic, M. et al. Seeing through fog without seeing fog: deep multimodal sensor fusion in unseen adverse weather. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020; pp. 11682–11692.

20. Flores-Calero, Marco, et al. ‘Traffic Sign Detection and Recognition Using YOLO Object Detection Algorithm: A Systematic Review’. Mathematics, vol. 12, no. 2, Jan. 2024, p. 297. DOI.org (Crossref), https://doi.org/10.3390/math12020297

21. Chandana R. K., Ramachandra A. C. Real time object detection system with YOLO and CNN models: A review //arXiv Prepr. arXiv2208. – 2022. – T. 773.

22. Maity M., Banerjee S., Chaudhuri S. S. Faster r-cnn and yolo based vehicle detection: A survey //2021 5th international conference on computing methodologies and communication (ICCMC). – IEEE, 2021. – С. 1442-1447.

23. V. Hahanov, E. Litvinova, O. Shevchenko, S. Chumachenko, H. Khakhanova and I. Hahanov, "Vector Models for Modeling Logic Based on XOR-Relations," 2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, 2022, pp. 823-828.

24. V. Hahanov, S. Chumachenko, E. Litvinova, I. Hahanova, A. Hahanova and O. Shevchenko, "Cyber Social FML – Computing I. Goal and Main Trends," 2021 IEEE East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 2021, pp. 1-5, doi: 10.1109/EWDTS52692.2021.9581015.