МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

Кафедра комп'ютерних наук та прикладної математики

"*До захисту допущена*"
Зав. кафедри комп'ютерних наук та прикладної математики
д.т н., проф. Ю.В. Турбал
«____» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

Розробка додатку для виявлення об'єктів в реальному часі з використанням нейронної мережі

Виконав: Чехрій Назар Романович _____
(прізвище, ім'я, по батькові) (підпис)

група ІПЗ-41

Керівник: к.т.н., доцент, доцент Остапчук О. П _____
(науковий ступінь, вчене звання, посада, прізвище, ініціали) (підпис)

Рівне – 2025

Національний університет водного господарства та природокористування

(повне найменування вищого навчального закладу)

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

Кафедра **комп'ютерних наук та прикладної математики**
Рівень вищої освіти **Бакалаврський (перший)**
Галузь знань 12 Інформаційні технології
(шифр і назва)
Спеціальність 121 Інженерія програмного забезпечення
(шифр і назва)

"Затверджую"
Завідувач кафедри
д.т н., проф. Ю.В. Турбал
3 лютого 2025р.

З А В Д А Н Н Я
НА КВАЛФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Чехрію Назару Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *" Розробка додатку для виявлення об'єктів в реальному часі з використанням нейронної мережі"*

керівник роботи *Остапчук Оксана Петрівна, к.т.н., доцент, доцент кафедри комп'ютерних наук та прикладної математики.*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада)

затверджені наказом по університету від *"18" квітня 2025 року С №-463.*

2. Термін подання роботи студентом *30 травня 2025 року.*

3. Вихідні дані до роботи: *технології, які необхідні для реалізації системи розпізнавання об'єктів та список основних завдань і вимог щодо основних функціональних можливостей.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *розробити прототип додатку розпізнавання об'єктів, інтегрувати моделі для виявлення та трекінгу, реалізувати графічний інтерфейс та функції перегляду результатів.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Мультимедійна презентація

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1	*доцент Остапчук О.П.*	*10.02.25*	*10.02.25*
Розділ 2	*доцент Остапчук О.П.*	*26.03.25*	*26.03.25*
Розділ 3	*доцент Остапчук О.П.*	*28.04.25*	*28.04.25*

7. Дата видачі завдання *03 лютого 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Вивчення літератури за обраною тематикою*	*01.02.25 – 20.02.25*	*виконав*
2	*Розробка прототипу додатку*	*23.02.25 – 17.03.25*	*виконав*
3	*Розробка функціональних можливостей додатку*	*05.03.25 – 28.03.25*	*виконав*
4	*Розробка перегляду, збереженню, аналізу результатів*	*20.03.25 – 25.03.25*	*виконав*
5	*Загальні висновки до роботи*	*20.03.25 – 15.05.25*	*виконав*
6	*Підготовка звіту кваліфікаційної роботи*	*01.04.25 – 20.05.25*	*виконав*
7	*Підготовка мультимедійної презентації*	*04.06.25 – 06.06.25*	*виконав*
8	*Підготовка до виступу*	*11.06.25 – 13.06.25*	*виконав*

Здобувач: _____ **Чехрій Н.Р.**
(підпис) (прізвище та ініціали)

Керівник: _____ **Остапчук О.П.**
(підпис) (прізвище та ініціали)

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 48 с., 8 рисунків, 13 джерел.

Мета роботи: розробка додатку, який не лише виявляє об’єкти на відео в реальному часі, але й надає користувачеві інструменти для збереження та аналізу даних розпізнаних об’єктів.

Об'єкт дослідження - процес виявлення об'єктів в реальному часі.

Предмет дослідження - методи та моделі глибинного навчання, зокрема нейронні мережі, які використовуються для розпізнавання об'єктів у реальному часі, а також їх інтеграція у розроблений додаток.

Методи дослідження - вивчення архітектури нейронних мереж, алгоритмів трекінгу та засобів реалізації графічних інтерфейсів.

Проведено детальний аналіз принципів роботи алгоритмів YOLO, DeepSORT та інших алгоритмів, зокрема їхніх переваг і обмежень у контексті завдань виявлення та відстеження об’єктів на відео. Досліджено методи інтеграції даних алгоритмів у додаток, який розроблений на мові програмування Python з урахуванням вимог до обробки відео в реальному часі. Реалізовано систему, що поєднує виявлення, трекінг, збереження інформації в базу даних та виведення результатів у графічному інтерфейсі.

Ключові слова: ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, YOLO (YOU ONLY LOOK ONCE), DEEPSORT, ВИЯВЛЕННЯ ОБ’ЄКТІВ, OPENCV, PYTHON.

ВСТУП

Нейронні мережі та машинне навчання стрімко проникають у всі сфери життя – від медицини та сільського господарства до промисловості, безпеки та побуту. Однією з ключових технологій, що лежать в основі багатьох інновацій, є виявлення об'єктів – процес, за якого комп'ютерна система здатна ідентифікувати та локалізувати об'єкти на зображеннях або відео в режимі реального часу. З розвитком глибокого навчання та нейронних мереж зросла точність та швидкість таких систем, що зробило можливим їх ефективне використання навіть на споживчому обладнанні.

Широке використання сучасних засобів, збільшення їх функціональних можливостей забезпечує розв'язання різних класів завдань. Одним із них є пошук і розпізнавання певного класу об'єктів на зображенні чи відеопотоці для подальшого його опрацювання [1].

Виявлення об'єктів – дуже цікавий напрямок, який вивчається та еволюціонує не перший десяток років. Зараз багато розробок у цій галузі побудовано на глибокому навчанні, яке має перевагу над стандартними алгоритмами, оскільки нейронні мережі можуть апроксимувати функції найчастіше краще у режимі реального часу та має надзвичайно широкий спектр застосувань – від систем безпеки та відеоспостереження до автономного транспорту, розумних міст.

Такі системи дозволяють зменшити навантаження на людей, автоматизувати рутинні завдання, підвищити ефективність обробки інформації та швидкість реагування на події. Реалізація подібних рішень потребує поєднання знань у галузях штучного інтелекту, програмування, цифрової обробки зображень та інтеграції з апаратними засобами.

У межах даної роботи здійснюється розробка додатку, що реалізує виявлення об'єктів в режимі реального часу. Основними компонентами системи є нейронна мережа, модуль захоплення відео, обробка результатів, а також функціонал збереження даних у базу даних та експорт у зручні формати.

Створення зручного графічного інтерфейсу користувача, що дозволяє керувати процесом розпізнавання та переглядати результати в інтерактивному режимі.

Актуальність теми зумовлена швидким розвитком комп'ютерного зору, який активно розвивається та знаходить застосування в завданнях автоматичного аналізу, моніторингу, спостереження та обробки візуальної інформації. Виявлення об'єктів у реальному часі є важливою складовою таких систем. Завдяки використанню нейронних мереж можна досягти високої точності та швидкості розпізнавання.

Метою цієї кваліфікаційної роботи є розробка додатку, який не лише виявляє об'єкти на відео в реальному часі, але й надає користувачеві інструменти збереженню та аналізу даних розпізнаних об'єктів.

Завдання, які необхідно виконати для досягнення поставленої мети:

1. Проаналізувати існуючі методи виявлення та трекінгу об'єктів, визначити вимоги до додатку.
2. Обрати відповідні алгоритми комп'ютерного зору інструменти розробки.
3. Реалізувати додаток з графічним інтерфейсом для розпізнавання об'єктів з відео або медіафайлів, інтегрувати систему з базою даних для збереження результатів виявлень та забезпечити можливість перегляду й керування історією виявлень.

Об'єктом дослідження є процес виявлення об'єктів в реальному часі.

Предметом дослідження є методи та моделі глибинного навчання, зокрема нейронні мережі, які використовуються для розпізнавання об'єктів у реальному часі, а також їх інтеграція в розроблений додаток.

Практична значущість додатку полягає в автоматичному виявленні та відстеженні об'єктів на відео, що дозволяє адаптувати його до широкого кола реальних задач у сферах безпеки, аналітики та автоматизації.

Система дозволяє в режимі реального часу розпізнавати об'єкти, відстежувати їх переміщення та автоматично зберігати ключові кадри. Це

відкриває можливість для автоматизованого моніторингу територій, виявлення порушень, контролю доступу тощо.

Завдяки інтеграції з базою даних та функцією експорту до Excel, система забезпечує зберігання результатів розпізнавання, що дозволяє вести історію подій, формувати звіти, проводити подальший аналіз і використовувати дані у роботі персоналу чи правоохоронних органів.

У першому розділі описані теоретичні основи реалізації систем виявлення об’єктів, зокрема важливість та аналіз алгоритмів і їхнього порівняння. У другому розділі розглядаються функціональні вимоги та деталі розробки додатку, включаючи вибір технологій. Третій розділ містить інформацію про програмну реалізацію та результати розробленого додатку.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РЕАЛІЗАЦІЇ СИСТЕМ ВИЯВЛЕННЯ ОБ'ЄКТІВ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ

1.1. Основні поняття та класифікація виявлення об'єктів у режимі реального часу

Розвиток глибокого навчання, особливо згорткових нейронних мереж, відкрив нові можливості для автоматичного виявлення об'єктів. Завдяки великим наборам даних та високопродуктивному апаратному забезпеченню, стало можливим створення моделей, які не лише точно розпізнають об'єкти, але й працюють в режимі реального часу.

Комп'ютерний зір наділяє машини здатністю інтерпретувати та розуміти візуальну інформацію з навколишнього світу. Використовуючи складні алгоритми і передові технології обробки зображень, системи комп'ютерного зору дозволяють комп'ютерам витягувати значущу інформацію із зображень і відео, імітуючи можливості людського зору з надзвичайною точністю і ефективністю.

Виявлення об'єктів – це завдання комп'ютерного зору , метою якого є визначення місцезнаходження об'єктів на цифрових зображеннях. Таким чином, це приклад штучного інтелекту , який полягає в навчанні комп'ютерів бачити так, як це роблять люди, зокрема шляхом розпізнавання та класифікації об'єктів відповідно до семантичних категорій [2].

Виявлення об'єктів – це метод, який використовує нейронні мережі для локалізації та класифікації об'єктів на зображеннях.

Локалізація зображення – це процес визначення правильного розташування одного або кількох об'єктів за допомогою обмежувальних рамок, які відповідають прямокутним формам навколо об'єктів. Цей процес іноді плутають з класифікацією зображень або розпізнаванням зображень, метою яких є передбачення класу зображення або об'єкта на зображенні в одній з категорій або класів [3].

Обробка зображень в основному зосереджені на роботі з двовимірними зображеннями, тобто як перетворити одне зображення в інше. Наприклад, попіксельні операції збільшення контрастності, операції з виділення країв, усунення шумів чи геометричні перетворення, такі як обертання зображення. Дані операції припускають, що обробка/аналіз зображення діють незалежно від вмісту самих зображень.

У комп'ютерному зорі зображення виражаються як неперервні функції на двовимірній координатній площині, представлені як $f(x,y)$. Після оцифрування зображення проходять два основні процеси, які називаються дискретизацією та квантуванням, які, коротко кажучи, разом перетворюють неперервну функцію зображення на дискретну сіткову структуру піксельних елементів. Потім комп'ютер може сегментувати зображення на дискретні області відповідно до візуальної подібності та близькості пікселів [2].

1.2. Принципи функціонування нейронних мереж у задачах виявлення об’єктів

У процесі виявлення об'єктів нейронна мережа виконує кілька завдань одночасно: спочатку вона аналізує зображення для виявлення потенційних областей, де можуть бути об’єкти, а потім класифікує кожну з цих областей та уточнює їхнє положення у вигляді координат обмежувальних рамок.

Функціонування таких мереж базується на ієрархічному представленні ознак: від простих на початкових шарах до складніших (текстури, контури об’єктів) на глибших рівнях. Навчання мережі відбувається за допомогою великого обсягу анотованих даних та оптимізаційних алгоритмів, які мінімізують похибку між прогнозованими та реальними результатами.

Простими словами, метою цього є визначення розташування об'єктів на заданому зображенні, що називається локалізацією об'єктів, та до якої категорії належить кожен об'єкт, що називається класифікацією об'єктів.

Згорткові нейронні мережі (англ. convolutional neural network, CNN) в машинному навчанні – це клас глибоких штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень.

Згорткові мережі взяли за основу біологічний процес, а саме схему з'єднання нейронів зорової кори тварин. Окремі нейрони кори реагують на стимули лише в обмеженій області зорового поля, відомій як рецептивне поле. Рецептивні поля різних нейронів частково перекриваються таким чином, що вони покривають усе зорове поле.

Згорткові нейронні мережі не потребують великої попередньої обробки даних у порівнянні з іншими алгоритмами класифікації зображень. Це означає, що мережа самостійно навчається виявляти фільтри, які в традиційних підходах створювалися вручну. Така незалежність від знань і людського втручання у процес формування ознак є перевагою нейронних мереж [7].

Згорткові нейронні мережі мали глибокий вплив на комп'ютерний зір, виходячи далеко за рамки базової класифікації зображень. Їхня здатність інтерпретувати візуальні дані відіграла ключову роль у виявленні об'єктів, сегментації, відеоаналізі та обробці даних у реальному часі [8].

Під час виявлення об'єктів нейронні мережі ідентифікують та знаходять кілька об'єктів на зображенні. Це завдання є складнішим, ніж класифікація, оскільки воно включає розпізнавання об'єктів та визначення їх точного розташування.

Архітектура згорткової нейронної мережі на основі регіонів та її наступні ітерації, Fast R-CNN та Faster R-CNN, зіграли в цьому важливу роль. Ці архітектури використовують комбінацію вибіркового пошуку для пропозиції регіонів та згорткових нейронних мереж для класифікації. Таким чином, значно підвищується точність та швидкість виявлення об'єктів.

У таких завданнях, як розпізнавання дій та виявлення аномалій у відео, згорткові нейронні мережі повинні розуміти часову динаміку та просторові особливості. Такі архітектури, як 3D-згорткові нейронні мережі, розширюють

традиційну 2D-згортку до трьох вимірів. Це дозволяє мережі вивчати як просторові, так і часові особливості.

Згорткові нейронні мережі, хоча й потужні, але стикаються з окремими труднощами у своєму застосуванні, особливо в таких сценаріях, як дефіцит даних, перенавчання та середовища з неструктурованими даними. Інноваційні методи та алгоритми навчання вирішують ці проблеми, підвищуючи стійкість та ефективність згорткових нейронних мереж [8].

Згорткові нейронні мережі продовжують розвиватися, відкриваючи нові горизонти нейронних мереж у штучному інтелекті та машинному навчанні.

Однак можна очікувати ще більшого розвитку нейронних мереж з точки зору:

- підвищеної обчислювальної ефективності, що робить їх більш життєздатними на менших пристроях;
- досягненню в обробці 3D-даних та складних часових рядів;
- підвищенню інтеграції з іншими сферами штучного інтелекту.

1.3. Важливість вибору правильної архітектури моделі

Однією з ключових передумов ефективного функціонування будь-якої системи комп'ютерного зору є правильний вибір алгоритму нейронної мережі. Від цього залежить не лише точність виявлення об'єктів, але й швидкодія, споживання ресурсів, масштабованість системи, її здатність до подальшого розширення та адаптації. Неправильно обрана модель може призвести до затримок в обробці, некоректних результатів, перевантаження пам'яті чи навіть повного припинення роботи системи під час реального використання.

Архітектура моделі визначає глибину, кількість параметрів, типи шарів, розміри вхідних даних, спосіб з'єднання шарів, а також наявність оптимізацій, таких як пропуски, нормалізація, згортки з розширенням тощо. У задачах виявлення об'єктів використовуються як «важкі» моделі – наприклад, Faster R-CNN або YOLO – так і «легкі» варіанти, розроблені спеціально для мобільних пристроїв чи вбудованих систем (наприклад, YOLOv5n, YOLOv8n, Tiny-YOLO).

Кожна з них має власні характеристики, які слід враховувати при впровадженні у реальну систему.

Вибір архітектури – це завжди компроміс між точністю та продуктивністю. У завданнях безпеки, де важливо не пропустити жодного об’єкта, доцільніше обрати потужні моделі, навіть якщо це призведе до зниження швидкості. Натомість у сценаріях, де швидкість є критичною – наприклад, у транспортних системах або відеонагляді в реальному часі – обирають моделі, що забезпечують високу частоту кадрів, навіть якщо ціною незначного зниження точності.

Ще одним важливим фактором є середовище, у якому функціонує система. Якщо вона працює на сервері з потужним GPU, то допустиме використання великих моделей, натомість для роботи на ноутбуках, Raspberry Pi чи інших енергоефективних пристроях – потрібні моделі з мінімальною кількістю параметрів.

Також варто враховувати, що сучасні фреймворки дозволяють гнучко конвертувати архітектури під різні апаратні платформи, що відкриває додаткові можливості оптимізації. Крім того, правильний вибір архітектури дозволяє уникнути перенавчання, забезпечити кращу генералізацію на нових даних і швидше проходити етапи адаптації або донавчання.

Таким чином, вибір архітектури – це не просто технічне рішення, а стратегічний етап, який визначає успішність усього проєкту. Ретельний аналіз характеристик моделі, потреб користувача та умов експлуатації є запорукою стабільної, продуктивної та практично корисної системи комп’ютерного зору.

1.4. Огляд сучасних алгоритмів виявлення об’єктів

У сучасному комп’ютерному зорі виявлення об'єктів є однією з ключових задач, що активно досліджується та вдосконалюється завдяки розвитку глибокого навчання. Протягом останнього десятиліття з'явилось кілька високопродуктивних алгоритмів, які суттєво підвищили точність і швидкість виявлення об'єктів у зображеннях та відео. Серед найпоширеніших підходів виділяються Faster R-CNN, YOLO (You Only Look Once) та SSD (Single Shot

Detector). Кожен з них має власну архітектуру, принцип дії та набір компромісів між точністю й обчислювальною складністю.

Розглянемо загальні характеристики, переваги та обмеження зазначених алгоритмів, що дозволяє краще зрозуміти їхню ефективність у задачах виявлення об’єктів, особливо у реальному часі.

Порівняємо дослідження трьох відомих алгоритмів виявлення об'єктів:

1. Faster R-CNN
2. YOLO (You Only Look Once)
3. SSD (Single Shot Detector)

Також здійснимо огляд поточного порівняння продуктивності цих часто використовуваних алгоритмів виявлення об'єктів.

1.4.1. Faster R-CNN

Модель Faster R-CNN була розроблена групою дослідників у Microsoft. Faster R-CNN – це глибока згорткова мережа, яка використовується для виявлення об'єктів і відображається користувачем як єдина, наскрізна, уніфікована мережа. Мережа може точно та швидко передбачати розташування різних об'єктів. Щоб по справжньому зрозуміти Faster R-CNN, також повинно бути знайомими з мережами, з яких вона розвинулася, а саме R-CNN та Fast R-CNN. Faster R-CNN є розширенням Fast R-CNN. Як випливає з назви, Faster R-CNN швидший за Fast R-CNN завдяки мережі пропозицій регіонів (RPN).

Faster R-CNN – це єдина уніфікована модель, архітектура якої складається з двох модулів:

- RPN: повністю згорткова нейронна мережа, яка сканує карту ознак і для кожної позиції одночасно прогнозує ймовірність наявності об’єкта та уточнює координати його обмежувальної рамки
- Faster R-CNN: згорткова нейронна мережа, яка виконує вилучення ознак із запропонованих областей та забезпечує одночасне визначення обмежувальних рамок і відповідних міток класів для виявлених об’єктів.

Обидва модулі працюють на одному виході глибокої нейронної мережі. RPN діє як механізм уваги для мережі Fast R-CNN, спрямовуючи її увагу на потенційно важливі області зображення, де з найбільшою ймовірністю можуть знаходитися об’єкти.

Faster R-CNN – одна з моделей, яка довела, що можна вирішувати складні проблеми комп'ютерного зору.

Наразі створюються нові моделі не лише для виявлення об'єктів, але й для семантичної сегментації, виявлення 3D-об'єктів тощо, що базуються на цій оригінальній моделі. Деякі запозичують RPN, деякі – R-CNN, інші просто надбудовують їх на обох.

1.4.2. YOLO (You Only Look Once)

YOLO – це алгоритм, розроблений спеціально для обробки в реальному часі. Він здійснює виявлення об’єктів за одне проходження зображення через нейронну мережу, що забезпечує високу швидкість обробки.

Він працює виключно на основі одноразового погляду на зображення, щоб побачити кілька об'єктів. Тому його називають YOLO, що означає «ви просто дивитеся один раз». Просто дивлячись на зображення один раз, швидкість виявлення становить певний час (45 кадрів/с). Швидкий YOLOv1 досягає 155 кадрів/с. Це ще один прогресивний метод виявлення об'єктів глибокого навчання, який був опублікований у CVPR 2016 року з більш ніж 2000 цитуваннями.

YOLO розділяє зображення на сітку. Для кожної сітки обчислюються деякі значення, такі як ймовірності класів та параметри обмежувальної рамки. Модель працює таким чином, що спочатку розділяє вхідне зображення на сітку комірок, де кожна комірка відповідає за прогнозування обмежувальної рамки, якщо центр обмежувальної рамки потрапляє в цю комірку. Кожна комірка сітки передбачає обмежувальну рамку, що включає координати x та y, ширину, висоту та достовірність. Прогнозування класу також базується на кожній комірці.

Алгоритм YOLO є одним з найкращих алгоритмів виявлення об'єктів з наступних причин:

- швидкість: Цей алгоритм покращує швидкість виявлення, оскільки він може передбачати об'єкти в режимі реального часу;
- висока точність: YOLO – це прогностичний метод, який забезпечує точні результати з мінімальними фоновими помилками.

Його використовують в різних сферах для виявлення світлофорів, людей, та тварин.

1.4.3. SSD (Single Shot Detector)

SSD – це одновимірний детектор. Він не має делегованої мережі пропозицій регіонів та прогнозує граничні рамки та класи безпосередньо з карт ознак за один прохід.

SSD також є одноетапним детектором, тобто не потребує попереднього виділення регіонів для аналізу.

Виявлення об'єктів SSD складається з двох частин:

Витяг карт ознак та застосування згорткових фільтрів для виявлення об'єктів.

Характеристики SSD наступні:

- невеликі згорткові фільтри для прогнозування класів об'єктів та зміщень до стандартних граничних блоків;
- окремі фільтри для стандартних блоків, щоб врахувати різницю у співвідношеннях сторін;
- багатомасштабні карти ознак для виявлення об'єктів.

SSD можна навчати від початку до кінця для кращої точності. SSD робить більше прогнозів і має краще покриття щодо розташування, масштабу та співвідношень сторін. Видаляючи пропозицію делегованої області та використовуючи зображення з нижчою роздільною здатністю, модель може працювати зі швидкістю реального часу та все ще перевершувати точність найсучаснішої Faster R-CNN.

1.4.4. Порівняння продуктивності

Сучасні алгоритми виявлення об'єктів значно відрізняються за точністю, швидкістю роботи та ресурсомісткістю. Найбільш популярними є YOLO (You Only Look Once), SSD (Single Shot Detector) та Faster R-CNN. Їх вибір залежить від конкретних вимог до системи – зокрема, чи потрібна обробка у режимі реального часу, чи пріоритетною є висока точність.

Розглянувши принципи роботи основних алгоритмів виявлення об’єктів – YOLO, SSD та Faster R-CNN – можна зробити узагальнений висновок щодо їхньої ефективності у практичних задачах. Кожен з них має свої переваги та обмеження, які стають критичними залежно від вимог до точності, швидкості та обчислювальних ресурсів.

Вибір між Faster R-CNN, SSD та YOLO залежить від конкретних випадків використання, вимог та пріоритетів. Кожна модель виявлення об'єктів має свої сильні та слабкі сторони.

1. Faster R-CNN:
   - переваги: Надійна точність та прецизійність у виявленні об'єктів, особливо менших розмірів;
   - недоліки: Повільніша швидкість логічного висновку порівняно з деякими іншими моделями.
2. SSD (Single Shot Detector):
   - переваги: Швидша швидкість логічного висновку порівняно з Faster R-CNN при збереженні хорошої точності;
   - недоліки: Можуть виникнути труднощі з виявленням дрібних об'єктів.
3. YOLO (You Only Look Once):
   - переваги: Висока швидкість висновку в режимі реального часу;
   - недоліки: Може погіршитися точність, особливо з меншими об'єктами, порівняно з повільнішими, але точнішими моделями, такими як Faster R-CNN.

Якщо точність є критично важливою, а швидкість не є головною проблемою, Faster R-CNN може бути кращим вибором. Якщо обробка в режимі реального часу є важливою, YOLO може бути більш доречним варіантом. SSD може запропонувати хороший баланс між швидкістю та точністю [5].

Отже, якщо головним критерієм є швидкодія та режим реального часу, найкращим вибором буде YOLO. Якщо ж важлива максимальна точність і є можливість використання потужного апаратного забезпечення – доцільно обрати Faster R-CNN. SSD, у свою чергу, є збалансованим варіантом, який добре працює на пристроях з обмеженими ресурсами.

1.5. Методи попередньої обробки зображень у задачах виявлення об’єктів

У задачах комп’ютерного зору, зокрема виявлення об’єктів у реальному часі, якість вхідних зображень відіграє вирішальну роль у точності та стабільності роботи нейронних мереж. Сирі зображення часто містять шуми, варіації освітлення, різні розміри та формати, що можуть негативно впливати на результати моделі. Тому перед подачею зображень до нейронної мережі необхідно здійснити їх попередню обробку.

Ключові кроки попередньої обробки зображень:

1. Зміна розміру зображення

Зміна розміру зображення відповідно до вхідних параметрів, очікуваних моделлю виявлення, дозволяє привести дані до стандартного формату та зменшити обчислювальне навантаження.

2. Нормалізація

Нормалізація значень пікселів шляхом перетворення інтенсивності пікселів цифрових зображень до бажаного діапазону значень, зазвичай [0, 1] або [-1, 1]. Такий підхід забезпечує узгоджену архітектуру та допомагає в процесі навчання моделей.

3. Зменшення шуму

Усунення небажаного фонового шуму на зображенні здійснюються за допомогою фільтрів, таких як гаусівський, медіанний або двосторонній фільтри. Це не лише очищає зображення, але й покращує яскравість зображення, що дозволяє ефективніше виділяти ознаки.

4. Регулювання контрастності

Налаштування контрастності для посилення основних характеристик. Ще до ефективних методів включають вирівнювання гістограми та адаптивне вирівнювання гістограми з обмеженням контрасту.

5. Перетворення колірного простору

Застосування математичних та геометричних перетворень для деформації об'єктів на зображеннях, таких як масштабування, обертання або переміщення. Ще це корисно у завданнях, зосереджених на розрізненні та сегментації кольорів.

6. Збільшення зображення

Генерація зображень за допомогою варіацій для підвищення різноманітності навчального набору даних. І ще це включає операції обертання, зміну масштабу, дзеркальне відображення, зміщення та додавання випадкового шуму.

7. Виявлення країв

Обробка контурів об'єктів на зображенні та визначення області, що містить об'єкти та методи які включають виявлення країв Канні, Собеля та Лапласа.

8. Порогове значення

Бінаризація зображень для поділу їх на сегменти, де обчислюється середнє значення інтенсивності пікселів. Це ефективно для сегментації об'єктів та їх відокремлення від фону.

9. Розмиття та збільшення різкості

Згладжування та усунення шуму для мінімізації контрасту та підвищення різкості вздовж меж об'єктів. Додатково, корекція освітлення для збільшення контрастності та експозиції деталей по краях та необхідних частинах зображень.

10. Морфологічні операції

Використання таких операцій, як розширення, ерозія, відкриття та закриття, для зміни розміру форми об'єктів. Вона також сприяє усуненню невеликого небажаного шуму та покращенню інформацію про межі об'єкта шляхом постобробки реконструйованого об'єму [8].

Застосування цих методів попередньої обробки забезпечує більш якісне навчання моделей виявлення об’єктів, підвищує їхню точність та робить їх більш стійкими до варіацій у вхідних даних. Зосередження на адаптивній або розширеній обробці зображень може забезпечити швидше навчання та стабільніші результати.

Оскільки цифрові зображення в реальних застосуваннях стають дедалі різноманітнішими, вкрай важливо вийти за рамки базових доповнень. Застосування передових алгоритмів і методів обробки зображень може значно покращити конвеєри аналізу зображень [9].

1.6. Проблеми та виклики при виявленні об’єктів у реальному середовищі

Незважаючи на стрімкий розвиток нейронних мереж та алгоритмів комп’ютерного зору, застосування систем виявлення об’єктів у реальному часі в практичних умовах пов’язане з рядом викликів. Розробники повинні враховувати технічні, алгоритмічні та контекстні обмеження, які можуть суттєво вплинути на точність, швидкодію та стабільність роботи системи.

У реальному середовищі рівень освітлення може постійно змінюватися – від природного світла до штучного або недостатнього. Це впливає на контрастність, кольори та видимість об’єктів, що знижує якість детекції. Для зменшення впливу цих факторів використовують алгоритми адаптивної нормалізації яскравості, проте навіть сучасні моделі не завжди справляються з екстремальними умовами.

Ще одним із складних сценаріїв для алгоритмів детекції є сцени з великою кількістю об’єктів, які частково перекривають один одного. У таких випадках алгоритми можуть помилково об’єднувати декілька об’єктів в один або зовсім не

виявляти деякі з них. Це особливо критично у завданнях безпеки, де кожен об’єкт має бути врахований.

Якщо камера не статична, а переміщується (наприклад, у випадку дронів або систем відеонагляду зі змінним кутом огляду), система має справлятися із динамічним фоном і швидкими змінами кадру. Це ускладнює як локалізацію об’єктів, так і їх подальший трекінг.

У реальних умовах системи можуть працювати на пристроях із невеликою обчислювальною потужністю. Це вимагає оптимізації моделі, використання "легких" архітектур, зменшення частоти кадрів або розміру зображень для збереження швидкодії.

У багатьох реальних відеопотоках присутній шум, розмиття, ефекти стиснення або сторонні елементи, які заважають точній обробці зображень. Перед подачею у модель необхідно виконати попередню обробку для покращення вхідних даних.

Таким чином, успішне застосування систем виявлення об’єктів у реальному часі передбачає не лише обрання правильної моделі, але й урахування особливостей середовища та адаптацію системи до змінних умов експлуатації.

РОЗДІЛ 2
РОЗРОБКА ДОДАТКУ ДЛЯ ВИЯВЛЕННЯ ОБ'ЄКТІВ

2.1. Формулювання проблеми та методи дослідження

У сучасних умовах автоматизації виникає необхідність у розробці ефективних систем комп'ютерного зору, здатних у режимі реального часу розпізнавати та відслідковувати об'єкти. Це актуально для низки галузей, зокрема: безпеки, моніторингу, логістики, виробництва та інтелектуального відеоспостереження.

Проблема полягає в розробці додатку для виявлення об'єктів, який не лише забезпечує високу точність та обробку відеопотоку в реальному часі, а й має можливість зберігати результати.

Методи дослідження:

1. Аналіз літературних джерел: ретельне вивчення сучасних алгоритмів виявлення та трекінгу об'єктів, методів попередньої обробки відеоданих.
2. Вибір нейронних мереж: вибір і розгортання моделей YOLO та DeepSORT для розроблення додатку.
3. Метод програмної реалізації: створення прототипу додатку на мові Python з використанням бібліотек OpenCV, Ultralytics YOLO, MySQL Connector та ін.

2.2. Визначення та вибір функціональних вимог

Функціональні вимоги до додатку, що розробляється, визначаються його основною метою – виявлення та відстеження об'єктів у відео в режимі реального часу із подальшою фіксацією результатів. Система повинна мати можливість приймати вхідне відео із камери або завантаженого відеофайлу, обробляти кожен кадр, здійснюючи виявлення об'єктів за допомогою нейронної мережі YOLO, і в реальному часі виводити результат на екран користувача. Для забезпечення точного трекінгу рухомих об'єктів використовується алгоритм DeepSORT, який дозволяє зберігати інформацію про ідентичність об'єктів між кадрами.

Програмне забезпечення має містити графічний інтерфейс, реалізований на основі бібліотеки Tkinter, що забезпечить користувачу зручне керування: вибір джерела відео, запуск та зупинку аналізу, а також перегляд результатів детекції. Кожне виявлення повинно зберігатися до бази даних MySQL, з фіксацією ідентифікатора об’єкта, класу, часу появи та координат на кадрі. Важливим функціоналом є можливість автоматичного збереження зображення об’єкта (скріншота) у момент його появи або за заданим інтервалом, а також зв’язування цих зображень із відповідними записами в базі даних.

Крім цього, система повинна надавати можливість експорту результатів у табличний формат Excel для подальшої обробки, звітності або аналізу. Також має бути реалізовано можливість налаштування параметрів аналізу, зокрема порогу впевненості детектора або вибору класів об’єктів, які необхідно виявляти. Таким чином, програмне забезпечення повинно бути гнучким, масштабованим та зручним у користуванні для розв’язання задач виявлення та моніторингу об’єктів у режимі реального часу.

2.2.1. Нейронна мережа YOLO (You Only Look Once)

У рамках даного додатку для виявлення об'єктів у режимі реального часу обрано архітектуру нейронної мережі YOLO (You Only Look Once). Основною причиною вибору саме цієї моделі є її висока швидкодія, яка забезпечується завдяки одноетапному підходу до виявлення.

YOLO відрізняється не лише високою швидкістю, а й хорошою точністю на практичних задачах, зокрема в умовах обмежених обчислювальних ресурсів. Сучасні версії YOLO, мають відкритий вихідний код, добре документовані API та активну спільноту, що значно полегшує інтеграцію моделі у прикладні системи. Крім того, вони підтримують запуск як на CPU, так і на GPU, що робить систему гнучкою у плані розгортання.

У даному додатку використано реалізацію YOLO через бібліотеку Ultralytics, яка забезпечує простий інтерфейс для завантаження попередньо навчених моделей, обробки відео та виведення результатів. Враховуючи всі ці

переваги, YOLO є оптимальним вибором для задачі виявлення об'єктів у реальному часі, що й обґрунтовує його використання у системі, що розробляється.

Архітектура YOLO (You Only Look Once)

Архітектура YOLO схожа на GoogleNet . Як показано нижче, вона має 24 згорткові шари, чотири шари максимального пулінгу та два повністю зв'язані шари.

Архітектура працює наступним чином:

- змінює розмір вхідного зображення до 448x448 перед проходженням через згорткову мережу;
- спочатку застосовується згортка 1x1 для зменшення кількості каналів, а потім згортка 3x3 для генерації кубоїдного вихідного сигналу;
- функція активації "під капотом" – це ReLU, за винятком останнього шару, який використовує лінійну функцію активації;
- деякі додаткові методи, такі як пакетна нормалізація та відсіювання, регуляризують модель та запобігають її перенавчанню.

Цей перший крок починається з поділу вихідного зображення на N x N комірок сітки однакової форми. Кожна комірка в сітці відповідає за локалізацію та прогнозування класу об'єкта, який вона охоплює, разом зі значенням ймовірності/довіри.

Наступний крок – визначити обмежувальні рамки, що відповідають прямокутникам, виділяючи всі об’єкти на зображенні. Можна мати стільки обмежувальних рамок, скільки об’єктів на даному зображенні.

YOLO визначає атрибути цих обмежувальних рамок, використовуючи один модуль регресії у наступному форматі:

$$Y = [pc, bx, by, bh, bw, c1, c2]$$

де Y – це кінцеве векторне представлення для кожної обмежувальної рамки;

pc відповідає ймовірнісному балу сітки, що містить об'єкт. Наприклад, усі сітки червоного кольору матимуть ймовірнісний бал вище нуля;

bx, by це координати x та y центру обмежувальної рамки відносно комірки огортальної сітки;

bh, bw відповідають висоті та ширині обмежувальної рамки відносно комірки огортальної сітки;

c1 і c2 відповідають двом класам, де клас це об’єкт. Можна мати стільки класів, скільки потрібно у вашому випадку використання.

У більшості випадків один об'єкт на зображенні може мати кілька кандидатів на сітку для прогнозування, навіть якщо не всі вони є релевантними. Мета IOU (значення від 0 до 1) полягає в тому, щоб відкинути такі сітки, залишивши лише ті, що є релевантними. Ось логіка, що лежить в основі де користувач визначає свій поріг вибору IOU, який може становити, наприклад 0,5 а потім YOLO обчислює IOU кожної комірки сітки, яка дорівнює площі перетину, поділеній на площу об'єднання. Зрештою, він ігнорує прогнозування комірок сітки, що мають IOU ≤ порогове значення, та розглядає ті, у яких IOU > порогове значення [10].

2.2.2. DeepSORT

Для реалізації функціональності трекінгу об’єктів у реальному часі в межах даного додатку обрано алгоритм DeepSORT (Simple Online and Realtime Tracking with a Deep Association Metric). Цей алгоритм є розширенням базового трекера SORT (Simple Online and Realtime Tracking), в якому використовувався фільтр Калмана та алгоритм угорського призначення для асоціації об'єктів між кадрами. У DeepSORT до цих методів додається компонент глибокого навчання – спеціальна вбудована модель для обчислення ознак зовнішнього вигляду об’єктів, що дозволяє покращити точність асоціації в умовах перекриття, різких змін руху або короткочасної втрати об’єкта в кадрі.

Перевага DeepSORT полягає у здатності стабільно відстежувати об’єкти навіть у складних умовах, де прості методи трекінгу дають збої. Алгоритм призначає унікальні ідентифікатори кожному об’єкту, що зберігаються упродовж усього часу перебування об’єкта в полі зору камери. Це дає змогу вести облік кількості об’єктів, аналізувати траєкторії руху та здійснювати подальшу аналітику.

У даному додатку DeepSORT інтегрується разом із моделлю YOLO. Спершу YOLO виконує детекцію об’єктів на кожному кадрі, після чого координати й ознаки передаються до DeepSORT, який забезпечує трекінг. Такий підхід поєднує швидкодію YOLO із високою стійкістю до втрат об’єктів, яку забезпечує DeepSORT, що в результаті забезпечує ефективну та надійну систему виявлення і стеження.

Алгоритм SORT є основою системи DeepSORT яка є найсучаснішим підходом до відстеження кількох об'єктів у відео. Алгоритм SORT, скорочено від Simple Online and Realtime Tracking – це простий, але ефективний алгоритм для виконання асоціації даних та ініціювання відстеження у сценарії відстеження кількох об'єктів.

Спочатку він використовує підхід на основі виявлення для асоціації об'єктів у послідовних кадрах, а потім застосовує фільтр Калмана для прогнозування майбутнього положення об'єктів та корекції асоціації на основі фактичних вимірювань.

Цей алгоритм широко використовується завдяки своїй ефективності та точності та служить основою для більш просунутої системи DeepSORT, яка покращує процес асоціації даних за допомогою методів глибокого навчання для подальшого підвищення продуктивності відстеження. Поєднуючи сильні сторони алгоритму SORT з глибоким навчанням, DeepSORT досяг найсучаснішої продуктивності в завданнях відстеження кількох об'єктів.

Виявлення та вилучення ознак

Глибоке сортування (SORT) починається з виявлення об'єктів, часто використовуючи згорткову нейронну мережу, таку як YOLO (You Only Look Once), для ідентифікації об'єктів у кадрі. Кожне виявлення пов'язане з багатовимірним дескриптором зовнішнього вигляду, витягнутим іншою CNN. Ці дескриптори кодують зовнішній вигляд виявлених об'єктів і використовуються для зіставлення [11].

2.2.3. Tkinter

Tkinter – це стандартний фреймворк графічного інтерфейсу Python, що робить його зручним для розробки графічних інтерфейсів користувача. Як кросплатформна бібліотека, Tkinter гарантує, що ваші програми виглядатимуть нативно у Windows, macOS та Linux. Незважаючи на критику щодо його застарілого вигляду, Tkinter залишається практичним вибором для швидкого створення функціональних та кросплатформних графічних програм [12].

Однією з головних причин вибору Tkinter є його інтегрованість у стандартну бібліотеку Python, що усуває потребу у встановленні сторонніх залежностей, а також забезпечує хорошу сумісність з іншими модулями та простоту в налаштуванні.

Tkinter дозволяє швидко створювати інтуїтивно зрозумілий графічний інтерфейс із кнопками, панелями, полями вводу, меню тощо. У контексті додатку він використовується для візуалізації результатів роботи нейронної мережі та трекера в реальному часі. Зокрема, інтерфейс дає змогу запускати або зупиняти процес виявлення, обирати джерело відеопотоку, переглядати поточні виявлення та зберігати дані. Tkinter забезпечує ефективну обробку взаємодії з користувачем без значного впливу на продуктивність основного процесу детекції.

2.2.4. MySQL

MySQL – це система керування базами даних. У реляційній базі даних дані зберігаються не всі скопом, а в окремих таблицях, завдяки чому досягається виграш в швидкості і гнучкості. Таблиці зв'язуються між собою за допомогою

відносин, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мову структурованих запитів плюс найбільш поширений стандартний мова, яка використовується для доступу до баз даних [13].

У розробці додатку виявлення об’єктів у режимі реального часу важливу роль відіграє організоване зберігання результатів. Для цієї мети у ньому використовується система керування реляційними базами даних MySQL. Основною перевагою MySQL є її стабільність, масштабованість, висока продуктивність і підтримка широкого спектра мов програмування, зокрема Python, що дозволяє легко інтегрувати її з основною логікою додатку.

У нашому додатку MySQL використовується для збереження інформації про кожен виявлений об’єкт: клас об’єкта, час його появи у кадрі, а також шлях до збереженого скріншоту, якщо такий наявний. Завдяки використанню SQL-запитів можна легко організувати фільтрацію, пошук або сортування результатів для подальшого аналізу, експорту чи візуалізації.

2.2.5. OpenCV

OpenCV (Open Source Computer Vision Library) – це відкрита бібліотека комп’ютерного зору та машинного навчання, яка широко використовується для обробки зображень, відео, виявлення об’єктів, розпізнавання облич, трекінгу, калібрування камер та інших завдань комп’ютерного зору. OpenCV підтримує мову програмування Python, що робить її особливо зручною для швидкої розробки прототипів у наукових і прикладних проєктах.

У межах даного проєкту OpenCV відіграє ключову роль, забезпечуючи:

- захоплення відео з веб-камери або відеофайлу;
- обробку кожного кадру для подальшого передавання його в нейронну мережу YOLO;
- накладання графічних елементів на відео (наприклад, прямокутників навколо об'єктів та тексту з їхніми назвами);

- вирізання частин зображення (скріншотів) для збереження об’єктів, що були виявлені;
- управління вікнами виведення.

Таким чином, OpenCV виступає фундаментальним інструментом для роботи з вхідними медіа-даними та візуалізації результатів виявлення й трекінгу об’єктів.

2.3. Робота з медіа різних форматів

У сучасних системах комп’ютерного зору важливу роль відіграє здатність працювати з різноманітними типами вхідних даних. Різні джерела візуальної інформації можуть представляти її у найрізноманітніших форматах – як у вигляді статичних зображень, так і у вигляді потокового або записаного відео. Саме тому універсальність щодо підтримки медіафайлів є невід’ємною складовою зручного та ефективного програмного забезпечення.

У розробленому додатку передбачена можливість обробки як зображень (у форматах .jpg, .jpeg, .png), так і відеофайлів (наприклад, .mp4, .avi). Це дозволяє користувачеві працювати з даними, отриманими з різних джерел: архівних записів відеоспостереження, камер відеореєстраторів, скріншотів з інших систем, демонстраційних зображень з інтернету, лабораторних наборів для тестування тощо. Такий підхід суттєво розширює сферу застосування програми, дозволяє адаптувати її до нестандартних сценаріїв і створює основу для дослідницької гнучкості.

На практиці обробка файлів різних типів дозволяє реалізовувати сценарії офлайн-аналізу, коли немає можливості підключення до камери, або коли користувач хоче дослідити поведінку системи на фіксованих кадрах. Наприклад, у випадках судово-медичного аналізу відео, навчання нейронних мереж на фіксованих зображеннях або верифікації результатів на контрольних прикладах.

З технічної точки зору, підтримка мультимедійного розмаїття є ознакою добре спроєктованого, універсального додатку. Це забезпечує кращу сумісність

з іншими програмами, які генерують відео або зображення, а також підвищує зручність користувача, який не обмежується певним стандартом.

Крім того, здатність обробляти статичні й динамічні зображення є важливою в навчальному та демонстраційному контексті. В освітніх або презентаційних цілях часто використовуються окремі ілюстрації або фрагменти відео, на яких потрібно показати роботу системи. Підтримка різних форматів дозволяє адаптуватися до таких умов без необхідності попередньої конвертації даних.

Отже, підтримка широкого спектра медіафайлів у системі виявлення об'єктів є не просто технічною зручністю, а й концептуально важливою особливістю, що забезпечує гнучкість, адаптивність та функціональну повноту програмного продукту.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ

3.1. Розробка функціональної частини системи розпізнавання об’єктів

У сучасних інформаційних системах все більше уваги приділяється застосуванню технологій комп’ютерного зору для автоматизації процесів спостереження та аналізу навколишнього середовища. Актуальною та перспективною темою є використання нейронних мереж для розпізнавання об’єктів у відео. У цьому розділі розглядається процес розробки функціональної частини системи, яка реалізує розпізнавання об’єктів у реальному часі та на основі медіафайлів із використанням сучасних нейронних мереж.

Кожен із режимів реалізовано з урахуванням взаємодії з користувачем: через інтуїтивно зрозумілий графічний інтерфейс можна легко перемикатись між сценаріями роботи. Додаток самостійно визначає тип завантаженого медіафайлу та виконує відповідну обробку, включаючи детекцію, трекінг та збереження результатів.

Важливо зазначити, що створення декількох режимів дозволяє краще протестувати систему на різних етапах розробки, виявити слабкі місця у логіці або продуктивності, а також забезпечити ширше коло використання додатку.

Розділення функціоналу на кілька окремих сценаріїв дозволяє зменшити навантаження на систему, оптимізувати використання ресурсів та зменшити ймовірність помилок при обробці складних потоків даних.

3.2. Імпортування бібліотек та моделей

Першим і одним з найважливіших кроків у розробці будь-якого програмного забезпечення є імпортування необхідних бібліотек. Це дозволяє отримати доступ до функціоналу, розробленого іншими фахівцями, що значно прискорює процес розробки та підвищує надійність системи. У моєму випадку

для реалізації додатку з виявлення об'єктів у реальному часі буде використано наступний набір бібліотек:

```
import cv2
from ultralytics import YOLO
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
import mysql.connector
from datetime import datetime
import os
from deep_sort_realtime.deepsort_tracker import DeepSort
from openpyxl import Workbook
from openpyxl.styles import numbers
from openpyxl.chart import BarChart, Reference
```

cv2 (OpenCV): Ця бібліотека є основою для роботи з зображеннями та відео. Вона надає широкий спектр функцій для зчитування відеопотоків, обробки кадрів, виконання графічних операцій (наприклад, малювання прямокутників навколо об'єктів) та відображення результатів. Без OpenCV реалізація більшості операцій комп'ютерного зору була б вкрай складною.

ultralytics.YOLO: Це спеціалізований модуль, який надає доступ до моделі YOLO. YOLO є однією з найсучасніших та найефективніших архітектур для виявлення об'єктів, яка забезпечує високу швидкість обробки даних при збереженні відмінної точності. Модуль *ultralytics.YOLO* спрощує завантаження попередньо навчених моделей, виконання висновків та доступ до результатів виявлення.

tkinter: Ця бібліотека є стандартним набором інструментів Python для створення графічних інтерфейсів користувача (GUI). *tkinter* буде використовуватися для побудови вікон додатку, кнопок, полів введення, а також для відображення відео з виявленими об'єктами.

mysql.connector: Цей конектор дозволяє додатку взаємодіяти з базою даних MySQL. Використання MySQL буде необхідним для зберігання інформації про виявлені об'єкти (наприклад, час виявлення, тип об'єкта), що дозволить в подальшому аналізувати зібрані дані, формувати звіти та вести статистику.

datetime, os: Бібліотеки *datetime* та *os* є стандартними модулями Python для роботи з датами, часом та файловою системою відповідно. *datetime* буде корисна для позначення часу виявлення об'єктів, а *os* – для збереження файлів, наприклад, скріншотів з виявленими об'єктами, або для навігації по каталогах при роботі з відеофайлами.

deep_sort_realtime: Ця бібліотека надає реалізацію алгоритму DeepSORT. DeepSORT є популярним алгоритмом для відстеження об'єктів, який використовує інформацію про зовнішній вигляд об'єкта та його рух для підтримки його ідентичності в часі.

openpyxl: Ця бібліотека призначена для роботи з файлами Microsoft Excel (.xlsx). Вона буде використовуватися для експорту зібраних даних (наприклад, інформації про виявлені об'єкти та їх переміщення) у формат Excel, що дозволить легко аналізувати дані та створювати різноманітні діаграми та звіти.

Ініціалізація моделей

```
model = YOLO("yolo11n.pt")
tracker = DeepSort(max_age=30)
```

Завантажується модель YOLO з заданою версією (*yolo11n.pt*) та ініціалізується DeepSORT для трекінгу об'єктів. Параметр *max_age* відповідає за те, скільки кадрів трек буде зберігатися без оновлень.

3.3. Графічний інтерфейс додатку

Реалізовано головне вікно додатку з графічним інтерфейсом (рис.3.1), створене за допомогою бібліотеки *tkinter*. Його метою є забезпечення користувача зручним і зрозумілим способом взаємодії з усіма основними функціями системи розпізнавання об'єктів. При запуску додаток відкриває компактне меню, у якому користувач може обрати потрібний режим роботи, переглянути історію виявлень або завершити.

Перед побудовою інтерфейсу викликається функція *init_db()*, яка гарантує, що база даних створена та готова до збереження інформації. Потім

ініціалізується головне вікно з назвою "Object Detection Menu", яке центрується на екрані за допомогою функції *center_window*. Ця функція розраховує координати для розміщення вікна по центру з урахуванням розміру екрана та самого вікна.

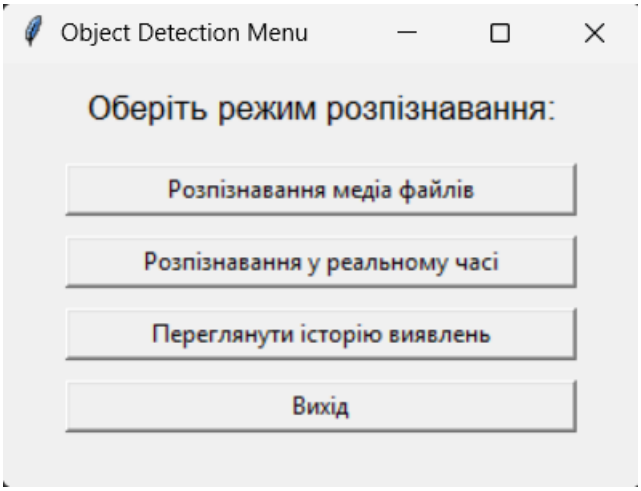


Рис. 3.1. Зображення головного меню

У самому вікні розміщено текстову мітку з інструкцією та кілька кнопок. Кожна кнопка виконує певну функцію:

"Розпізнавання медіа файлів" викликає функцію *choose_media()*, що відкриває діалог вибору відео або зображення для обробки.

"Розпізнавання у реальному часі" запускає обробку відеопотоку з веб-камери, але перед тим приховує головне вікно *(root.withdraw())*, щоб не заважало виведенню кадрів OpenCV.

"Переглянути історію виявлень" відкриває нове вікно з таблицею, збережених у базі даних, об’єктів.

"Вихід" завершує роботу додатку, закриваючи головне вікно.

Таким чином, реалізовано зручну точку входу до системи, яка забезпечує логічну організацію всіх функцій і дозволяє користувачу легко перемикатися між режимами. Це робить додаток зручним для користувача та придатним для застосування навіть людьми без досвіду роботи з Python або терміналом.

3.4. Розробка режимів та обробки виявлення об’єктів

Одним з ключових етапів є розробка режимів виявлення об'єктів, що дозволить користувачеві адаптувати функціонал додатку до конкретних потреб та сценаріїв використання. Різні завдання вимагають різного підходу до обробки візуальних даних, тому замість єдиного режиму було створено два основні варіанти. Це не тільки підвищує універсальність додатку, а також його ефективність у широкому спектрі застосувань.

3.4.1. Режим реального часу

Режим реального часу є серцем додатку, оскільки він реалізує його основну функціональність – виявлення та відстеження об'єктів у реальному часі. Цей режим дозволяє користувачеві підключитися до веб-камери або іншого відеоджерела та миттєво бачити результати роботи системи комп'ютерного зору.

Далі представлено фрагмент коду для реалізації цього режиму:

```
def detect_objects_video(root):
cap = cv2.VideoCapture(0)
while True:
ret, frame = cap.read()
if not ret:
break
results = model(frame, verbose=False)
process_results_with_tracking(results, frame)
cv2.imshow("Video Detection", frame)
cv2.waitKey(1)
if cv2.getWindowProperty
("Video Detection", cv2.WND_PROP_VISIBLE) < 1:
break
cap.release()
cv2.destroyAllWindows()
root.deiconify()
```

Функція *detect_objects_video(root)* (де *root* є посиланням на головне вікно Tkinter для оновлення GUI) буде відповідати за всю логіку цього процесу.

cap = cv2.VideoCapture(0) - тут ініціалізується об'єкт для захоплення відеопотоку за допомогою бібліотеки OpenCV.

Число 0 вказує на те, що додаток намагається захопити відео з основної веб-камери, підключеної до системи. Якщо у вас кілька камер, ви можете спробувати 1, 2 і так далі.

while True - запускається нескінченний цикл для обробки кадрів.

ret, frame = cap.read() - зчитується один кадр з камери.

if not ret - якщо кадр не зчитується (камера відключена/помилка), виходить з циклу.

results = model(frame, verbose=False) - виконується виявлення об'єктів на кадрі за допомогою YOLO.

process_results_with_tracking(results, frame) – обробляється результати, виконується трекінг DeepSORT та малюється рамки на кадрі.

cv2.imshow("Video Detection", frame) - відображається оброблений кадр у вікні OpenCV.

cv2.waitKey(1) - оновлюється вікно та чекаємо 1 мс.

if cv2.getWindowProperty("Video Detection", cv2.WND_PROP_VISIBLE) < 1 - якщо користувач закрив вікно відео, виходить з циклу.

cap.release() - звільняються ресурси веб-камери.

cv2.destroyAllWindows() - закривається всі вікна OpenCV.

root.deiconify() - повертається до головного вікна додатку.

Цей код забезпечує безперервне виявлення, відстеження та візуалізацію об'єктів з камери (рис.3.2), а також коректне завершення роботи.

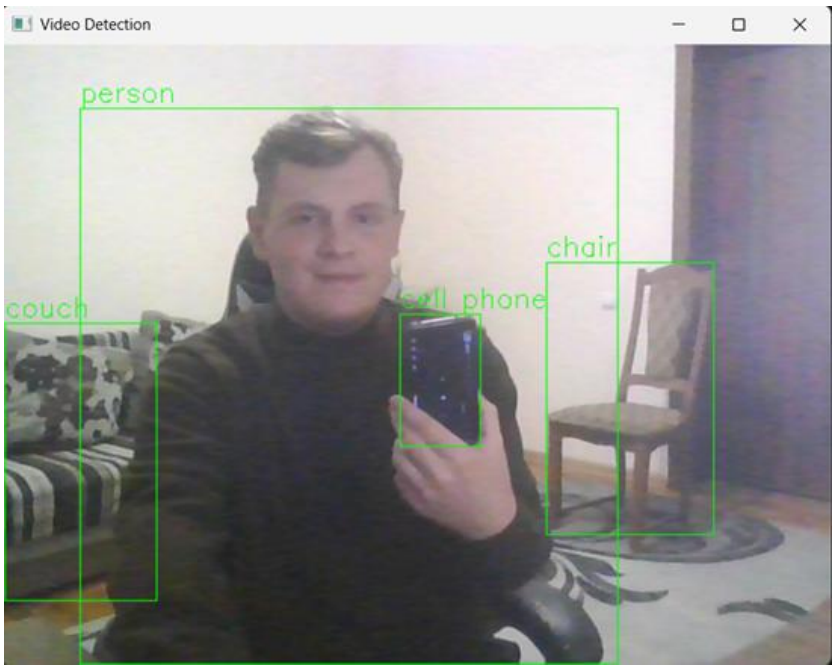


Рис. 3.2. Розпізнавання в реальному часі

3.4.2. Розпізнавання з файлу

У системах комп'ютерного зору, що фокусуються на динамічному аналізі, режим обробки медіафайлів є важливою, хоча й другорядною функціональністю. Він доповнює основний режим реального часу, дозволяючи користувачеві аналізувати попередньо записані зображення та відео. Цей режим особливо корисний для ретроспективного аналізу даних, тестування алгоритмів або демонстрації можливостей системи без потреби в активному підключенні до камери.

Обробка зображень:

```
if ext in ['.jpg', '.jpeg', '.png']:
image = cv2.imread(media_path)
image = cv2.resize(image, (800, 600))
results = model(image, verbose=False)
process_results_with_tracking(results, image)
cv2.imshow("Media Detection", image)
```

Якщо завантажений файл має розширення .jpg, .jpeg або .png, зображення зчитується за допомогою *cv2.imread()*, після чого масштабується до розміру 800×600 пікселів функцією *cv2.resize()*. Далі зображення передається на обробку об’єктів YOLO через виклик *model(image, verbose=False)*. Отримані результати

передаються у функцію *process_results_with_tracking()* для подальшої обробки з урахуванням відстеження об’єктів.

Зображення з результатами відображається у вікні за допомогою *cv2.imshow(),* яке залишається відкритим до моменту закриття користувачем.

Обробка відео:

```
cap = cv2.VideoCapture(media_path)
while cap.isOpened():
ret, frame = cap.read()
if not ret:
break
frame = cv2.resize(frame, (800, 600))
results = model(frame, verbose=False
process_results_with_tracking(results, frame)
cv2.imshow("Media Detection", frame)
cv2.waitKey(1)
```

Для відеофайлів ініціалізується завантаження відео з файлу *(cv2.VideoCapture(media_path)).*

Додаток в циклі зчитує кожен кадр, змінює його розмір, виконує виявлення та трекінг об’єктів, і відображає результат (рис.3.3). Цикл триває до кінця відео або закриття вікна користувачем. Після обробки відеофайлу ресурси звільняються, а вікна закриваються.

Рис. 3.3. Вигляд розпізнавання медіа файлів

Вибір медіафайлу:

```
def choose_media():
media_path = filedialog.askopenfilename(filetypes=
[("Media files", "*.jpg *.jpeg *.png *.mp4 *.avi")])
if media_path:
detect_objects_media(media_path)
```

Функція *choose_media()* інтегрує обидва режими з користувацьким інтерфейсом. Вона відкриває стандартне діалогове вікно вибору файлу *(filedialog.askopenfilename)*, фільтруючи лише підтримувані типи медіа (.jpg, .jpeg, .png, .mp4, .avi).

Якщо користувач вибирає файл, його шлях передається до *detect_objects_media()* для подальшої обробки.

3.5. Обробка та збереження результатів YOLO з трекінгом

Після того як об’єкти були виявлені на відеокадрі за допомогою моделі YOLO, наступним важливим етапом є обробка цих результатів, відстеження руху об'єктів у часі та збереження корисної інформації. Це необхідно для розпізнавання об’єктів, які переміщуються в кадрі, уникнення дублювання інформації про один і той самий об’єкт, створення повноцінної історії подій з візуальними матеріалами, аналітики на основі зафіксованих даних.

Функція *process_results_with_tracking()*, поєднує обробку результатів нейронної мережі, трекінг, збереження скріншотів і запис у базу даних. Також можна здійснювати перегляд історії (рис.3.4) та експорт даних у Excel-файл із побудовою діаграми.

Цей модуль забезпечує повний цикл роботи з виявлення об’єктів: від обробки кадру й виявлення унікальних об’єктів до створення візуальних матеріалів і фіксації в базі з можливістю подальшого аналізу. Завдяки поєднанню DeepSORT, MySQL та OpenCV, система стає гнучкою, надійною та придатною для практичного застосування, зокрема в системах безпеки, моніторингу, підрахунку відвідувачів тощо.

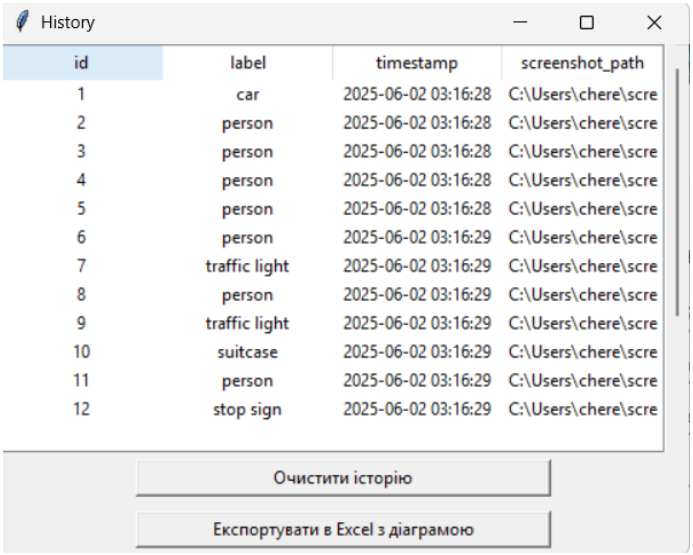


Рис. 3.4. Вигляд перегляду історії

3.5.1. Визначення координат виявлених об’єктів

На першому етапі функція *process_results_with_tracking()* обробляє вхідні результати з моделі YOLO.

```
boxes = result.boxes
if boxes is None:
continue
for box in boxes:
x1, y1, x2, y2 = map(int, box.xyxy[0])
conf = float(box.conf[0])
cls_id = int(box.cls[0])
label = model.names[cls_id]
detections_in_frame.append
(([x1, y1, x2 - x1, y2 - y1], conf, label))
```

Для кожного об’єкта отримуються координати обмежувальної рамки у форматі x1, y1, x2, y2, що визначають прямокутник, у якому знаходиться об’єкт.

Усі ці дані збираються у список *detections_in_frame*, який використовується для подальшого трекінгу.

Цей етап є підготовчим для трекінгу: визначаються просторові та категорійні ознаки кожного об’єкта.

3.5.2. Трекінг об’єктів у реальному часі з використанням DeepSORT

Після формування списку виявлених об’єктів, дані передаються до алгоритму трекінгу DeepSORT, який виконує ідентифікацію кожного об’єкта з присвоєнням йому унікального *track_id*.

```
track_id = track.track_id
l, t, r, b = map(int, track.to_ltrb())
label = track.det_class
```

Це дозволяє системі запам’ятовувати об’єкти між кадрами та уникати повторного збереження одного і того ж об’єкта.

Також відсіювання непідтверджених об’єктів (наприклад, тимчасові об’єкти або помилкові виявлення, що зникають швидко), актуалізацію позицій кожного об’єкта відповідно до його переміщення в кадрі.

Таким чином, система фіксує саме "стійкі" об'єкти, які реально присутні у відео протягом певного часу.

3.5.3. Визначення нових об’єктів, збереження скріншотів

Коли трекінг ідентифікує новий *track_id*, тобто об'єкт, який з'явився вперше, то зображення цього об'єкта вирізається з кадру (*cropped*) відповідно до координат.

```
screenshot_path = None
if track_id not in recent_ids:
recent_ids.add(track_id)
cropped = frame[t:b, l:r]
screenshot_dir = "screenshots"
os.makedirs(screenshot_dir, exist_ok=True)
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
screenshot_filename = f"{label}_{timestamp}.jpg"
screenshot_path=os.path.abspath
(os.path.join(screenshot_dir, screenshot_filename))
cv2.imwrite(screenshot_path, cropped)
save_detection_to_db(label, screenshot_path)
```

У каталозі *screenshots* створюється новий JPEG-файл. Назва файлу містить мітку часу та назву об'єкта, що дозволяє легко ідентифікувати його пізніше.

Встановлюється повний шлях до цього файлу (*screenshot_path*), який згодом буде записаний до бази даних.

```
cv2.rectangle(frame, (l, t), (r, b), (0, 255, 0), 1)
cv2.putText(frame, f"{label}", (l, t - 5),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 1)
```

На поточному кадрі виводиться прямокутник навколо об'єкта та його назва (*label*), що дозволяє користувачу одразу побачити результат роботи системи. Цей процес виконується лише один раз для кожного унікального об’єкта протягом сесії, завдяки механізму перевірки *recent_ids*.

3.5.4. Збереження інформації до бази даних

Одночасно з візуальним представленням результатів, система зберігає ключову інформацію в базу MySQL (рис.3.5). У таблицю *detections* записується назва об’єкта (*label*), дата, час виявлення (автоматично через поле *timestamp*) і шлях до скріншоту. З'єднання з базою здійснюється через функцію *connect_db()*, яка повертає активне з’єднання до бази з параметрами доступу. При першому запуску додатку викликається *init_db()*, яка створює таблицю, якщо вона ще не існує.

Таким чином, кожне виявлення надійно фіксується в базі, що дозволяє зберігати історію, аналізувати частоту появи певних об'єктів і вести облік.

id	label	timestamp	screenshot_path
1	per…	2025-06-…	C:\Users\chere\…
2	couch	2025-06-…	C:\Users\chere\…
3	chair	2025-06-…	C:\Users\chere\…
NULL	NULL	NULL	NULL

Рис. 3.5. Вигляд бази даних

3.5.5. Перегляд збережених об’єктів та експорт до Excel

У процесі функціонування системи обробки відео з використанням YOLO та DeepSORT зберігаються важливі дані про розпізнані об’єкти: їх класи, час

виявлення, а також знімки фрагментів кадрів, де ці об’єкти були зафіксовані. Щоб забезпечити зручний доступ до цієї інформації, у додатку реалізовано два корисні функціональні модулі: перегляд історії розпізнань і експорт результатів до Excel. Вони спрямовані на підвищення аналітичних можливостей додатку та створення більш прозорого і контрольованого процесу розпізнавання.

Перегляд історії

Відкривається нове вікно з таблицею (рис.3.6), що показує всі записи з бази даних. Інтерфейс побудовано на базі Tkinter з використанням віджетів Treeview, що дозволяють створювати зручну табличну структуру з можливістю прокрутки. Інформація представлена у вигляді стовпців: ID, назва об'єкта, час фіксації, шлях до зображення. Можна прокручувати список, переглядати деталі та відслідковувати порядок подій.

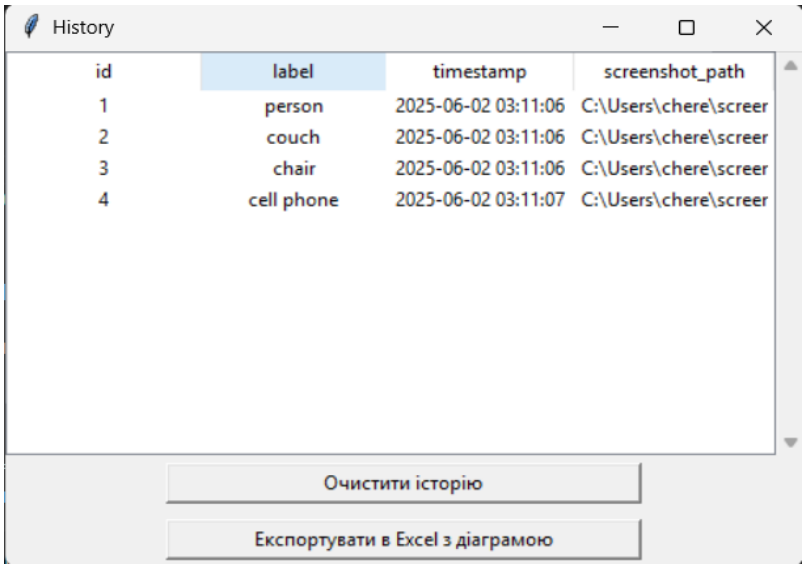


Рис. 3.6. Вигляд перегляду історії

Користувач може швидко переглянути список виявлень, орієнтуватися у часовій послідовності, а за потреби – скористатися шляхом до файлу для відкриття відповідного зображення.

Експорт у Excel

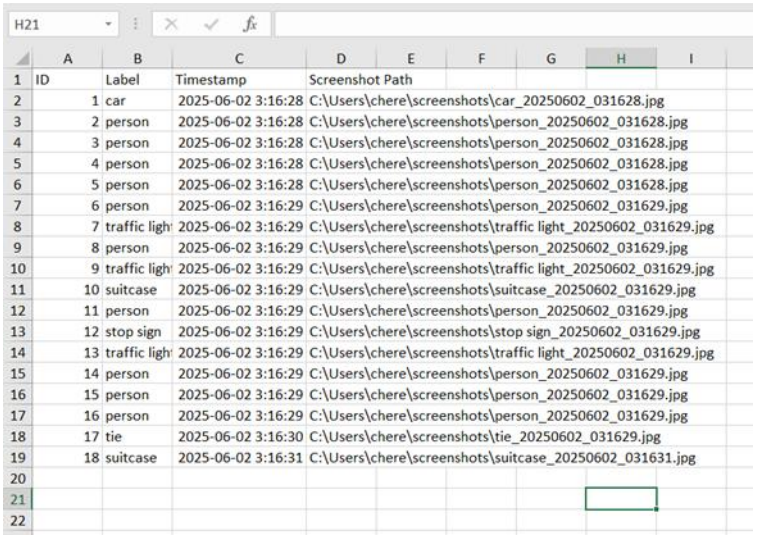
Для користувачів, які бажають проводити подальший аналіз зібраних даних або створювати звітність, реалізовано можливість експорту інформації у формат Microsoft Excel.

```
def export_to_excel():
conn = connect_db()
wb = Workbook()
ws = wb.active
ws.title = "Detections"
ws.append(["ID", "Label", "Timestamp", "Screenshot Path"])
for row in rows:
ws.append(row)
chart = BarChart()
data = Reference(chart_ws, min_col=2, max_row=len(label_count)+1)
cats = Reference(chart_ws, min_col=1, max_row=len(label_count)+1)
chart.add_data(data, titles_from_data=True)
chart.set_categories(cats)
chart.title = "Кількість об'єктів по категоріях"
chart_ws.add_chart(chart, "E5")
```

Механізм роботи функції:

- по команді система знову звертається до бази даних і отримує повний список записів;
- створюється новий .xlsx-файл, до якого по рядках записуються всі дані у форматі таблиці;
- окрім табличної частини, автоматично будується стовпчикова діаграма, яка наочно демонструє кількість виявлень об’єктів за класами. Це дозволяє швидко виявити, які об’єкти траплялися найчастіше.

Дані записуються у вигляді таблиці (рис.3.7), а також автоматично будується стовпчикова діаграма.



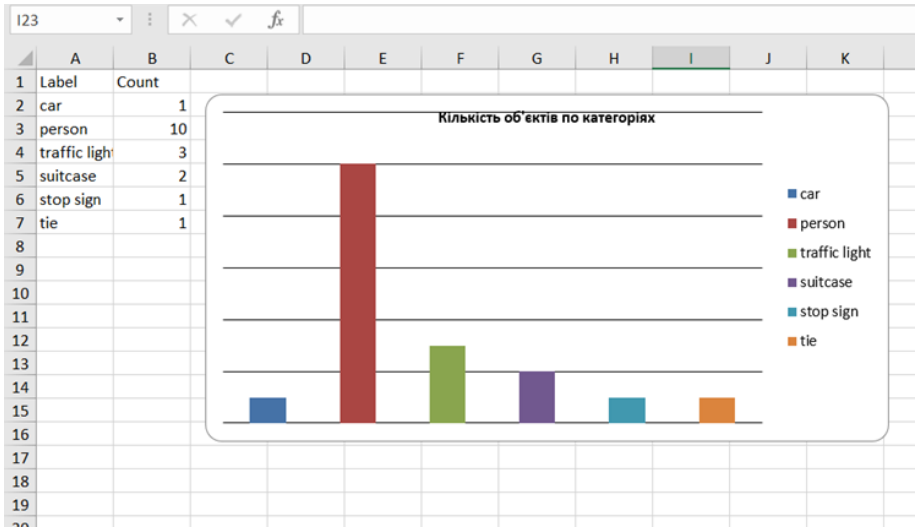
	A	B	C	D
1	ID	Label	Timestamp	Screenshot Path
2	1	car	2025-06-02 3:16:28	C:\Users\chere\screenshots\car_20250602_031628.jpg
3	2	person	2025-06-02 3:16:28	C:\Users\chere\screenshots\person_20250602_031628.jpg
4	3	person	2025-06-02 3:16:28	C:\Users\chere\screenshots\person_20250602_031628.jpg
5	4	person	2025-06-02 3:16:28	C:\Users\chere\screenshots\person_20250602_031628.jpg
6	5	person	2025-06-02 3:16:28	C:\Users\chere\screenshots\person_20250602_031628.jpg
7	6	person	2025-06-02 3:16:29	C:\Users\chere\screenshots\person_20250602_031629.jpg
8	7	traffic ligh	2025-06-02 3:16:29	C:\Users\chere\screenshots\traffic light_20250602_031629.jpg
9	8	person	2025-06-02 3:16:29	C:\Users\chere\screenshots\person_20250602_031629.jpg
10	9	traffic ligh	2025-06-02 3:16:29	C:\Users\chere\screenshots\traffic light_20250602_031629.jpg
11	10	suitcase	2025-06-02 3:16:29	C:\Users\chere\screenshots\suitcase_20250602_031629.jpg
12	11	person	2025-06-02 3:16:29	C:\Users\chere\screenshots\person_20250602_031629.jpg
13	12	stop sign	2025-06-02 3:16:29	C:\Users\chere\screenshots\stop sign_20250602_031629.jpg
14	13	traffic ligh	2025-06-02 3:16:29	C:\Users\chere\screenshots\traffic light_20250602_031629.jpg
15	14	person	2025-06-02 3:16:29	C:\Users\chere\screenshots\person_20250602_031629.jpg
16	15	person	2025-06-02 3:16:29	C:\Users\chere\screenshots\person_20250602_031629.jpg
17	16	person	2025-06-02 3:16:29	C:\Users\chere\screenshots\person_20250602_031629.jpg
18	17	tie	2025-06-02 3:16:30	C:\Users\chere\screenshots\tie_20250602_031629.jpg
19	18	suitcase	2025-06-02 3:16:31	C:\Users\chere\screenshots\suitcase_20250602_031631.jpg

Рис. 3.7. Вигляд історії в Excel

Переваги даного підходу:

- дані у форматі Excel легко інтегруються у зовнішні звіти, а також можуть бути передані іншим користувачам або збережені для архіву;
- діаграма дозволяє отримати узагальнене уявлення про розподіл об’єктів у часі.

Діаграма виводить кількість об'єктів кожного типу (наприклад: 10 людей, 1 машина тощо) (рис.3.8).



	A	B
1	Label	Count
2	car	1
3	person	10
4	traffic ligh	3
5	suitcase	2
6	stop sign	1
7	tie	1

Рис. 3.8. Вигляд діаграми кількості об’єктів в Excel

Це значно спрощує подальший аналіз, підготовку звітів або презентацій.

Обидва додаткові інструменти – перегляд історії та експорт до Excel є важливими, оскільки дозволяють користувачеві не тільки взаємодіяти з даними в реальному часі, але й працювати з ними. Вони значно підвищують зручність використання, та створюють додаткову аналітичну цінність, що є важливою перевагою у системі розпізнавання об’єктів.

ВИСНОВКИ

У даній кваліфікаційній роботі було здійснено комплексне дослідження, розробку та реалізацію додатку для виявлення та трекінгу об’єктів у режимі реального часу з використанням глибинного навчання. Результатом стала повнофункціональна система, яка поєднує сучасні алгоритми комп’ютерного зору з практичними інструментами збереження, візуалізації та аналізу даних.

По-перше, було детально проаналізовано сучасні методи детекції об’єктів, зокрема алгоритми YOLO, SSD та Faster R-CNN. На основі порівняльного аналізу за критеріями точності, швидкодії та обчислювальної ефективності було обрано модель YOLO як найоптимальнішу для завдань реального часу. Особливу увагу приділено її архітектурі, принципам роботи та застосуванню у прикладних задачах.

По-друге, досліджено методи трекінгу об’єктів у відеопотоці та впроваджено алгоритм DeepSORT, що забезпечує надійне відстеження рухомих об’єктів із збереженням їх ідентичності упродовж кадрів. Це дозволило реалізувати точну систему моніторингу, здатну коректно працювати навіть в умовах динамічних змін сцени.

По-третє, розроблено інтуїтивно зрозумілий графічний інтерфейс користувача на базі Tkinter, який дозволяє керувати процесом детекції, переглядати історію виявлень, зберігати скріншоти та експортувати результати у формат Excel. Для фіксації даних про виявлені об’єкти інтегровано реляційну базу даних MySQL, що забезпечує зручне збереження, пошук та аналіз інформації.

У результаті створено стабільний, продуктивний та зручний у використанні додаток, здатний працювати як з реальним відеопотоком, так і з медіафайлами. Реалізація системи підтвердила ефективність поєднання YOLO та DeepSORT у завданнях реального часу, а також практичну придатність розробки для використання у системах відеоспостереження, безпеки, аналітики руху та інших прикладних сферах.

Таким чином, в процесі виконання кваліфікаційної роботи було досягнуто поставленої мети – створено ефективний, зручний та надійний додаток для виявлення і відстеження об’єктів у реальному часі. Отримані результати підтверджують доцільність застосування сучасних методів нейронних мереж.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методи пошуку та розпізнавання об’єктів у відеозображеннях на мобільній платформі ios в реальному часі URL: https://science.lpnu.ua/sites/default/files/journal-paper/2020/feb/21040/var1ksm-19-26-36.pdf (дата звернення: 15.02.2025)
2. What is object detection? URL: https://www.ibm.com/think/topics/object-detection (дата звернення: 16.02.2025)
3. YOLO Object Detection Explained URL: https://www.datacamp.com/blog/yolo-object-detection-explained (дата звернення: 16.02.2025)
4. Faster R-CNN vs YOLO vs SSD — Object Detection Algorithms URL: https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc (дата звернення: 18.02.2025)
5. Object Detection with ssd, Faster RCNN, yolo URL: https://medium.com/@javadghasemi7/object-detection-with-ssd-faster-rcnn-yolo-ce29b5c6a045 (дата звернення: 20.02.2025)
6. Згорткова нейронна мережа URL: https://uk.wikipedia.org/wiki/%D0%97%D0%B3%D0%BE%D1%80%D1%82%D0%BA%D0%BE%D0%B2%D0%B0_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0 (дата звернення: 10.03.2025)
7. Convolutional Neural Networks (CNNs): A Deep Dive URL: https://viso.ai/deep-learning/convolutional-neural-networks/ (дата звернення: 15.03.2025)
8. Introduction to Object Detection Using Image Processing URL: https://www.geeksforgeeks.org/introduction-to-object-detection-using-image-processing/ (дата звернення: 16.03.2025)
9. Image Preprocessing Best Practices To Optimize Your AI Workflows URL: https://voxel51.com/learn/master-image-preprocessing-best-practices (дата звернення: 08.04.2025)

10. YOLO Object Detection Explained URL: https://www.datacamp.com/blog/yolo-object-detection-explained (дата звернення: 09.04.2025)
11. Mastering Deep Sort: The Future of Object Tracking Explained URL: https://www.ikomia.ai/blog/deep-sort-object-tracking-guide (дата звернення: 11.04.2025)
12. Python GUI Programming With Tkinter URL: https://realpython.com/python-gui-tkinter/ (дата звернення: 15.04.2025)
13. База даних MySQL URL: https://promoter.net.ua/articles/baza-danix-mysql.html (дата звернення: 17.04.2025)