МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

кафедра «Інформаційні системи та мережі»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи на тему:

Інформаційна система ідентифікації об'єктів з безпілотних літальних апаратів

Студента групи **ІТ-31сп Козака О.О.**

(шифр, прізвище та ініціали)

Керівник роботи _____ (___Тарас БАСЮК____)

Консультант _____ (_____)

_____ (_____)

Нормоконтроль _____ (__Андрій ВАСИЛЮК__)

Завідувач кафедри ІСМ _____ (**Дмитро ДОСИН**)

«_04__» __червня_____ **2025 р.**

ЛЬВІВ – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп’ютерних наук та інформаційних технологій
Кафедра «Інформаційні системи та мережі»
Спеціальність 126 "Інформаційні системи та технології"
Перший (бакалаврський) рівень вищої освіти
ОПП "Інтелектуальні інформаційні технології"

«ЗАТВЕРДЖУЮ»
Завідувач кафедри ІСМ _____
«_04_» __червня___ 2025 р.

ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу студента групи ІТ-31сп

_____Козака Олександра Олеговича_____

(прізвище, ім’я, по батькові)

1. Тема роботи Інформаційна система ідентифікації об'єктів з безпілотних літальних апаратів_____

затверджена наказом по НУ «ЛП» від «_11_»__березня___2025р. №_865-4-08_____

2. Термін здачі студентом закінченої роботи___30.05.2025р._____

3. Вихідні дані для роботи: методичні вказівки до виконання бакалаврської кваліфікаційної роботи для студентів освітньо-професійної програми спеціальності 126 «Інформаційні системи та технології» першого (бакалаврського) рівня вищої освіти, ресурси інтернету, літературні джерела

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити): провести аналіз предметної області, провести аналіз існуючих рішень, побудувати концептуальну модель інформаційної системи ідентифікації об’єкітв з безпілотних літальних апаратів, обрати програмні засоби, технології та методи для реалізації системи, реалізувати систему

5. Перелік графічного матеріалу: системи аналоги, дерево цілей, ієрархічна структура, IDEF0 діаграма, ієрархія процесів, графічні вікна системи

6. Перелік програмних продуктів, які належить використати в процесі розроблення роботи (проекту): AllFusion pocess Modeler, Pycharm IDE, Microsoft Edge, MS Word

7. Консультування роботи, із зазначенням розділів роботи

Розділ	Консультанти	Підпис, дата	
		завдання видав	завдання отримав

8. Дата, коли видано завдання _____24.02.2025 р._____

Керівник _____
(підпис)

Завдання отримав до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Етапи кваліфікаційної роботи	Термін виконання етапів роботи	Примітки
1	Аналітичний огляд літературних та інших джерел	24.02.2025 – 08.03.2025	Виконано
2	Вивчення базових алгоритмів прийняття рішень та вибір варіанту для розробки	08.03.2025 – 16.03.2025	Виконано
3	Системний аналіз об’єкта дослідження	16.03.2025 – 30.03.2025	Виконано
4	Побудова архітектури системи та проектування бази даних	30.03.2025 – 20.04.2025	Виконано
5	Тестування системи та верифікація результатів	20.04.2025 – 07.05.2025	Виконано
6	Оформлення кваліфікаційної роботи	08.05.2025	Виконано

Студент _____
(підпис)

Керівник роботи _____
(підпис)

ЗМІСТ

ВСТУП 6
РОЗДІЛ 1 9
Аналітичний огляд літературних та інших джерел 9
1.1. Методологічні засади дослідження 9
1.2. Методи та алгоритми розпізнавання об'єктів з БПЛА 11
1.2.1. Традиційні методи комп'ютерного зору 12
1.2.2. Методи на основі глибокого навчання 14
1.2.3. Специфічні технології для БПЛА 17
1.2.4. Виклики й перспективи розвитку методів ідентифікації об'єктів з БПЛА ... 22
1.3. Аналіз відомих програмних систем 23
Висновок до першого розділу 28
РОЗДІЛ 2 30
Аналітичний огляд літературних та інших джерел 30
2.1. Дерево цілей 31
2.2. Конкретизація функціонування системи 43
2.3. Побудова ієрархії процесів (функцій, задач) 56
Висновок до другого розділу 58
РОЗДІЛ 3 60
Програмні засоби розв’язання задачі 60
3.1 Вибір та обґрунтування засобів розв’язання задачі 60
3.1.1. Аналіз мов програмування для реалізації системи 60
3.1.2. Обґрунтування вибору Python, як основної мови розробки 67
3.1.3. Аналіз моделей розпізнавання 68
3.1.4. Обґрунтування вибору YOLOv11 76
3.2 Технічні характеристики обраних програмних засобів розроблення 78
Висновки до третього розділу 81
РОЗДІЛ 4 82
Практична реалізація 82

4.1 Опис створеного програмного засобу 82
4.1.1. Опис механізмів 82
4.1.2. Структура бази даних 91
4.2 Інструкція користувача 94
4.3 Аналіз контрольного прикладу 96
Висновок до четвертого розділу 106
ВИСНОВКИ 107
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 110
АНОТАЦІЯ 113
ANNOTATION 115
ДОДАТКИ 117
Додаток А код файлу yolo.py 117

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку цифрових технологій та автоматизації зростає потреба у високоточному зборі, аналізі та інтерпретації візуальних даних, що надходять із безпілотних літальних апаратів (БПЛА). Такі апарати широко застосовуються в різних сферах – від військової розвідки та моніторингу інфраструктури до сільського господарства, охорони довкілля, надзвичайних ситуацій і безпеки. Проте ефективність використання БПЛА безпосередньо залежить від здатності оперативно ідентифікувати об’єкти, що потрапляють у поле зору, та швидко приймати рішення на основі цих даних.

Актуальність розробки інформаційної системи для автоматизованого розпізнавання об'єктів з використанням глибокого навчання зумовлена як потребою в прискоренні обробки великого обсягу візуальної інформації, так і необхідністю зменшення людського впливу на процес ідентифікації. Сучасні технології комп’ютерного зору, зокрема нейронні мережі типу YOLO, дозволяють досягти високої точності детекції в режимі реального часу, що є критично важливим для динамічних задач, де від швидкості й достовірності розпізнавання залежить ефективність усієї системи.

Водночас багато існуючих рішень або є вузькоспеціалізованими, складними у розгортанні, або не адаптованими до специфіки зйомки з БПЛА. Це створює потребу в розробці індивідуального програмного забезпечення, що враховує особливості реального застосування, дозволяє працювати з фото, відео та відеопотоком, підтримує інтерактивний вебінтерфейс та зберігає результати у структурованому вигляді.

Мета і задачі дослідження. З урахуванням проведеного аналізу предметної області, технічних вимог та сучасних тенденцій у сфері комп’ютерного зору, метою даної бакалаврської кваліфікаційної роботи є створення веборієнтованої інформаційної системи, здатної здійснювати автоматизовану ідентифікацію об’єктів на основі зображень, відео та відеопотоків, отриманих із безпілотних літальних апаратів, з використанням нейромережевих алгоритмів глибокого навчання.

Для досягнення поставленої мети необхідно виконати такі *основні завдання:*

- проаналізувати предметну область, визначити актуальні підходи до використання БПЛА для збору візуальної інформації та методи її обробки;
- здійснити огляд сучасних методів комп'ютерного зору та моделей глибокого навчання для задач об'єктної детекції;
- проаналізувати наявні програмні рішення, що реалізують схожий функціонал, визначити їх переваги та обмеження;
- виконати системний аналіз майбутньої системи: побудувати дерево цілей, визначити функціональні блоки, розробити контекстну та деталізовані діаграми IDEF0;
- обґрунтувати вибір мов програмування, фреймворків та моделі YOLOv11 для реалізації системи;
- реалізувати інформаційну систему з підтримкою обробки фото, відео та відеопотоку, зберігання результатів і виведення даних у вебінтерфейсі;
- протестувати розроблену систему на вхідному датасеті, продемонструвати точність, функціональність та приклад використання;
- підготувати інструкцію користувача, структуру бази даних та технічну документацію відповідно до стандартів.

Об'єкт дослідження – це процес автоматизованої ідентифікації об'єктів на зображеннях, відео та відеопотоці, отриманих з безпілотних літальних апаратів.

Предмет дослідження – це методи, моделі та програмні засоби комп'ютерного зору і глибокого навчання, що використовуються для побудови інформаційної системи розпізнавання об'єктів з БПЛА.

Практичне значення одержаних результатів. Результатом дослідження стала створена програмна інформаційна система, яка дозволяє користувачам здійснювати розпізнавання об'єктів на фото, відео та відеопотоці з БПЛА через вебінтерфейс. Система підтримує авторизацію користувачів, візуалізацію результатів, що робить її придатною до використання у прикладних галузях: цивільний моніторинг, охорона, сільське господарство, екологічний контроль тощо. Отримані результати можуть бути

основою для подальшого розвитку системи, зокрема впровадження аналітики розпізнаних об’єктів, розширення типів вхідних даних або підключення до зовнішніх джерел потокового відео.

Апробація роботи. Основні результати та висновки дослідження були представлені на XX міжнародній науково-практичній конференції «Innovative scientific research: latest theories, modern methods and practices», що відбулася 20–23 травня 2025 року в місті Севілья, Іспанія.

РОЗДІЛ 1

Аналітичний огляд літературних та інших джерел

1.1. Методологічні засади дослідження

Розпізнавання об'єктів – це процес, який здійснює ідентифікацію та класифікацію об'єктів на зображенні чи відеопослідовності. Розпізнавання є невід'ємною частиною процесу автоматизованого аналізу даних, отриманих з безпілотних літальних апаратів (БПЛА). Процедура розпізнавання об'єктів полягає у використанні певних алгоритмів та технологій машинного навчання, які дозволяють системі визначати та класифікувати різноманітні об'єкти. Безсумнівно, інформаційна система використовує не лише алгоритми виявлення, а й методи попередньої обробки зображень, які слугують для підвищення якості розпізнавання. Розпізнавання об'єктів потрібне для: моніторингу сільськогосподарських угідь, контролю інфраструктури, аналізу лісових масивів, пошуково-рятувальних операцій, військової розвідки та безпеки периметру[1].

Розпізнавання об'єктів з БПЛА поділяється на такі методи:

- методи на основі класичного машинного навчання – використовують ручне виділення ознак (SIFT, SURF, HOG) та традиційні класифікатори (SVM, Random Forest, Boosting);
- методи глибокого навчання – засновані на використанні нейронних мереж різної архітектури (CNN, R-CNN, YOLO, SSD);
- гібридні методи – поєднують класичне машинне навчання з глибоким для оптимізації процесу розпізнавання.

Залежно від того, які підходи використовуються в процесі розпізнавання, системи можуть базуватися:

- на сегментації зображень – коли кожен піксель зображення класифікується і присвоюється певному класу об'єктів. Прикладами таких алгоритмів можуть слугувати: U-Net, SegNet, DeepLab, Mask R-CNN. У всіх цих прикладах система виконує поділ зображення на осмислені регіони, що дозволяє отримати контури об'єктів з піксельною точністю;

- на детектуванні об'єктів з використанням обмежувальних рамок. В цьому випадку система, яка виявляє об'єкти, може показати, що вона знайшла об'єкт одним із способів: окреслити його прямокутником або визначити координати. Архітектури, які використовуються для детектування: YOLO, SSD, R-CNN, Faster R-CNN;
- на класифікації зображень, коли система визначає наявність об'єкта на зображенні, але не вказує його точне положення. Для таких задач використовуються CNN, ResNet, DenseNet та інші мережі, які забезпечують високу точність визначення класу об'єкта.

Найбільш відомими прикладами архітектур, які забезпечують ефективне розпізнавання об'єктів з БПЛА, є YOLO (You Only Look Once) і Faster R-CNN. Ці архітектури дозволяють виявляти об'єкти на зображеннях з високою точністю та швидкістю, що критично важливо для обробки даних в режимі реального часу.

Всі ці методи розпізнавання так само можна поділити на однозадачні та багатозадачні. Однозадачне розпізнавання – базовий, традиційний метод виявлення з використанням тільки одного типу задачі (наприклад, лише класифікація). Найбільш поширеним прикладом багатозадачного розпізнавання є системи, які одночасно виконують детектування об'єктів і їх класифікацію. Можна виділити такі категорії задач:

- Класифікація – визначення класу об'єкта на зображенні.
- Локалізація – визначення положення об'єкта на зображенні.
- Сегментація – визначення контурів об'єкта з піксельною точністю.
- Трекінг – відстеження руху об'єкта в послідовності кадрів.

Однозадачні методи поділяються на:

- алгоритмічні – базуються на традиційних методах обробки зображень: виділення країв, порогова обробка, морфологічні операції тощо;

- засновані на машинному навчанні – для вилучення ознак і класифікації використовуються алгоритми навчання. Недоліками є: необхідність великих наборів даних; тривалий час навчання; потреба в обчислювальних ресурсах;
- гібридні – поєднують алгоритмічні та машинно-навчальні підходи. В їх основі лежить аналіз переваг кожного з підходів та їх інтеграція для підвищення ефективності розпізнавання.

Тема роботи пов'язана із створенням інформаційної системи розпізнавання об'єктів з безпілотних літальних апаратів. З огляду на це, її актуальність зумовлена тим, що в нашому світі зростає використання БПЛА для різноманітних задач, відповідно, зростає потреба в автоматизованих системах аналізу отриманих даних. На сьогоднішній день ще не було розроблено такого універсального рішення для розпізнавання об'єктів з БПЛА, яке б забезпечувало високу точність при різних умовах зйомки, працювало в режимі реального часу та мало низьке енергоспоживання. Більшість із загальновідомих та популярних систем лишаються ресурсоємними і непридатними для використання безпосередньо на борту БПЛА. Натомість менш популярні рішення, які забезпечують обробку даних на борту, мають низьку точність розпізнавання. Саме тому актуальною задачею є створення інформаційної системи розпізнавання об'єктів з безпілотних літальних апаратів, яка поєднує високу точність, швидкодію та енергоефективність.

1.2. Методи та алгоритми розпізнавання об'єктів з БПЛА

Розпізнавання об'єктів з безпілотних літальних апаратів (БПЛА) – це автоматизований процес виявлення, класифікації та локалізації об'єктів на зображеннях або відеопотоках, отриманих з борту безпілотного апарата. Ця галузь комп'ютерного зору набуває особливої актуальності в контексті збільшення використання БПЛА в різних сферах людської діяльності – від військових операцій до сільського господарства, моніторингу навколишнього середовища та пошуково-рятувальних місій.

Основною складністю розпізнавання об'єктів з БПЛА є особливості отримання зображень: різні кути огляду, висота зйомки, погодні умови, динамічний характер руху платформи, обмеження щодо обчислювальних ресурсів на борту та інші фактори, які впливають на якість та обробку даних. Відповідно, алгоритми, які застосовуються для такого розпізнавання, повинні враховувати ці особливості та забезпечувати стабільність роботи у різноманітних умовах.

Методи розпізнавання об'єктів з БПЛА можна класифікувати за різними критеріями, проте найбільш загальним є поділ на традиційні методи комп'ютерного зору та методи, засновані на глибокому навчанні.

1.2.1. Традиційні методи комп'ютерного зору

Традиційні методи комп'ютерного зору базуються на аналізі візуальних ознак зображень без використання методів глибокого навчання. Вони включають підходи, які застосовувалися ще до широкого впровадження нейронних мереж.

Методи на основі виділення ознак

Ці методи аналізують зображення шляхом виділення характерних ознак об'єктів, що дозволяє їх ідентифікувати та класифікувати. Основними компонентами таких методів є:

- *SIFT (Scale-Invariant Feature Transform)* – алгоритм для виділення ключових точок зображення, які інваріантні до масштабу, обертання та частково до змін освітлення. SIFT працює шляхом виділення точок інтересу на різних масштабах зображення, обчислення дескрипторів для цих точок та їх порівняння з дескрипторами об'єктів, які потрібно розпізнати[2].
- *SURF (Speeded-Up Robust Features)* – модифікація SIFT, спрямована на підвищення швидкодії. SURF використовує інтегральні зображення та приблизні обчислення для прискорення виділення ознак, зберігаючи при цьому хорошу точність розпізнавання[2].
- *HOG (Histogram of Oriented Gradients)* – метод, який фокусується на розподілі градієнтів інтенсивності та напрямків країв у локальних областях

зображення. HOG особливо ефективний для розпізнавання об'єктів, що мають характерні контури, наприклад, транспортних засобів або людей[2].

- *ORB (Oriented FAST and Rotated BRIEF)* – комбінація детектора FAST та дескриптора BRIEF, яка забезпечує високу швидкість роботи при задовільній точності, що особливо важливо для систем реального часу, які часто використовуються на БПЛА[2].

Перевагами методів на основі виділення ознак є:

- Відносно низькі вимоги до обчислювальних ресурсів
- Відсутність необхідності у великих наборах даних для навчання
- Висока швидкість роботи, що важливо для обробки даних у реальному часі

Однак ці методи мають обмеження при роботі зі складними сценами, змінним освітленням та різними ракурсами зйомки, характерними для БПЛА.

Методи сегментації зображень

Сегментація зображення передбачає поділ зображення на області, які відповідають різним об'єктам або їх частинам. У контексті розпізнавання з БПЛА використовуються такі методи сегментації.

Порогова сегментація – найпростіший метод, який розділяє зображення на області на основі значень інтенсивності пікселів. Для аерофотознімків цей метод може використовуватися для виділення водойм, рослинності або будівель.

Кластеризація K-means – групування пікселів за кольором, яскравістю та іншими характеристиками в K кластерів. Метод дозволяє розділити зображення на області з подібними характеристиками[3].

Watershed – алгоритм, який розглядає зображення як топографічну поверхню, де значення яскравості відповідають висоті. Це дозволяє визначити межі об'єктів як вододіли між басейнами на цій поверхні[3].

GrabCut – ітеративний алгоритм, який використовує графові моделі для сегментації об'єктів переднього плану від фону. Ця техніка особливо корисна для виділення об'єктів зі складним фоном[3].

Методи сегментації часто використовуються як попередня обробка перед застосуванням більш складних алгоритмів розпізнавання, оскільки вони дозволяють зменшити обсяг даних для аналізу та зосередитися на найбільш важливих частинах зображення.

Методи на основі шаблонів

Ці методи базуються на порівнянні частин зображення з заздалегідь визначеними шаблонами об'єктів.

Template Matching – пошук фрагментів зображення, які відповідають шаблону об'єкта. Для кожної позиції у вхідному зображенні обчислюється міра подібності з шаблоном[4].

Cascade Classifiers – каскадні класифікатори, такі як алгоритм Віоли-Джонса, які використовують простіші класифікатори на початкових етапах для швидкого відкидання областей, що не містять об'єктів інтересу, і більш складні класифікатори на пізніших етапах[4].

Chamfer Matching – метод, який шукає відповідності між контурами об'єктів та шаблонів, використовуючи карти відстаней Chamfer. Це дозволяє ефективно знаходити об'єкти з характерними контурами на аерофотознімках[4].

Методи на основі шаблонів показують хороші результати при розпізнаванні об'єктів стандартної форми, наприклад, транспортних засобів або будівель, але можуть бути недостатньо гнучкими для розпізнавання об'єктів зі змінною формою або при істотних змінах ракурсу зйомки.

1.2.2. Методи на основі глибокого навчання

Методи глибокого навчання, особливо згорткові нейронні мережі (CNN), зробили революцію в галузі комп'ютерного зору та значно підвищили точність розпізнавання об'єктів, у тому числі з БПЛА.

Згорткові нейронні мережі для класифікації

Згорткові нейронні мережі для класифікації зображень фокусуються на визначенні класу об'єкта на зображені.

AlexNet – одна з перших успішних архітектур CNN, яка значно покращила точність класифікації зображень порівняно з традиційними методами. Архітектура складається з п'яти згорткових шарів, за якими йдуть повнозв'язні шари.

VGG – мережа з дуже глибокою архітектурою (16-19 шарів), яка використовує маленькі згорткові фільтри (3×3) для підвищення точності класифікації. VGG досягає високої точності, але вимагає значних обчислювальних ресурсів.

ResNet – архітектура з залишковими з'єднаннями, яка дозволяє ефективно навчати дуже глибокі мережі (до 152 шарів і більше). ResNet і його варіанти часто використовуються як основа для більш складних моделей розпізнавання об'єктів.

MobileNet та *EfficientNet* – легкі архітектури, розроблені спеціально для мобільних і вбудованих пристроїв, які можуть використовуватися на борту БПЛА з обмеженими обчислювальними ресурсами.

Ці мережі навчаються на великих наборах даних, таких як ImageNet, і можуть бути адаптовані для розпізнавання конкретних об'єктів на аерофотознімках за допомогою передавального навчання.

Об'єктні детектори

Об'єктні детектори не лише класифікують зображення, але й локалізують об'єкти на ньому, зазвичай шляхом визначення обмежувальних рамок.

R-CNN (Region-based CNN) та його покращені версії (Fast R-CNN, Faster R-CNN) – двоетапні детектори, які спочатку пропонують регіони-кандидати, а потім класифікують їх та уточнюють обмежувальні рамки. Ці моделі досягають високої точності, але можуть бути занадто повільними для аналізу в реальному часі.

YOLO (You Only Look Once) – одноетапний детектор, який розглядає детекцію об'єктів як проблему регресії, передбачаючи одночасно обмежувальні рамки та класи об'єктів для всього зображення за один прохід. Різні версії YOLO (v3, v4, v5) пропонують різний баланс між швидкістю та точністю, що робить їх популярними для аналізу відео з БПЛА в реальному часі.

SSD (Single Shot MultiBox Detector) – ще один одноетапний детектор, який використовує кілька шарів різного масштабу для виявлення об'єктів різних розмірів.

SSD балансує між швидкістю та точністю, що робить його придатним для систем з обмеженими ресурсами.

EfficientDet – сімейство детекторів об'єктів, які масштабуються від легких моделей для мобільних пристроїв до великих моделей для серверів. EfficientDet використовує архітектуру EfficientNet як основу і двонапрямлену мережу ознак (BiFPN) для ефективного об'єднання ознак різних масштабів.

Об'єктні детектори особливо корисні для виявлення невеликих об'єктів на аерофотознімках високої роздільної здатності, таких як автомобілі, люди або тварини.

Семантична та екземплярна сегментація

На відміну від детекторів, які лише визначають обмежувальні рамки, методи сегментації визначають точні контури об'єктів.

Fully Convolutional Networks (FCN) – архітектура, яка адаптує класифікаційні мережі для задач сегментації, замінюючи повнозв'язні шари на згорткові. FCN генерує карти сегментації, де кожен піксель класифікується до певного класу.

U-Net – мережа з архітектурою енкодер-декодер та зв'язками між відповідними шарами енкодера і декодера. U-Net особливо ефективна для сегментації медичних зображень та аерофотознімків, де важливо зберегти деталі контурів.

Mask R-CNN – розширення Faster R-CNN, яке додає гілку для передбачення масок об'єктів паралельно з класифікацією та уточненням обмежувальних рамок. Mask R-CNN може виконувати екземплярну сегментацію, розрізняючи окремі екземпляри об'єктів одного класу.

DeepLab – сімейство моделей, які використовують розширені згортки (dilated convolutions) для збільшення поля сприйняття без втрати роздільної здатності. DeepLab досягає високої точності сегментації і часто використовується для аналізу аерофотознімків.

Методи сегментації особливо важливі для завдань, де потрібно визначити точні межі об'єктів, наприклад, для моніторингу змін ландшафту, аналізу рослинності або точного землеробства.

1.2.3. Специфічні технології для БПЛА

Розпізнавання об'єктів з безпілотних літальних апаратів (БПЛА) характеризується специфічними викликами та вимогами, що зумовлює необхідність адаптації існуючих методів комп'ютерного зору або розробки спеціалізованих підходів. Дана секція представляє комплексний огляд технологічних рішень, що забезпечують ефективне розпізнавання об'єктів з БПЛА.

Мультисенсорний підхід

Сучасні БПЛА часто оснащуються комплексом різнотипних сенсорів, що дозволяє інтегрувати гетерогенні дані для підвищення точності та надійності розпізнавання. Найбільш поширеними є наступні комбінації:

RGB + Thermal – інтеграція даних від стандартних оптичних камер з тепловізійними зображеннями забезпечує можливість виявлення об'єктів, що можуть бути невидимими в одному з спектрів. Даний підхід демонструє підвищену ефективність у пошуково-рятувальних операціях, системах безпеки та охорони територій, особливо при обмеженій видимості або в нічний час[5].

RGB + Multispectral – доповнення RGB-зображень даними мультиспектральних камер, що фіксують випромінювання в декількох діапазонах електромагнітного спектру, надає додаткові дискримінативні ознаки для класифікації рослинності, аналізу стану ґрунтів, моніторингу сільськогосподарських угідь та екологічних досліджень[6].

RGB + LiDAR – синтез оптичних зображень з даними лазерного сканування уможливлює створення прецизійних тривимірних моделей місцевості та об'єктів на ній, що суттєво підвищує точність детекції, класифікації та сегментації, особливо в умовах складної геометрії сцени[7].

Процес інтеграції даних з гетерогенних сенсорів потребує застосування спеціалізованих алгоритмів злиття даних, що функціонують на різних рівнях абстракції: від раннього злиття на рівні вхідних даних до пізнього злиття на рівні рішень. Незважаючи на підвищену обчислювальну складність, мультисенсорний підхід забезпечує значне покращення робастності систем розпізнавання об'єктів.

Алгоритми відстеження об'єктів

Відстеження об'єктів є критичним завданням при аналізі відеопотоку з БПЛА, особливо для моніторингу динамічних сцен та рухомих об'єктів. Основними алгоритмічними підходами є:

Калманівська фільтрація – рекурсивний метод, що використовує послідовність вимірювань для оцінки невідомих змінних стану з мінімізацією середньоквадратичної похибки. Калманівські фільтри та їх модифікації (розширений фільтр Калмана, незапахований фільтр Калмана) широко застосовуються для прогнозування траєкторії руху об'єктів.

KCF (Kernelized Correlation Filters) – алгоритм, що базується на застосуванні кореляційних фільтрів з ядровим перетворенням для відстеження об'єктів у відеопослідовності. KCF характеризується високою обчислювальною ефективністю, що дозволяє реалізувати відстеження об'єктів у режимі реального часу на вбудованих системах з обмеженими ресурсами.

SORT (Simple Online and Realtime Tracking) та DeepSORT – алгоритми, що поєднують детектори об'єктів з методами асоціації даних для одночасного відстеження множини об'єктів. DeepSORT додатково імплементує екстракцію глибоких ознак для покращення асоціації об'єктів між суміжними кадрами, що підвищує стійкість до короткочасних оклюзій та змін зовнішнього вигляду об'єктів.

SiamRPN та його модифікації – трекери на основі сіамських нейронних мереж, що здійснюють порівняння шаблону об'єкта з потенційними позиціями в поточному кадрі. Архітектури на основі регіональних пропозиційних мереж (RPN) демонструють оптимальний баланс між швидкодією та точністю відстеження.

Впровадження зазначених алгоритмів дозволяє БПЛА підтримувати неперервне спостереження за об'єктами інтересу навіть при тимчасовій втраті візуального контакту або значних змінах ракурсу спостереження.

Методи обробки з урахуванням ракурсу та висоти

Специфіка аеріальної зйомки з БПЛА вимагає розробки спеціалізованих методів, що враховують особливості спостереження з повітря:

Multi-view recognition – методи, що інтегрують інформацію з множини зображень одного об'єкта, отриманих з різних ракурсів під час польоту БПЛА. Застосування підходів на основі об'єднання ознак або рішень для різних точок спостереження суттєво підвищує надійність класифікації об'єктів.

Scale-adaptive detection – алгоритми, що адаптуються до варіацій масштабу об'єктів, зумовлених зміною висоти польоту БПЛА. Імплементація пірамідального представлення зображень або архітектур нейронних мереж з мультимасштабними ознаками забезпечує інваріантність до змін відстані до об'єктів.

Orientation-aware models – моделі, що враховують просторову орієнтацію об'єктів, критично важливу для розпізнавання з повітряних суден, оскільки вигляд об'єктів з висоти птичого польоту суттєво відрізняється від наземних спостережень. Інкорпорація механізмів ротаційної еквіваріантності або інваріантності в архітектури нейронних мереж підвищує їх ефективність для аеріальних зображень.

Domain adaptation – методи адаптації моделей, навчених на наземних зображеннях, для функціонування з аерофотознімками. Застосування техніки переносу навчання та доменної адаптації вирішує проблему дефіциту анотованих даних, отриманих з БПЛА, особливо для специфічних класів об'єктів.

Зазначені методи дозволяють подолати труднощі, пов'язані з особливостями аеріальної зйомки, та суттєво підвищити точність розпізнавання об'єктів з БПЛА.

Технології обробки даних в реальному часі

Ефективне функціонування БПЛА потребує мінімізації затримки при обробці даних, що досягається впровадженням наступних технологій:

Edge Computing – розміщення обчислювальних потужностей безпосередньо на борту БПЛА для проведення первинної обробки та аналізу даних без необхідності їх передачі на наземну станцію. Сучасні бортові комп'ютери (NVIDIA Jetson, Intel Neural Compute Stick, спеціалізовані FPGA) забезпечують достатню продуктивність для виконання алгоритмів комп'ютерного зору при відносно низькому енергоспоживанні.

Model Compression – методи компресії моделей машинного навчання, що включають квантизацію (зниження точності представлення вагових коефіцієнтів),

обрізання (елімінацію надлишкових нейронів або зв'язків) та дистиляцію знань (навчання компактної моделі імітувати поведінку об'ємної моделі). Застосування цих технік дозволяє зменшити розмір моделей та прискорити їх інференс на бортових обчислювальних пристроях.

TensorRT, OpenVINO, TFLite – фреймворки оптимізації та розгортання моделей глибокого навчання на пристроях з обмеженими ресурсами, що забезпечують значне прискорення процесу інференсу на бортових комп'ютерах БПЛА.

SLAM для візуальної навігації – алгоритми одночасної локалізації та картографування, що дозволяють БПЛА паралельно будувати мапу навколишнього середовища та визначати своє положення в просторі, використовуючи переважно візуальні дані. Сучасні імплементації ORB-SLAM, LSD-SLAM та їх модифікації функціонують у режимі реального часу на бортових комп'ютерах БПЛА.

Спеціалізовані архітектури нейронних мереж для БПЛА

Для оптимального функціонування на БПЛА розроблено спеціалізовані архітектури нейронних мереж:

Lightweight CNN – легкі архітектури згорткових нейронних мереж, оптимізовані для пристроїв з обмеженими обчислювальними ресурсами. MobileNet та його варіанти використовують глибинно-роздільні згортки для редукції кількості параметрів; EfficientDet оптимізує баланс між точністю та обчислювальною складністю для завдань детекції об'єктів; SlimYOLO та Tiny-YOLO представляють собою компактні версії популярних детекторів YOLO.

Attention-based models for aerial imagery – моделі з механізмами уваги, спеціально адаптовані для аерофотознімків, що забезпечують фокусування на найбільш інформативних регіонах зображення. Імплементація просторових модулів уваги виділяє релевантні області на зображенні; канальні механізми уваги визначають значущість різних каналів ознак; архітектури на основі трансформерів ефективно моделюють довгі залежності в даних.

Few-shot and Zero-shot Learning – методи, що уможливлюють розпізнавання нових класів об'єктів з мінімальною кількістю прикладів або за їх відсутності, що

особливо цінно для БПЛА, які можуть зустріти непередбачені об'єкти під час виконання місій.

Технології для подолання складних умов зйомки

БПЛА часто функціонують у несприятливих умовах, що вимагає спеціалізованих підходів до обробки зображень:

Dehazing and Defogging – алгоритми покращення видимості на зображеннях, отриманих в умовах туману або димки. Сучасні методи на основі глибокого навчання елімінують атмосферні артефакти з зображень у режимі реального часу.

Low-light Enhancement – методи підвищення якості зображень при недостатньому освітленні, критичні для цілодобового моніторингу. RetinexNet та його модифікації покращують контраст та деталізацію в темних регіонах; Zero-DCE (Zero-Reference Deep Curve Estimation) оптимізує темні зображення без еталонних зразків; URetinex-Net інтегрує класичну теорію Retinex з глибоким навчанням.

Motion Blur Correction – методи компенсації розмиття, спричиненого рухом БПЛА або об'єктів спостереження. Deblurring GAN застосовують генеративно-змагальні мережі для відновлення чітких зображень; DeblurNet та його варіанти оптимізовані для швидкої обробки на пристроях з обмеженими ресурсами.

Weather-robust Detection – методи детекції об'єктів, стійкі до варіативних погодних умов, включаючи опади та різке освітлення. Стратегії аугментації даних забезпечують навчання моделей для різноманітних погодних сценаріїв; доменна рандомізація нівелює розбіжності між синтетичними та реальними даними; самоконтрольоване навчання уможливлює адаптацію до нових умов без додаткового анотування даних.

Інтеграція з системами прийняття рішень

Сучасні технології розпізнавання об'єктів інтегруються з системами прийняття рішень БПЛА:

Event-driven recognition – системи, що активуються у відповідь на специфічні події, оптимізуючи використання обчислювальних ресурсів та енергії. Алгоритми

детекції змін ідентифікують значущі модифікації сцени для ініціації детального аналізу; методи детекції аномалій виявляють нетипові патерни, що потребують уваги.

Information-theoretic planning – алгоритми планування траєкторії БПЛА на основі інформаційної цінності потенційних спостережень. Активне сприйняття оптимізує збір даних для максимізації інформаційного приросту; планування оптимального наступного ракурсу визначає найбільш інформативну позицію для спостереження.

Multi-agent coordination – методи координації групи БПЛА для ефективного розподілу завдань розпізнавання. Розподілене відстеження об'єктів делегує завдання між кількома БПЛА; кооперативне сенсорне сприйняття оптимізує розташування апаратів для отримання комплементарної інформації; алгоритми консенсусу забезпечують узгодженість результатів розпізнавання між різними агентами.

Імплементація представлених технологій суттєво розширює функціональні можливості систем розпізнавання об'єктів на БПЛА, підвищуючи їх продуктивність, надійність та сферу застосування у різноманітних практичних сценаріях.

1.2.4. Виклики й перспективи розвитку методів ідентифікації об'єктів з БПЛА

Незважаючи на значний прогрес у галузі розпізнавання об'єктів з БПЛА, залишається низка викликів.

Обмеження обчислювальних ресурсів – більшість сучасних алгоритмів глибокого навчання вимагають значних обчислювальних ресурсів, які можуть бути недоступні на борту БПЛА. Це стимулює розробку легких та ефективних архітектур нейронних мереж.

Розпізнавання дрібних об'єктів – об'єкти на аерофотознімках часто займають невелику частину зображення, що ускладнює їх виявлення та класифікацію. Розвиток спеціалізованих методів для роботи з дрібними об'єктами є актуальним напрямком досліджень.

Змінні умови зйомки – погода, освітлення, час доби, сезон року значно впливають на вигляд об'єктів на аерофотознімках. Розробка алгоритмів, стійких до таких змін, залишається важливим завданням.

Обробка даних в реальному часі – багато застосувань вимагають аналізу відеопотоку в реальному часі, що накладає жорсткі обмеження на швидкість роботи алгоритмів.

Перспективними напрямками розвитку методів розпізнавання об'єктів з БПЛА такі.

Нейроморфні обчислення – використання спеціалізованих нейроморфних процесорів для ефективної реалізації нейронних мереж на борту БПЛА.

Самоконтрольоване навчання – методи, які дозволяють навчати моделі без великої кількості анотованих даних, що особливо важливо для специфічних завдань, де таких даних може бути недостатньо.

Активне навчання – підходи, які дозволяють моделі самостійно вибирати найбільш інформативні приклади для навчання, що зменшує потребу в великих анотованих наборах даних.

Інтеграція з іншими системами – поєднання систем розпізнавання об'єктів з системами автономної навігації, планування маршрутів та прийняття рішень для створення повністю автономних БПЛА здатних виконувати складні завдання.

Таким чином, методи та алгоритми розпізнавання об'єктів з БПЛА є критично важливим компонентом для розширення можливостей безпілотної авіації та її застосування в різних галузях. Поєднання традиційних методів комп'ютерного зору з сучасними підходами глибокого навчання дозволяє досягти високої точності та надійності розпізнавання в складних умовах експлуатації.

1.3. Аналіз відомих програмних систем

На сьогодні розроблено множину програмних засобів для здійснення ідентифікації об'єктів з БПЛА. Серед найбільш релевантних для нашої теми засобів можна виділити: DroneDeploy, Pix4D, Kespry Drone Intelligence Platform, DJI Terra та MATLAB Computer Vision Toolbox.

DroneDeploy – одна з найпопулярніших платформ для аналізу та обробки даних, отриманих з БПЛА. Система дозволяє створювати ортофотоплани, 3D-моделі

місцевості та проводити аналіз об'єктів на них. Програмне забезпечення має функції автоматичного розпізнавання об'єктів, можливість вимірювання параметрів ідентифікованих об'єктів, створення карт рельєфу та рослинності. На рисунку 1.1 представлено інтерфейс платформи DroneDeploy.

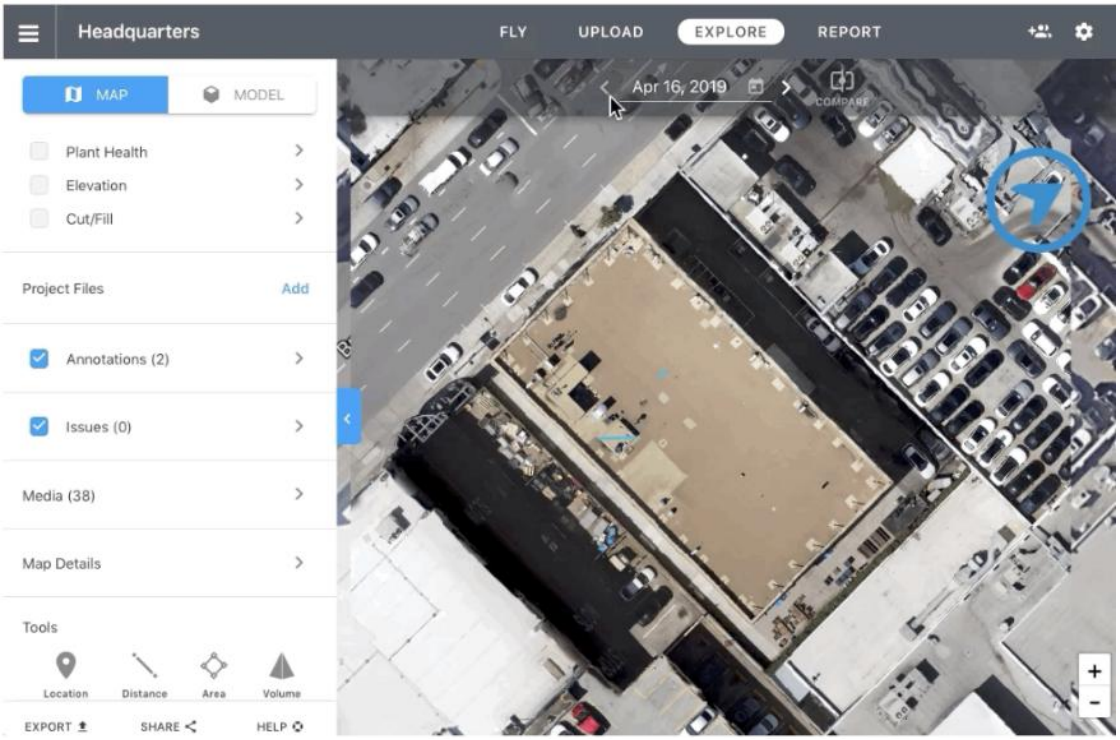

Рис. 1.1. Інтерфейс системи DroneDeploy

Перевагами системи є: інтуїтивно зрозумілий інтерфейс, хмарна обробка даних, що не потребує потужного обладнання для користувача, наявність мобільного додатку для планування польотів та перегляду результатів, висока швидкість обробки даних, автоматичне генерування звітів та можливість інтеграції з іншими програмними рішеннями. До недоліків можна віднести високу вартість підписки для професійного використання, обмежену функціональність при роботі офлайн, залежність від якості інтернет-з'єднання та деякі обмеження при обробці надвеликих масивів даних [8].

Pix4D – програмний комплекс, спеціалізований на фотограмметричній обробці даних з БПЛА. Система забезпечує точне картографування, створення цифрових моделей місцевості та автоматичне розпізнавання об'єктів. Особливістю Pix4D є висока точність створюваних моделей та можливість їх геоприв'язки.

Система використовує машинне навчання для автоматичної класифікації об'єктів на знімках. Точність ідентифікації об'єктів досягає 95%, а обробка даних середнього об'єму займає близько 30 хвилин. На рисунку 1.2 зображено інтерфейс системи Pix4D.

Рис. 1.2. Інтерфейс системи Pix4D

Переваги Pix4D включають: підтримку різноманітних форматів даних, детальну документацію та навчальні матеріали, потужні інструменти для аналізу та вимірювання, а також спеціалізовані модулі для різних галузей застосування. Недоліками є висока вартість ліцензії, вимоги до апаратного забезпечення для обробки великих проектів та відносно складна крива навчання для початківців[9].

Kespry Drone Intelligence Platform – хмарна платформа для автоматичного збору, обробки та аналізу даних з БПЛА, спеціалізована на комерційних застосуваннях. Система орієнтована на використання в таких сферах, як видобуток корисних копалин, страхування, будівництво та енергетика.

Kespry забезпечує автоматичне планування польотів, збір просторових даних та генерацію аналітичних звітів. Платформа інтегрує дані в єдине середовище для оцінки об'єктів, об'ємних розрахунків, аналізу пошкоджень тощо. Однією з ключових особливостей є можливість збору даних без активної участі оператора, що мінімізує людський фактор.

Переваги Kespry Drone Intelligence Platform включають: автоматизацію всього процесу збору та обробки даних, високу точність аналітики, простий у використанні інтерфейс, безпечне зберігання даних у хмарі, можливість інтеграції із системами управління підприємствами. Серед недоліків варто зазначити високу вартість рішення, залежність від інтернет-з'єднання для повноцінної роботи та обмежену гнучкість налаштувань для спеціалізованих сценаріїв використання. На рисунку 1.3 продемонстровано інтерфейс Kespry Drone Intelligence Platform [10].

Рис. 1.3. Інтерфейс системи Kespry Drone Intelligence Platform

DJI Terra – програмне забезпечення для планування польотів, картографування, 3D-моделювання та аналізу даних з дронів. Система розроблена для інтеграції з БПЛА компанії DJI та забезпечує повний цикл обробки даних – від польоту до створення цифрових моделей місцевості.

DJI Terra дозволяє створювати високоточні ортофотоплани, 3D-реконструкції інфраструктурних об'єктів та цифрові моделі поверхні. Платформа підтримує обробку великих масивів даних у режимі реального часу, що значно пришвидшує процес прийняття рішень [11].

Рис. 1.4. Інтерфейс системи DJI Terra

Переваги DJI Terra включають: глибоку інтеграцію з популярними моделями дронів DJI, високу швидкість обробки даних, можливість створення місій для автоматичних польотів за заданими параметрами, підтримку обробки даних без підключення до Інтернету, гнучкі інструменти для побудови карт і моделей. До недоліків можна віднести високу вартість ліцензії на повну версію, обмеження на підтримку моделей дронів сторонніх виробників, а також високу вимогливість до апаратного забезпечення при обробці великих проектів. На рисунку 1.5 зображено інтерфейс програмного забезпечення DJI Terra.

MATLAB Computer Vision Toolbox – набір інструментів для розробки алгоритмів комп'ютерного зору та обробки зображень в середовищі MATLAB. Цей інструментарій часто використовується для створення систем ідентифікації об'єктів з БПЛА, особливо в академічному та дослідницькому середовищі. На рисунку 1.6 зображено інтерфейс MATLAB Computer Vision Toolbox.

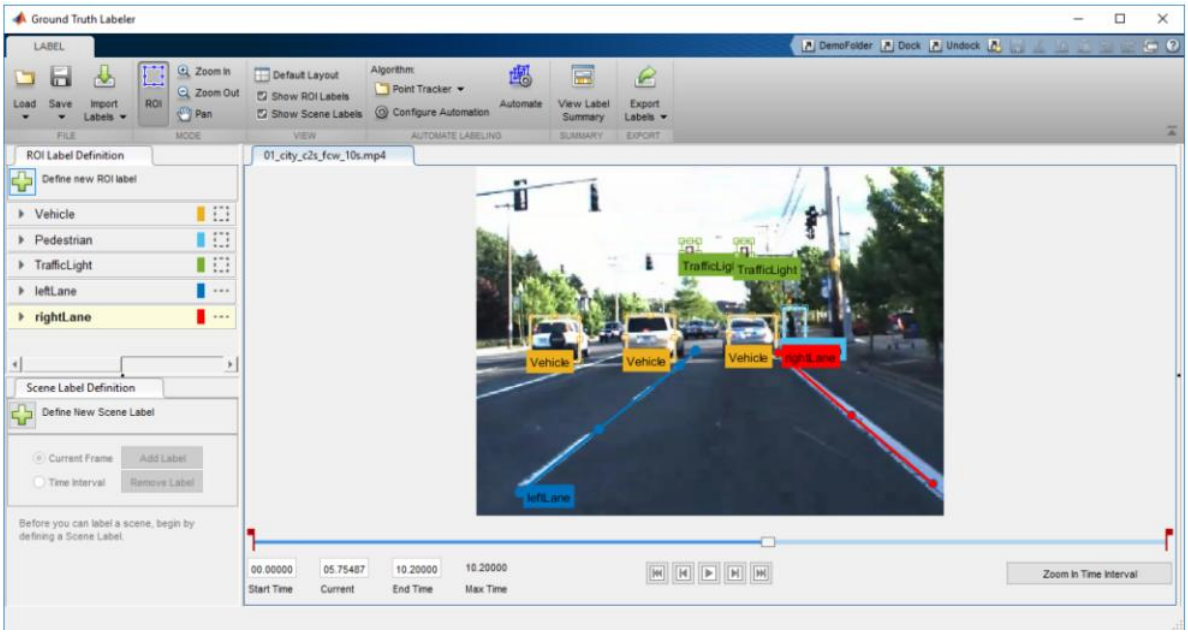

Рис. 1.5. Інтерфейс MATLAB Computer Vision Toolbox

Toolbox включає функції для виявлення та відстеження об'єктів, сегментації зображень, калібрування камери та відновлення 3D-сцен. MATLAB також підтримує інтеграцію з глибокими нейронними мережами через Deep Learning Toolbox [12].

Переваги MATLAB Computer Vision Toolbox: широкий набір готових функцій та алгоритмів, висока гнучкість при розробці власних алгоритмів, потужне середовище для експериментів та тестування, відмінна документація та навчальні матеріали, можливість компіляції коду у самостійні додатки. Недоліки: висока вартість ліцензії, вимагає знань MATLAB, не оптимізований для розробки виробничих систем, порівняно нижча швидкодія при обробці великих масивів даних порівняно з низькорівневими мовами програмування.

Висновок до першого розділу

У першому розділі було здійснено ґрунтовний аналіз методологічних засад розпізнавання об'єктів з використанням безпілотних літальних апаратів. Розглянуто основні підходи до розпізнавання, серед яких виокремлено методи класичного машинного навчання, методи глибокого навчання та гібридні технології. Особливу увагу приділено аналізу архітектур нейронних мереж, алгоритмів сегментації, класифікації та детектування об'єктів у специфічних умовах аерофотозйомки.

Встановлено, що застосування традиційних методів забезпечує швидкість обробки даних при обмежених обчислювальних ресурсах, однак поступається методам глибокого навчання за точністю та стійкістю до змін умов зйомки. Методи глибокого навчання, зокрема архітектури YOLO, Faster R-CNN, U-Net та DeepLab, демонструють високу ефективність у вирішенні завдань розпізнавання об'єктів, проте потребують значних обчислювальних ресурсів, що ускладнює їх використання на борту БПЛА.

Проаналізовано існуючі програмні рішення для розпізнавання об'єктів з БПЛА, серед яких DroneDeploy, Pix4D, Kespry Drone Intelligence Platform, DJI Terra та MATLAB Computer Vision Toolbox. Виявлено, що більшість із них характеризується або високою вартістю, або високими вимогами до технічної підготовки користувача, або обмеженнями у застосуванні для специфічних задач.

На основі проведеного огляду обґрунтовано доцільність розробки нової інформаційної системи розпізнавання об'єктів з безпілотних літальних апаратів, яка повинна забезпечувати високу точність і швидкість обробки даних при обмежених ресурсах, бути адаптивною до різних умов зйомки та доступною для широкого кола користувачів.

РОЗДІЛ 2

Аналітичний огляд літературних та інших джерел

Розробка інформаційної системи потребує не лише технічної реалізації, а й ґрунтовного аналізу предметної області, в якій ця система функціонуватиме. Системний аналіз дозволяє структурувати початкову ідею, виявити ключові завдання, врахувати всі можливі обмеження та сформулювати вимоги до майбутнього програмного продукту. Це дає змогу уникнути помилок на етапі проєктування та забезпечити узгодженість між очікуваннями користувачів і функціональністю системи.

У контексті даної роботи об'єктом дослідження є інформаційна система ідентифікації об'єктів з безпілотних літальних апаратів (БПЛА). Сучасні БПЛА широко використовуються для моніторингу територій, пошуку об'єктів, охорони інфраструктурних об’єктів та в багатьох інших сферах. У зв’язку з цим виникає потреба в автоматизованих засобах аналізу відеоінформації, зокрема, у швидкій та точній ідентифікації об’єктів у реальному часі.

Метою системного аналізу у межах даного проєкту є визначення вимог до функціонування системи, вибір ефективних підходів до обробки зображень і виявлення об’єктів, а також формування загальної структури майбутньої програмної системи. Особливу увагу приділено здатності системи працювати з потоковим відео, точності розпізнавання, обмеженням у ресурсах (наприклад, при використанні на портативних пристроях), а також зручності у використанні для кінцевих користувачів.

Системний підхід у цьому випадку базується на таких принципах:

- *Цілісність* – система розглядається як єдине функціональне ціле, що включає модулі збору відеоданих з БПЛА, обробки інформації, ідентифікації об’єктів та виведення результатів у зрозумілому для користувача форматі;
- *Ієрархічність* – архітектура системи побудована з дотриманням принципу багаторівневої організації. Верхній рівень включає загальні функції, пов’язані з ціллю системи – ідентифікацією об’єктів у реальному часі, розпізнавання звантажених відео та фото. Нижчі рівні деталізують окремі підсистеми:

користувацький інтерфейс, модуль розпізнавання, систему обробки відеопотоку, підсистему збереження та передачі даних тощо. Такий підхід дозволяє полегшити аналіз, проєктування і подальшу розробку за рахунок розподілу складності та забезпечення чіткої структуризації функціональних елементів;

- *Гнучкість і адаптивність* – система проєктується з урахуванням майбутніх змін у вимогах або умовах експлуатації. Передбачається можливість модернізації: інтеграція нових алгоритмів глибинного навчання для покращення точності розпізнавання, розширення типів об'єктів, що підлягають ідентифікації, або перехід до інших джерел відеоданих. Це забезпечує тривалу актуальність розробки та її готовність до роботи в динамічному середовищі застосування;
- *Формалізація* – вона реалізується шляхом побудови структурованих моделей, зокрема дерева цілей, контекстної діаграми IDEF0 та її декомпозиції[13]. Ці моделі використовуються для графічного представлення взаємозв'язків між основними модулями системи: збору відеопотоку з БПЛА, попередньої обробки даних, ідентифікації об'єктів, виведення результатів та взаємодії з користувачем. Такий підхід дозволяє чітко окреслити функціональні обов'язки кожного модуля, визначити, які дані надходять на вхід системи, які обробляються усередині, а які – передаються далі;
- *Практична орієнтованість* – функціональні можливості системи орієнтовані на вирішення прикладних завдань у реальних умовах, таких як виявлення об'єктів на складній місцевості, в умовах обмеженої видимості або в режимі реального часу під час виконання розвідувальних, моніторингових чи рятувальних операцій.

Таким чином, системний аналіз виконує роль основи, на якій базується технічне завдання, визначається архітектура системи, обираються технології реалізації та формується бачення фінального продукту.

2.1. Дерево цілей

На початковому етапі системного аналізу важливо сформувати чітке уявлення про загальну мету розробки та деталізувати її через ієрархічну структуру підцілей і

задач. Для цього застосовується метод побудови дерева цілей, що дозволяє визначити логічну послідовність дій, необхідних для реалізації проєкту, а також встановити взаємозв'язки між ключовими компонентами системи. Такий підхід дає змогу послідовно перейти від абстрактної мети до конкретних технічних завдань, забезпечуючи системність і контрольованість процесу проєктування. Особливо важливим це є у складних інформаційних системах, де взаємодіють різні підсистеми, а до кінцевого результату висуваються численні, часто суперечливі вимоги.

Дерево цілей – це ієрархічна графічна або логічна модель, яка відображає головну мету системи, розбиває її на підцілі, задачі та функції, що необхідні для досягнення бажаного результату. У структурі дерева цілей основна мета розміщується на найвищому рівні, після чого здійснюється її послідовна декомпозиція до рівня конкретних операцій або критеріїв оцінювання. Така структура є зручним інструментом для стратегічного планування, розробки технічного завдання та прийняття обґрунтованих інженерних рішень[14]. Дерево цілей дозволяє забезпечити узгодженість між різними аспектами системи, уникнути логічних суперечностей і виявити критичні елементи, від яких залежить успішність реалізації проєкту. Окрім того, воно є основою для подальшого використання формальних методів оцінки, зокрема, методу аналітичної ієрархії, який дозволяє визначити пріоритетність критеріїв та оптимізувати вибір проєктних рішень.

У межах даного дослідження на рисунку 2.1 побудоване дерево цілей, котре відображає цілісну структуру розробки інформаційної системи ідентифікації об'єктів з безпілотних літальних апаратів (БПЛА). Центральним елементом і, відповідно, головною метою системи є створення програмного рішення, яке здатне забезпечити автоматизоване виявлення, ідентифікацію та класифікацію об'єктів на основі відеопотоку, отриманого з бортових камер БПЛА. Такий підхід дозволяє підвищити ефективність виконання завдань моніторингу, пошуку, охорони територій, реагування на надзвичайні ситуації та виконання інших оперативних функцій у режимі реального часу.

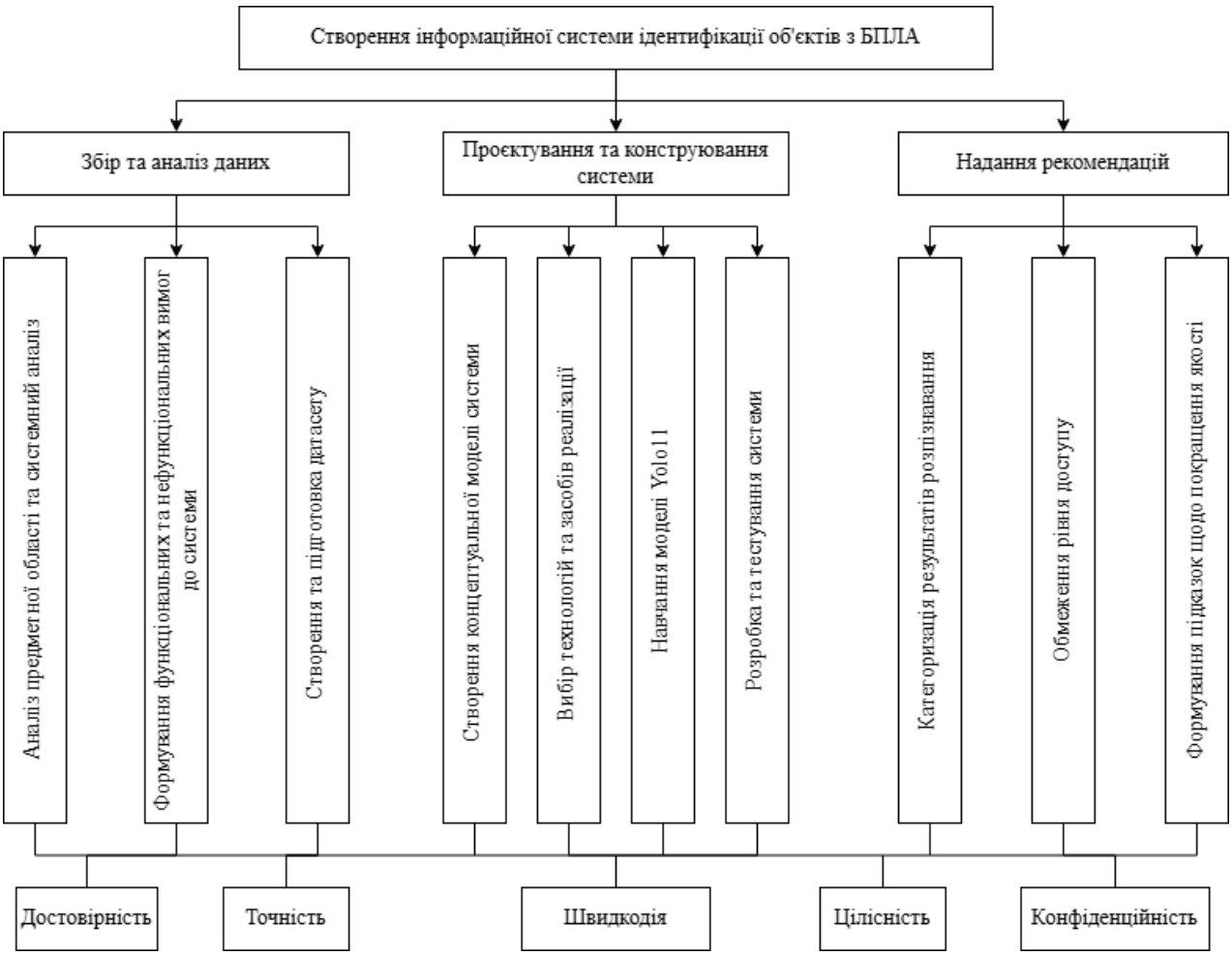

Рис. 2.1 Дерево цілей

Досягнення поставленої мети потребує врахування цілого комплексу вимог – від попередньої підготовки даних до реалізації функціоналу з формування рекомендацій для користувача. Загальна мета системи зумовлює необхідність реалізації трьох ключових функціональних блоків, які утворюють основні гілки дерева цілей.

Першим блоком виступає збір та аналіз даних, що охоплює підготовчий етап, який закладає основу для функціональної частини системи. До його основних завдань належать вивчення предметної області, аналіз існуючих рішень, визначення проблемних аспектів, а також формування функціональних і нефункціональних вимог. Важливою підціллю цього блоку є створення навчального датасету, що передбачає добір, маркування та попередню обробку зображень і відеофрагментів, які використовуватимуться для навчання моделі розпізнавання. Цей етап відіграє

вирішальну роль у забезпеченні достовірності, повноти та відповідності вхідних даних поставленим задачам системи.

Другим функціональним блоком, який становить ядро розробки системи, – це проєктування та конструювання. У межах цього етапу формується концептуальна модель системи, визначається її логічна структура та архітектура, обираються технології реалізації. Особливу увагу приділено вибору та навчання моделі комп'ютерного зору (у даному випадку – YOLOv11), що забезпечує розпізнавання об'єктів у відеопотоці з БПЛА. Результатом цього етапу є безпосереднє створення та тестування програмного забезпечення, яке включає реалізацію внутрішніх алгоритмів, побудову інтерфейсу користувача та перевірку коректності функціонування системи в цілому. Цей блок цілей забезпечує досягнення високих показників точності, швидкодії та цілісності програмного продукту.

Третім та останнім функціональним блок є напрямок, пов'язаний з наданням рекомендацій. Його мета полягає у підвищенні прикладної ефективності системи та її зручності для користувача. У рамках цього етапу реалізуються механізми класифікації розпізнаних об'єктів, впроваджуються засоби управління доступом до результатів аналізу, а також формуються інтелектуальні підказки щодо подальших дій або поліпшення якості введення. Ці функції дозволяють адаптувати систему до потреб різних категорій користувачів і забезпечують необхідний рівень конфіденційності та керованості доступу до чутливої інформації.

Таким чином, структура дерева цілей дозволяє послідовно організувати розробку системи ідентифікації об'єктів з БПЛА та створює основу для подальшого формального аналізу ефективності реалізації окремих цілей на основі багатокритеріального підходу.

На найнижчому рівні дерева цілей розміщено критерії оцінювання, за якими визначається успішність реалізації проєкту. Вони логічно випливають із підцілей трьох основних напрямків: збір та аналіз даних, проєктування та конструювання системи та надання рекомендацій. Для даної інформаційної системи до таких

ключових критеріїв належать: точність, швидкодія, достовірність, цілісність і конфіденційність.

Точність – здатність системи правильно виявляти та класифікувати об’єкти у відеопотоці з мінімальною кількістю помилкових спрацювань. Висока точність є критичною для використання системи в умовах змінного освітлення, руху та часткового перекриття об’єктів.

Швидкодія – ефективність обробки відеоданих у реальному часі. Забезпечення низької затримки між надходженням зображення і результатом обробки необхідне для оперативного прийняття рішень.

Достовірність – відповідність результатів ідентифікації реальному стану об’єктів. Вона залежить як від якості навчального датасету, так і від стійкості моделі до різних умов зйомки.

Цілісність – узгодженість роботи всіх компонентів системи, включаючи модулі збору, аналізу та виведення даних. Вона визначає стабільність функціонування та можливість масштабування.

Конфіденційність – захист даних та обмеження доступу до результатів. Актуально для систем, що працюють із чутливою інформацією, зокрема в оборонній чи цивільній сфері.

Ці критерії формують основу для подальшого застосування методу аналітичної ієрархії з метою вибору оптимальної архітектури системи.

Наступним етапом є обґрунтування вибору доцільного типу інформаційної системи для реалізації задач автоматизованої ідентифікації об’єктів з БПЛА. З цією метою використано метод аналітичної ієрархії (МАІ), який дозволяє формалізувати процес прийняття рішень шляхом побудови ієрархічної моделі з подальшим попарним порівнянням критеріїв та альтернатив. Це дає змогу визначити вагомість кожного критерію та обґрунтувати вибір архітектурного рішення[15].

У межах дослідження сформовано трирівневу структуру: на першому рівні – загальна мета (вибір оптимального типу системи для ефективної ідентифікації об’єктів у відеопотоці з БПЛА); на другому – п’ять ключових критеріїв: достовірність,

точність, швидкодія, цілісність і конфіденційність; на третьому – чотири альтернативи реалізації: інформаційно-аналітична, інформаційно-керуюча, система підтримки прийняття рішень та інформаційна система. Ієрархія, зображена на рисунку 2.2, є основою для подальшого порівняльного аналізу.

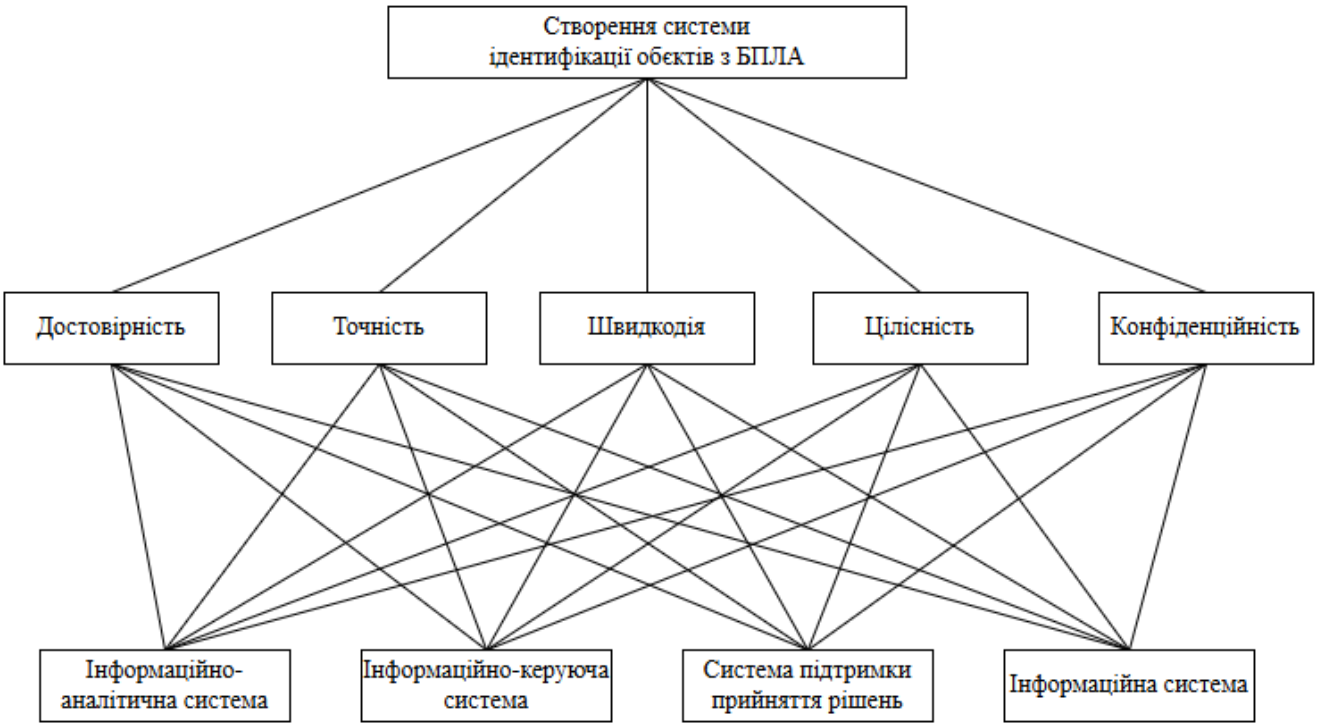

Рис. 2.2 Аналіз ієрархій

Для реалізації методу аналітичної ієрархії на наступному етапі необхідно визначити відносну значущість кожного з критеріїв, сформульованих на другому рівні ієрархічної структури. З цією метою здійснюється побудова матриці попарного порівняння (табл. 2.1), у якій кожен критерій оцінюється відносно інших за ступенем важливості щодо досягнення загальної цілі. Оцінювання виконується відповідно до шкали пріоритетів, запропонованої Т. Сааті, що базується на дев’ятибальній градації: від 1 – для рівнозначних критеріїв, до 9 – для критеріїв із абсолютною перевагою одного над іншим. Такий підхід дозволяє об’єктивно кількісно виразити експертні оцінки та на їх основі обчислити вагові коефіцієнти, що характеризують відносну важливість кожного критерію в контексті досягнення поставленої мети.

Таблиця 2.1

Матриця попарного порівняння критеріїв

№	Критерій	1	2	3	4	5	Вектор пріоритетів
1	Достовірність	1	2	3	2	4	0.34
2	Точність	1/2	1	2	2	3	0.24
3	Швидкодія	1/3	1/2	1	1	2	0.16
4	Цілісність	1/2	1/2	1	1	2	0.15
5	Конфіденційність	1/4	1/3	1/2	1/2	1	0.11

Результати розрахунку векторів пріоритетів показують, що найбільшу значущість для розробки даної інформаційної системи мають критерії достовірність (0.34) та точність (0.24). Це зумовлено високими вимогами до коректності розпізнавання об’єктів у режимі реального часу під час роботи з відеопотоком з БПЛА. Критерії швидкодія (0.16), цілісність (0.15) та конфіденційність (0.11) мають нижчу вагу, однак залишаються важливими для забезпечення стабільності функціонування системи, збереження структури даних та захисту результатів.

На наступному етапі буде здійснено попарне порівняння альтернативних варіантів реалізації інформаційної системи відповідно до кожного з визначених критеріїв. Для цього буде сформовано окремі матриці оцінювання, що дозволить кількісно виразити перевагу тієї чи іншої архітектури системи у контексті кожного критерію. За результатами цих порівнянь буде обчислено інтегральні ваги альтернатив, на основі яких здійснюється остаточний вибір найдоцільнішого варіанта

реалізації, що найбільш повно відповідає цілі створення системи ідентифікації об’єктів з БПЛА.

У таблиці 2.2 представлено порівняльну оцінку альтернатив за критерієм достовірності, що відображає здатність системи точно відображати реальний стан об’єктів. Найвищу пріоритетність отримала інформаційна система, яка завдяки використанню методів штучного інтелекту забезпечує найвищий рівень достовірності в умовах динамічного середовища.

Таблиця 2.2

Матриця попарного порівняння критерію «Достовірність»

№	Альтернатива	1	2	3	4	Вектор пріоритетів
1	Інформаційно-аналітична система	1	1/2	1/3	1/4	0.10
2	Інформаційно-керуюча система	2	1	1/2	1/3	0.18
3	Система підтримки прийняття рішень	3	2	1	1/2	0.29
4	Інформаційна система	4	3	2	1	0.43

У таблиці 2.3 здійснено порівняння альтернатив за критерієм точності, що характеризує здатність системи мінімізувати хибні позитивні та хибні негативні спрацювання. Інформаційна система продемонструвала найкращий результат завдяки самонавчанню та гнучкому реагуванню на зміну вхідних даних.

Таблиця 2.3

Матриця попарного порівняння критерію «Точність»

№	Альтернатива	1	2	3	4	Вектор пріоритетів
1	Інформаційно-аналітична система	1	1/2	1/3	1/4	0.09
2	Інформаційно-керуюча система	2	1	1/2	1/3	0.17
3	Система підтримки прийняття рішень	3	2	1	1/2	0.28
4	Інформаційна система	4	3	2	1	0.46

У таблиці 2.4 представлено порівняльну оцінку за критерієм швидкодії. Цей показник є критичним для систем реального часу. Інформаційно-керуюча система отримала найвищу вагу завдяки здатності забезпечувати оперативну обробку інформації та швидке реагування.

Таблиця 2.4

Матриця попарного порівняння критерію «Швидкодія»

№	Альтернатива	1	2	3	4	Вектор пріоритетів
1	Інформаційно-аналітична система	1	1/3	1/2	1/3	0.12
2	Інформаційно-керуюча система	3	1	2	1	0.42
3	Система підтримки прийняття рішень	2	1/2	1	1/2	0.25
4	Інформаційна система	4	1	2	1	0.21

Таблиця 2.5 демонструє оцінку альтернатив за критерієм цілісності, що враховує узгодженість компонентів системи. Система підтримки прийняття рішень і інформаційна система показали найкращі результати завдяки своїй архітектурній гнучкості та здатності до інтеграції.

Таблиця 2.5

Матриця попарного порівняння критерію «Цілісність»

№	Альтернатива	1	2	3	4	Вектор пріоритетів
1	Інформаційно-аналітична система	1	2	1/2	1/2	0.22
2	Інформаційно-керуюча система	1/2	1	1/2	1/3	0.15
3	Система підтримки прийняття рішень	2	2	1	1	0.32
4	Інформаційна система	2	3	1	1	0.31

У таблиці 2.6 представлено порівняльну оцінку за критерієм конфіденційності. Найвищу оцінку отримала інформаційна система, яка завдяки сучасним методам шифрування та адаптивному доступу забезпечує високий рівень захисту інформації.

Таблиця 2.6

Матриця попарного порівняння критерію «Конфіденційність»

№	Альтернатива	1	2	3	4	Вектор пріоритетів
1	Інформаційно-аналітична система	1	2	1/2	1/3	0.20
2	Інформаційно-керуюча система	1/2	1	1/2	1/3	0.15

Продовження табл. 2.6

3	Система підтримки прийняття рішень	2	2	1	1/2	0.28
4	Інформаційна система	3	3	2	1	0.37

На основі побудованих матриць попарного порівняння альтернатив за кожним із п’яти ключових критеріїв — достовірність, точність, швидкодія, цілісність та конфіденційність (таблиці 2.2–2.6) — було визначено локальні вектори пріоритетів. Ці вектори відображають відносну перевагу кожної з чотирьох альтернативних архітектур інформаційної системи (інформаційно-аналітична, інформаційно-керуюча, система підтримки прийняття рішень, класична інформаційна система) у межах кожного окремого критерію. Визначені локальні пріоритети є результатом нормалізації значень у відповідних матрицях, що забезпечує єдиний масштаб для подальшого узагальнення оцінок.

Наступним етапом застосування методу аналітичної ієрархії (МАІ) став процес ієрархічного синтезу — обчислення інтегральної ваги для кожної альтернативи. Для цього вектори локальних пріоритетів по кожному критерію було зважено на відповідні вагові коефіцієнти самих критеріїв, попередньо отримані внаслідок побудови та узагальнення матриці попарних порівнянь критеріїв першого рівня ієрархії (таблиця 2.1). Іншими словами, кожен локальний пріоритет альтернативи множився на глобальну вагу критерію, а результати підсумовувалися для отримання загального оцінного значення — інтегрального пріоритету альтернативи. Таким чином реалізовано поетапний перехід від часткових оцінок до загального рішення.

Інтегральні оцінки всіх чотирьох альтернативних типів інформаційних систем представлено в таблиці 2.7. Зведені результати ієрархічного синтезу дозволяють візуально й кількісно порівняти ступінь відповідності кожної альтернативи загальній меті проєкту. Обрана альтернатива, що отримала максимальне значення інтегральної оцінки, визнається найбільш придатною для реалізації поставленої задачі —

створення ефективної інформаційної системи для ідентифікації об’єктів на основі даних з безпілотних літальних апаратів. Зроблений вибір має обґрунтовану математичну базу, яка враховує багатокритеріальний підхід до прийняття рішень та складність предметної області.

Таблиці 2.7

Ієрархічний синтез критеріїв та типів систем

Критерій / Система	Достовірність	Точність	Швидкодія	Цілісність	Конфіденційність	Підсумок
Інформаційно-аналітична система	0.034	0.0216	0.0192	0.033	0.022	0.1298
Інформаційно-керуюча система	0.0612	0.0408	0.0672	0.0225	0.0165	0.2082
Система підтримки прийняття рішень	0.0986	0.0672	0.04	0.048	0.0308	0.2846
Інформаційна система	0.1462	0.1104	0.0336	0.0465	0.0407	**0.3774**

У результаті проведених розрахунків було визначено загальну ефективність кожного з варіантів реалізації інформаційної системи з урахуванням усіх п’яти критеріїв. На основі ієрархічного синтезу стало зрозуміло, що найвищу інтегральну оцінку отримала інформаційна система, що свідчить про її найбільшу відповідність поставленим вимогам. Зокрема, вона найкраще задовольняє умови щодо достовірності, точності, швидкодії та конфіденційності, що є важливими для задач ідентифікації об’єктів з БПЛА.

З огляду на проведений аналіз, було зроблено висновок, що найбільш доцільним варіантом для реалізації розроблюваної системи є саме інформаційна система, яка забезпечує баланс між точністю, швидкістю роботи та надійністю функціонування в умовах реального середовища.

2.2. Конкретизація функціонування системи

У процесі системного аналізу важливо не лише визначити цілі та вимоги до інформаційної системи, а й сформувати структуроване уявлення про її функціонування. Для цього застосовуються методи функціонального моделювання, зокрема один із найпоширеніших – метод IDEF0 (Integration Definition for Function Modeling). Цей підхід дозволяє формально описати функції системи, виявити зв’язки між ними, а також встановити взаємодію з зовнішнім середовищем.

Нотація IDEF0 представляє систему у вигляді набору функціональних блоків, кожен з яких виконує певну операцію. Вхідна, вихідна, керуюча та забезпечувальна інформація позначається стрілками, що входять у блок з різних сторін: зліва – вхідні дані, справа – результати, зверху – керуючі впливи, знизу – механізми реалізації. Така структура дозволяє зрозуміти логіку роботи системи на всіх рівнях її деталізації[16].

Побудова моделі IDEF0 починається з контекстної діаграми (A-0), яка відображає загальну функцію системи на найвищому рівні абстракції. Вона включає лише один блок, що описує основне призначення системи, та показує її взаємодію з зовнішнім середовищем через відповідні інформаційні потоки. Контекстна діаграма є основою для подальшої декомпозиції системи на підфункції, що реалізуються в межах діаграми A0.

У межах бакалаврської кваліфікаційної роботи було побудовано контекстну діаграму інформаційної системи ідентифікації об’єктів з безпілотних літальних апаратів (рис. 2.3). Головною функцією системи є здійснення ідентифікації об’єктів у режимі реального часу на основі відеопотоку з БПЛА.

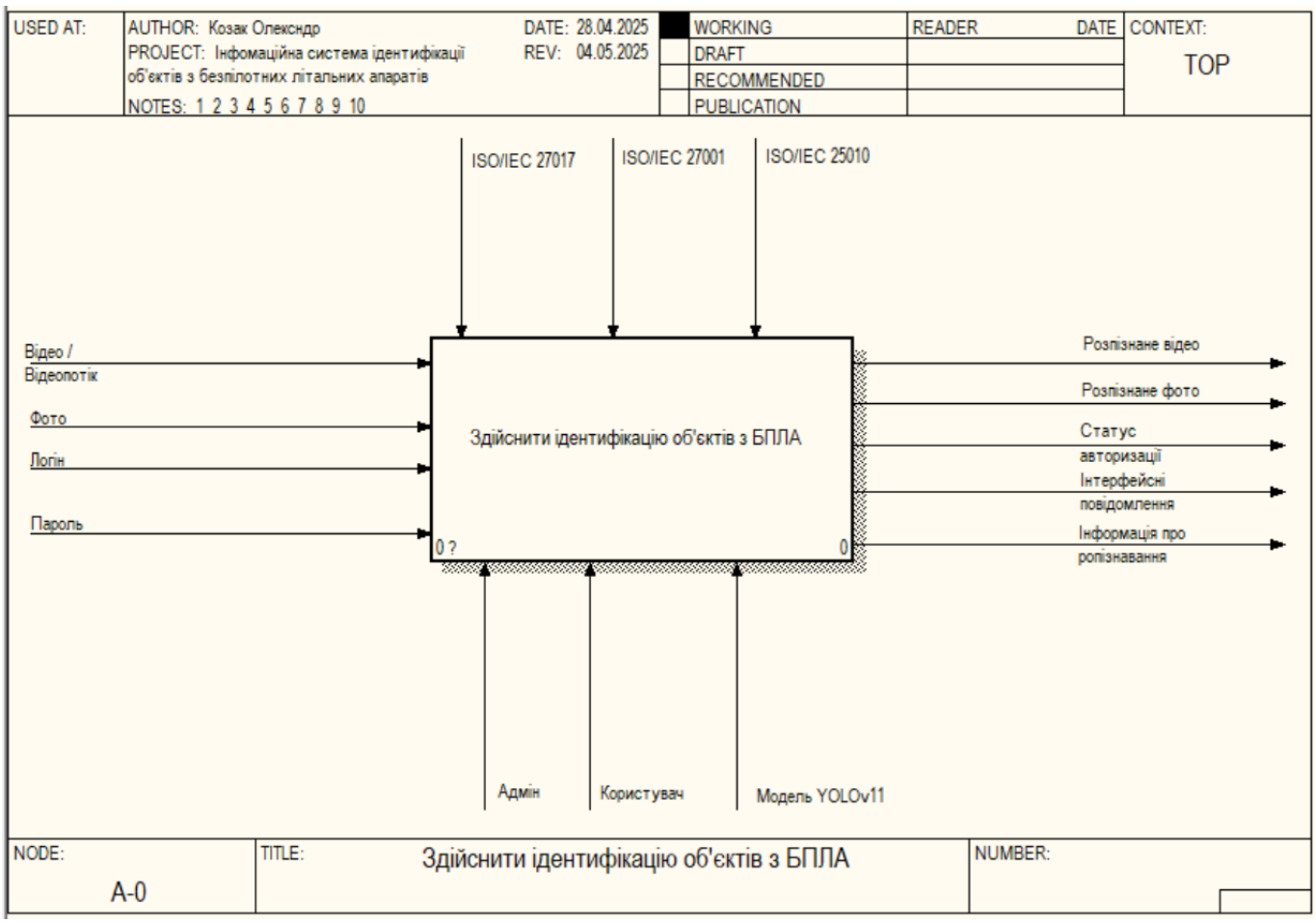

Рис. 2.3 Контекстна діаграма

Основна функція інформаційної системи – «Здійснити ідентифікацію з БПЛА» – представлена у вигляді центрального блоку контекстної діаграми IDEF0. Цей процес охоплює ключові етапи взаємодії користувача з системою, зокрема автентифікацію, передачу зображень або відео, обробку отриманих даних і видачу результатів розпізнавання. Центральна функція є ядром системи і служить об’єднуючим елементом між усіма вхідними, керуючими, вихідними та ресурсними компонентами.

Зліва від основної функції розташовано вхідні дані, які активують процес роботи системи. До них належать логін і пароль користувача, що використовуються для автентифікації, а також зображення, відео або відеопотік, що надходять від безпілотного літального апарата. Ці дані є основою для розпізнавання та обробки, забезпечуючи запуск алгоритмів ідентифікації об’єктів на отриманих носіях.

У верхній частині діаграми подано керуючі впливи, які задають вимоги до роботи системи відповідно до міжнародних стандартів. Зокрема, стандарт ISO/IEC 27017

регламентує безпеку у хмарному середовищі[17], ISO/IEC 27001 встановлює загальні вимоги до систем управління інформаційною безпекою, а ISO/IEC 25010 визначає критерії якості програмного забезпечення, включаючи функціональність, надійність, зручність використання, сумісність тощо[18]. Сукупно ці нормативні документи задають рамки розробки й експлуатації системи, визначаючи необхідний рівень захищеності, відповідності стандартам і якісної реалізації.

Праворуч від основної функції зображено вихідні дані – результати, що генерує система у відповідь на вхідну інформацію. До таких результатів належать розпізнані фото або відео, які містять візуально ідентифіковані об’єкти; статус авторизації, який свідчить про успішну або неуспішну автентифікацію користувача; інтерфейсні повідомлення, що забезпечують зворотний зв’язок системи з користувачем; а також структурована інформація про розпізнані об’єкти, включаючи тип, кількість та просторове розташування. Вихідні дані є цінним джерелом інформації для подальшого аналізу, зберігання або ухвалення рішень.

У нижній частині діаграми позначено механізми – ресурси, які забезпечують реалізацію основного процесу. Серед них – користувач, який ініціює процес ідентифікації, вводить облікові дані й передає медіафайли до обробки; адміністратор, що відповідає за налаштування, підтримку й контроль коректної роботи системи; а також модель глибокого навчання YOLOv11, яка виконує безпосередню ідентифікацію об’єктів на фото, відео або у відеопотоці з високою точністю та швидкістю. Взаємодія між цими елементами забезпечує стабільне й ефективне функціонування системи.

Наступним кроком буде розроблено декомпозицію для контекстної діаграми «Здійснити ідентифікацію об’єктів з БПЛА», яка деталізує основну функцію системи на чотири ключові підпроцеси. Цей розподіл дозволяє точніше зрозуміти функціональну структуру системи, встановити логіку взаємодії між її компонентами та визначити зв’язки між даними, стандартами і ресурсами. Декомпозиція представлена у вигляді діаграми IDEF0 на рисунку 2.4.

Першим функціональним блоком є «Виконати автентифікацію». На вхід надходять облікові дані користувача – логін і пароль, що вводяться через інтерфейс системи. Управління цим процесом забезпечується нормативом ISO/IEC 27001[19], який визначає вимоги до захищеної обробки інформації, а також ISO/IEC 25010, що регламентує якість програмного забезпечення. На виході формується статус авторизації, який передається до наступних блоків і визначає, чи має користувач дозвіл на подальше використання системи. Ресурсом для виконання цієї функції є сам користувач, який ініціює авторизацію, а також адміністратор, що забезпечує налаштування системи доступу.

Другим функціональним блоком є «Перевірити статус авторизації», який виконується після автентифікації користувача для контролю прав доступу до функціоналу системи. Блок приймає статус авторизації як вхідні дані з попереднього процесу та, згідно з вимогами стандарту ISO/IEC 27017, забезпечує перевірку дозволів на подальші дії. У випадку підтвердження прав доступу користувач може передати дані до одного з наступних блоків – «Обробити та розпізнати фото» або «Обробити та розпізнати відео / відеопотік». Якщо ж прав доступу немає користувач отримує інтерфейсе повідомлення про заборону у доступі.

Третій блок – «Обробити та розпізнати фото» активується у разі, якщо користувач завантажує окреме зображення для аналізу. Вхідними даними є фото та статус авторизації. Управляючими елементами для цього процесу виступають ті самі стандарти ISO/IEC 27017 та ISO/IEC 25010 котрий забезпечує вимоги до якості програмного забезпечення, включаючи точність і функціональність розпізнавання. На вихід подається розпізнаний об'єкт та його точність, що будуть використані в наступному блоці формування результатів. Механізмом реалізації цієї функції є модель глибокого навчання YOLOv11, яка виконує розпізнавання зображень та користувач, який виступає джерелом даних.

Четвертий функціональний блок – «Обробити та розпізнати відео/відеопотік» відповідає за ідентифікацію об’єктів на динамічних відеоданих. На вхід надходить відео або відеопотік, а також статус авторизації користувача. Управління реалізується

тими ж міжнародними стандартами, що й попередні блоки. Вихідними даними є розпізнані об’єкти та відповідна оцінка їх точності. Як і в попередньому випадку, для реалізації використовуються модель YOLOv11, а також участь користувача як джерела даних.

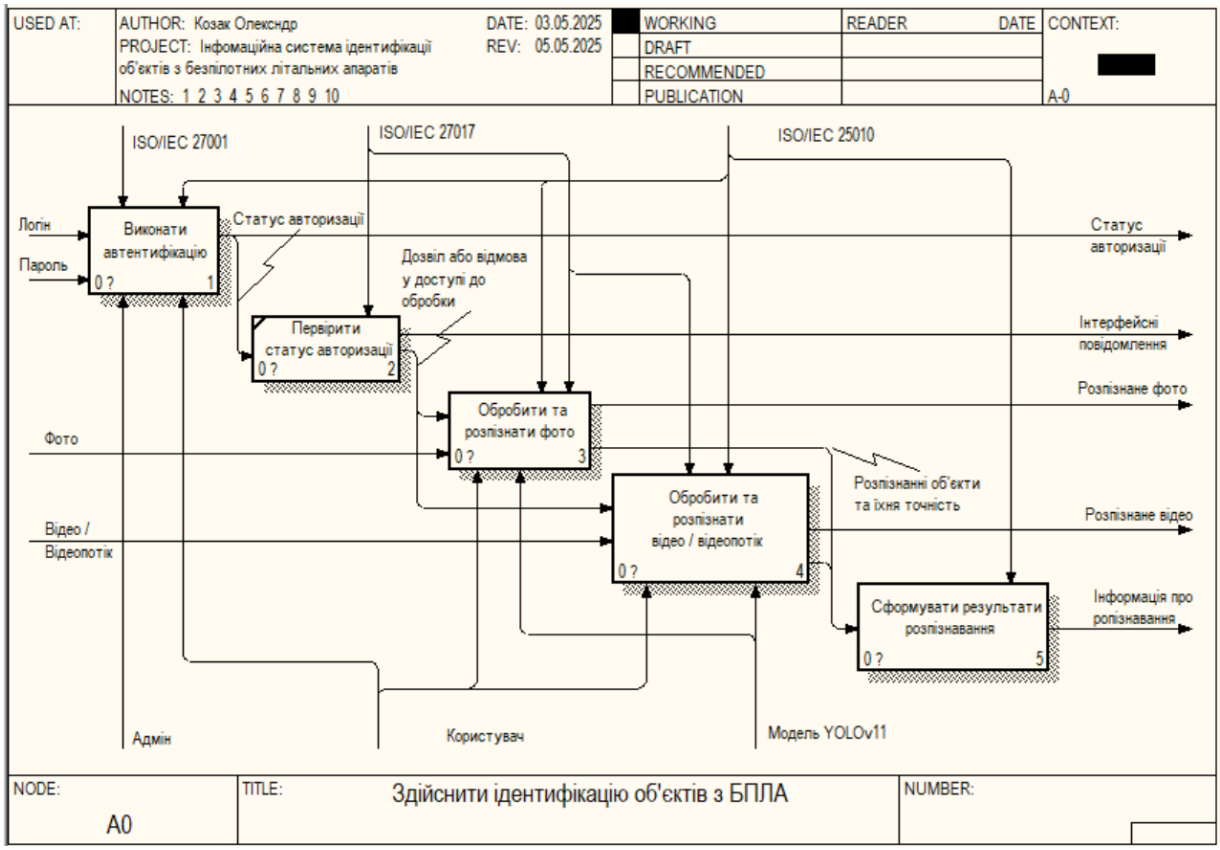

Рис. 2.4 Діаграма декомпозиції першого рівня

Заключний блок – «Сформувати результати розпізнавання» – узагальнює дані з попередніх двох процесів і забезпечує вивід результатів для користувача. Вхідними елементами є розпізнані об’єкти з фото або відео, а також статус авторизації. Управляючими стандартами виступає ISO/IEC 25010, який гарантує якість, структурованість і доступність інформації. На вихід система формує такі результати: розпізнані фото та відео, інформацію про розпізнані об’єкти, інтерфейсні повідомлення, а також фінальний статус авторизації. Цей етап завершує логіку ідентифікації об’єктів з БПЛА, надаючи користувачу всю необхідну інформацію у зручному форматі.

Таким чином, діаграма на рисунку 2.4 демонструє повний цикл функціонування інформаційної системи – від автентифікації до отримання результатів. Кожен блок чітко взаємодіє з іншими через потоки даних, керуючих елементів і ресурсів, що створює надійну, логічну та стандартизовану архітектуру рішення.

Наступним кроком стала розробка декомпозиції функціонального блоку «Виконати автентифікацію», що входить до складу контекстної функції «Здійснити ідентифікацію об’єктів з БПЛА». Вказана декомпозиція, представлена на рисунку 2.5, дає змогу деталізувати логіку обробки облікових даних та перевірки доступу до інформаційної системи. Такий підхід забезпечує структуроване представлення процесів і дозволяє сформулювати вимоги до реалізації кожного з них відповідно до стандартів безпеки та якості.

Функція аутентифікації включає три послідовних підпроцеси. Першим є приймання облікових даних користувача, на якому система отримує логін і пароль, введені через інтерфейс. Ці дані надходять на вхід як активуючий сигнал. Ініціатором процесу виступає користувач, а якість цього етапу регулюється відповідно до стандарту ISO/IEC 25010. Результатом є підготовка даних до перевірки.

Другий підпроцес полягає у перевірці правильності облікових даних. Внутрішній механізм системи співставляє введену інформацію з даними, збереженими в базі. Тут особливого значення набуває безпечність і достовірність операцій, що забезпечуються за допомогою керуючих впливів у вигляді стандартів ISO/IEC 27001 і ISO/IEC 25010. Адміністратор, як технічний механізм, відповідає за підтримку бази користувачів і контроль за правильністю функціонування системи.

Завершальним етапом є формування повідомлення про результат авторизації. На цьому етапі система генерує вихідну інформацію у вигляді статусу авторизації – позитивного або негативного – та передає її користувачеві. Цей підпроцес є важливим для забезпечення зворотного зв’язку і також реалізується відповідно до вимог безпеки та якості. Таким чином, розроблена декомпозиція блоку «Аутентифікація користувача» відображає логічну послідовність дій, які дозволяють здійснити контрольований доступ до системи.

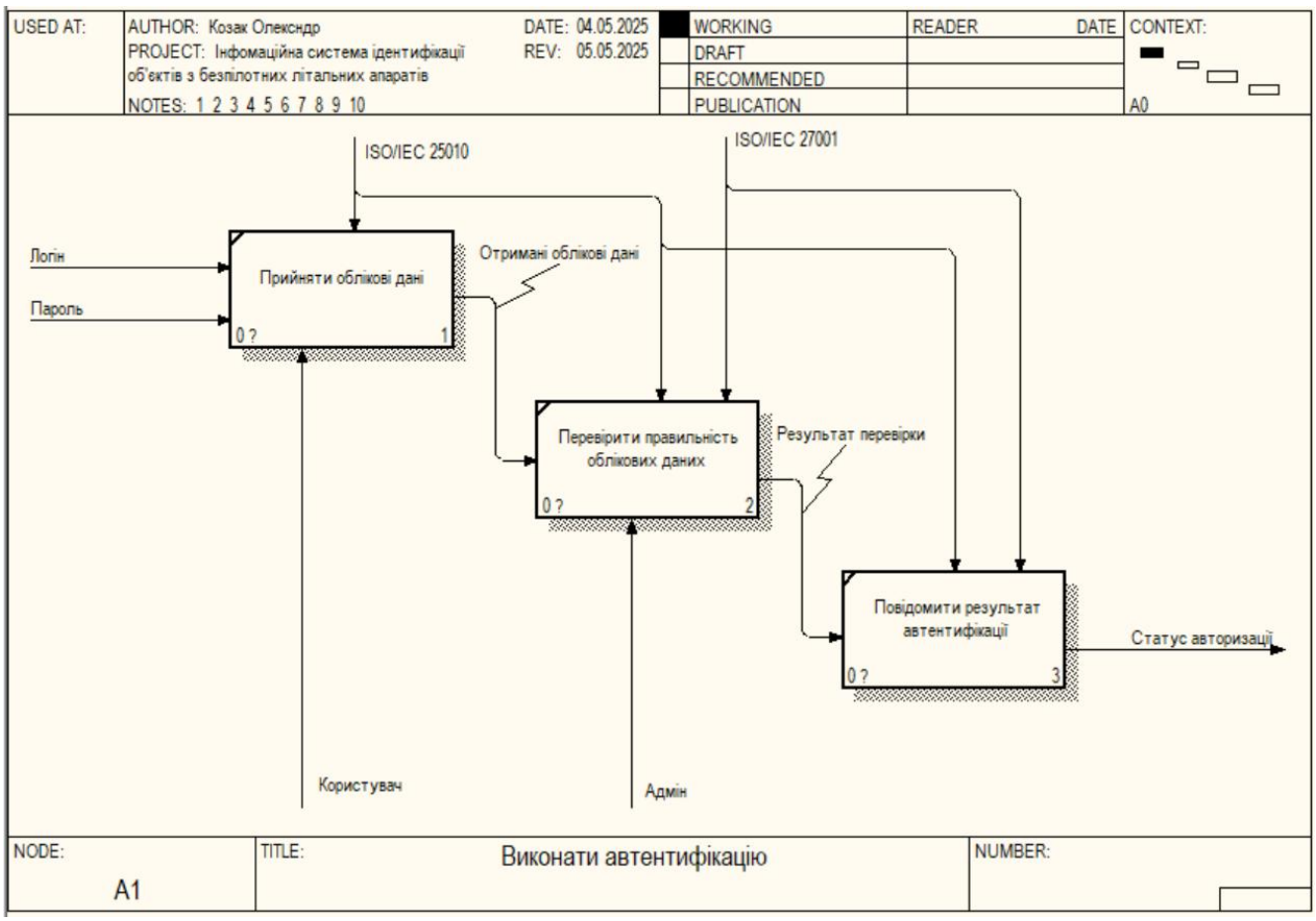

Рис. 2.5 Діаграма декомпозиції другого рівня процесу «Виконати автентифікацію»

На наступному етапі структурного проєктування інформаційної системи було виконано декомпозицію функціонального блоку другого рівня «Обробити та розпізнати фото», що входить до складу функціональної підсистеми «Ідентифікувати об'єкти з БПЛА». Вказана діаграма, подана на рисунку 2.6, ілюструє поетапний процес обробки зображення, яке надходить від користувача для аналізу, та відображає логічну послідовність перетворень з метою отримання розпізнаного результату. Такий підхід забезпечує чітку структуризацію процедур, дозволяє формалізувати вимоги до кожного з підпроцесів та забезпечує відповідність розроблюваної системи нормативним вимогам ISO/IEC 27017 (інформаційна безпека в хмарних сервісах) та ISO/IEC 25010 (якість програмного забезпечення).

Структура функціонального блоку складається з чотирьох підпроцесів, які виконуються послідовно. Першим кроком є реалізація процесу «Отримати фото». На цьому етапі система приймає вхідне зображення, що було надане користувачем. У ролі

механізму на цьому етапі виступає сам користувач, який ініціює обробку, а також нормативно визначені політики доступу.

Другим підпроцесом є «Виконати передобробку зображення», який передбачає застосування алгоритмів корекції зображення, зокрема зміну розміру, нормалізацію яскравості, фільтрацію шумів та інші перетворення, що підвищують якість вхідних даних для подальшого аналізу. Даний етап має вирішальне значення для забезпечення точності розпізнавання, оскільки формує технічно придатне до аналізу зображення. Управляючим впливом цього процесу є вимоги до якості програмного забезпечення, сформульовані в ISO/IEC 25010.

Третій підпроцес – «Розпізнати об’єкти» – реалізує безпосереднє виявлення цільових об'єктів на зображенні за допомогою моделі глибокого навчання YOLOv11, яка забезпечує детекцію в реальному часі з високим рівнем точності. На цьому етапі система визначає положення, клас і ймовірність наявності об’єктів у межах кадру. На цьому етапі система приймає дозвіл на його розпізавання відповідно до статусу авторизації. Доступ до подальших обчислень дозволяється лише при підтвердженні автентичності користувача. Вихідними результатами є структуровані дані щодо розпізнаних об’єктів, які передаються до наступного підпроцесу.

Завершальним кроком є «Згенерувати розпізнане фото». Цей процес забезпечує візуалізацію результатів розпізнавання у вигляді графічного накладення меж виявлених об’єктів та відповідних міток (ідентифікаторів і точності класифікації) безпосередньо на зображення. Такий підхід підвищує інформативність результату для кінцевого користувача, забезпечуючи його зручне сприйняття та можливість подальшого аналізу.

Таким чином, побудована декомпозиція функціонального блоку «Обробити та розпізнати фото» забезпечує детальний опис послідовних процедур обробки вхідного зображення, формує основу для реалізації функціональності розпізнавання в рамках інформаційної системи ідентифікації об’єктів з БПЛА та гарантує її відповідність сучасним міжнародним стандартам.

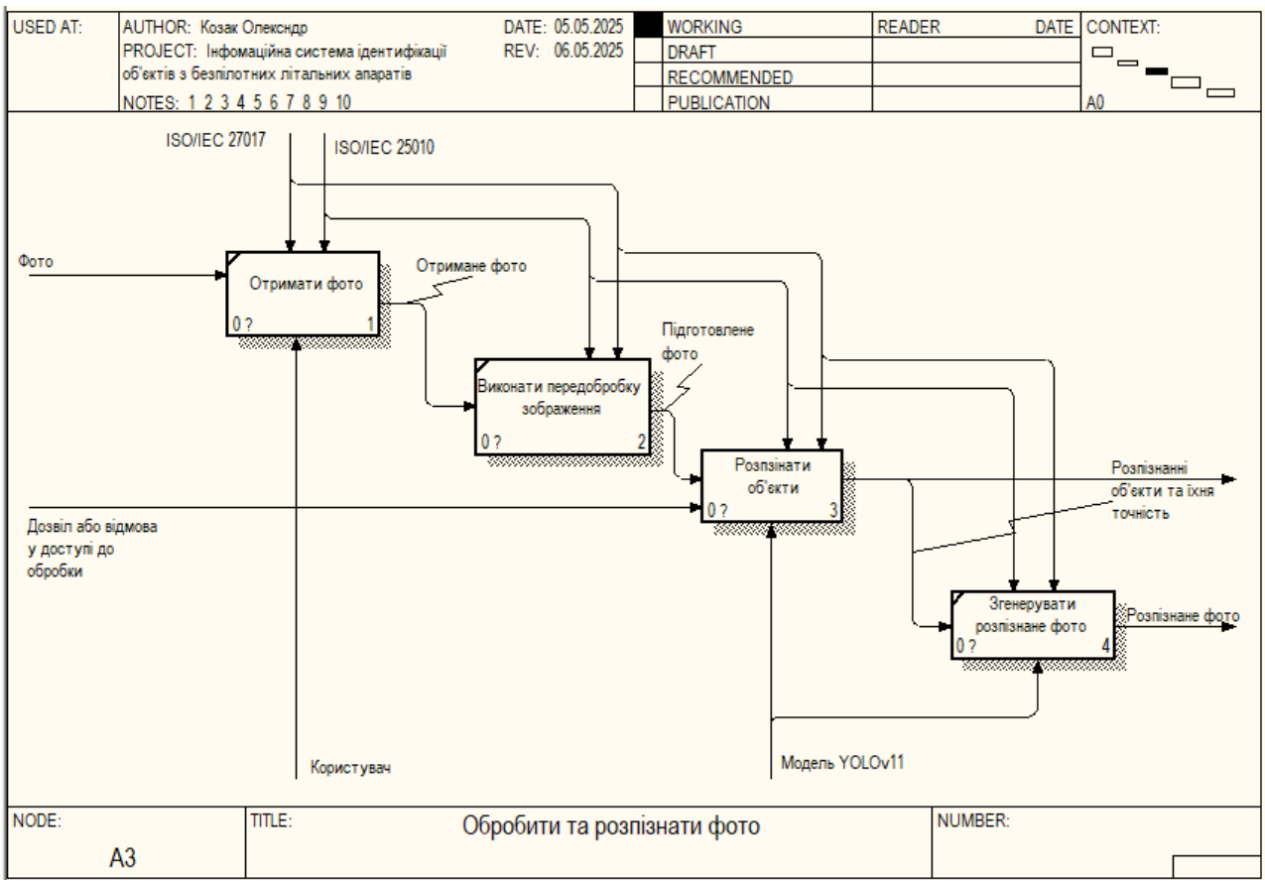

Рис. 2.6 Діаграма декомпозиції другого рівня процесу «Обробити та розпізнати фото»

У межах етапу структурного проєктування інформаційної системи було виконано декомпозицію функціонального блоку другого рівня «Обробити та розпізнати відео / відеопотік», який є складовою підсистеми «Ідентифікувати об’єкти з БПЛА». Відповідна діаграма, представлена на рисунку 2.7, демонструє покроковий процес обробки відеоданих, що надходять від користувача або зовнішнього джерела, з метою виявлення та візуалізації об'єктів у межах кожного кадру. Такий підхід дозволяє впорядкувати обробку мультимедійного потоку, забезпечити наочність результатів та дотримання вимог міжнародних стандартів, зокрема ISO/IEC 27017 (інформаційна безпека у хмарному середовищі) та ISO/IEC 25010 (якість програмних продуктів).

Функціональний блок складається з послідовності шести підпроцесів, що забезпечують повний цикл аналізу відео. Першим є підпроцес «Ініціалізувати відео / відеопотік», у межах якого система підключається до джерела відео та виконує перевірку прав доступу користувача. Успішна ініціалізація дозволяє перейти до

подальшої обробки потоку. Керуючими впливами на цьому етапі виступають політики автентифікації та безпеки, що регламентуються стандартом ISO/IEC 27017.

Наступним кроком є процес «Виділити кадри з відео», який передбачає розбиття потоку на окремі зображення відповідно до заданої частоти або змін у сцені. Це дозволяє зменшити обчислювальне навантаження та сфокусувати аналіз лише на релевантних кадрах.

Третій підпроцес – «Виконати попередню обробку кадрів» – забезпечує покращення якості відеокадрів шляхом застосування операцій масштабування, фільтрації, нормалізації тощо. Така обробка є критично важливою для забезпечення високої точності подальшого розпізнавання.

На наступному етапі – «Виконати розпізнавання об’єктів» – здійснюється ідентифікація об’єктів за допомогою глибокої нейронної мережі YOLOv11. Система виявляє об’єкти на кожному кадрі, визначає їхнє розташування, клас і ступінь ймовірності. Отримані результати структуровано зберігаються для подальшої обробки.

Після розпізнавання виконується підпроцес «Згенерувати розпізнане відео», в межах якого результати детекції інтегруються у відеопотік. Кожен кадр збагачується візуальними мітками – рамками, назвами об’єктів і значеннями ймовірності – що підвищує зручність сприйняття даних кінцевим користувачем.

Таким чином, діаграма декомпозиції функціонального блоку «Обробити та розпізнати відео / відеопотік» відображає логічну й технічну структуру процедур аналізу відео в межах інформаційної системи ідентифікації об’єктів з БПЛА. Це дозволяє не лише деталізувати архітектуру функціоналу, але й гарантувати відповідність розробки сучасним вимогам безпеки, якості та ефективності.

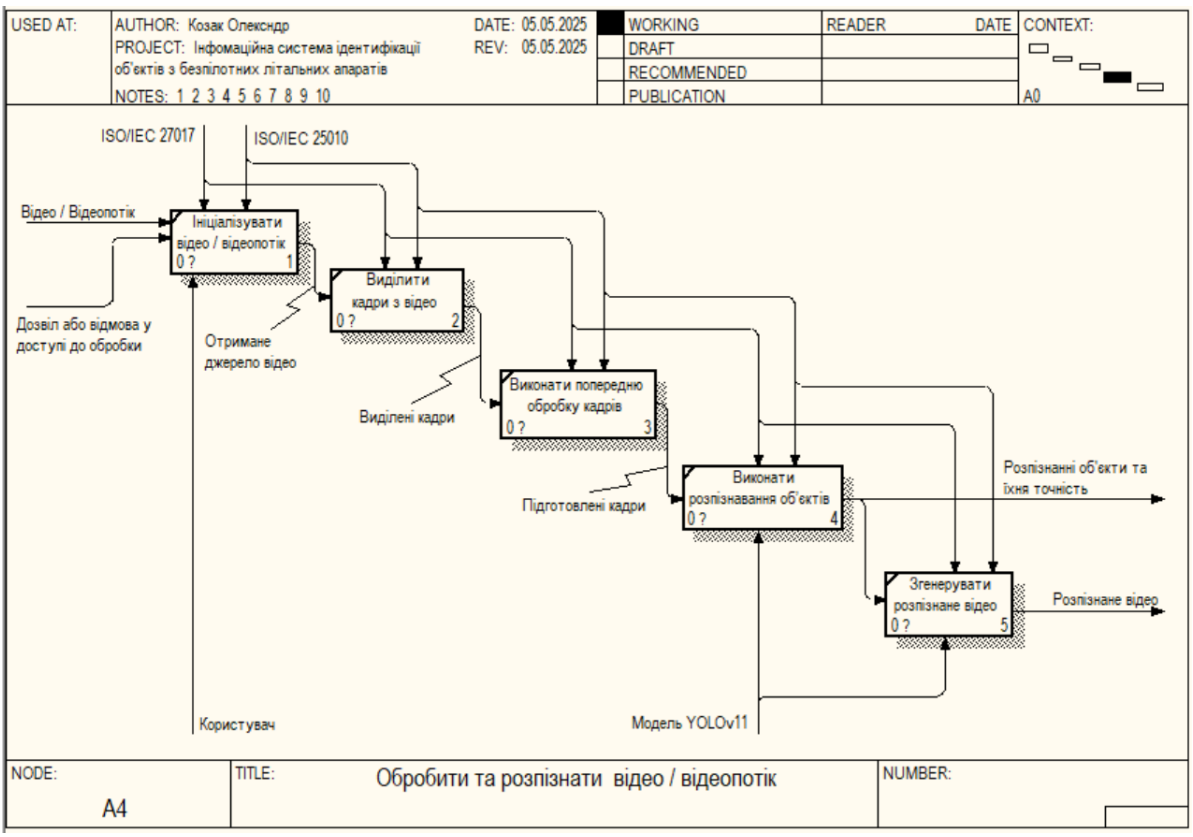

Рис. 2.7 Діаграма декомпозиції другого рівня процесу «Обробити та розпізнати відео / відеопотік»

Наступник кроком виступає реалізація проєктування інформаційної системи здійснено декомпозицію функціонального блоку другого рівня «Сформувати результати розпізнавання», який входить до складу підсистеми «Ідентифікувати об’єкти з БПЛА». Відповідна функціональна діаграма, подана на рисунку 2.8, відображає логічну послідовність обробки результатів, отриманих внаслідок виконання розпізнавання за допомогою нейронної мережі YOLOv11. Основною метою даного блоку є підготовка детекційної інформації до подальшого виведення у призначеному для користувача інтерфейсі з дотриманням вимог стандарту ISO/IEC 25010 щодо якості програмного забезпечення.

Функціональний блок складається з трьох послідовних підпроцесів. Перший підпроцес – «Зібрати розпізнані дані» – передбачає агрегування об’єктів, ідентифікованих під час обробки вхідних мультимедійних даних (зображення, відеофайл або відеопотік). У межах цього підпроцесу формується набір первинних

детекційних характеристик, зокрема координати розташування об’єктів, їхні класи та рівень достовірності розпізнавання.

Другий підпроцес – «Структурувати результати для відображення» – виконує логічну організацію зібраної інформації відповідно до вимог подальшої візуалізації. Зокрема, здійснюється формування структурованих масивів даних, які включають класифіковані об’єкти, значення ймовірностей, часові позначки, що забезпечує ефективну адаптацію результатів для представлення в графічному чи табличному вигляді в інтерфейсі користувача.

Завершальним етапом є підпроцес «Вивести результати користувачу», в межах якого сформовані результати передаються на рівень презентації у вигляді інтерфейсних повідомлень, а також візуалізацій із нанесеними графічними позначками. Результатом цього підпроцесу є подання користувачу повної, візуально зрозумілої інформації про виявлені об’єкти, їх кількість, типи та рівень точності розпізнавання.

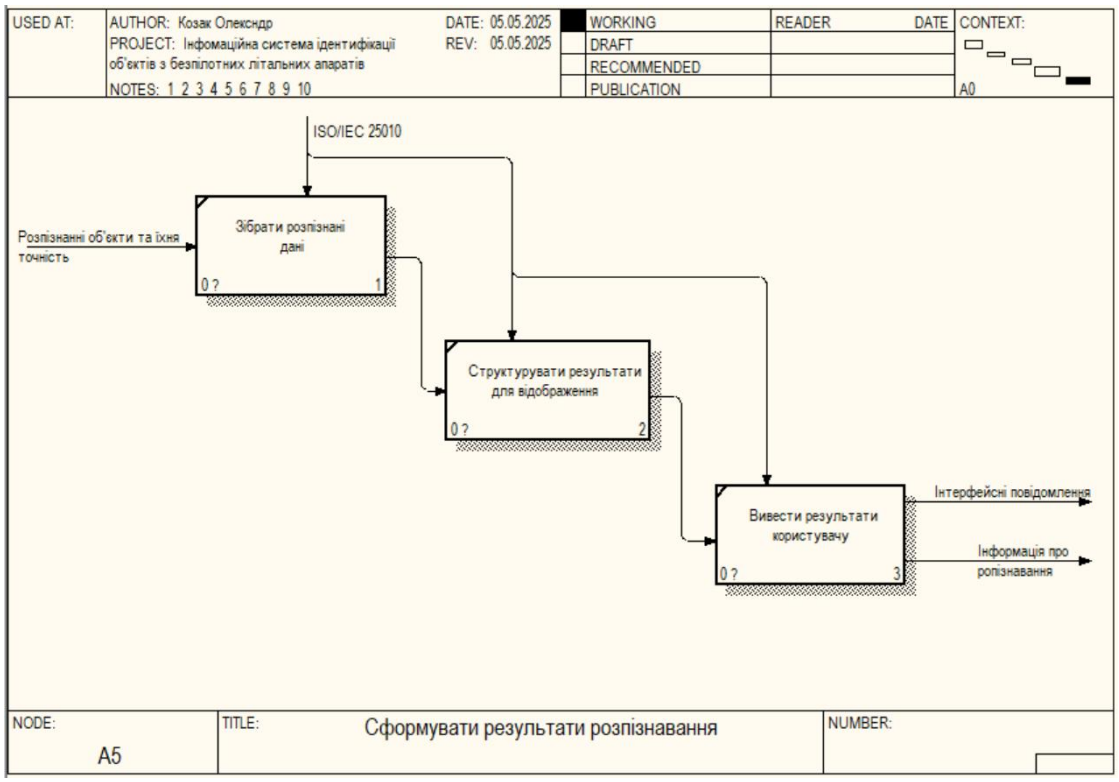

Рис. 2.8 Діаграма декомпозиції другого рівня процесу «Сформувати результати розпізнавання»

Таким чином, функціональна декомпозиція блоку «Сформувати результати розпізнавання» забезпечує чітке представлення логіки обробки результатів розпізнавання, гарантує їх відповідність вимогам якості та орієнтованість на користувача, що є критично важливим для побудови ефективної інформаційної системи ідентифікації об’єктів з безпілотних літальних апаратів.

Останнім кроком у межах етапу проєктування інформаційної системи виконано декомпозицію функціонального блоку другого рівня «Розпізнати об’єкти», що є складовою підсистеми «Ідентифікувати об’єкти з БПЛА». Відповідна функціональна діаграма, наведена на рисунку 2.9, відображає логічну послідовність процесів комп’ютерного розпізнавання об’єктів на зображеннях за допомогою моделі YOLOv11. Основна мета цього функціонального блоку полягає в забезпеченні автоматизованої ідентифікації об’єктів та підготовці достовірних даних для подальшого аналізу і візуалізації, із урахуванням вимог стандартів ISO/IEC 25010 (якість програмного забезпечення) та ISO/IEC 27017 (інформаційна безпека).

Функціональний блок складається з трьох послідовних підпроцесів. Перший підпроцес – «Ініціалізувати модель YOLOv11» – виконує підключення та конфігурацію попередньо навченої моделі розпізнавання, перевіряє її працездатність, відповідність технічним параметрам та перевіряє наявність доступу на обробку зображення. Також забезпечує готовність до обробки зображення відповідно до стандарту ISO/IEC 25010.

Другим етапом є підпроцес «Застосувати модель для розпізнавання», у межах якого здійснюється безпосередня обробка підготовленого зображення з використанням нейронної мережі. Система визначає місцеположення, клас та ймовірність кожного об’єкта, і формує набір попередніх результатів детекції.

Завершальним підпроцесом виступає «Відфільтрувати та структурувати результати», де здійснюється видалення нерелевантних або надто неточних розпізнань, а також логічна організація даних для подальшої передачі у вигляді розпізнаних об’єктів з відповідними параметрами точності. Також в цьому

функціональному блоці забезпечується інформаційна безпека відповідно до стандарту ISO/IEC 27017.

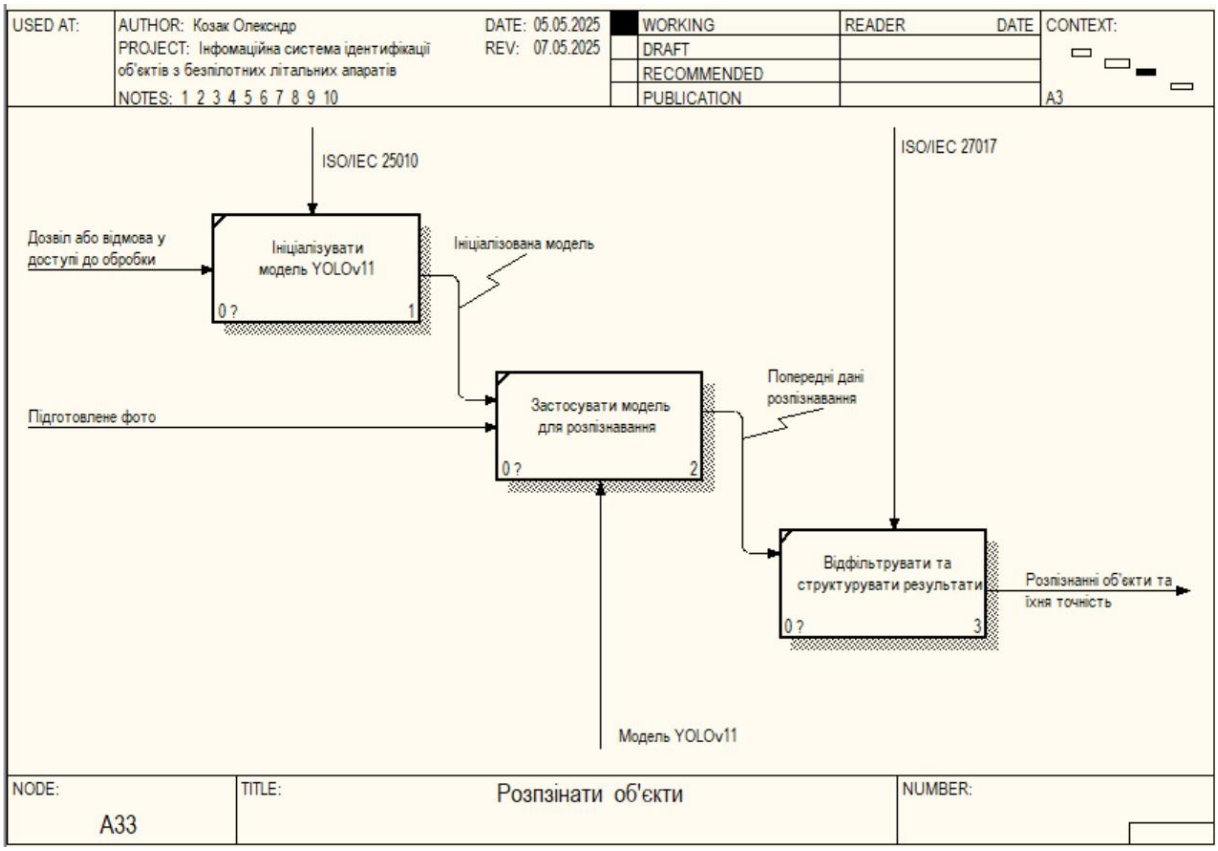

Рис. 2.9 Діаграма декомпозиції другого рівня процесу «Розпізнати об’єкти»

Таким чином, функціональна декомпозиція блоку «Розпізнати об’єкти» надає цілісне уявлення про механізми глибинного розпізнавання зображень, підвищує прозорість і контрольованість процесу виявлення об’єктів та слугує основою для забезпечення надійності й точності в межах розроблюваної інформаційної системи.

2.3. Побудова ієрархії процесів (функцій, задач)

Для побудови моделі функціонування інформаційної системи ідентифікації об’єктів з безпілотних літальних апаратів було сформовано ієрархічну структуру функцій, яка відображає логіку реалізації основних процесів системи. Така структура дозволяє здійснити деталізацію загальної цільової функції до конкретних підзадач, забезпечуючи повне уявлення про функціональні механізми її реалізації.

На верхньому рівні ієрархії (контекстна діаграма А-0) визначено загальну цільову функцію – «Здійснити ідентифікацію об’єктів з БПЛА». Вона декомпонується на шість основних підфункцій першого рівня, які реалізують логіку обробки даних від моменту автентифікації користувача до виведення результатів розпізнавання:

- Виконати автентифікацію – забезпечує перевірку достовірності користувача та надає доступ до функціональних можливостей системи;
- Обробити та розпізнати фото – передбачає завантаження зображень, їхню попередню обробку та проведення розпізнавання за допомогою моделі YOLOv11;
- Обробити та розпізнати відео / відеопотік – включає сегментацію відео, виділення ключових кадрів та їх обробку з подальшим розпізнаванням об’єктів;
- Сформувати результати розпізнавання – відповідає за логічну організацію детекційних результатів, їх структурування та передачу для відображення;
- Перевірити статус авторизації – реалізує механізм перевірки прав доступу до окремих функцій системи на основі автентифікаційних даних;
- Генерувати розпізнане відео – виконує створення відеофайлів з нанесеними графічними позначками виявлених об’єктів на кадрах.

Кожна з основних функцій у свою чергу деталізується на підфункції другого і третього рівнів, що забезпечує всебічне охоплення функціоналу системи. Наприклад, функція «Обробити та розпізнати фото» включає такі підпроцеси:

Отримати фото (вхідні дані);

- Виконати передобробку зображення (зміна розміру, нормалізація);
- Ініціалізувати модель YOLOv11;
- Застосувати модель для розпізнавання;
- Відфільтрувати та структурувати результати.

Такий підхід до проєктування ієрархії функцій дозволяє досягти чіткої структуризації процесів, логічного розмежування обов’язків між компонентами

системи, а також формує підґрунтя для подальшої реалізації масштабованої, надійної та орієнтованої на користувача інформаційної системи.

На рисунку 2.10 представлено побудовану ієрархію функцій у вигляді структурної схеми, реалізованої засобами функціонального моделювання IDEF0.

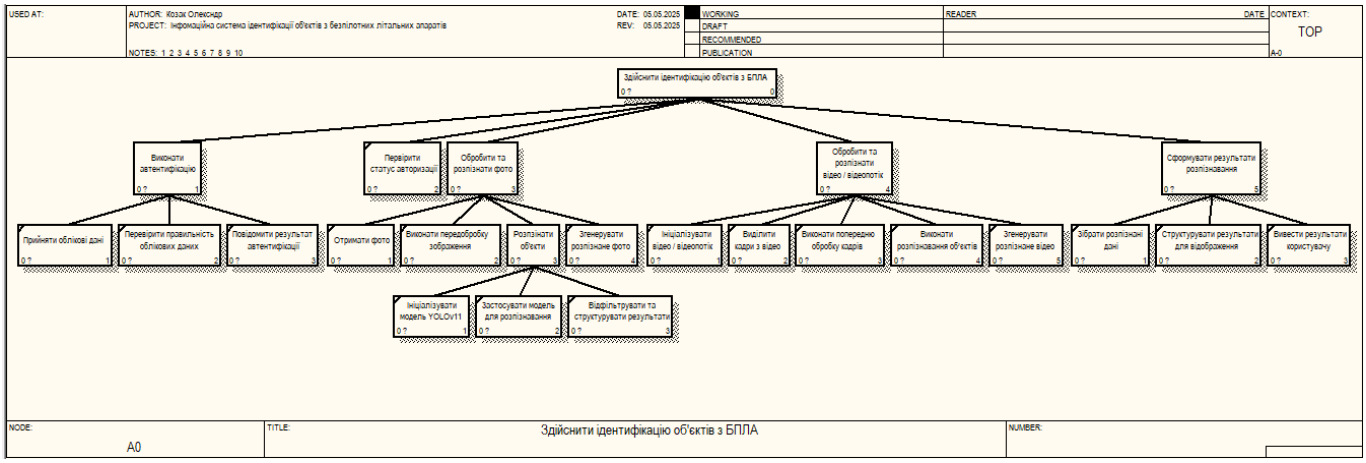

Рис. 2.10 Ієрархія процесів

Висновок до другого розділу

У цьому розділі було проведено системний аналіз досліджуваної теми – «Інформаційна система ідентифікації об’єктів з безпілотних літальних апаратів». Здійснено аналіз ключових понять і принципів, що лежать в основі проєктування подібних інформаційних систем. Особливу увагу приділено застосуванню системного підходу, який включає принципи цілісності, ієрархічності, гнучкості, формалізації та практичної орієнтованості. В межах розділу проаналізовано реалізацію кожного з принципів у контексті розробки даної системи.

Було побудовано дерево цілей, на вершині якого знаходиться основна мета – створення інформаційної системи, що забезпечує автоматизовану ідентифікацію об'єктів на основі відеопотоку з БПЛА. Дерево цілей деталізує основну мету через підцілі, які відображають ключові функціональні напрямки системи: збір даних, обробку відеопотоку, алгоритмічну ідентифікацію об’єктів, взаємодію з користувачем, а також збереження і передавання результатів. Така структура дозволяє

логічно розподілити задачі розробки системи та врахувати її складність на різних рівнях.

З метою формалізації функціональної структури було застосовано метод функціонального моделювання IDEF0. Побудовано контекстну діаграму, яка демонструє загальну функцію системи на найвищому рівні абстракції та її взаємодію з зовнішніми елементами, зокрема користувачем, нормативними вимогами й технічними засобами. Контекстна діаграма відображає вхідні, вихідні, керуючі та забезпечувальні потоки, що дозволяє цілісно представити роботу системи.

На основі контекстної діаграми проведено декомпозицію системи до нижчих рівнів деталізації. Визначено основні функціональні блоки системи: захоплення відеоданих з БПЛА, попередню обробку зображень, аналіз та ідентифікацію об’єктів, збереження результатів та формування звітів. Для кожної функції вказано її входи, виходи, механізми реалізації та керуючі впливи, що забезпечує чітке розуміння логіки функціонування системи на всіх рівнях.

Побудована модель дозволяє узгоджено описати всі взаємозв’язки в межах системи та створити основу для наступного етапу – проєктування архітектури та реалізації програмних компонентів.

РОЗДІЛ 3

Програмні засоби розв’язання задачі

3.1 Вибір та обґрунтування засобів розв’язання задачі

Процес проєктування та реалізації інформаційної системи ідентифікації об’єктів з безпілотних літальних апаратів вимагає ретельного вибору відповідних програмних засобів, які б забезпечували надійність, масштабованість, інтеграцію з технологіями комп’ютерного зору, а також можливість створення вебзастосунку з інтуїтивно зрозумілим інтерфейсом. Враховуючи складність задачі, доцільним є використання сучасних інструментів, які підтримують обробку зображень і відеопотоків, реалізацію алгоритмів глибокого навчання та побудову клієнт-серверної архітектури.

Обґрунтування вибору засобів реалізації здійснюється на основі аналізу їхніх функціональних можливостей, технічних характеристик, сумісності з іншими компонентами системи, а також відповідності сучасним стандартам програмної інженерії. У цьому підрозділі наведено порівняння програмних технологій, які розглядалися на етапі проєктування, та аргументовано обрано ті, що найбільш оптимально підходять для поставлених завдань.

3.1.1. Аналіз мов програмування для реалізації системи

Мова програмування C++ є однією з найпотужніших системних мов, яка поєднує можливості процедурного та об'єктно-орієнтованого програмування. Завдяки високій продуктивності, прямому доступу до пам’яті та контролю за управлінням ресурсами, вона широко використовується в розробці систем реального часу, ігрових рушіїв, вбудованих систем, а також у галузі комп’ютерного зору. Розвинений синтаксис і можливість оптимізації під конкретну платформу роблять її привабливою для інженерних завдань з підвищеними вимогами до обчислювальних ресурсів[20].

Однією з найбільш відомих бібліотек для обробки зображень у C++ є OpenCV, яка містить широкий спектр функцій для виявлення об’єктів, фільтрації, обробки потоків відео, побудови дескрипторів тощо. Завдяки тому, що OpenCV розроблена саме на C++, вона забезпечує максимальну продуктивність при використанні

низькорівневих функцій, зокрема при роботі з великими зображеннями або відео з високою частотою кадрів. Крім того, C++ підтримує використання паралельного програмування та апаратного прискорення (CUDA), що дозволяє ефективно розподіляти навантаження під час виконання задач комп'ютерного зору в реальному часі[21].

Втім, використання C++ для побудови комплексної інформаційної системи, яка поєднує веб-інтерфейс, користувацьку автентифікацію, збереження результатів та гнучке управління структурою даних, є досить складним завданням. Створення клієнт-серверної архітектури на C++ вимагає підключення додаткових інструментів, таких як Boost.Asio для роботи з мережею, або фреймворків на кшталт Wt, які є значно менш поширеними порівняно з фреймворками мов високого рівня. Крім того, розробка графічного або веб-інтерфейсу користувача в C++ потребує значного обсягу коду, додаткової конфігурації та глибших знань системного програмування.

Ще одним важливим аспектом є складність інтеграції сучасних нейронних мереж у проєкти, реалізовані мовою C++. Хоча існують готові реалізації моделей глибокого навчання, такі як Darknet для YOLOv4, підтримка новітніх версій моделей часто вимагає значних зусиль з боку розробника. Це стосується як оновлення, так і перенавчання моделей або експериментування з їх архітектурами. У зв'язку з цим, розробка систем, що активно використовують сучасні досягнення у сфері глибинного навчання, може вимагати додаткових ресурсів та інструментів для повноцінної реалізації в середовищі C++.

Підсумовуючи, мова програмування C++ є ефективною та потужною платформою для створення високопродуктивних рішень у галузі комп'ютерного зору. Завдяки низькорівневому доступу до системних ресурсів і високій швидкодії, вона ідеально підходить для розробки продуктивних алгоритмів обробки відео та зображень. Проте розробка повнофункціональної інформаційної системи, що охоплює як алгоритмічну частину, так і серверну логіку, інтерфейс користувача та інтеграцію з сучасними бібліотеками машинного навчання, потребує додаткових інструментів і ресурсів при використанні C++

Java це – одна з найпоширеніших мов програмування, яка завдяки своїй стабільності, переносимості та суворій об’єктно-орієнтованій моделі довгий час застосовується у розробці промислових програмних систем. Основною перевагою Java є її незалежність від платформи, що досягається завдяки використання віртуальної машини Java (JVM)[22]. Це дозволяє запускати програмний код на будь-якому пристрої, де встановлено JVM, що особливо цінно для побудови розподілених або масштабованих систем.

З огляду на вимоги до реалізації інформаційної системи, яка має включати веб-інтерфейс, клієнт-серверну взаємодію та підтримку складних обчислень, Java пропонує багатий набір фреймворків і бібліотек, зокрема Spring Framework, який є галузевим стандартом для створення серверних додатків. Використання Spring Boot дозволяє швидко створювати REST API, інтегрувати бази даних та реалізовувати логіку автентифікації. Також Java має потужний засіб роботи з БД – JDBC – та численні ORM-рішення (Hibernate, JPA), які суттєво спрощують доступ до даних[23].

Разом з тим, при переході до задач, пов’язаних з комп’ютерним зором та глибоким навчанням, Java демонструє обмежену гнучкість порівняно з іншими мовами. Попри існування бібліотек на кшталт DeepLearning4J, JavaCV або ND4J, спектр доступних інструментів не охоплює повною мірою сучасні рішення в галузі нейронних мереж. Наприклад, адаптація моделі YOLO (You Only Look Once), що активно використовується у комп’ютерному зорі, в середовищі Java вимагає використання сторонніх інструментів або попереднього перетворення моделей у сумісні формати (ONNX, TensorFlow SavedModel), що ускладнює інтеграцію.

Крім того, більшість актуальних дослідницьких розробок та бібліотек, пов’язаних із розпізнаванням зображень, орієнтовані на Python, а не Java, що створює додаткові бар’єри при реалізації систем, пов’язаних із машинним навчанням[24]. Це стосується як підтримки нових версій моделей, так і документації, прикладів використання та активності спільноти.

Таким чином, хоча Java є ефективною для створення масштабованої серверної логіки та організації доступу до баз даних, її використання в задачах, де основний

акцент зроблено на комп’ютерний зір і обробку відеопотоків з безпілотних літальних апаратів у режимі реального часу, може потребувати значних додаткових зусиль. Реалізація таких систем вимагає спеціалізованих бібліотек і засобів, що не є типовими для стандартного Java-стека, що ускладнює розробку та інтеграцію алгоритмів глибинного навчання й відеоаналітики.

Мова програмування C# – це мова, що була розроблена компанією Microsoft у межах платформи .NET. Вона поєднує концепції об'єктно-орієнтованого, типобезпечного та компонентно-орієнтованого програмування, і завдяки своїй строгій структурі, високому рівню абстракції та зручному синтаксису широко застосовується у створенні як настільних, так і веб-застосунків[25]. Інтеграція із середовищем розробки Visual Studio надає можливість розширювати функціонал для побудови графічних інтерфейсів, підключення до баз даних, управління сервісами та розгортання програмного забезпечення.

Особливістю мови C# є її тісна інтеграція з фреймворком .NET, що надає розробнику велику кількість вбудованих бібліотек для реалізації функціоналу на різних рівнях системи – від файлового доступу й обробки мережевого трафіку до створення інтерфейсів користувача та побудови багатопотокових застосунків[26]. Для створення веб-застосунків на C# активно використовується платформа ASP.NET Core, яка дозволяє будувати гнучкі REST API, виконувати маршрутизацію, автентифікацію, а також інтегрувати системи обробки даних і логіки бізнес-процесів.

Щодо застосування у сфері комп’ютерного зору, C# підтримує підключення до бібліотек, реалізованих мовами нижчого рівня (зокрема C++), завдяки механізму P/Invoke або через використання спеціалізованих обгорток. Існують сторонні рішення, такі як Emgu CV – оболонка для OpenCV на платформі .NET, яка дозволяє реалізувати базові задачі комп’ютерного зору: зчитування відео з камери, ідентифікація об’єктів, сегментація, фільтрація зображень тощо. Проте функціонал таких обгорток обмежений і часто відстає від можливостей вбудованого API.

Для реалізації нейронних мереж у C# розроблена бібліотека ML.NET, яка надає базові можливості навчання моделей, класифікації, регресії, а також обробки

зображень. Утім, підключення моделей, збережених у популярних форматах (наприклад, ONNX), потребує додаткових налаштувань і не завжди підтримує новітні архітектури глибокого навчання, зокрема моделі типу YOLO. Це обмежує можливість реалізації найсучасніших підходів до ідентифікації об’єктів у середовищі C#.

Слід також зазначити, що C# орієнтований переважно на розробку настільних або корпоративних систем у середовищі Windows, і хоча платформа .NET Core підтримує кросплатформність, основна частина розширень, пов’язаних із комп’ютерним зором, розрахована саме на Windows-інфраструктуру. Це створює додаткові обмеження під час розгортання системи у середовищах Linux або на edge-пристроях, що нерідко використовуються у сфері безпілотних технологій.

Таким чином, мова програмування C# є зручною, структурованою та потужною платформою для побудови графічних інтерфейсів, розробки веб-сервісів і обробки даних. Проте в умовах задач, що вимагають активного використання глибокого навчання, розпізнавання об'єктів на зображеннях і потоковому відео, її можливості потребують розширення через сторонні компоненти, які не завжди відповідають вимогам сучасних алгоритмів комп’ютерного зору.

Мова програмування JavaScript є динамічною інтерпретованою мовою програмування, яка спочатку була розроблена як інструмент для додавання інтерактивності до веб-сторінок, проте з часом еволюціонувала в повноцінну платформу для створення як клієнтських, так і серверних веб-застосунків. Завдяки екосистемі, що постійно розширюється, зокрема таким технологіям, як Node.js, JavaScript став універсальним середовищем для побудови сучасних веб-систем[27].

У контексті розробки інформаційної системи для розпізнавання об'єктів з БПЛА JavaScript відіграє важливу роль передусім у реалізації інтерфейсу користувача. Бібліотеки й фреймворки такі як React, Vue.js та Angular дозволяють створювати динамічні застосунки з однією сторінкою, які забезпечують інтерактивну візуалізацію результатів розпізнавання, обробку зображень на клієнтській стороні та передачу даних до серверної частини. У поєднанні з Node.js JavaScript також може виконувати

роль серверного середовища, що дозволяє реалізувати обробку запитів, управління файлами, автентифікацію користувачів і збереження інформації у базі даних.

Щодо машинного навчання, то JavaScript підтримує запуск нейронних мереж безпосередньо у браузері завдяки таким бібліотекам, як TensorFlow.js. Це відкриває можливості для розпізнавання об'єктів на клієнтській стороні без передачі зображень на сервер, що важливо в умовах обмеженої пропускної здатності мережі або вимог до конфіденційності даних. Однак варто зазначити, що обчислювальні можливості браузера значно поступаються серверним GPU-системам, і тому застосування JavaScript у сфері глибокого навчання залишається обмеженим переважно демонстраційними або спрощеними сценаріями.

Крім того, основна частина високопродуктивних бібліотек комп’ютерного зору, а саме: OpenCV, PyTorch, Detectron2 або YOLO. Ці бібліотеки реалізовані на інших мовах, і їх інтеграція в JavaScript-проєкт потребує побудови проміжного рівня взаємодії через REST API або використання WebAssembly. Це збільшує складність архітектури, розділяє обробку між клієнтом і сервером, а також створює додаткові вимоги до безпеки та синхронізації.

Таким чином, JavaScript це ефективним інструментом для створення інтерактивної клієнтської частини інформаційної системи, що дозволяє реалізовувати динамічні веб-інтерфейси, обробку подій користувача та візуалізацію результатів розпізнавання. Його застосування у складі повнофункціональної архітектури дає змогу організувати зручну взаємодію між користувачем і серверною частиною, а також забезпечити підтримку сучасних веб-стандартів. У поєднанні з серверними технологіями, такими як Node.js, JavaScript може також виступати як інструмент для реалізації обробки запитів, маршрутизації, керування доступом та обміну даними між модулями інформаційної системи.

Мова програмування Python це – сучасний мова, яка завдяки своїй універсальності, зрозумілому синтаксису та потужній екосистемі бібліотек стала галузевим стандартом у сферах штучного інтелекту, комп’ютерного зору й глибокого навчання. Особливістю Python є те, що вона підтримується широкою дослідницькою

спільнотою, постійно оновлюється, а також має численні офіційні та неофіційні пакети, що охоплюють майже всі сфери програмної інженерії[28].

У рамках розробки інформаційної системи ідентифікації об’єктів з безпілотних літальних апаратів використання Python є технічно доцільним і обґрунтованим. Одним із ключових завдань системи є виявлення та класифікація об’єктів на відеопотоці, що надходить з БПЛА. Для реалізації цього функціоналу у Python доступна бібліотека OpenCV, яка дозволяє зчитувати відео з камер або файлів, розбивати його на кадри, проводити попередню обробку (зміна розміру, нормалізація, фільтрація), а також виконувати візуалізацію результатів у реальному часі. Бібліотека підтримує роботу з апаратним забезпеченням БПЛА через стандартні інтерфейси, що забезпечує гнучкість у налаштуванні системи.

Ключовим компонентом системи є інтеграція моделі глибокого навчання YOLOv11, яка використовується для точного виявлення об’єктів військової техніки. Python забезпечує повну підтримку цієї моделі через бібліотеки PyTorch та Ultralytics, що дозволяє здійснювати завантаження навченої моделі, виконання інференсу на окремих кадрах або потоках, а також оптимізацію її продуктивності з використанням GPU. Бібліотека torchvision додатково полегшує підготовку датасетів, анотацію, перетворення зображень та створення навчального циклу. Для візуалізації виявлених об’єктів використовуються графічні засоби OpenCV, які дозволяють наносити межі, ідентифікатори та інші метадані на зображення.

Для реалізації веб-інтерфейсу системи застосовується фреймворк Django, який дозволяє швидко побудувати повноцінний серверний застосунок із підтримкою автентифікації, завантаження файлів, перегляду результатів розпізнавання та обліку користувачів. Завдяки Django REST Framework реалізовано взаємодію з клієнтською частиною через HTTP-запити, а також забезпечено інтеграцію з фронтендом. Статичні файли, зображення та результати обробки зберігаються у файловій системі або базі даних, яка також управляється за допомогою Python (наприклад, SQLite або PostgreSQL через ORM).

Ще однією перевагою Python є можливість розгортання системи в контейнеризованому середовищ, що полегшує масштабування, оновлення і підтримку. Бібліотеки Python мають відкритий код і дозволяють гнучко адаптувати алгоритми під конкретні умови польоту, параметри відео з БПЛА, змінні погодні умови або типи об’єктів, що розпізнаються.

Таким чином, Python є не просто зручним інструментом для програмування, а повноцінною платформою для реалізації складних інтелектуальних систем, таких як ідентифікація об'єктів із БПЛА. Поєднання потужних бібліотек комп’ютерного зору, фреймворків глибокого навчання, засобів для обробки відео та серверних технологій робить її ідеальною мовою для реалізації даної інформаційної системи.

3.1.2. Обґрунтування вибору Python, як основної мови розробки

Одним з ключових етапів створення інформаційної системи ідентифікації об’єктів з безпілотних літальних апаратів є вибір мови програмування, що забезпечує ефективну реалізацію обчислювальних процедур, гнучку інтеграцію компонентів системи та підтримку сучасних бібліотек для машинного навчання й комп’ютерного зору. Після аналізу низки кандидатів, таких як C++, Java, JavaScript та C#, було обґрунтовано доцільність використання Python як основної мови розробки.

Однією з головних причин вибору Python є його виняткова підтримка інструментів для машинного навчання та комп’ютерного зору.

Бібліотека OpenCV з відкритим вихідним кодом відіграє ключову роль в обробці візуальних даних, забезпечуючи широкий спектр інструментів для роботи із зображеннями та відеопотоками. Зокрема, вона надає функціональність для реалізації етапів попередньої обробки, таких як фільтрація шуму, нормалізація яскравості, зміна розміру зображень, а також сегментація кадрів із відео. Завдяки інтеграції з бібліотеками NumPy та Pillow, створюється продуктивне середовище, що дозволяє здійснювати складні операції над медіа даними із високою точністю, що є необхідним для підготовки вхідної інформації до подальшого аналізу алгоритмами комп’ютерного зору.

Також до переваг Python можна віднести його інтеграція з вебфреймворками, що дає змогу реалізувати повноцінну інформаційну систему з доступом через вебінтерфейс. У даному проєкті використовується фреймворк Django, що є високорівневим серверним середовищем на Python. Django підтримує побудову MVC-архітектури (Model–View–Controller), реалізацію REST API через Django REST Framework, обробку авторизації, захист даних користувача, а також легку інтеграцію з інтерфейсною частиною, створеною за допомогою HTML, CSS та JavaScript.

Python також підтримує роботу з такими базами даних: SQLite, PostgreSQL, MySQL, MongoDB та іншими, що дозволяє інтегрувати логіку зберігання результатів розпізнавання або авторизаційних даних. Django має вбудовану систему ORM (Object-Relational Mapping), котра значно спрощує роботу з базами без написання сирого SQL-коду.

Окремо варто наголосити на підтримці GPU (CUDA) – через PyTorch та Ultralytics Python забезпечує запуск моделі YOLOv11 на графічному процесорі, що критично важливо для швидкої обробки великого обсягу зображень або відео у режимі реального часу. Це робить Python придатним для проєктів, де пріоритетом є як точність розпізнавання, так і швидкодія.

Таким чином, вибір Python у поєднанні з Django, PyTorch, OpenCV та іншими бібліотеками повністю відповідає потребам розроблюваної інформаційної системи. Він забезпечує ефективну реалізацію логіки прийняття рішень, обробки зображень, розпізнавання об’єктів, керування користувачами та взаємодії з клієнтською частиною. Окрім цього, Python має можливість масштабувати систему, адаптувати її до нових моделей машинного навчання та зберігати гнучкість під час розширення функціональності.

3.1.3. Аналіз моделей розпізнавання

У галузі комп’ютерного зору однією з ключових задач є детекція об’єктів, що передбачає визначення їхніх координат на зображенні та класифікацію. Для цього застосовуються глибокі згорткові нейронні мережі, які дозволяють досягати високої

точності навіть за умов значного шуму, часткового перекриття об'єктів або складного фону. Найбільш поширеними сучасними архітектурами, що використовуються для вирішення таких задач, є YOLO, SSD, Faster R-CNN та Detectron2. Кожна з них має власні архітектурні підходи, продуктивні характеристики та обмеження, що визначають доцільність їх використання в різних типах систем.

YOLO (You Only Look Once) є однією з найвідоміших та найефективніших архітектур для об'єктного розпізнавання, що поєднує високу точність із надзвичайною швидкістю роботи[29]. Головна ідея цієї архітектури полягає в тому, що модель розглядає задачу ідентифікації, як єдину регресійну проблему, яка перетворює вхідне зображення безпосередньо у координати об'єктів і їх класи. На відміну від двоетапних моделей, де спочатку визначаються області інтересу, а потім виконується класифікація, YOLO виконує все «за один погляд», що й обумовило її назву.

Архітектурно, модель розбиває вхідне зображення на сітку фіксованого розміру (наприклад, 13×13), і кожна клітинка цієї сітки прогнозує декілька обмежувальних рамок (bounding boxes), разом із класами об'єктів і відповідними ймовірностями. Такий підхід дозволяє досягти дуже високої швидкодії – у реальному часі можна обробляти десятки або навіть сотні кадрів за секунду, залежно від конфігурації моделі та апаратного забезпечення. Швидкість є критичним чинником для задач, де обробка даних відбувається в потоковому режимі, зокрема під час роботи з відео з БПЛА.

Однією з ключових переваг YOLO є її простота в реалізації та розгортанні. Починаючи з YOLOv3, а особливо з версій YOLOv5 і далі, модель стала надзвичайно зручною для кастомізації. Розробники можуть змінювати кількість класів, структуру мережі, а також проводити тренування на власних наборах даних без глибоких знань у галузі глибокого навчання. Це досягається завдяки проекту Ultralytics YOLO, що надає зручний API та готові скрипти для тренування, валідації й інференсу моделей.

YOLO також вирізняється високою гнучкістю щодо масштабування. Існують модифікації з різним співвідношенням між швидкістю та точністю: YOLOv5s (small), YOLOv5n (nano), YOLOv5m (medium), YOLOv5l (large) та YOLOv5x (extra large). Це

дає змогу обирати конфігурацію залежно від обчислювальних ресурсів. У YOLOv8 і пізніших версіях реалізовано підтримку ONNX, TensorRT, OpenVINO та інших форматів експорту, що розширює можливості інтеграції моделі в мобільні, вбудовані або серверні системи.

З технічного погляду, модель базується на глибокій згортковій нейронній мережі з блоками типу CSPNet, що підвищує точність без значного ускладнення архітектури. Для підвищення загальної якості розпізнавання використовуються сучасні прийоми регуляризації, аугментації та втрат (loss functions), наприклад CIoU loss та Mosaic augmentation. У поєднанні це забезпечує YOLO високу середню точність (mAP), яка досягає 70–75% на стандартних наборах типу COCO, що є конкурентним результатом навіть у порівнянні з більш повільними двоетапними моделями.

Ще однією перевагою YOLO є активна спільнота розробників і підтримка з боку відкритих проєктів. Це забезпечує постійне оновлення, документування та появу нових інструментів для використання моделі у різних програмних середовищах, зокрема Python, C++, TensorRT та навіть у вебзастосунках. Зважаючи на всі ці чинники, YOLO є універсальним інструментом, який підходить як для досліджень, так і для побудови прикладних систем, включаючи системи моніторингу, безпеки, автономного керування та, зокрема, ідентифікації об’єктів з БПЛА.

У межах розробки моєї системи модель YOLOv11 використовується як основний інструмент для виявлення та класифікації об'єктів на зображеннях та у відеопотоці. Вона пройшла попереднє навчання на власному датасеті, адаптованому до специфіки зйомки з безпілотника, що дозволило досягти прийнятного балансу між точністю та швидкістю. Враховуючи доступність коду, підтримку GPU, простоту кастомізації та ефективну інтеграцію з Python-середовищем, вибір на користь YOLOv11 є повністю обґрунтованим з технічної та практичної точки зору.

SSD (Single Shot MultiBox Detector) – це ще один представник одноетапних моделей об'єктного розпізнавання, яка була розроблена з метою досягнення високої продуктивності при збереженні достатнього рівня точності. На відміну від багатьох двоетапних підходів, SSD здійснює ідентифікацію об’єктів за один прохід крізь

нейронну мережу, що дозволяє використовувати її для обробки зображень у режимі реального часу[30].

Основна особливість SSD полягає у використанні багатошарових ознак (multi-scale feature maps) для ідентифікації об'єктів на різних масштабах. Модель базується на згортковій нейронній мережі, наприклад VGG16 або ResNet, і додає кілька додаткових згорткових шарів, які працюють із зображеннями зменшеного розміру. Це дозволяє ефективно виявляти як великі, так і малі об’єкти, що особливо важливо у випадках змішаних сцен або зйомки з висоти.

SSD використовує анкор-бокси (anchor boxes) з різними співвідношеннями сторін для кожної точки на сітці ознак. Для кожного такого боксу модель прогнозує ймовірність наявності об’єкта та уточнені координати. Такий підхід знижує обчислювальні витрати, оскільки не потребує попередньої генерації регіонів інтересу, як у Faster R-CNN.

Однією з переваг SSD є її швидкість: навіть на середніх відеокартах модель здатна обробляти від 30 до 80 кадрів за секунду, що робить її конкурентоспроможною в мобільних або середовищах, що обмежені у ресурсах. SSD може бути запущена без використання високопродуктивного GPU, що важливо для вбудованих систем або пристроїв з обмеженою енергоефективністю (наприклад, дрони, смартфони або міні-ПК). Проте модель має і певні обмеження. У складних сценах, де об’єкти перекривають один одного або займають дуже малу частину зображення, точність SSD може бути помітно нижчою, ніж у двоетапних моделей. Крім того, вона менш гнучка в адаптації до спеціалізованих датасетів, порівняно з сучасними версіями YOLO, через відсутність зручних інструментів тренування та обмежену підтримку форматів експорту.

SSD демонструє хорошу продуктивність у завданнях, де важлива низька затримка і прийнятна точність, особливо у випадках, коли використання потужного GPU є недоцільним або неможливим. Модель часто використовується у мобільних застосунках, автономних системах, а також у системах відеоспостереження.

У межах аналізу для даної роботи SSD розглядався як потенційна альтернатива YOLOv11. Проте через вищу точність, кращу адаптацію до кастомних наборів даних і активнішу підтримку спільноти, вибір було зроблено на користь останньої версії YOLO.

Faster R-CNN (Region-Based Convolutional Neural Network) є класичною двоетапною архітектурою для об'єктного розпізнавання, яка вважається еталонною у багатьох дослідницьких задачах завдяки своїй високій точності. Ця модель не є одношаровим детектором, як YOLO чи SSD, а виконує розпізнавання у два послідовні етапи: спочатку визначає потенційні області, де можуть знаходитися об'єкти, а вже потім проводить класифікацію й уточнення координат[31].

Головною особливістю Faster R-CNN є використання мережі регіональних пропозицій (Region Proposal Network, RPN). Вона генерує набір ймовірних областей, що містять об’єкти, на основі згорткових ознак, сформованих глибокою нейронною мережею (наприклад, ResNet50, ResNet101). Ці регіони надалі обробляються за допомогою механізму RoI (Region of Interest) Pooling, що дозволяє «витягнути» локальні ознаки з кожної запропонованої області та передати їх на подальшу класифікацію.

Саме дворівнева структура дозволяє цій моделі забезпечувати високу якість розпізнавання об’єктів, особливо у випадках, коли вони перекривають один одного, мають складний фон або малі розміри. Faster R-CNN демонструє стабільну точність (mAP > 75%) на основних еталонних наборах зображень (наприклад, Pascal VOC, COCO) та часто використовується як базовий підхід у наукових публікаціях.

Проте ці переваги досягаються ціною обчислювальної складності. Для кожного зображення модель генерує велику кількість регіонів і обробляє кожен окремо, що значно уповільнює інференс. Середня швидкість роботи – близько 5–10 FPS на GPU середнього рівня, а на слабших пристроях цей показник є ще нижчим. Це обмежує застосування Faster R-CNN у реальному часі, особливо коли йдеться про відеопотік або обробку зображень у польових умовах.

Ще одним аспектом, який варто врахувати, є ресурсомісткість та складність конфігурування. Модель потребує більше оперативної пам’яті, ретельного підбору параметрів навчання, а також використання розширених механізмів попередньої обробки. Це робить її менш зручною для інтеграції у легковагові прикладні системи або вебзастосунки, де важлива простота та швидкодія.

Незважаючи на це, Faster R-CNN залишається одним із найточніших рішень для задач, де пріоритет надається якості розпізнавання, а не швидкості. Модель часто застосовують у медичному аналізі, аналізі супутникових зображень, криміналістиці, де результат повинен бути максимально точним, а час обробки – другорядним.

У контексті розробки системи ідентифікації об'єктів з БПЛА, де ключовим є реагування в реальному часі на зміну зображення або відео, застосування Faster R-CNN є технічно можливим, проте недоцільним. Тому модель було розглянуто на етапі аналізу, однак не включено до остаточного стеку технологій.

Detectron2 – це сучасний фреймворк для задач комп’ютерного зору, розроблений дослідницьким підрозділом Facebook AI Research (FAIR). На відміну від окремих моделей, таких як YOLO або SSD, Detectron2 є повноцінною платформою, яка дозволяє реалізовувати широкий спектр задач, включаючи об’єктну детекцію, семантичну і інстанс-сегментацію, а також keypoint detection (виявлення ключових точок на тілі, обличчі, предметах). Його модульна структура дає змогу швидко тестувати й поєднувати різні архітектури, наприклад, Faster R-CNN, RetinaNet, Mask R-CNN, Cascade R-CNN, що робить цей фреймворк дуже потужним у науково-дослідницьких та експериментальних проєктах[32].

Detectron2 побудовано на базі PyTorch і має зручний механізм конфігурацій, який дозволяє змінювати модель, бекбон, гіперпараметри й логіку тренування без переписування коду. Він підтримує як навчання з нуля, так і донавчання на попередньо натренованих моделях. Фреймворк орієнтований на глибоку кастомізацію – можна змінювати механізми втрат (loss functions), додавати нові типи head-структур, вводити кастомні трансформації або оптимізатори.

Однак цей високий рівень гнучкості супроводжується зростанням складності реалізації. Навіть базова настройка Detectron2 вимагає глибшого розуміння принципів PyTorch, структури COCO-анотацій, формату навчальних датасетів і внутрішньої архітектури фреймворку. Для менш досвідчених користувачів цей бар'єр може бути суттєвим, особливо у випадках, коли час на розгортання обмежений.

Щодо технічних характеристик, Detectron2 показує надзвичайно високу точність на класичних наборах даних (COCO, Cityscapes, LVIS), особливо у випадках складних зображень із великою кількістю об'єктів або деталізацією (наприклад, сегментація об'єктів). Його основні архітектури (Mask R-CNN, Cascade R-CNN) стабільно перевершують YOLO і SSD у показниках mAP, особливо на дрібних класах. Проте ці результати досягаються за рахунок збільшеної ресурсомісткості – як у плані пам'яті, так і часу інференсу (зазвичай 10–25 FPS навіть на високопродуктивних GPU).

З практичного погляду, Detectron2 найкраще підходить для офлайн-аналізу, серверних рішень або наукових експериментів, де головним є не швидкість, а точність і розширюваність системи. Його рідко використовують для розгортання на edge-пристроях або у вебінтерфейсах через складність інтеграції й залежність від складної екосистеми.

У межах проєкту інформаційної системи розпізнавання об'єктів з БПЛА фреймворк Detectron2 розглядався як одна з альтернатив. Проте через його громіздкість, тривалий час налаштування і високі вимоги до апаратного забезпечення, а також відсутність потреби у сегментації об'єктів або розширеній аналітиці, було вирішено не включати його до фінального стеку технологій. Попри високу точність, ця модель є надмірною для задачі швидкої обробки зображень і відео в режимі реального часу.

Далі розглянемо у таблиці 3.1 порівняння характеристики цих чотирьох поширених архітектур об'єктного розпізнавання за основними критеріями ефективності. Показник точності відображає середню ефективність моделі при різних значеннях порогу перетину (IoU) та є основним критерієм для оцінки якості виконання задачі розпізнавання об'єктів. *Швидкість* відображає кількість кадрів, які

модель може обробити за секунду, що напряму впливає на її здатність працювати у реальному часі. Параметр *придатності до реального часу* оцінює загальну відповідність моделі вимогам до затримок у розпізнаванні при обробці потокового відео або зображень. *Ресурсомісткість* вказує на необхідні обчислювальні ресурси, зокрема потребу в потужному графічному процесорі для досягнення заявленої продуктивності. Останній критерій – *гнучкість адаптації* – характеризує, наскільки просто можна адаптувати модель до кастомного датасету, змінити кількість класів або застосувати модель у нових умовах. Значення критеріїв у таблиці мають якісний характер (висока, середня, низька) і базуються на узагальнених тестах, проведених для моделей у схожих середовищах розгортання.

Таблиця 3.1

Порівняльна характеристика моделей розпізнавання об'єктів

Модель	Точність	Швидкість	Придатність до реального часу	Ресурсомісткість	Гнучкість адаптації
YOLOv11	Висока	Висока	Відмінна	Низька–середня	Висока
SSD	Середня	Висока	Добра	Низька	Середня
Faster R-CNN	Дуже висока	Низька	Обмежена	Висока	Висока
Detectron2	Дуже висока	Середня	Обмежена	Висока	Висока

Аналіз результатів порівняння показує, що моделі типу YOLO, зокрема новітні версії (YOLOv8–YOLOv11), демонструють оптимальний баланс між точністю, швидкодією та адаптивністю. Зазначені моделі є найбільш придатними для розв'язання задач розпізнавання об'єктів у режимі реального часу, що має принципове значення для інформаційних систем, орієнтованих на обробку потокових даних, отриманих із безпілотних літальних апаратів.

У порівнянні з YOLO, модель SSD хоч і забезпечує високу швидкість, проте поступається в точності, особливо в складних умовах, коли на зображенні присутні об’єкти різного масштабу або з частковим перекриттям. Її доцільно застосовувати у менш вимогливих до якості системах або в мобільних застосунках.

Моделі Faster R-CNN і Detectron2 забезпечують високу точність і глибоку деталізацію, проте мають вищу обчислювальну складність та обмеженість у роботі в режимі реального часу. Вони найчастіше використовуються у серверних рішеннях для офлайн-аналізу, де продуктивність не є визначальним фактором.

З огляду на потреби розроблюваної системи, де головними критеріями є швидкість розпізнавання, можливість використання на GPU, гнучка адаптація до власного датасету, а також відкритість коду та підтримка індустріальних стандартів, найбільш доцільним є використання моделі YOLOv11, яка поєднує високу ефективність та зручність інтеграції в Python-середовище.

3.1.4. Обґрунтування вибору YOLOv11

Для побудови ефективної інформаційної системи ідентифікації об'єктів з безпілотних літальних апаратів (БПЛА) критично важливим є вибір оптимальної моделі комп’ютерного зору, яка забезпечувала б високу швидкість обробки, достатню точність розпізнавання та можливість адаптації до специфіки власного набору даних. З урахуванням вищенаведених вимог, на підставі порівняльного аналізу сучасних підходів до ідентифікації об’єктів, було обґрунтовано доцільність використання архітектури YOLOv11, як базової моделі системи розпізнавання.

Архітектура YOLOv11 належить до класу одноетапних моделей глибокого навчання, в яких процес виявлення та класифікації об’єктів виконується за один прохід через згорткову нейронну мережу. У порівнянні з двоетапними моделями, такими як Faster R-CNN або Detectron2, вона характеризується значно меншою затримкою обробки (Latency), що дозволяє забезпечити обробку вхідних фото- та відеоданих у режимі, наближеному до реального часу. Це є суттєвою перевагою при роботі з БПЛА, де швидкодія системи прямо впливає на якість прийняття рішень.

YOLOv11, як представник останнього покоління моделей серії YOLO, включає в себе низку технічних покращень: оновлену архітектуру neck-блоків, оптимізовані head-компоненти, більш ефективну функцію втрат (CIoU), а також підтримку inference через PyTorch, ONNX, TensorRT і OpenVINO. Ці особливості сприяють зниженню обчислювальної складності без втрати точності та дозволяють легко інтегрувати модель у вже існуюче середовище Python/Django. Така інтеграція уможливлює реалізацію централізованого вебінтерфейсу для взаємодії користувача із системою, з обробкою запитів у серверній частині та представленням результатів розпізнавання через веб браузер.

Додатковою перевагою YOLOv11 є підтримка повноцінного донавчання на власному датасеті. У межах даного дослідження було сформовано спеціалізований набір зображень, отриманих із камер БПЛА, що містив об’єкти, актуальні для розв’язуваної задачі. Анотація даних здійснювалася у форматі YOLO, що забезпечило повну сумісність із вимогами моделі. Застосування донавчання дозволило досягти високих показників точності на кастомному наборі без потреби у масштабному навчанні «з нуля».

З технічного погляду, реалізація обробки відео здійснюється з використанням GPU-прискорення на базі технології CUDA. Модель демонструє стабільну продуктивність у межах 50–60 кадрів за секунду при роздільній здатності 720p, що є прийнятним значенням для більшості задач реального часу. Крім того, експортування моделі у формат ONNX відкриває можливості для подальшої оптимізації роботи системи на пристроях котрі мають обмеженні ресурси.

Одним із важливих аспектів вибору YOLOv11 є її відповідність практичним вимогам: відкритість коду, активна підтримка з боку спільноти розробників, наявність документації та утиліт для навчання, валідації й тестування. Крім того, на відміну від SSD та Faster R-CNN, модель YOLOv11 забезпечує кращу адаптивність до змін у датасеті, простоту інтеграції у вебархітектуру та вищу ефективність при обмежених ресурсах.

Таким чином, застосування YOLOv11 у межах побудови інформаційної системи ідентифікації об’єктів з БПЛА є технічно виправданим, оскільки вона повністю відповідає вимогам до точності, швидкодії, масштабованості, підтримки апаратного прискорення та сумісності з обраним стеком технологій. Це дозволило досягти високої якості розпізнавання, забезпечити обробку відеопотоків у реальному часі та реалізувати надійну інтеграцію з вебінтерфейсом системи.

3.2 Технічні характеристики обраних програмних засобів розроблення

Було здійснено всебічний аналіз сучасних мов програмування, фреймворків, бібліотек та допоміжних інструментів, які найкраще підходять для побудови інформаційної системи ідентифікації об’єктів на основі зображень, отриманих з безпілотних літальних апаратів (БПЛА). У ході аналізу оцінювались такі ключові аспекти: продуктивність системи, зручність інтеграції компонентів, підтримка машинного навчання та комп’ютерного зору, доступність документації, активність спільноти, можливість масштабування, а також відповідність специфіці веб-орієнтованих систем.

Оскільки система повинна бути реалізована як веб-застосунок з можливістю обробки даних у реальному часі, було поставлено вимоги до високої стабільності серверної частини, ефективної реалізації механізмів обробки зображень, інтеграції з глибинними нейронними мережами, а також підтримки масштабованої архітектури. Особливу увагу приділено модульності системи, щоб окремі її частини – веб-інтерфейс, модуль розпізнавання об’єктів, база даних – могли розвиватись незалежно.

У рамках цього аналізу було розглянуто мови програмування Python, JavaScript, C++ та відповідні фреймворки, такі як Django, Flask, OpenCV, TensorFlow, PyTorch. Python було обрано як основну мову розробки серверної частини через її простоту, високу популярність, а також велику кількість потужних бібліотек для аналізу даних, комп’ютерного зору та реалізації глибинного навчання.

Для побудови серверної логіки та управління маршрутизацією запитів було обрано фреймворк Django, який має вбудовану ORM-систему для роботи з

реляційними базами даних, підтримку REST API через Django REST framework, а також забезпечує високий рівень безпеки. Django забезпечує зручне створення прототипів і розгортання проєкту завдяки використанню системи шаблонів та вбудованих засобів для автоматизованого управління міграціями бази даних.

Модуль обробки зображень реалізований на основі бібліотеки OpenCV. Вона дозволяє виконувати широкий спектр операцій, таких як масштабування, нормалізація яскравості, фільтрація шумів, зміна формату зображень, перетворення колірних просторів, побудова гістограм тощо. Для додаткової обробки використовуються також бібліотеки NumPy (операції над масивами пікселів) та Pillow (PIL) для базових маніпуляцій із зображеннями у форматах JPEG, PNG, BMP тощо.

Для виявлення та класифікації об’єктів на зображеннях з БПЛА використано сучасну архітектуру YOLOv11 – модель глибинного навчання, яка поєднує високу точність виявлення з можливістю обробки відео- та фотоданих у реальному часі. YOLOv11 є наступником популярної серії моделей YOLO (You Only Look Once) і має вдосконалену структуру, оптимізовану для високошвидкісного розпізнавання зображень. Модель була реалізована за допомогою фреймворку PyTorch – гнучкого інструмента для побудови, навчання та тестування нейронних мереж. Для полегшення інтеграції моделі в систему використовувався Ultralytics YOLO – офіційний репозиторій з відкритим кодом, який підтримує зручну розмітку даних, тренування моделей та тестування точності.

Фронтенд-секція реалізована з використанням HTML5, CSS3, JavaScript, а також фреймворку Bootstrap, що забезпечив адаптивну верстку веб-інтерфейсу для відображення результатів розпізнавання. Завдяки Bootstrap було досягнуто зручності взаємодії користувача з системою, незалежно від типу пристрою. Веб-інтерфейс дозволяє завантажувати зображення, переглядати об’єкти на зображенні, отримувати інформацію про тип, кількість та координати виявлених об’єктів.

Обмін даними між фронтендом і серверною частиною реалізовано через REST API, що забезпечує стандартизовану взаємодію та можливість майбутньої інтеграції з мобільними застосунками або іншими інформаційними системами. REST API

реалізовано на основі Django REST framework, який спрощує створення серіалізаторів, контролерів та обробників запитів.

Для зберігання метаданих, результатів розпізнавання та інформації про зображення було обрано вбудовану реляційну базу даних SQLite, яка інтегрується з Django за замовчуванням. SQLite є ефективним рішенням на етапі прототипування завдяки простоті розгортання, відсутності необхідності у конфігурації окремого сервера бази даних. У перспективі система може бути легко адаптована до більш продуктивної СКБД, такої як PostgreSQL або MySQL, з використанням механізмів міграції Django.

Процес навчання моделі YOLOv11 охоплював підготовку датасету, його аугментацію, розмітку зображень, навчання на GPU, а також валідацію на тестовій вибірці. Усі етапи реалізовані засобами Ultralytics із можливістю тонкого налаштування гіперпараметрів.

На основі проведеного аналізу, для реалізації системи були обрані такі основні програмні засоби:

- Python – основна мова програмування для реалізації логіки системи, комп’ютерного зору, обробки зображень та взаємодії з моделлю машинного навчання;
- Django – фреймворк для створення серверної частини веб-додатку, забезпечення маршрутизації, авторизації, бази даних, API;
- OpenCV, NumPy, Pillow – бібліотеки для обробки зображень та попередньої обробки даних;
- PyTorch, Ultralytics – фреймворки для побудови та навчання моделі розпізнавання об’єктів YOLOv11;
- HTML, CSS, JavaScript, Bootstrap – засоби створення адаптивного веб-інтерфейсу користувача;
- SQLite – легка реляційна СКБД для зберігання даних на етапі прототипування.

Комплексне використання цих інструментів дозволило створити функціональну, модульну та масштабовану систему ідентифікації об’єктів, яка здатна працювати в реальному часі з високою точністю та інтегрується у веб-середовище з зручним користувацьким інтерфейсом.

Висновки до третього розділу

У третьому розділі бакалаврської кваліфікаційної роботи здійснено безпосередню реалізацію інформаційної системи для ідентифікації об’єктів із застосуванням даних, отриманих з безпілотних літальних апаратів. Основну увагу зосереджено на виборі архітектури програмної системи, реалізації її окремих функціональних модулів та інтеграції алгоритмів машинного навчання.

У ході реалізації було обґрунтовано вибір мови програмування Python, яка завдяки своїй простоті, високій продуктивності та широкій екосистемі бібліотек (зокрема OpenCV, Ultralytics, Torch тощо) є оптимальним рішенням для задач комп’ютерного зору та глибокого навчання. Як серверний фреймворк обрано Django, який забезпечує зручну організацію коду, ефективну взаємодію з базами даних, підтримку автентифікації користувачів та розширювану структуру вебзастосунків.

Вибір моделі YOLOv11, яка є сучасним розвитком архітектури YOLO, дозволив досягти високої точності та швидкості розпізнавання об’єктів. Це є критично важливим для оперативного аналізу зображень, отриманих з БПЛА, що можуть містити цільову інформацію про об'єкти інфраструктури, техніку, людей тощо.

Узагальнюючи, можна зазначити, що реалізована система повністю відповідає вимогам до сучасних інформаційних систем з ідентифікації об'єктів. Вона поєднує в собі гнучкість вебплатформи, потужність алгоритмів глибокого навчання та зручність інтерфейсу для кінцевого користувача.

РОЗДІЛ 4

Практична реалізація

4.1 Опис створеного програмного засобу

4.1.1. Опис механізмів

З метою реалізації програмного рішення бакалаврської кваліфікаційної роботи для інформаційної системи ідентифікації об’єктів з безпілотних літальних апаратів було розроблено такі механізми:

- Модуль ініціалізації та завантаження моделі YOLO;
- Модуль розпізнавання об'єктів на зображеннях;
- Модуль розпізнавання об'єктів на відео;
- Модуль обробки відео з веб-камери;
- Модуль взаємодії з користувачем через веб-інтерфейс;

Модуль, який призначений для ініціалізації та завантаження моделі YOLO відповідає за початкове завантаження нейронної мережі YOLOv11, яка використовується для розпізнавання об’єктів на зображеннях, відео та потокових кадрах з камери. Основна задача – одноразово зчитати ваги моделі з диску, розмістити її у відеокарті (GPU) та встановити базові параметри обробки. У модулі використовуються сторонні бібліотеки torch, ultralytics, os та cv2. Завантаження моделі виконується наступним чином:

```
model = YOLO(os.path.join(settings.BASE_DIR, 'detection_app', 'best.pt'))
if not torch.cuda.is_available():
    raise RuntimeError("CUDA (GPU) не доступна. Ініціалізація моделі заборонена
без GPU.")
model.to('cuda')
model.conf = 0.4
model.iou = 0.5
model.max_det = 100
```

На початку створюється об’єкт моделі YOLO, в який передається шлях до попередньо натренованого файлу вагів best.pt. Далі виконується перевірка доступності графічного процесора: якщо він не виявлений, ініціалізація припиняється і викидається помилка. Це обмеження запроваджене через те, що модель обробляє

зображення великого розміру, що є надто ресурсомістким для центрального процесора (CPU).

Після підтвердження доступності GPU, модель переноситься на графічний процесор методом to('cuda'). Після цього встановлюються ключові параметри:

- conf = 0.4 – поріг впевненості для виявлення об’єкта. Якщо значення нижче – об'єкт вважається недостовірним.
- iou = 0.5 – поріг перекриття (IoU), який використовується для пригнічення накладань (NMS).
- max_det = 100 – максимальна кількість об’єктів, які можуть бути виявлені на одному зображенні.

Таке розділення дозволяє один раз підготувати модель і уникати багаторазового її завантаження при кожному новому запиті від користувача. Завдяки цьому зменшується час відповіді сервера та підвищується ефективність всієї системи.

Модуль, що призначений для розпізнавання об’єктів на зображеннях реалізує функціональність аналізу та виявлення об’єктів на статичних зображеннях. Він приймає шлях до зображення, зчитує його, масштабує у разі необхідності, передає його в попередньо завантажену модель YOLOv11, а потім зберігає результат з нанесеними рамками. Основна функція модуля має наступний вигляд:

```
def detect_image(image_path):
    img = cv2.imread(image_path)
    if img is None:
        raise Exception(f"Не вдалося прочитати зображення: {image_path}")

    h, w = img.shape[:2]
    if max(h, w) > 1280:
        scale = 1280 / max(h, w)
        img = cv2.resize(img, (int(w * scale), int(h * scale)))

    results = model(img)[0]
    result_img = results.plot()
    result_dir = os.path.join(settings.MEDIA_ROOT, 'results')
    os.makedirs(result_dir, exist_ok=True)
    result_filename = f"result_{Path(image_path).stem}.jpg"
    result_path = os.path.join(result_dir, result_filename)
    cv2.imwrite(result_path, result_img)
    return result_path, results
```

На початку функція використовує метод cv2.imread, щоб зчитати зображення з диску у вигляді масиву пікселів. Якщо файл не вдається прочитати – викидається виключення.

Після зчитування перевіряється розмір зображення: якщо воно перевищує 1280 пікселів по ширині або висоті, то зображення масштабується пропорційно. Це дозволяє пришвидшити обробку та зменшити навантаження на GPU.

Далі обробка зображення виконується через модель YOLOv11. Результат results = model(img)[0] містить координати об’єктів, їх класи та впевненості. Метод results.plot() автоматично візуалізує детекції, додаючи кольорові рамки та підписи.

Остаточний результат зберігається у підкаталозі results каталогу медіа-файлів, що дозволяє легко відображати або завантажувати оброблене зображення через веб-інтерфейс. Назва результату формується на основі імені початкового зображення, щоб уникнути конфліктів.

Функція повертає:

- шлях до обробленого зображення;
- об’єкт results, що містить інформацію про знайдені класи, впевненості та координати рамок.

Таким чином, модуль реалізує повний цикл: завантаження → обробка → збереження → повернення результату.

Наступним розлягнемо модуль, що призначений для автоматизованої обробки відеофайлів з метою виявлення об'єктів. Метою даного модуля є покадрове виявлення об’єктів на відео за допомогою нейромережевої моделі YOLOv11 натренованої на власному датасеті. Його реалізація базується на концепції багатопотокового програмування, що дозволяє паралельно виконувати ключові етапи обробки: зчитування, аналіз кадрів та збереження результатів. Такий підхід значно підвищує швидкодію системи при роботі з відеофайлами великого розміру або тривалості.

На початку відкривається відеофайл та зчитуються його основні характеристики – ширина, висота, кількість кадрів і частота оновлення:

```
cap = cv2.VideoCapture(video_path)
```

```
if not cap.isOpened():
    raise Exception("Не вдалося відкрити відеофайл")

width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(cv2.CAP_PROP_FPS))
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
```

Для відкриття відеофайлу використовується клас VideoCapture з бібліотеки OpenCV, якому передається шлях до вхідного відео. Якщо файл не вдалося відкрити (наприклад, файл не існує або формат не підтримується), програма викидає виняток з поясненням помилки.

Після успішного відкриття, з відео зчитуються технічні параметри:

- width – ширина кадру у пікселях;
- height – висота кадру;
- fps – частота кадрів (кількість кадрів за секунду);
- total_frames – загальна кількість кадрів у відеофайлі.

Ці значення використовуються в подальшій обробці для масштабування зображень, створення вихідного відеофайлу з правильними параметрами та відображення прогресу обробки.

Наступним етапом є перевірка роздільної здатності відео та, за необхідності, масштабування кадрів для оптимізації обчислювальних витрат при подальшій обробці.

```
target_size = 1280
resize_factor = 1.0
if width > target_size or height > target_size:
    resize_factor = target_size / max(width, height)
    new_width = int(width * resize_factor)
    new_height = int(height * resize_factor)
else:
    new_width, new_height = width, height
```

Для покращення швидкодії системи на етапі попередньої обробки здійснюється перевірка, чи перевищує роздільна здатність вхідного відео задане граничне значення target_size, яке встановлено на рівні 1280 пікселів. Це обмеження обрано як компроміс між якістю вхідних даних та швидкістю виконання детекції.

У разі перевищення встановлюється коефіцієнт масштабування resize_factor, який обчислюється як відношення максимально допустимого розміру до поточного

значення ширини або висоти відео. За цим коефіцієнтом розраховуються нові значення ширини (new_width) та висоти (new_height) кадру, які згодом будуть використані під час зчитування та обробки кожного окремого кадру.

Якщо ж вхідне відео має роздільну здатність, що не перевищує граничне значення, масштабування не виконується, і розміри кадру залишаються незмінними.

Таким чином, даний етап забезпечує динамічне налаштування розміру кадрів відповідно до технічних обмежень обчислювального середовища, що дозволяє забезпечити стабільну роботу системи на комп'ютерах із різною продуктивністю.

Наступним етапом є підготовка середовища для збереження результатів обробки, зокрема створення директорії для вихідних відеофайлів та ініціалізація об'єкта для запису результатного відео.

```
    result_dir = os.path.join(settings.MEDIA_ROOT, 'results')
    os.makedirs(result_dir, exist_ok=True)
    output_filename =
f"result_video_{Path(video_path).stem}_{int(time.time())}.mp4"
    output_path = os.path.join(result_dir, output_filename)
    fourcc = cv2.VideoWriter_fourcc(*'avc1')
    out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))
```

У першому рядку формується шлях до підкаталогу results, який зберігається у кореневій директорії медіафайлів веб-додатку. Метод os.makedirs(..., exist_ok=True) гарантує, що у разі відсутності вказаного каталогу він буде створений автоматично; якщо директорія вже існує – помилка не виникатиме.

Далі формується ім'я вихідного відеофайлу. Для цього використовується назва вхідного файлу (без розширення), до якої додається мітка часу, що унеможливлює конфлікти при повторній обробці файлів з однаковими назвами. Розширення файлу встановлюється .mp4, що є сучасним форматом з широкою підтримкою.

Для створення відеозапису використовується об'єкт cv2.VideoWriter, у який передаються наступні параметри:

- output_path – шлях до файлу з результатом;
- fourcc – чотирибуквений кодек, що вказує на використання формату стиснення avc1 (H.264);
- fps – частота кадрів, що відповідає оригінальному відео;

- (width, height) – розмір кадру, який зберігатиметься безпосередньо у результатному відео (без масштабування).

Таким чином, цей етап гарантує коректне створення вихідного середовища для запису результатів детекції у відеоформаті, що повністю відповідає початковим технічним параметрам вхідного відео.

Переходимо до наступного етапу, на якому здійснюється ініціалізація структур для організації багатопотокової обробки відеокадрів: створюються черги для передачі даних між потоками, а також керуючий об’єкт для синхронізації зупинки роботи.

```
frame_queue = queue.Queue(maxsize=30)
result_queue = queue.Queue(maxsize=30)
stop_event = Event()
```

У цьому фрагменті використовується стандартна бібліотека queue, яка забезпечує потокобезпечну передачу даних між різними потоками. Створюються дві окремі черги:

- frame_queue – призначена для зберігання кадрів, які зчитуються з відеофайлу та очікують обробки;
- result_queue – слугує буфером для збереження вже оброблених кадрів із нанесеними результатами детекції.

Обидві черги мають обмеження у 30 елементів (maxsize=30), що дає змогу уникнути надмірного накопичення кадрів у пам’яті та підтримує контрольований обсяг даних під час обробки. Це особливо важливо при роботі з довгими або високоякісними відеофайлами.

Об’єкт stop_event, який створюється за допомогою класу Event з модуля threading, використовується як глобальний сигнал завершення роботи потоків. Його використання дозволяє вчасно зупинити фонові процеси (наприклад, зчитування кадрів або запис результатів) при досягненні кінця відео або виникненні критичної помилки.

Таким чином, цей етап створює основу для асинхронної взаємодії між трьома незалежними потоками, які будуть виконувати зчитування, обробку та збереження кадрів.

Модуль, призначений для обробки відеопотоку з веб-камери в реальному часі. Він реалізує механізм асинхронної обробки відеокадрів, які надходять безперервно з веб-камери користувача. Основна мета – забезпечити можливість розпізнавання об’єктів у режимі реального часу з мінімальною затримкою та збереженням продуктивності.

Цей компонент активно використовує багатопоточність та черги для обробки кадрів незалежно від швидкості зчитування або передачі результатів. На відміну від обробки відеофайлів, тут потрібно постійно підтримувати живий потік даних, адаптуючи систему до роботи у "streaming"-режимі.

Головна функція модуля – detect_webcam_frame(frame)

```
def detect_webcam_frame(frame):
    try:
        if processing_thread is None or not processing_thread.is_alive():
            initialize_processing_thread()
        h, w = frame.shape[:2]
        original_size = (h, w)
        frame_id = int(time.time() * 1000)
        if max(h, w) > 960:
            scale = 960 / max(h, w)
            resized_frame = cv2.resize(frame, (int(w * scale), int(h * scale)))
        else:
            resized_frame = frame.copy()
        if not frame_queue.full():
            frame_queue.put((resized_frame, frame_id, original_size))
        else:
            print("⚠️ Черга обробки заповнена, кадр пропущено")
        try:
            result_frame, detections, _ = result_queue.get_nowait()
            print(f"🔍 CPU: {psutil.cpu_percent(interval=0.1)}%, Об'єктів: {len(detections)}")
            return result_frame, detections
        except queue.Empty:
            print("🔄 Очікування результатів обробки...")
            return frame, []
    except Exception as e:
        print(f"❌ Помилка під час детекції кадру веб-камери: {e}")
        return frame, []
```

Загальна логіка функції

Ініціалізація потоку обробки хдійснюється перед початком роботи перевіряється, чи активний фоновий потік обробки (processing_thread). Якщо він не існує або

завершився – викликається функція initialize_processing_thread(), яка запускає окремий демон-потік, у якому відбувається батчова обробка відеокадрів.

Підготовка кадру

З вхідного кадру зчитуються розміри, після чого кадр масштабується до роздільної здатності не більше ніж 960 пікселів по більшій стороні – це оптимізує швидкодію. Генерується унікальний ідентифікатор кадру frame_id, що базується на поточному часі.

Передача кадру на обробку відбувається таким чином, кадр поміщається у вхідну чергу frame_queue, якщо вона не заповнена. У протилежному випадку кадр пропускається, і виводиться відповідне попередження.

Отримання результату обробки відбувається з result_queue, котрий намагається негайно зчитати оброблений кадр та виявлені об’єкти. Якщо результати вже є, вони повертаються разом із кадром. Якщо черга пуста – повертається необроблений кадр, і користувач отримує повідомлення про очікування.

Обробка винятків

Усі помилки перехоплюються, записуються у консоль, і в разі збоїв система продовжує передавати оригінальний кадр без зупинки роботи потоку.

Супутні компоненти модуля

Функція process_frames_worker() це фоновий потік, який асинхронно обробляє кадри з frame_queue батчами, виконує inference через YOLO-модель, формує результати з виявленими об’єктами та передає їх до result_queue.

Функція initialize_processing_thread() відповідає за створення і запуск потоку process_frames_worker, якщо він ще не працює. Потік запускається як демон, щоб не блокувати завершення програми.

Функція stop_processing_thread() забезпечує коректне завершення роботи потоку та очищення черг, якщо потрібно зупинити систему або перезапустити обробку.

Модуль дозволяє інтегрувати обробку потокового відео з веб-камери у вебінтерфейс або десктоп-додаток, де необхідно здійснювати розпізнавання об'єктів у

реальному часі. Завдяки розподіленню обов’язків між потоками та чергами, забезпечується висока швидкість роботи, навіть за умов інтенсивного відеопотоку.

Тепер розглянемо останій з модулів, призначений для взаємодії користувача з системою через веб-інтерфейс. Він представлень є основною точкою входу у систему зі сторони користувача. Він приймає запити з інтерфейсу (через форми або API), передає отримані дані у відповідні функції для обробки зображення чи відео, та формує відповідь, що включає результати роботи нейронної мережі YOLOv11.

У даному модулі реалізовано кілька функцій представлення, проте головною є detect_objects, яка обробляє зображення, надіслані користувачем, і передає їх до модуля детекції.

Головна функція модуля – detect_objects(request)

```
@csrf_exempt
def detect_objects(request):
    if request.method == 'POST' and request.FILES.get('image'):
        uploaded_file = request.FILES['image']
        file_name = f"{int(time.time())}_{uploaded_file.name}"
        file_path = os.path.join(settings.MEDIA_ROOT, 'uploads', file_name)

        os.makedirs(os.path.dirname(file_path), exist_ok=True)
        with open(file_path, 'wb') as f:
            for chunk in uploaded_file.chunks():
                f.write(chunk)

        result_path, results = detect_image(file_path)
        response_data = {
            'status': 'success',
            'result_image_url': f"/media/results/{Path(result_path).name}",
            'detections': []
        }
        if results:
            for box in results.boxes:
                cls_id = int(box.cls.item())
                conf = float(box.conf.item())
                label = results.names[cls_id]
                response_data['detections'].append({
                    'label': label,
                    'confidence': round(conf, 2)
                })
        return JsonResponse(response_data)
    return JsonResponse({'status': 'error', 'message': 'Invalid request'},
status=400)
```

Загальна логіка функції

Прийом вхідного запиту Функція реагує на POST-запити, які містять зображення (request.FILES.get('image')). Завдяки декоратору @csrf_exempt, запит можна надіслати навіть без токена CSRF, що зручно для API або frontend-викликів.

Зображення зберігається на сервері в директорії media/uploads. Назва файлу доповнюється міткою часу, щоб уникнути конфліктів при одночасному завантаженні кількох зображень.

Передача на обробку відбувається завдяки функції detect_image передається повний шлях до збереженого зображення. Вона повертає шлях до результатного зображення та об’єкт results, що містить інформацію про знайдені об’єкти.

Відповідь формується у форматі JSON і містить такі дані: статус (success), посилання на оброблене зображення з результатами (result_image_url) та масив об’єктів (detections), які містять назву класу (label) та рівень впевненості (confidence).

Обробка помилкових запитів виконується, якщо запит не є POST або не містить файлу зображення, повертається повідомлення про помилку зі статусом HTTP 400.

У модулі також можуть бути реалізовані подібні функції для обробки відео та кадрів з веб-камери (наприклад, process_video, process_webcam), які працюють за схожим принципом:

- приймають відеофайл або зображення з frontend-інтерфейсу;
- викликають відповідну функцію з модуля yolo.py;
- формують відповідь із результатами.

Загалом цей модуль представлень (views.py) слугує мостом між користувацьким інтерфейсом та нейронною мережею. Він пов’язує HTTP-запити з функціональністю, реалізованою у модулі yolo.py, забезпечуючи гнучкий API для інтеграції з веб-застосунками або клієнтськими системами.

4.1.2. Структура бази даних

Для забезпечення збереження та структурованої обробки даних у межах розробленої інформаційної системи ідентифікації об’єктів з безпілотних літальних апаратів реалізовано базу даних на основі вбудованого ORM-фреймворку Django.

Побудована база даних відповідає принципам реляційної моделі і дозволяє забезпечити ефективне керування користувачами, зберігання історії запитів на розпізнавання та результуючих даних.

Ключовим елементом бази даних є сутність CustomUser, яка реалізує розширену модель користувача, що продемонстровано на рисунку 4.1. Вона успадковує всі базові поля від стандартної абстрактної моделі AbstractUser, зокрема ім’я користувача, електронну пошту, хешований пароль, а також дати створення та останньої активності. У розширеній реалізації додатково передбачено поле bio, що дозволяє зберігати довільну текстову інформацію про користувача. Це дає змогу персоналізувати взаємодію користувача із системою та, за потреби, використовувати додаткові атрибути без внесення змін до основної моделі автентифікації. Кожен обліковий запис є унікальним і може мати кілька пов’язаних запитів до модуля розпізнавання.

	cid	name	type	notnull	dflt_value	pk
1	0	id	INTEGER	1	NULL	1
2	1	password	varchar(128)	1	NULL	0
3	2	last_login	datetime	0	NULL	0
4	3	is_superuser	bool	1	NULL	0
5	4	username	varchar(150)	1	NULL	0
6	5	first_name	varchar(150)	1	NULL	0
7	6	last_name	varchar(150)	1	NULL	0
8	7	email	varchar(254)	1	NULL	0
9	8	is_staff	bool	1	NULL	0
10	9	is_active	bool	1	NULL	0
11	10	date_joined	datetime	1	NULL	0
12	11	bio	TEXT	0	NULL	0

Рис. 4.1 Структура таблиці користувачів

Другим основним компонентом є сутність RecognitionHistory, що фіксує інформацію про кожну спробу розпізнавання, здійснену користувачем, графічне відображення якої продемонстровано на рисунку 4.2. Кожен запис у цій таблиці містить посилання на користувача (через зовнішній ключ), що дозволяє реалізувати зв’язок типу «один-до-багатьох». Окрім імені файлу (file_name) та типу носія (media_type), у записі зберігаються також текстові поля з розпізнаними класами

(recognized_classes) та відповідними показниками достовірності (confidence_scores). Таке представлення дозволяє зберігати результати у компактній формі – наприклад, як JSON-рядки – що спрощує виведення результатів на інтерфейсі.

Сутність RecognitionHistory додатково містить поле timestamp, яке автоматично фіксує дату та час створення запису, що важливо для ведення журналу активності. Крім того, у полі result_image зберігається зображення з візуалізованими результатами розпізнавання, яке генерується після успішного проходження алгоритму ідентифікації. Це зображення слугує як зручна форма представлення для користувача – з накладеними рамками об'єктів, назвами класів і відсотками достовірності.

	cid	name	type	notnull	dflt_value	pk
1	0	id	INTEGER	1	NULL	1
2	1	file_name	varchar(255)	1	NULL	0
3	2	media_type	varchar(10)	1	NULL	0
4	3	recognized_classes	TEXT	1	NULL	0
5	4	confidence_scores	TEXT	1	NULL	0
6	5	timestamp	datetime	1	NULL	0
7	6	result_image	varchar(100)	0	NULL	0
8	7	user_id	bigint	1	NULL	0

Рис. 4.2 Структура таблиці історії розпізнаваня

Модель CustomUser має один до багатьох зв'язок з моделлю RecognitionHistory, що відображає реальний сценарій взаємодії – коли один користувач може здійснити багато сеансів розпізнавання, кожен із яких супроводжується окремим набором результатів. Така структура є гнучкою, масштабованою та відповідає потребам системи щодо подальшого аналізу, фільтрації результатів, формування звітності або статистичної обробки.

Таким чином, запропонована модель бази даних забезпечує повноцінне збереження історії розпізнавань, підтримку персоналізованого доступу до даних та можливість масштабування функціональності без порушення цілісності схеми. Завдяки застосуванню ORM-технології, структура зберігає узгодженість між рівнем

коду та фізичним представленням у базі даних, що сприяє підвищенню гнучкості подальшої розробки та підтримки системи.

4.2 Інструкція користувача

Вступ

Розроблена інформаційна система призначена для автоматизованого розпізнавання об'єктів на фото, відео або відеопотоці, що надходять з безпілотних літальних апаратів (БПЛА). Програмне рішення орієнтоване на використання у веб-середовищі й забезпечує простий та доступний інтерфейс для обробки візуальних даних з метою виявлення цільових об’єктів за допомогою технологій комп’ютерного зору. Застосування даної системи підвищує ефективність аналізу візуальної інформації, що може використовуватись у сфері моніторингу, охорони, управління інфраструктурою, сільського господарства тощо.

Система функціонує з використанням моделі глибокого навчання YOLOv11, яка натренована на спеціалізованому наборі даних, що відображає характерні об’єкти для зйомки з БПЛА. Користувачі системи мають змогу авторизуватися, подавати на обробку фото- або відеоматеріали та отримувати результати розпізнавання в інтерактивному форматі. Для захисту доступу реалізовано механізм автентифікації, що забезпечує безпечне використання програмного засобу.

Загальні відомості про програму

Інформаційна система отримала назву «Розпізнавання Зображень та Відео». Програмне забезпечення розроблено з використанням мови програмування Python та фреймворку Django для створення серверної частини веб-застосунку. Для формування інтерактивного інтерфейсу використано технології HTML, CSS і JavaScript. Система функціонує у браузері, не потребує встановлення на клієнтському пристрої й є доступною з будь-якого ПК або ноутбука, що має підключення до мережі Інтернет.

Основними функціями системи є:

- реєстрація нового користувача;
- авторизація за логіном і паролем;

- розпізнавання об'єктів на зображеннях;
- обробка відеофайлів і відеопотоків у реальному часі;
- розпізнавання об’єктів на оброблених відео та відеопотоків

перегляд отриманих результатів у графічному та текстовому вигляді.

Усі дані про виконані операції зберігаються в базі даних, що дозволяє здійснювати подальший аналіз або перегляд результатів. Система не потребує складного додаткового обладнання та сумісна з усіма поширеними веб-браузерами.

Класи вирішуваних завдань

Система виконує низку завдань, пов’язаних з виявленням об’єктів на вхідних медіаданих. Передусім, реалізовано функціональність реєстрації користувача, яка передбачає створення облікового запису з логіном, паролем і базовою інформацією. Після реєстрації користувач може авторизуватися та отримати доступ до головного інтерфейсу.

Головна сторінка системи містить інструменти для завантаження зображення, запуску розпізнавання та виведення результатів. Результат подається у вигляді модифікованого зображення з нанесеними рамками навколо знайдених об’єктів та списку класів із відповідними рівнями впевненості (confidence).

Додатково реалізовано можливість обробки відеофайлів. Користувач має змогу завантажити файл визначеного формату, після чого система виконує обробку кожного кадру та виводить результати у зручному форматі.

Ще однією функціональною можливістю є розпізнавання відеопотоку в реальному часі. Користувач має змогу ініціювати підключення до камери або потоку з дрона, після чого система розпочинає обробку кадрів із незначною затримкою та виводить розпізнані об’єкти в реальному часі.

Опис основних характеристик і особливостей програми

Система побудована з використанням технологій відкритого коду та забезпечує ефективне розпізнавання об’єктів у реальному часі завдяки моделі YOLOv11, яка оптимізована для роботи з GPU. Її інтеграція в вебзастосунок дозволяє отримувати

результати без потреби встановлення окремого програмного забезпечення на локальний пристрій.

Головною перевагою системи є її інтерактивність та простота використання. Інтерфейс дозволяє здійснювати повний цикл: від автентифікації до перегляду результатів розпізнавання. Застосування хмарної архітектури (при необхідності) дозволяє масштабувати систему та інтегрувати її у більш складні інформаційні середовища.

Розпізнавання можливе для трьох основних типів медіа: статичних зображень, відеофайлів і відеопотоку. Система демонструє стабільну роботу на тестових наборах даних і показує високі результати точності розпізнавання, навіть за умов змішаного освітлення, різної висоти польоту та динамічного фону.

Відомості про функціональні обмеження на застосування

На поточному етапі розгортання система має локальний характер: усі дані обробляються на сервері, де розміщено модель розпізнавання. Для коректної роботи необхідне стабільне підключення до Інтернету, що забезпечує взаємодію між клієнтом та сервером.

Функціонування системи обмежене розміром медіафайлів, які можуть бути оброблені – перевищення встановленого порогу може призвести до відхилення запиту. У майбутніх версіях передбачається реалізація автоматичної оптимізації завантажених даних та розширення допустимих форматів.

Для запуску вебзастосунку на стороні розробника необхідна інсталяція середовища Python, відповідних бібліотек (зокрема Django, torch, opencv-python, ultralytics) та налаштування серверного середовища (наприклад, через Gunicorn і Nginx). Клієнтська частина не потребує встановлення додаткового програмного забезпечення, достатньо стандартного веббраузера.

4.3 Аналіз контрольного прикладу

Після завершення розробки інформаційної системи ідентифікації об’єктів з безпілотних літальних апаратів було здійснено перевірку її працездатності шляхом

тестування основних функціональних сценаріїв. Метою цього розділу було оцінити коректність роботи програмного забезпечення, відповідність реалізованих функцій поставленим вимогам, а також зафіксувати результати взаємодії користувача із системою в типових умовах експлуатації.

Під час першого звернення до вебзастосунку користувач автоматично перенаправляється на сторінку входу до облікового запису (рис. 4.3). На цій сторінці передбачено два варіанти подальших дій: авторизація в разі наявності зареєстрованого облікового запису або перехід до реєстраційної форми для створення нового профілю. Такий підхід дозволяє забезпечити персоналізований доступ до функціоналу системи та зберігати історію взаємодії для кожного користувача окремо.

Для здійснення входу користувачеві необхідно заповнити відповідні поля форми, вказавши логін (що відповідає унікальному імені користувача в системі) та пароль, встановлений під час реєстрації. Після підтвердження правильності введених даних, система надає користувачеві доступ до головної сторінки, з якої починається безпосередня робота з функціоналом розпізнавання об’єктів.

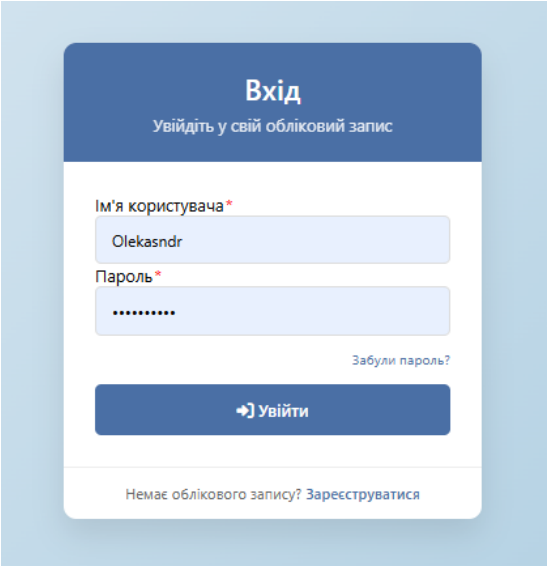

Рис. 4.3 Сторінка авторизації користувача

Наступним кроком знайомства з інформаційною системою буде знайомство з сторінкою реєстрації нового користувача, яка представлена на рисунку 4.4. Цей інтерфейс забезпечує можливість створення персонального облікового запису, що є необхідною умовою для отримання доступу до функціональних можливостей

інформаційної системи. Реєстрація дозволяє системі ідентифікувати кожного користувача, зберігати результати його розпізнавань та забезпечувати конфіденційність оброблюваних даних.

Форма реєстрації містить чотири обов’язкові поля: "Ім’я користувача", "Email адреса", "Пароль" і "Підтвердження пароля". Кожне поле валідується як на клієнтському, так і на серверному рівні, що дозволяє перевірити правильність введених даних, уникнути повторного використання імен користувачів, а також забезпечити відповідність введених паролів. Після успішного заповнення форми та натискання кнопки "Зареєструватися", обліковий запис створюється в базі даних, і користувач отримує можливість перейти до авторизації.

Додатково передбачено навігаційне посилання "Увійти", яке забезпечує зручний перехід на сторінку авторизації у випадку, якщо реєстрація вже була виконана раніше. Така структура інтерфейсу відповідає сучасним вимогам до юзабіліті, мінімізує кількість дій, необхідних для входу до системи, та сприяє ефективній взаємодії користувача з вебзастосунком.

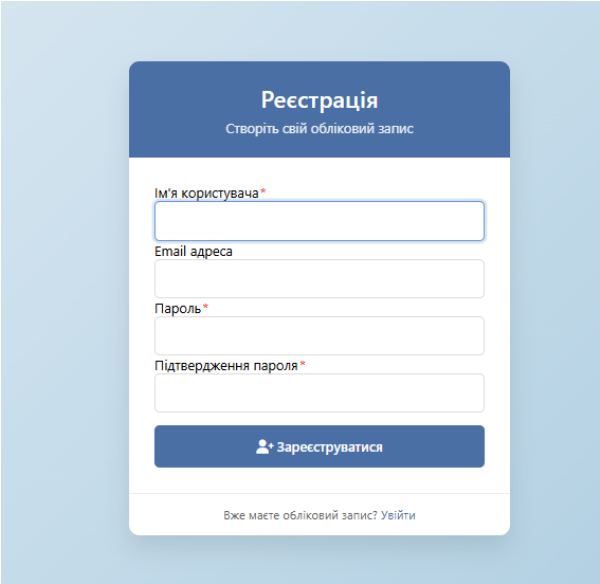

Рис. 4.4 Сторінка реєстрації користувача

Наступним кроком після успішної авторизації є перехід користувача на головну сторінку вебзастосунку, яка представлена на рисунку 4.5. Даний інтерфейс виконує роль основного навігаційного екрана системи та забезпечує зручний доступ до

ключових функцій – зокрема розпізнавання об’єктів на зображеннях безпосередньо на цій сторінці, а також можливість перейти до окремого розділу для розпізнавання відео та відеопотоків за допомогою відповідної навігаційної кнопки. Такий підхід до структурування інтерфейсу дозволяє користувачеві швидко обрати потрібний сценарій використання системи, зважаючи на тип вхідних даних.

Інтерфейс головної сторінки поділено на два функціональні блоки: «Розпізнавання зображень» та «Розпізнавання відео». Перший блок дозволяє користувачу здійснити розпізнавання об’єктів на статичних зображеннях. Завдяки інтерактивному полю, реалізованому у вигляді області для перетягування файлу або вибору його з локального пристрою, користувач має змогу швидко завантажити необхідне зображення. Після цього процес розпізнавання активується натисканням кнопки «Розпізнати об'єкти», що призводить до обробки зображення моделлю YOLOv11 та виводу результатів з анотаціями.

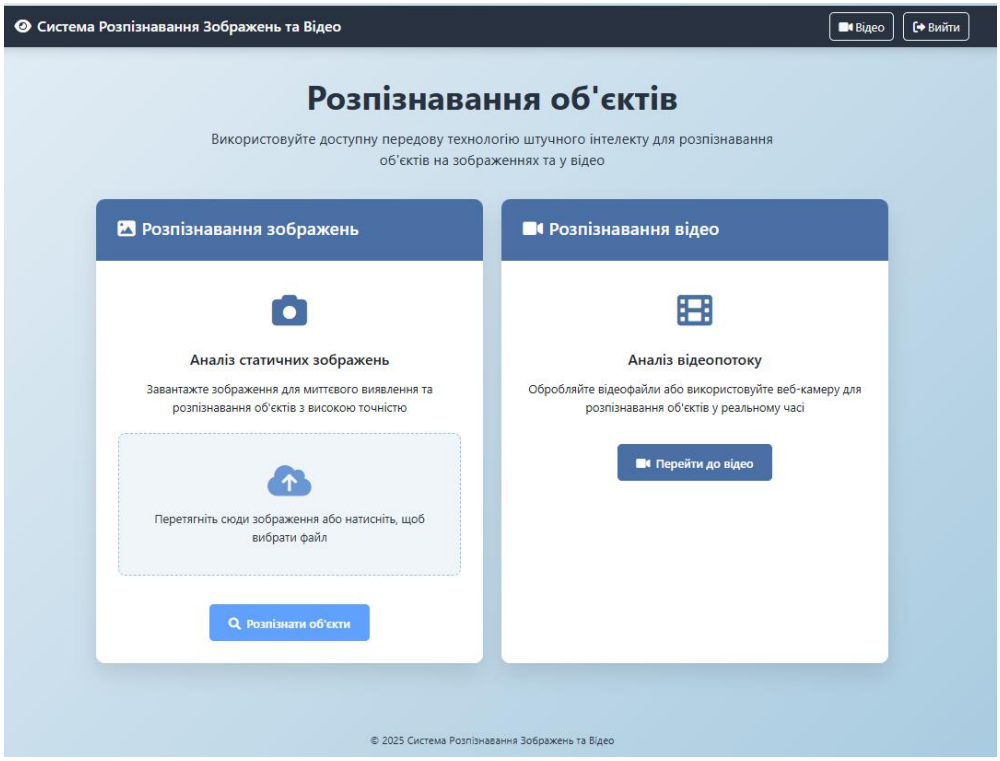

Рис. 4.5 Головна сторінка системи

Другий блок – «Розпізнавання відео» – реалізує перехід до окремої сторінки, призначеної для аналізу відеофайлів або потокового відео. Користувач може натиснути кнопку «Перейти до відео», після чого користувач опиняється на іншій

сторінці вебзастосунку. Подібна структуризація інтерфейсу забезпечує логічний поділ функціоналу системи, підвищує зручність користування та адаптована до широкого спектру практичних сценаріїв використання.

Після завантаження зображення та активації функції розпізнавання шляхом натискання кнопки «Розпізнати об’єкти», система виконує етап обробки вхідного зображення. Цей процес виконується за допомогою попередньо навченої нейромережної моделі YOLOv11.

На рисунку 4.6 представлено результат виконання розпізнавання: у верхній частині екрану відображається початкове зображення, що було завантажене користувачем. Після обробки, нижче автоматично формується нове зображення з візуалізованими результатами розпізнавання – кожен виявлений об’єкт позначається прямокутною рамкою з відповідною назвою класу та рівнем впевненості моделі (у даному прикладі – «APC, 0.88» armored personnel carrier – бронетранспортер).

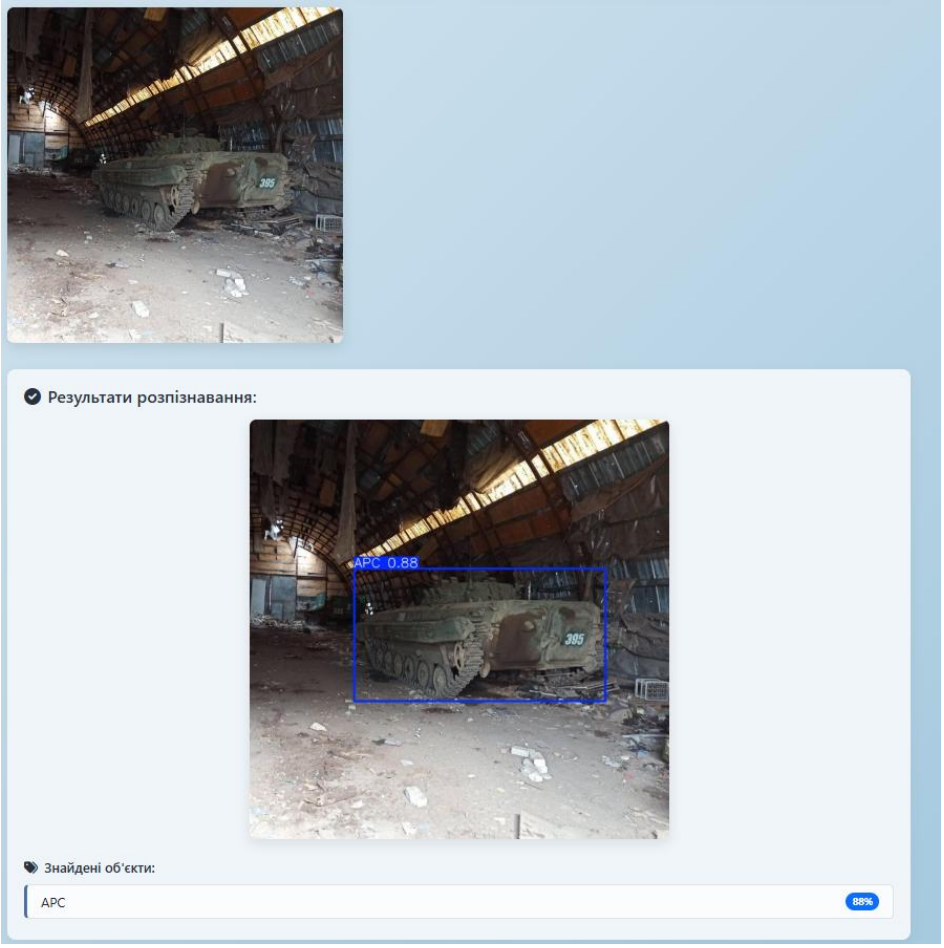

Рис. 4.6 Результат розпізнавання об’єктів на фото

Під візуалізацією результатів також наведено текстовий блок із загальним списком розпізнаних об’єктів. Це забезпечує користувача як наочною, так і структурованою інформацією про вміст зображення. Такий підхід поєднує візуальну інтерпретацію з можливістю автоматизованого аналізу виявлених цілей, що є особливо корисним у задачах ситуаційного моніторингу, розвідки або класифікації технічних засобів на місцевості.

Результати виводяться динамічно після завершення обробки, без необхідності оновлення сторінки або повторного введення даних. Це покращує інтерактивність та загальну зручність роботи з системою, а також дозволяє оперативно отримати необхідну інформацію для подальших рішень.

Перейдімо до наступної сторінки вебзастосунку котра відповідає за розпізнавання відео та відео в реальному часі. Окремою складовою функціоналу інформаційної системи є можливість розпізнавання об’єктів не лише на статичних зображеннях, але й у відеофайлах або відео в реальному часі. Для реалізації цієї задачі створено відповідну сторінку, інтерфейс якої зображено на рисунку 4.7.

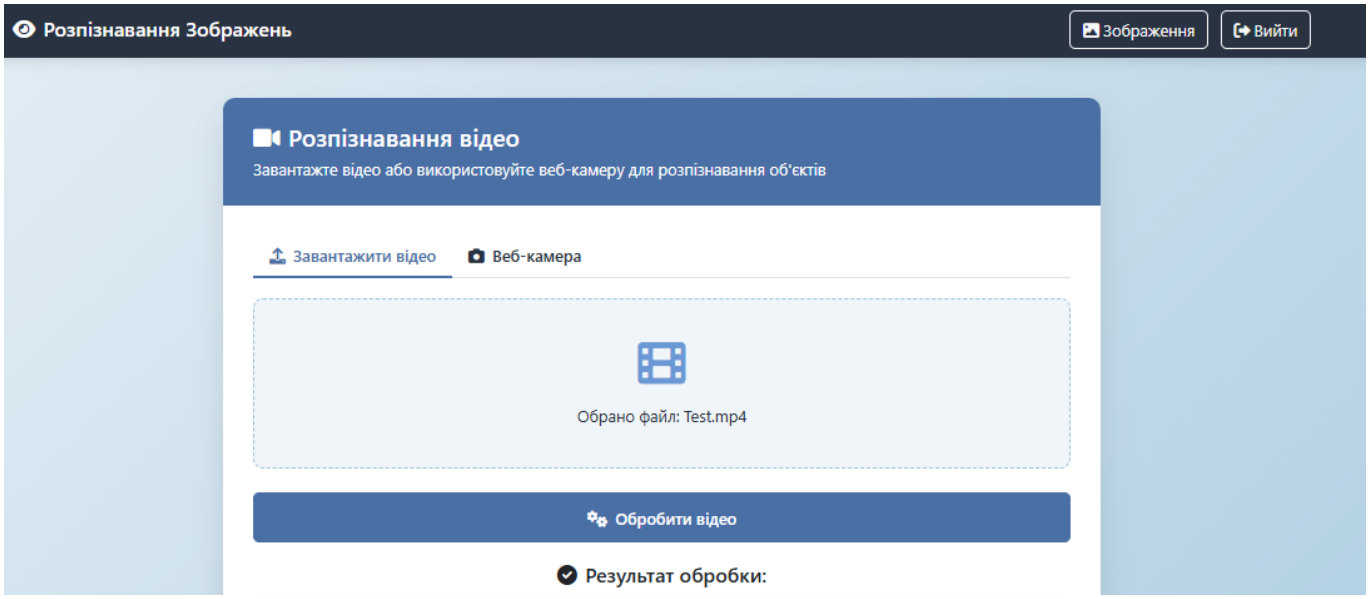

Рис. 4.7 Сторінка розпізнавання об’єктів з відео

Цей інтерфейс забезпечує два основні режими роботи:

- Завантаження відео з локального пристрою для подальшої обробки;

- Підключення до вебкамери або іншого пристрою для виконання розпізнавання у реальному часі.

Інтерактивна область завантаження дозволяє користувачу перетягнути відеофайл у відведене поле або скористатися кнопкою вибору файлу. У разі переходу до режиму «Веб-камера» активується потік із підключеного пристрою, який аналізується в реальному часі за допомогою моделі YOLOv11.

Процес ініціюється натисканням кнопки «Обробити відео», після чого система автоматично виконує обробку кожного кадру, визначаючи та класифікуючи об’єкти згідно з навченою моделлю. Такий підхід дозволяє застосовувати розроблену систему у практичних сценаріях моніторингу, відеоспостереження, а також для швидкого виявлення цілей на динамічних відеопотоках.

Після вибору відеофайлу користувач ініціює процес розпізнавання натисканням кнопки «Обробити відео», що активує серверну логіку застосунку. У цей момент система виконує серію покадрових обчислень із використанням попередньо навченої моделі комп’ютерного зору YOLOv11. Модель здійснює аналіз кожного кадру з метою виявлення на ньому об’єктів, що відповідають заздалегідь визначеним класам, та визначає рівень ймовірності для кожного розпізнаного об'єкта.

Інтерфейс результатів розпізнавання відео представлено на рисунку 4.8. У верхній частині відображено інтерактивний відеопрогравач, що містить оброблений варіант відео з нанесеними анотаціями – прямокутниками навколо виявлених об’єктів із відповідним класом (наприклад, Military Truck) та значенням довіри моделі (наприклад, 0.81). Така візуалізація є ключовим елементом зручного сприйняття результатів та дозволяє швидко інтерпретувати зміст кадрів навіть для непідготовленого користувача.

Під відео розташований функціональний блок «Статистика розпізнавання», який надає зведену інформацію про всі класи об’єктів, що були виявлені у відео. У цьому блоці виводиться перелік категорій (наприклад, Military Truck, APC) разом з відсотковим співвідношенням присутності кожного класу у загальній кількості виявлених об'єктів. Цей компонент системи виконує роль простого аналітичного

інструменту, який може використовуватись для оцінки характеру обстановки або формування оперативних висновків.

З технічної точки зору, усі результати обробки зберігаються у структурованій формі. Оброблене відео з нанесеними позначками, а також допоміжні файли (звіти, мета-дані) автоматично зберігаються в каталозі media/ кореневого каталогу Django-проєкту. Це рішення забезпечує:

- можливість повторного перегляду результатів;
- архівацію оброблених даних без втрати вмісту;
- доступ до матеріалів через адміністраторську панель або окремий функціонал.

Таким чином, реалізована сторінка забезпечує повний цикл обробки відеоданих – від завантаження користувачем до виводу структурованих результатів та збереження їх для подальшої роботи. Інтерфейс і функціонал є інтуїтивно зрозумілими, що підвищує ефективність використання системи в реальних умовах.

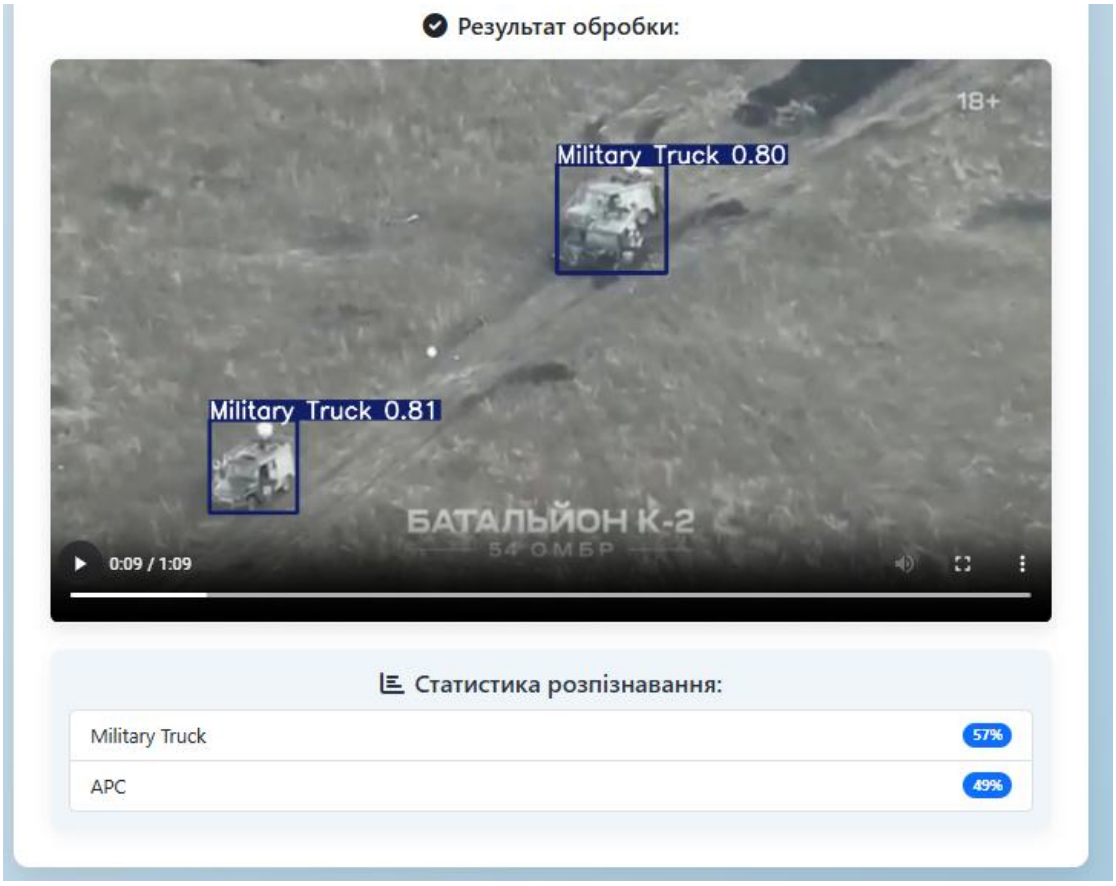

Рис. 4.8 Розпізнавання об’єктів з відео

Перейдімо до останньої сторінки вебзастосунку, котра відповідає за розпізнавання відео в реальному часі. Цей функціонал забезпечує можливість підключення до вебкамери або іншого відеопристрою, що дозволяє здійснювати обробку та аналіз потоку в реальному часі. Після натискання кнопки «Запустити камеру» система ініціює зчитування відеопотоку та автоматично передає його на обробку попередньо навченою моделлю комп'ютерного зору YOLOv11. Інтерфейс цієї сторінки продемонстровано на рисунку 4.9.

На екрані користувача відображається відеопотік із нанесеними анотаціями – прямокутниками навколо виявлених об'єктів із зазначенням їхнього класу та рівня ймовірності. Такий підхід забезпечує оперативне виявлення цілей у динамічному середовищі та може застосовуватись у сценаріях моніторингу, відеоспостереження або в умовах використання безпілотних літальних апаратів.

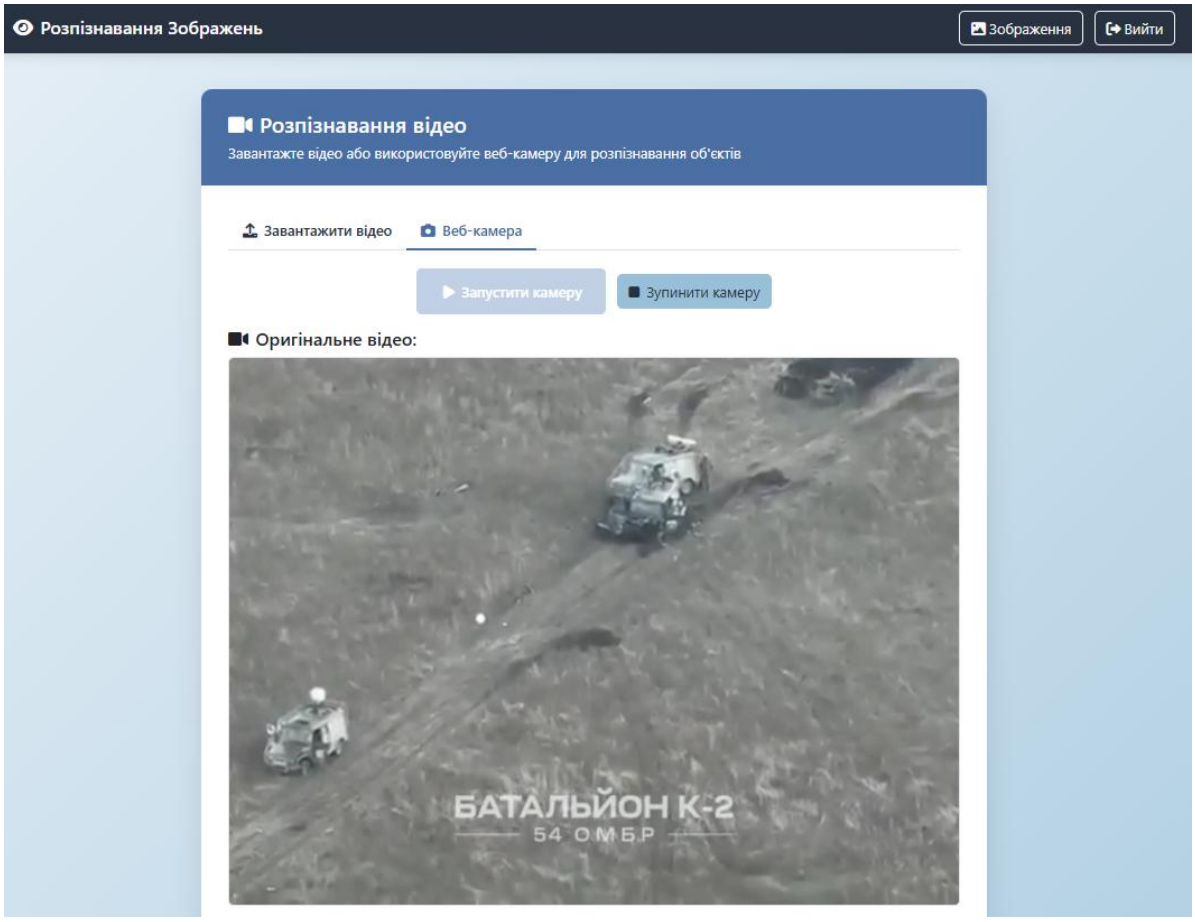

Рис. 4.9 Відображення нативного відео у реальному часі

Після увімкнення камери система розпочинає виводити результати, що продемонстровані на рисунку 4.10 зображено інтерфейс з результатами розпізнавання об’єктів у реальному часі. Відеопотік, що транслюється з вебкамери або іншого підключеного пристрою, виводиться з нанесеними анотаціями – синім прямокутником навколо виявленого об'єкта з підписом його класу (у цьому випадку – Military Truck) та рівнем довіри моделі (0.86).

Під відео розміщено інформаційний блок «Статистика розпізнавання в реальному часі», який відображає узагальнені дані про кількість розпізнаних об’єктів, частоту обробки кадрів (FPS), а також відсоткову ймовірність для кожного класу. Це дозволяє користувачу оперативно оцінити ефективність роботи системи та загальну ситуацію в кадрі.

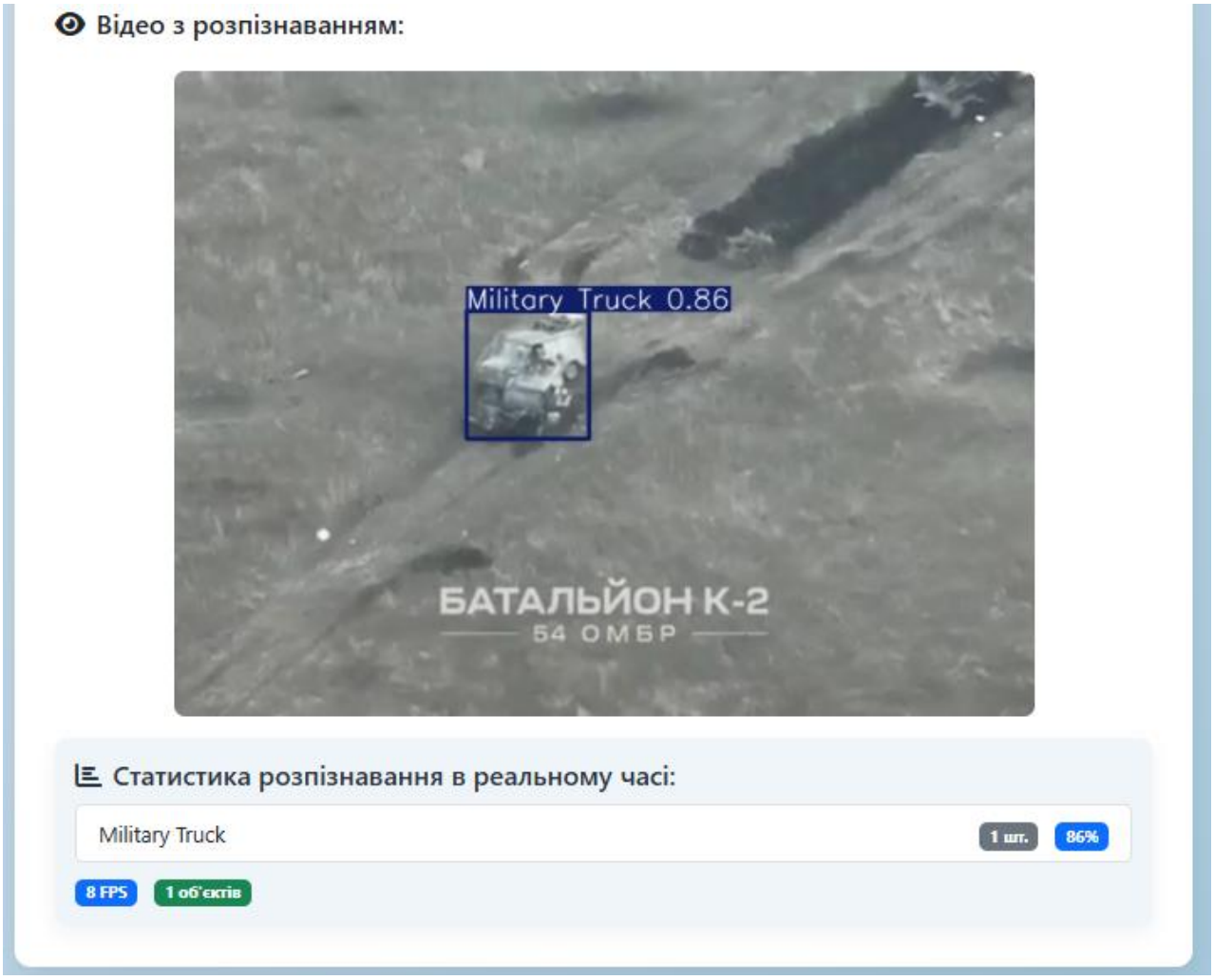

Рис. 4.10 Розпізнавання об’єктів у реальному часі з вебкамери

Висновок до четвертого розділу

У четвертому розділі бакалаврської кваліфікаційної роботи було здійснено детальний опис практичної реалізації інформаційної системи, призначеної для автоматизованої ідентифікації об’єктів на зображеннях, відеофайлах та відеопотоці, що надходять з безпілотних літальних апаратів. Реалізоване програмне рішення поєднує методи комп’ютерного зору з сучасними технологіями веброзробки, що забезпечує його адаптивність до широкого спектра прикладних завдань у сфері моніторингу, розвідки та аналізу даних з повітряних платформ.

У межах реалізації було розроблено набір модулів, кожен з яких виконує окрему функціональну роль: модуль ініціалізації та завантаження моделі YOLOv11; модулі обробки зображень, відеофайлів та відеопотоків у режимі реального часу; компоненти для асинхронної обробки даних; а також модуль вебпредставлення для інтерактивної взаємодії з користувачем. Кожен з компонентів був реалізований із дотриманням принципів модульності, повторного використання коду та масштабованості. Основою системи є глибока нейронна мережа YOLOv11, що демонструє високу точність та швидкість розпізнавання об’єктів, що є критичними параметрами для обробки великого обсягу візуальної інформації у динамічних умовах.

Результати експериментального тестування підтвердили коректність реалованої логіки, відповідність розробленого функціоналу попередньо сформульованим вимогам, а також продемонстрували високу ефективність системи в умовах обробки реальних медіаданих. Описані інтерфейсні компоненти дозволяють здійснювати повноцінну взаємодію користувача з системою – від авторизації до отримання результатів розпізнавання у візуальному та текстовому форматах.

ВИСНОВКИ

Сучасні умови воєнного протистояння та зростання ролі технологій у сфері безпеки вимагають впровадження інтелектуальних рішень для аналізу даних, що надходять з безпілотних літальних апаратів (БПЛА). Особливої важливості набуває завдання ідентифікації військової техніки, транспортних засобів і супутніх об’єктів на зображеннях та у відеопотоці, отриманих із повітря. Це одна з базових задач комп’ютерного зору, реалізація якої потребує не лише високої точності та швидкодії, але й можливості обробки даних у реальному часі. У зв’язку з цим постає необхідність створення адаптованих програмних рішень, здатних стабільно та ефективно функціонувати в умовах оперативного використання БПЛА для військової розвідки, нагляду чи аналітики.

Результатом бакалаврської кваліфікаційної роботи було розроблення інформаційної системи ідентифікації об'єктів з безпілотних літальних апаратів, що здатна обробляти зображення, відео та відеопотік з використанням сучасних методів глибокого навчання. Розроблена система поєднує засоби комп’ютерного зору та вебпрограмування, що забезпечує як ефективність розпізнавання, так і зручність доступу для користувачів.

У процесі дослідження предметної області було проведено аналіз сучасного стану технологій комп’ютерного зору, методів розпізнавання об’єктів та інструментів обробки даних з БПЛА. Розглянуто найпоширеніші архітектури глибокого навчання, зокрема YOLO, SSD, Faster R-CNN, Detectron2, та визначено переваги їх застосування в задачах реального часу. Встановлено, що модель YOLO, зокрема її оновлені версії, є найбільш придатною до розгортання в ресурсозалежних середовищах, де критичною є швидкість обробки та точність розпізнавання. Огляд аналогічних систем продемонстрував, що більшість існуючих рішень орієнтовані на окремі задачі або потребують складної інтеграції, що зумовило доцільність створення власного рішення з урахуванням конкретних вимог до зручності та адаптивності.

У другому розділі роботи було виконано системний аналіз об’єкта дослідження. Сформульовано загальну ціль створення інформаційної системи та побудовано дерево

цілей із визначенням підцілей та критеріїв успішної реалізації. За допомогою методу аналітичної ієрархії (AHP) було формалізовано структуру оцінювання ключових характеристик системи, а також встановлено пріоритети при виборі архітектури. Для структурованого уявлення про функціонування системи використано нотацію IDEF0 – побудовано контекстну діаграму, а також декомпозицію процесів до другого та третього рівнів. Це дало змогу описати логіку роботи майбутнього застосунку на концептуальному рівні, зокрема етапи автентифікації, обробки медіаданих, інтеграції з моделлю та формування результатів.

У межах розробки архітектури системи було визначено основні технічні та програмні засоби реалізації. Здійснено порівняльний аналіз мов програмування та інструментів реалізації глибокого навчання. У результаті обрано Python як основну мову програмування за рахунок її гнучкості, розвиненої екосистеми бібліотек для машинного навчання (PyTorch, OpenCV, NumPy) та зручності інтеграції з фреймворком Django, що було використано для створення вебінтерфейсу. Для розробки клієнтської частини застосовано HTML, CSS та JavaScript. Модель YOLOv11, натренована на власному датасеті, було обрано як основний інструмент розпізнавання, оскільки вона поєднує високу точність, швидкість інференсу та зручність адаптації до прикладної задачі.

У четвертому розділі здійснено повну реалізацію інформаційної системи, яка включає набір функціональних модулів: реєстрація та авторизація користувачів, завантаження зображень і відео, обробка відеопотоків у реальному часі, розпізнавання об’єктів за допомогою моделі YOLOv11 та виведення результатів розпізнавання. Крім того, реалізовано збереження історії всіх запитів користувача в базі даних. Структура бази побудована з урахуванням гнучкого зв’язку між користувачами та результатами розпізнавання, що забезпечує зручність подальшого аналізу. Система має вебінтерфейс, доступний авторизованим користувачам, та підтримує інтерактивну взаємодію через інтуїтивно зрозуміле середовище.

Результати тестування розробленої системи підтвердили правильність реалізації функціоналу. Усі основні сценарії використання – автентифікація, розпізнавання

фото, відео та відеопотоків, збереження та вивід результатів – працюють коректно відповідно до поставлених вимог. Графічний інтерфейс забезпечує швидкий доступ до функцій, а використання власної моделі YOLOv11 гарантує високу якість обробки зображень і кадрів навіть у складних умовах, що наближує систему до реального практичного застосування.

Таким чином, у межах виконання бакалаврської кваліфікаційної роботи було повністю досягнуто поставлену мету та реалізовано всі заплановані завдання. Створена інформаційна система демонструє практичну ефективність і може бути використана у прикладних задачах моніторингу, охорони об’єктів, екологічного нагляду, а також як базова платформа для подальшого розвитку. У перспективі систему можна вдосконалити через розширення функціоналу, зокрема додавання підтримки геоприв’язки об’єктів, інтеграції з хмарними платформами, мобільними додатками, а також впровадження більш глибоких методів аналітики виявлених об'єктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Szeliski, R. (2010). *Computer vision: Algorithms and applications* (Draft ed., September 3, 2010). Springer.
2. Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). *ORB: An efficient alternative to SIFT or SURF*. In ICCV 2011.
3. Rother, C., Kolmogorov, V., & Blake, A. (2004). *GrabCut – Interactive Foreground Extraction using Iterated Graph Cuts*. ACM Transactions on Graphics, 23(3), 309–314.
4. Brunelli, R. (2009). *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley.
5. Zhao, L., Yang, K., & Huang, X. (2023). "RGB-Thermal Data Fusion Approaches for Search and Rescue Operations Using UAVs". Drones, 7(2), 112-130. https://www.mdpi.com/2504-446X/7/2/112
6. Tsouros, D.C., Bibi, S., & Sarigiannidis, P.G. (2023). "Precision Agriculture: A Comprehensive Review on Framework and Applications of RGB and Multispectral UAV Imaging". Drones, 7(1), 41-59. https://www.mdpi.com/2504-446X/7/1/41
7. Caltagirone, L., Bellone, M., Svensson, L., & Wahde, M. (2019). *LIDAR–camera fusion for road detection using fully convolutional neural networks*. Robotics and Autonomous Systems, 111, 125–131. https://doi.org/10.1016/j.robot.2018.10.007
8. DroneDeploy Ground App Feature List [Електронний ресурс] // DroneDeploy Help Center. – 2024. – Режим доступу: https://help.dronedeploy.com/hc/en-us/articles/17081309471127-DroneDeploy-Ground-App-Feature-List
9. Pix4D Documentation [Електронний ресурс] // *Pix4D Support Center*. – 2025. – Режим доступу: https://support.pix4d.com/hc
10. Kespry Drone Intelligence Platform // Firmatek. – [Електронний ресурс]. – Режим доступу: https://firmatek.com/services/drone-solutions/
11. DJI. DJI Terra – Центр завантажень [Електронний ресурс]. – Режим доступу: https://www.dji.com/global/downloads/products/dji-terra#doc

12. MathWorks. (2024). *Computer Vision Toolbox documentation*. MathWorks. Retrieved May 7, 2025, from https://www.mathworks.com/help/vision/

13. Баранов, П. А., & Жучков, В. В. (2016). *Проектування інформаційних систем*. Київ: КНЕУ.

14. Шершньова З. Є. Стратегічне управління: підручник. – Київ: КНЕУ, 2004. – 221 с.

15. Гуреєв, С. П., & Кривошеїн, А. А. (2011). *Метод аналітичних ієрархій у задачах підтримки прийняття рішень*. Харків: ХНУРЕ.

16. Роженко О. В. Формування бізнес-моделі підприємства: – Кривий Ріг: ДонНУЕТ, 2018. – с. 28-29

17. International Organization for Standardization. (2015). *ISO/IEC 27017:2015 – Information technology – Security techniques – Code of practice for information security controls based on* https://www.iso.org/standard/43757.html

18. International Organization for Standardization. (2011). *ISO/IEC 25010:2011 — Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. https://www.iso.org/standard/35733.html

19. International Organization for Standardization. (2013). *ISO/IEC 27001:2013 — Information technology — Security techniques — Information security management systems* https://www.iso.org/standard/54534.html

20. Learn C++ Programming [Електронний ресурс] – Режим доступу: https://www.programiz.com/cpp-programming

21. Schildt H. C++: A Beginner's Guide. – New York: McGraw-Hill Education, 2011. – 4 p.

22. JDBC Java Database Connectivity [Електронний ресурс] – Режим доступу: https://www.oracle.com/ru/database/technologies/jdbc-migration.htm

23. Java Sound API [Електронний ресурс] – Режим доступу: https://www.oracle.com/java/technologies/java-sound-api.htm

24. How can I get started developing Java programs with the Java Development Kit (JDK)? [Електронний ресурс] – Режим доступу: https://java.com/en/download/help/develop.html#:~:text=Writing%20Java%20applets%20and%20applications,experienced%20programmers%20to%20get%20started

25. C#/.NET History Lesson [Електронний ресурс] – Режим доступу: http://jameskovacs.com/2007/09/07/cnet-history-lesson/

26. Create a Windows Forms app in Visual Studio with C# [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2019

27. The History of JavaScript: Everything You Need to Know [Електронний ресурс] – Режим доступу: https://www.springboard.com/blog/history-of-javascript/

28. Harwani B. Qt5 Python GUI Programming Cookbook. – Birmingham: Packt Publishing, 2018. – 9 p.

29. Ultralytics. YOLOv11: Key features and model details [Електронний ресурс] // Ultralytics Documentation. – 2024. – Режим доступу: https://docs.ultralytics.com/models/yolo11

30. Liu W. SSD: Single Shot MultiBox Detector [Електронний ресурс] / GitHub Repository. – 2017. – Режим доступу: https://github.com/weiliu89/caffe/tree/ssd

31. Girshick R. Faster R-CNN [Електронний ресурс] / GitHub Repository. – 2015. – Режим доступу: https://github.com/rbgirshick/py-faster-rcnn

32. Facebook AI Research. Detectron2 Documentation [Електронний ресурс]. – 2024. – Режим доступу: https://detectron2.readthedocs.io

АНОТАЦІЯ

Козак О.О., Басюк Т.М.,(керівник). Інформаційна система ідентифікації об’єктів з безпілотних літальних апаратів. Бакалаврська кваліфікаційна робота. – Національний університет «Львівська політехніка», Львів, 2025.

Розширена анотація

У сучасному світі автоматизований аналіз візуальної інформації набуває дедалі більшої важливості в широкому спектрі прикладних задач. Одним із перспективних напрямів є використання безпілотних літальних апаратів (БПЛА) для отримання зображень та відео з метою подальшої обробки. Ефективна ідентифікація об’єктів на таких медіаданих потребує застосування високопродуктивних моделей комп’ютерного зору, які забезпечують розпізнавання в реальному часі [1].

У роботі розглянуто архітектурний підхід до побудови веборієнтованої інформаційної системи, яка виконує автоматичне розпізнавання об’єктів на зображеннях, відео та потоковому відео, отриманих з БПЛА. Система реалізована мовою програмування Python із використанням фреймворку Django. Для реалізації функціоналу комп’ютерного зору використано модель YOLOv11, яка була адаптована до власного датасету [2]. Також забезпечено обробку результатів розпізнавання та виведення в інтерфейс користувача у зручному форматі.

У процесі розробки було побудовано дерево цілей, виконано декомпозицію функціональних блоків за допомогою IDEF0, проаналізовано сучасні моделі комп’ютерного зору, зокрема YOLOv5–v11, SSD, Faster R-CNN [3]. Система пройшла тестування на різних типах вхідних даних (зображення, відео, стрім), що дозволило оцінити її точність, стабільність і продуктивність.

Об’єкт дослідження – це процес автоматизованої ідентифікації об’єктів на зображеннях, відео та відеопотоці, отриманих з безпілотних літальних апаратів.

Предмет дослідження – це методи, моделі та програмні засоби комп’ютерного зору і глибокого навчання, що використовуються для побудови інформаційної системи розпізнавання об’єктів з БПЛА.

Мета дослідження – створення інформаційної системи, що поєднує сучасні алгоритми розпізнавання з доступністю вебтехнологій для обробки даних з БПЛА.

Результатом є система, здатна працювати в умовах реального часу, адаптована до прикладних задач моніторингу, охорони, агроаналізу та технічної інспекції.

Ключові слова – розпізнавання об’єктів, комп’ютерний зір, YOLOv11, вебзастосунок, глибоке навчання, безпілотні літальні апарати, відеоаналіз.

Перелік використаних джерел:

1. Szeliski, R. (2010). Computer vision: Algorithms and applications (Draft ed., September 3, 2010). Springer. https://szeliski.org/Book/
2. Ultralytics. (2024). *YOLOv11: Key features and model details*. Ultralytics Documentation. Retrieved May 7, 2025, from https://docs.ultralytics.com/models/yolo11/#key-features
3. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In *ECCV 2016*. https://doi.org/10.1007/978-3-319-46448-0_2

ANNOTATION

Kozak O.O., Basyuk T.M., (head). Information system for identifying objects from unmanned aerial vehicles. Bachelor's qualification work. - Lviv Polytechnic National University, Lviv, 2025.

Extended annotation

In today's world, automated analysis of visual information is becoming increasingly important in a wide range of applied tasks. One of the promising directions is the use of unmanned aerial vehicles (UAVs) to obtain images and videos for further processing. Effective identification of objects on such media requires the use of high-performance computer vision models that provide real-time recognition [1].

The paper considers the architectural approach to building a web-oriented information system that performs automatic recognition of objects in images, video and streaming video obtained from UAVs. The system is implemented in the Python programming language using the Django framework. To implement the functionality of computer vision, a YOLOv11 model was used, which was adapted to its own dataset [2]. Processing of the results of recognition and output to the user interface in a convenient format is also provided.

In the process of development, a tree of goals was built, decomposition of functional blocks was performed using IDEF0, modern models of computer vision were analyzed, in particular YOLOv5-v11, SSD, Faster R-CNN [3]. The system was tested on various types of input data (image, video, stream), which made it possible to evaluate its accuracy, stability and performance.

Object of the research – the process of automated identification of objects in images, video, and video streams obtained from unmanned aerial vehicles (UAVs).

Subject of the research – methods, models, and software tools of computer vision and deep learning used for the development of an object recognition information system based on UAV data.

The purpose of the study is to create an information system that combines modern recognition algorithms with the availability of web technologies for processing data from UAVs.

The result is a system capable of working in real-time conditions, adapted to the applied tasks of monitoring, protection, agricultural analysis and technical inspection.

Keywords - object recognition, computer vision, YOLOv11, web application, deep learning, unmanned aerial vehicles, video analysis.

List of sources used:

1. Szeliski, R. (2010). Computer vision: Algorithms and applications (Draft ed., September 3, 2010). Springer. https://szeliski.org/Book/
2. Ultralytics. (2024). YOLOv11: Key features and model details. Ultralytics Documentation. Retrieved May 7, 2025, from https://docs.ultralytics.com/models/yolo11/#key-features
3. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In ECCV 2016. https://doi.org/10.1007/978-3-319-46448-0_2

ДОДАТКИ

Додаток А код файлу yolo.py

```
import os
import cv2
import torch
from pathlib import Path
from tqdm import tqdm
from ultralytics import YOLO
from django.conf import settings
import threading
import queue
import time
from concurrent.futures import ThreadPoolExecutor
from threading import Event
import psutil
# Глобальні черги для обміну даними між потоками
input_queue = queue.Queue(maxsize=10)  # Черга для вхідних кадрів
output_queue = queue.Queue(maxsize=10)  # Черга для оброблених кадрів
processing_active = threading.Event()  # Флаг для контролю активності обробки
processing_active.set()  # За замовчуванням активно
# Завантаження моделі один раз при імпорті та переміщення на GPU, якщо
доступний
model = YOLO(os.path.join(settings.BASE_DIR, 'detection_app', 'best.pt'))
if not torch.cuda.is_available():
    raise RuntimeError("CUDA (GPU) не доступна. Ініціалізація моделі
заборонена без GPU.")
model.to('cuda')
# Налаштування параметрів для швидшої роботи
model.conf = 0.5  # поріг впевненості
model.iou = 0.5  # поріг IoU для NMS
model.max_det = 100  # максимальна кількість виявлень
def detect_image(image_path):
    try:
        # Читаємо зображення
        img = cv2.imread(image_path)
        if img is None:
            raise Exception(f"Не вдалося прочитати зображення: {image_path}")
        # Змінюємо розмір для швидшої обробки, якщо зображення занадто велике
        h, w = img.shape[:2]
        if max(h, w) > 1280:
            scale = 1280 / max(h, w)
            img = cv2.resize(img, (int(w * scale), int(h * scale)))
        # 1. Детекція
        results = model(img)[0]  # беремо перший результат
        # 2. Візуалізація результатів
        result_img = results.plot()
        # 3. Збереження обробленого зображення
        result_dir = os.path.join(settings.MEDIA_ROOT, 'results')
        os.makedirs(result_dir, exist_ok=True)
        result_filename = f"result_{Path(image_path).stem}.jpg"
        result_path = os.path.join(result_dir, result_filename)
        cv2.imwrite(result_path, result_img)
        return result_path, results
    except Exception as e:
```

```
        print(f"❌ Помилка під час детекції зображення: {e}")
        return None, None

def detect_video(video_path):
    try:
        # 1. Відкриваємо відео для читання
        cap = cv2.VideoCapture(video_path)
        if not cap.isOpened():
            raise Exception("Не вдалося відкрити відеофайл")
        # 2. Отримуємо параметри відео
        width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        fps = int(cap.get(cv2.CAP_PROP_FPS))
        total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
        # Перевіримо розмір відео і зменшимо його для швидшої обробки, якщо
потрібно
        target_size = 1280  # цільова висота або ширина для обробки
        resize_factor = 1.0
        if width > target_size or height > target_size:
            resize_factor = target_size / max(width, height)
            new_width = int(width * resize_factor)
            new_height = int(height * resize_factor)
            print(f"Зменшення розміру кадрів для обробки: {width}x{height} ->
{new_width}x{new_height}")
        else:
            new_width, new_height = width, height
        # 3. Готуємо тимчасовий вихідний файл
        result_dir = os.path.join(settings.MEDIA_ROOT, 'results')
        os.makedirs(result_dir, exist_ok=True)
        output_filename =
f"result_video_{Path(video_path).stem}_{int(time.time())}.mp4"
        output_path = os.path.join(result_dir, output_filename)
        # 4. Налаштовуємо відеозапис
        fourcc = cv2.VideoWriter_fourcc(*'avc1')  # або 'avc1' для h264
        out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))
        # 5. Створюємо черги та події для міжпотокової комунікації
        frame_queue = queue.Queue(maxsize=30)  # черга для зчитаних кадрів
        result_queue = queue.Queue(maxsize=30)  # черга для оброблених кадрів
        stop_event = Event()  # подія для зупинки потоків

        # Рахуємо кадри
        processed_frames = 0
        all_detections = []
        # 6. Функція для зчитування кадрів у окремому потоці
        def read_frames():
            nonlocal processed_frames
            frame_count = 0
            try:
                while cap.isOpened() and not stop_event.is_set():
                    ret, frame = cap.read()
                    if not ret:
                        break
                    frame_count += 1
                    # Зменшуємо розмір кадру для швидшої обробки
                    if resize_factor < 1.0:
```

```
                        resized_frame = cv2.resize(frame, (new_width,
new_height))
                        frame_queue.put((resized_frame, frame, frame_count))
                    else:
                        frame_queue.put((frame, None, frame_count))
                    processed_frames += 1
            except Exception as e:
                print(f"Помилка в потоці зчитування кадрів: {e}")
            finally:
                # Сигналізуємо про завершення
                frame_queue.put((None, None, -1))
                print("Зчитування кадрів завершено")
        # 7. Функція для обробки кадрів у окремому потоці
        def process_frames():
            try:
                batch_size = 8  # розмір батчу для обробки
                batch_frames = []
                batch_originals = []
                batch_indices = []
                while not stop_event.is_set():
                    # Отримуємо кадр з черги
                    frame, original, frame_idx = frame_queue.get()
                    # Перевіряємо на сигнал завершення
                    if frame is None:
                        break
                    # Додаємо кадр до батчу
                    batch_frames.append(frame)
                    batch_originals.append(original if original is not None
else frame)
                    batch_indices.append(frame_idx)
                    # Якщо зібрали батч або це останній кадр
                    if len(batch_frames) >= batch_size or frame_queue.qsize()
== 0:
                        # Обробляємо батч разом
                        results = model(batch_frames, verbose=False)
                        # Обробляємо кожен результат
                        for i, (res, orig_frame, frame_idx) in
enumerate(zip(results, batch_originals, batch_indices)):
                            # Візуалізуємо результати на оригінальному кадрі
                            result_frame = res.plot()
                            # Якщо був ресайз, повертаємо до оригінального
розміру
                            if resize_factor < 1.0:
                                result_frame = cv2.resize(result_frame,
(width, height))
                            # Збираємо інформацію про детекції
                            frame_detections = []
                            for box in res.boxes:
                                cls_id = int(box.cls.item())
                                conf = float(box.conf.item())
                                label = res.names[cls_id]
                                frame_detections.append({
                                    'label': label,
                                    'confidence': conf,
                                    'frame': frame_idx
                                })
```

```
                            # Додаємо до результатів
                            result_queue.put((result_frame, frame_detections,
frame_idx))
                        # Очищаємо батч
                        batch_frames = []
                        batch_originals = []
                        batch_indices = []
            except Exception as e:
                print(f"Помилка в потоці обробки кадрів: {e}")
            finally:
                # Сигналізуємо про завершення
                result_queue.put((None, None, -1))
                print("Обробка кадрів завершено")
        # 8. Функція для запису результатів
        def write_results():
            nonlocal all_detections
            try:
                results_count = 0
                frames_buffer = {}  # буфер для збереження порядку кадрів
                next_frame_idx = 1  # наступний індекс кадру для запису
                progress_bar = tqdm(total=total_frames, desc="Обробка відео")
                while not stop_event.is_set():
                    # Отримуємо результат
                    result_frame, detections, frame_idx = result_queue.get()
                    # Перевіряємо на сигнал завершення
                    if result_frame is None and frame_idx == -1:
                        break
                    # Додаємо в буфер
                    frames_buffer[frame_idx] = (result_frame, detections)
                    # Записуємо кадри по порядку
                    while next_frame_idx in frames_buffer:
                        frame, dets = frames_buffer.pop(next_frame_idx)
                        out.write(frame)
                        if dets is not None:
                            all_detections.append(dets)
                        progress_bar.update(1)
                        next_frame_idx += 1
                        results_count += 1
                # Записуємо залишок кадрів
                for idx in sorted(frames_buffer.keys()):
                    frame, dets = frames_buffer[idx]
                    out.write(frame)
                    if dets is not None:
                        all_detections.append(dets)
                    progress_bar.update(1)
                    results_count += 1
                progress_bar.close()
                print(f"Записано {results_count} кадрів")
            except Exception as e:
                print(f"Помилка в потоці запису результатів: {e}")
        # 9. Запускаємо потоки через ThreadPoolExecutor
        with ThreadPoolExecutor(max_workers=3) as executor:
            read_thread = executor.submit(read_frames)
            process_thread = executor.submit(process_frames)
            write_thread = executor.submit(write_results)
            try:
```

```
                # Чекаємо завершення всіх потоків
                read_thread.result()
                process_thread.result()
                write_thread.result()
            except Exception as e:
                print(f"Помилка при виконанні потоків: {e}")
                stop_event.set()  # зупиняємо всі потоки у випадку помилки
        # 10. Закриваємо ресурси
        cap.release()
        out.release()
        print(f"Обробку відео завершено! Знайдено {len(all_detections)}
детекцій")
        return output_path, all_detections
    except Exception as e:
        print(f"❌ Помилка під час детекції відео: {e}")
        import traceback
        traceback.print_exc()
        return None, []
# Глобальні змінні для реалізації багатопотоковості
frame_queue = queue.Queue(maxsize=20)  # Черга для необроблених кадрів
result_queue = queue.Queue(maxsize=20)  # Черга для оброблених результатів
processing_thread = None
thread_active = threading.Event()
thread_active.set()  # За замовчуванням активний
def process_frames_worker():
    """Потік для батчової обробки кадрів з веб-камери"""
    print("🚀 Потік обробки кадрів запущено (batch mode)")
    device = 'cuda' if torch.cuda.is_available() else 'cpu'
    if device == 'cuda' and str(model.device) != 'cuda':
        model.to(device)
        print(f"Модель переміщено на {device}")
    batch_size = 8  # Скільки кадрів обробляти за раз
    frames_batch = []
    ids_batch = []
    sizes_batch = []
    while thread_active.is_set():
        try:
            # Отримуємо кадр з черги
            frame_data = frame_queue.get(timeout=0.2)
            if frame_data is None:
                continue
            frame, frame_id, original_size = frame_data
            frames_batch.append(frame)
            ids_batch.append(frame_id)
            sizes_batch.append(original_size)
            # Якщо набрали повний батч або черга пуста
            if len(frames_batch) >= batch_size or frame_queue.empty():
                with torch.no_grad():
                    results = model(frames_batch, verbose=False)

                # Обробляємо кожен результат окремо
                for res, frame_id, original_size in zip(results, ids_batch,
sizes_batch):
                    result_frame = res.plot()
                    # Повертаємо до оригінального розміру
```

```
                    h, w = result_frame.shape[:2]
                    orig_h, orig_w = original_size
                    if h != orig_h or w != orig_w:
                        result_frame = cv2.resize(result_frame, (orig_w,
orig_h))
                    detections = []
                    for box in res.boxes:
                        cls_id = int(box.cls.item())
                        conf = float(box.conf.item())
                        label = res.names[cls_id]
                        bbox = box.xyxy[0].tolist()
                        # Масштабуємо координати назад
                        if h != orig_h or w != orig_w:
                            scale_x = orig_w / w
                            scale_y = orig_h / h
                            bbox[0] *= scale_x
                            bbox[1] *= scale_y
                            bbox[2] *= scale_x
                            bbox[3] *= scale_y
                        detections.append({
                            'label': label,
                            'confidence': conf,
                            'bbox': bbox
                        })
                    if not result_queue.full():
                        result_queue.put((result_frame, detections,
frame_id))
                    else:
                        try:
                            result_queue.get_nowait()
                        except queue.Empty:
                            pass
                        result_queue.put((result_frame, detections,
frame_id))
                frames_batch = []
                ids_batch = []
                sizes_batch = []
        except queue.Empty:
            continue
        except Exception as e:
            print(f"❌ Помилка в батчовій обробці: {e}")
            import traceback
            traceback.print_exc()
    print("🛑 Потік обробки кадрів завершено")
    if device == 'cuda':
        torch.cuda.empty_cache()
def detect_webcam_frame(frame):
    """Розпізнавання об'єктів на кадрі з веб-камери з використанням
багатопотоковості"""
    try:
        # Перевіряємо стан потоку і перезапускаємо, якщо потрібно
        if processing_thread is None or not processing_thread.is_alive():
            initialize_processing_thread()
        # Отримуємо оригінальні розміри
        h, w = frame.shape[:2]
```

```
        original_size = (h, w)
        frame_id = int(time.time() * 1000)
        # Зменшуємо розмір для швидшого розпізнавання
        if max(h, w) > 960:
            scale = 960 / max(h, w)
            resized_frame = cv2.resize(frame, (int(w * scale), int(h *
scale)))
        else:
            resized_frame = frame.copy()  # Створюємо копію для безпеки
        # Додаємо кадр до черги для обробки, якщо є місце
        if not frame_queue.full():
            frame_queue.put((resized_frame, frame_id, original_size))
        else:
            # Якщо черга заповнена, пропускаємо кадр
            print("⚠️ Черга обробки заповнена, кадр пропущено")
        # Перевіряємо, чи є результати в черзі
        try:
            result_frame, detections, _ = result_queue.get_nowait()
            print(f"🔍 CPU: {psutil.cpu_percent(interval=0.1)}%, Об'єктів:
{len(detections)}")
            return result_frame, detections
        except queue.Empty:
            # Якщо результатів немає, використовуємо оригінальний кадр
            print("🔄 Очікування результатів обробки...")
            return frame, []
    except Exception as e:
        print(f"❌ Помилка під час детекції кадру веб-камери: {e}")
        import traceback
        traceback.print_exc()
        # У випадку помилки повертаємо оригінальний кадр
        return frame, []
```