НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і

спеціалізованих комп'ютерних систем

«На правах рукопису»
УДК 004.8

До захисту допущено:

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____2025р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Системне програмування та спеціалізовані комп'ютерні системи»

зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Метод та модель автономного управління БПЛА»

Виконав:
студент ІІ курсу, групи КВ-31мн
Шевченко Олександр Валерійович _____

Науковий керівник:
Професор кафедри СПіСКС, д.т.н., професор,
Терейковський Ігор Анатолійович _____

Рецензент:

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і

спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-наукова програма **«Системне програмування та**

спеціалізовані комп'ютерн системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Шевченко Олександр Валерійович

1. Тема дисертації «Метод та модель автономного управління БПЛА», науковий керівник дисертації Терейковський Ігор Анатолійович, д.т.н., професор, затверджені наказом по університету від «21» березня 2025 р. №1219-с

2. Термін подання студентом дисертації: 15 травня 2025 р.

3. Об'єкт дослідження: процес управління безпілотним пристроєм

4. Вихідні дані: нейромережева модель та система управління безпілотним пристроєм

5. Перелік завдань, які потрібно розробити:

1) Аналіз існуючих рішень та обґрунтування теми дипломного проєкту;
2) Інструменти та методи розробки;
3) Аналіз та розробка програмного продукту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: презентація

7. Орієнтовний перелік публікацій:
   - XVII Наукова конференція магістрантів та аспірантів ПМК-2024, Київ, 24 листопада 2024.
   - 2-га Міжнародна науково-практична конференція «Інформаційні системи та технології: результати і перспективи (IST 2025)», Київ, 5 березня 2025
   - V Міжнародна науково-практична конференція «EUROPEAN CONGRESS OF SCIENTIFIC DISCOVERY», Мадрид (Іспанія), 28–30 квітня 2025
   - Наукова стаття у журналі «Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки», Том 36 (75), № 3, 2025 (категорія «Б» згідно з переліком МОН України)

8. Дата видачі завдання 25.10.2023

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Вивчення літератури за тематикою МД	15.01.2025	
2	Розроблення та узгодження технічного завдання	25.01.2025	
3	Аналіз існуючих рішень	01.02.2025	
4	Підготовка матеріалів першого розділу	10.02.2025	
5	Підготовка матеріалів другого розділу	01.03.2025	
6	Підготовка матеріалів третього розділу	20.03.2025	
7	Підготовка матеріалів четвертого розділу	10.04.2025	
8	Підготовка графічної частини МД	15.04.2025	
9	Оформлення документації МД	22.04.2025	
10	Попередній розгляд МД на кафедрі	25.04.2025	

Студент Олександр ШЕВЧЕНКО

Науковий керівник Ігор ТЕРЕЙКОВСЬКИЙ

РЕФЕРАТ

Актуальність теми. З огляду на зростаючу роль безпілотних літальних апаратів у військових, розвідувальних та гуманітарних місіях, питання розробки автономних систем керування, здатних до незалежного сприйняття навколишнього середовища та ухвалення рішень у реальному часі, є актуальним. Класичні методи не забезпечують достатньої гнучкості та точності в складних динамічних умовах. Застосування нейромережевих моделей детекції та трекінгу цілей відкриває нові можливості для створення інтелектуальних систем управління БПЛА з підвищеним ступенем автономності.

Об'єктом дослідження є процес автономного керування безпілотним літальним апаратом у задачах виявлення та супроводу об'єктів.

Предметом дослідження є методи та програмні моделі комп'ютерного зору на основі глибокого навчання для виявлення і трекінгу цілей у реальному часі на борту БПЛА.

Метою роботи є розробка та дослідження нейромережевої моделі, що дозволяє реалізувати автономне виявлення та трекінг об'єктів з борту БПЛА в режимі реального часу з мінімальною затримкою та високою точністю.

Наукова новизна полягає в наступному:

Удосконалено архітектуру нейромережевого модуля виявлення та трекінгу цілей, що на відміну від відомих, за рахунок об'єднання YOLOv8 і DeepSORT та оптимізації моделі з використанням ONNX і TensorRT, дозволяє досягти роботи в реальному часі на платформах з обмеженими ресурсами.

Запропоновано механізм попередньої обробки відеопотоку та динамічної буферизації, адаптований до умов обмеженого каналу передачі даних та апаратної продуктивності, що дозволило забезпечити стабільність виявлення та супроводу цілей без втрати точності.

Практична цінність отриманих результатів полягає у можливості впровадження розробленої системи автономного виявлення та трекінгу цілей у тактичних БПЛА, які працюють у реальному бойовому середовищі або в умовах радіозаглушень.

Апробація роботи. Основні положення роботи були представлені на XVII Науковій конференції магістрантів та аспірантів ПМК-2024 (м. Київ, 24 листопада 2024 р.), 2-й Міжнародній науково-практичній конференції «Інформаційні системи та технології: результати і перспективи (IST 2025)» (м. Київ, 5 березня 2025 р.), V Міжнародній науково-практичній конференції «European Congress of Scientific Discovery» (м. Мадрид, Іспанія, 28–30 квітня 2025 р). Крім того, матеріали дисертаційного дослідження опубліковані у науковому журналі категорії «Б» – Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки, т. 36(75), №3, 2025.

Структура та обсяг. Магістерська дисертація складається зі вступу, п’яти розділів, висновків, переліку використаних джерел, додатків. У вступі обґрунтовано актуальність теми, сформульовано мету, завдання, наукову новизну. У першому розділі проведено аналіз існуючих методів автономного керування БПЛА. У другому – описано математичне забезпечення та вибір архітектури. У третьому – змодельовано логіку функціонування системи. У четвертому – реалізовано програмне забезпечення та проведено тестування. У п’ятому – систематизовано експериментальні результати, графіки, таблиці та зроблено аналітичні висновки. Робота викладена на 97 сторінках, містить 47 рисунки, 1 таблию, 7 додатки.

Ключові слова: безпілотник, YOLOv8, DeepSORT, трекінг, комп’ютерний зір, нейронна мережа, TensorRT, автономне керування, відеоаналіз.

ABSTRACT

Relevance of the topic. With the increasing role of unmanned aerial vehicles (UAVs) in military, reconnaissance, and humanitarian missions, the development of autonomous control systems capable of perceiving the environment and making real-time decisions has become critically important. Classical control methods do not provide sufficient flexibility or accuracy in dynamic environments. In this context, the application of deep learning-based object detection and tracking models opens new opportunities for the creation of intelligent UAV control systems with a high degree of autonomy.

The object of the study is the process of autonomous control of a UAV in tasks of object detection and tracking.

The subject of the study is the methods and software models of computer vision based on deep neural networks for real-time detection and tracking onboard UAVs.

The aim of the thesis is to design and evaluate a neural network-based system that enables autonomous object detection and tracking from a UAV in real time with high precision and minimal latency.

Scientific novelty includes the following:

1. An improved architecture of the detection and tracking module was developed. In contrast to existing solutions, the integration of YOLOv8 and DeepSORT with ONNX and TensorRT optimization enables real-time performance on resource-constrained embedded platforms.
2. A preprocessing and buffering mechanism for the video stream was proposed, adapted to the limited bandwidth and computational capabilities of edge devices, which ensures stable object detection and tracking without accuracy degradation.

The practical value of the obtained results lies in the ability to implement the developed autonomous detection and tracking system in tactical UAVs

deployed in combat or GNSS-jammed environments. The results can also be used for swarm UAVs, autonomous patrolling platforms, and civil monitoring systems.

Thesis presentation. The main ideas and results of this work were presented at the XVII Scientific Conference of Master's and Postgraduate Students PMK-2024 (Kyiv, November 24, 2024), 2nd International Scientific and Practical Conference "Information Systems and Technologies: Results and Prospects (IST 2025)" (Kyiv, March 5, 2025), V International Scientific and Practical Conference "European Congress of Scientific Discovery" (Madrid, Spain, April 28–30, 2025). In addition, the results of the thesis have been published in a scientific journal included in the List of Scientific Professional Publications of Ukraine (Category "B"): Scientific Notes of the V.I. Vernadsky Taurida National University. Series: Technical Sciences, Vol. 36 (75), No. 3, 2025.

Structure and volume. The master's thesis consists of an introduction, five chapters, conclusions, a list of references, and appendices. The introduction outlines the problem's relevance, objectives, research tasks, and scientific novelty. Chapter one presents a review of existing UAV control methods. Chapter two describes the mathematical background and architecture selection. Chapter three covers system modeling and logic. Chapter four includes the implementation of the software and testing. Chapter five summarizes experimental results, graphs, tables, and analytical conclusions. The thesis comprises 97 pages, includes 47 figures, 1 tables, and 7 appendices.

Keywords: unmanned aerial vehicle, YOLOv8, DeepSORT, tracking, computer vision, neural network, TensorRT, autonomous control, video analytics.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БПЛА — безпілотний літальний апарат

ШІ – Штучний інтелект

ByteTrack — трекер з двоетапною логікою обробки детекцій

CNN — Convolutional Neural Network, згорткова нейронна мережа

DeepSORT — Deep Simple Online and Realtime Tracking, трекер з використанням векторів ознак

F1-score — гармонічне середнє між Precision і Recall

FP16/FP32/INT8 — формати знижених розрядностей представлення чисел (Half/Single Precision/Integer)

FPS — Frames Per Second, кількість кадрів на секунду

FSM — Finite State Machine, скінченний автомат керування

IMU — Inertial Measurement Unit, інерціальна вимірювальна система

Jetson Nano / Xavier NX — платформи NVIDIA для edge-обчислень

Kalman Filter — фільтр Калмана, математичний інструмент для прогнозування стану системи

mIoU — Mean Intersection over Union, середнє перекриття по класах

ONNX — Open Neural Network Exchange, формат для уніфікованого представлення нейромереж

PyTorch — фреймворк глибокого навчання

ReID — Re-identification, повторна ідентифікація об’єктів за ознаками

ROI — Region of Interest, область інтересу на зображенні

SORT — Simple Online and Realtime Tracking, базовий алгоритм трекінгу

TensorRT — програмна платформа NVIDIA для високошвидкісного інференсу

YOLO — You Only Look Once, модель детекції об’єктів у реальному часі

YOLOv8 — восьма версія архітектури YOLO з удосконаленою структурою та точністю

ВСТУП

Сучасні виклики, пов’язані з безпекою, обороною та інтелектуалізацією технічних систем, зумовлюють стрімкий розвиток безпілотних літальних апаратів (БПЛА). У контексті військового застосування потреба в таких автономних системах є критичною, адже це забезпечує оперативність, знижує ризики для персоналу та підвищує ефективність виконання розвідувальних і бойових задач. Метою цієї роботи є розробка ефективного методу та моделі автономного управління БПЛА, яка реалізує функції виявлення та трекінгу військових об'єктів у реальному часі із застосуванням нейромережевих технологій. Для досягнення цієї мети було поставлено наступні завдання: дослідити існуючі методи керування БПЛА, реалізувати та протестувати нейромережеву модель, оцінити її ефективність в умовах обмежених обчислювальних ресурсів. Об’єктом дослідження є процес автономного керування безпілотними літальними апаратами в умовах змінного середовища. Предметом дослідження є нейромережеві моделі, алгоритми комп’ютерного зору та методи інтеграції систем виявлення і трекінгу цілей для забезпечення стабільної автономної навігації та супроводу об'єктів. Розроблена модель дозволяє значно підвищити ефективність виявлення та супроводу об'єктів у реальному часі на борту БПЛА з обмеженими обчислювальними ресурсами.

Наукова новизна роботи полягає у поєднанні передових моделей глибокого навчання YOLOv8 та DeepSORT в одному нейромережевому модулі для автономного виявлення та трекінгу об'єктів в реальному часі. Запропоновано метод підготовки тренувальної вибірки з урахуванням специфіки військових об'єктів, а також реалізовано аугментаційні стратегії, що підвищують стійкість до змін освітлення та ракурсів. Розроблена система забезпечує високу точність (90.1%) та стабільну роботу на малопотужному обладнанні.

1. АНАЛІЗ ВІДОМИХ РІШЕНЬ ЗА ТЕМАТИКОЮ ДИПЛОМУ

1.1. Постановка задачі

Безпілотні літальні апарати (БПЛА) посідають важливе місце в сучасних технологічних системах, особливо у військовій справі, розвідці, моніторингу інфраструктури та забезпеченні безпеки. Їхнє призначення полягає в автономному або дистанційному виконанні завдань, що передбачають збір, обробку та передачу даних, виявлення цілей, навігацію та супровід об’єктів. Основною перевагою БПЛА є можливість діяти в умовах, небезпечних або недоступних для людини, а також виконання місій із високою точністю та оперативністю.

Автономне керування БПЛА відіграє ключову роль у підвищенні ефективності таких систем. Воно дозволяє апарату діяти незалежно від оператора, приймати рішення на основі сенсорних даних, аналізувати обстановку та адаптувати свої дії відповідно до змін навколишнього середовища. Це особливо важливо в умовах бою, де затримка сигналу, втрата зв’язку чи перешкоди можуть унеможливити дистанційне управління.

Для успішної реалізації автономного управління необхідно враховувати низку вимог. Система має забезпечувати стабільну навігацію, точне розпізнавання та відстеження об’єктів, своєчасну реакцію на зміни ситуації, а також здатність працювати на обмежених апаратних ресурсах. Окрім того, важливою є здатність до адаптації до нових умов без потреби в донавчанні або зовнішньому втручанні. У процесі розвитку безпілотних літальних апаратів важливою є класифікація систем за рівнем автономності, що дозволяє визначити потенціал кожного рівня у контексті бойових задач. На рисунку 1.1 представлено чотири основні рівні автономного управління

БПЛА — від ручного до повністю автономного режиму, що наочно демонструє еволюцію можливостей таких систем в умовах реального застосування.

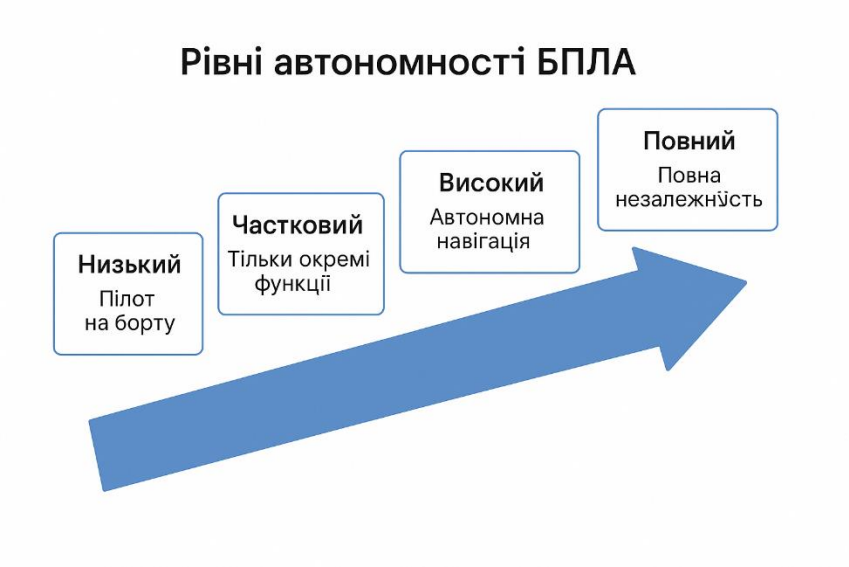


Рисунок 1.1 – Рівні автономності БПЛА

Як видно з рисунка 1.1, поступовий перехід до вищих рівнів автономності супроводжується зростанням складності алгоритмів управління, а також вимог до надійності систем розпізнавання та ухвалення рішень. Класифікація БПЛА за рівнем автономності дозволяє виділити кілька категорій — від повністю керованих оператором до повністю автономних. На початкових рівнях автономності БПЛА виконують лише окремі функції самостійно, наприклад стабілізацію польоту або автоматичне повернення на базу. Вищі рівні передбачають повну незалежність — від виявлення цілей до ухвалення рішень про дії. Однак, автономне керування пов’язане з низкою викликів. БПЛА мають працювати в умовах обмеженої видимості, за наявності перешкод, змінної геометрії цілі, під впливом радіоелектронної боротьби або в режимах, де втрата

навігаційного сигналу є ймовірною. Системи повинні бути стійкими до таких факторів та забезпечувати надійність функціонування навіть у критичних сценаріях.

Типові сценарії використання автономних БПЛА включають патрулювання, пошуково-рятувальні операції, супровід конвоїв, моніторинг ліній фронту, виявлення і супровід рухомих цілей. У кожному з них автономність дає змогу досягти оперативності, непомітності та мінімізації людського фактору. У таких умовах особливо цінним є використання інтелектуальних алгоритмів аналізу відео, розпізнавання об’єктів та прогнозування їхньої поведінки, що і визначає актуальність створення систем на основі глибокого навчання.

1.2. Аналіз відомих рішень в області автономного управління БПЛА

Методи управління безпілотними літальними апаратами постійно еволюціонують у відповідь на нові виклики, серед яких — зростаюча складність середовища, необхідність роботи в реальному часі, обмеження по енергоспоживанню та підвищені вимоги до точності й адаптивності [1, 2, 3]. Існуючі рішення, які застосовуються для автономного або напівавтономного керування БПЛА, можна умовно поділити на три великі класи: традиційні алгоритми, реактивні моделі та інтелектуальні системи.

Традиційні підходи включають алгоритми жорстко запрограмованої логіки, такі як пропорційно-інтегрально-диференціальне (PID) регулювання, а також навігаційні схеми, що базуються на GPS, інерціальних модулях (IMU) та попередньо заданих маршрутах. Вони забезпечують надійну стабілізацію та контроль, проте мають обмежену здатність до адаптації, що знижує їхню ефективність у складних та змінних умовах.

Реактивні моделі ґрунтуються на принципі прямого реагування на вхідні сенсорні дані. До них відносяться системи ухилення від перешкод, трекінг рухомих цілей за допомогою оптичного потоку або аналізу поточної сцени. Перевагою таких систем є швидкість реагування, однак вони рідко забезпечують стратегічне планування або глибоке розуміння ситуації, через що мають обмеження в контексті довготривалих місій або складних сценаріїв з багатьма невизначеними факторами.

Інтелектуальні методи управління стали можливими завдяки появі глибоких нейронних мереж, алгоритмів глибинного навчання та збільшенню обчислювальної потужності компактних пристроїв. Зокрема, архітектури на основі YOLO (You Only Look Once) [4] забезпечують високошвидкісне виявлення об'єктів у відеопотоці, тоді як DeepSORT [5] дозволяє надійно супроводжувати ці об’єкти в реальному часі. Залежно від складності задач та ступеня автономності, методи керування поділяються на традиційні та інтелектуальні. Рисунок 1.2 відображає цю класифікацію, де особливу увагу приділено методам, що використовуються у комп’ютерному зорі та адаптивному плануванні, які є актуальними для бойових умов.

Рисунок 1.2 – Класифікація методів керування БПЛА

На рисунку 1.2 зображено класифікацію сучасних методів управління БПЛА за підходом до обробки інформації та ступенем автономності. Згідно з представленою схемою, традиційні та реактивні методи поступаються інтелектуальним рішенням, особливо у динамічних умовах або при наявності непередбачуваних ситуацій. Інтелектуальні моделі демонструють вищу адаптивність, гнучкість та здатність до роботи в умовах часткової втрати даних, шумів або спотворень.

Попри очевидні переваги новітніх систем, важливим залишається питання їхньої інтеграції з уже існуючими платформами. У багатьох випадках ефективним є гібридний підхід, що поєднує класичні PID-регулятори для стабілізації та інтелектуальні модулі для вищого рівня управління. Це дозволяє забезпечити баланс між швидкодією, точністю та ресурсозалежністю.

Таким чином, аналіз існуючих методів управління БПЛА свідчить про необхідність поєднання кращих практик класичного управління з можливостями, які відкривають сучасні технології глибокого навчання. Подальший розвиток полягає у створенні адаптивних, обчислювально ефективних та масштабованих рішень, що здатні працювати в реальних польових умовах із високим ступенем автономності.

Класичні методи керування безпілотними літальними апаратами залишаються актуальними попри стрімкий розвиток інтелектуальних систем. Їхня стабільність, обґрунтованість і простота реалізації дозволяють використовувати їх у багатьох практичних сценаріях, особливо коли мова йде про стабілізацію, регулювання положення у просторі та виконання простих повторюваних маневрів. Найбільш поширеними є PID-регулятори,

методи прогнозного управління (Model Predictive Control, MPC) [2] та системи, побудовані на основі нечіткої логіки (fuzzy logic) [3].

PID-регулятори забезпечують стабільну реакцію системи на відхилення від заданого значення. Принцип дії полягає у зваженому сумуванні пропорційної похибки, інтегралу похибки за часом та її похідної. Це дозволяє забезпечити не лише швидку реакцію, а й забезпечити плавність керування. Вони широко застосовуються у низькорівневих системах управління — наприклад, для утримання висоти, стабілізації нахилу або обертання дрона. Основним недоліком PID є чутливість до змін середовища, складність тюнінгу для нелінійних систем та повна відсутність адаптації.

На рисунку 1.3 наведено схематичне порівняння класичних підходів за ключовими критеріями ефективності. З рисунка видно, що методи класичного регулювання є ефективними у простих середовищах, однак їхня придатність до складних сценаріїв суттєво обмежена.

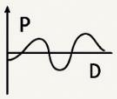
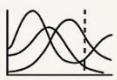
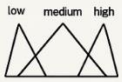
МЕТОД	СУТЬ ТА ПРИНЦИП РОБОТИ	ОСОБЛИВОСТІ
 PID-регулятор	Регулювання за пропорційною, інтегральною та диференціальною складовими відхилення	Простота реапізації, висока швидкодія Вимоги до чіткої настройки проблеми з адаптаціею
 Модельно-прогнозне керування	Використання моделі для прогнозування й оптимізації керування на основі «ковзного горизонту»	Урахування обмежень робота у багатоканальних системах Високі обчислювальні витрати
 Нечітка логіка	Нечіткі правила перетворюють нечіткі значення на чітке керуюче рішення	Адаптивність, робота у складних умовах Висока складність розробки бази

Рисунок 1.3 – Порівняння класичних методів керування БПЛА

Model Predictive Control (MPC) є потужнішим підходом, який дозволяє враховувати обмеження системи та передбачати її майбутній стан. Метод базується на формуванні оптимального керуючого сигналу шляхом розв’язання задачі оптимізації з урахуванням динамічної моделі системи на заданому часовому горизонті. MPC здатен забезпечити багатоканальне управління та працювати з нелінійними моделями, що робить його перспективним для автономних БПЛА. Проте високі обчислювальні витрати обмежують його використання у малопотужних системах без оптимізації або попередньої дискретизації задачі.

Fuzzy logic або нечітка логіка дозволяє створювати адаптивні системи на основі набору правил, які не потребують точного математичного опису. Замість числових моделей використовуються лінгвістичні змінні типу «високий», «низький», «швидкий» тощо, а керування здійснюється на основі логіки, подібної до людського мислення. Це дає змогу створювати системи, що можуть працювати в умовах невизначеності або неповноти даних. Основною проблемою таких систем є складність налаштування та обмежена масштабованість для складних задач.

Кожен з цих класичних підходів має свою нішу. PID-регулятори — це найкращий вибір для базових стабілізаційних задач, MPC — для складних, стратегічно орієнтованих сценаріїв, де доступні ресурси для оптимізації, а нечітка логіка — для адаптивних задач із нечітко визначеною математичною моделлю [2]. Часто вони комбінуються: PID — для стабілізації, а нейромережа — для високорівневого планування. Це дозволяє побудувати гібридні системи, що поєднують надійність класичних рішень із гнучкістю інтелектуальних підходів.

У бойових умовах ефективність класичних методів різко знижується, що пов’язано з потребою швидкої адаптації, роботи в умовах завад, обмеженої видимості та агресивного середовища. Проте вони залишаються

корисними як фундамент для побудови багаторівневих систем управління, де кожен рівень відповідає за свій аспект поведінки БПЛА.

Застосування нейромереж у сфері автономного управління безпілотними літальними апаратами стало революційним етапом, який суттєво змінив уявлення про можливості дронів [1]. Завдяки глибокому навчанню та архітектурам комп’ютерного зору, системи управління отримали здатність до самонавчання, аналізу складного середовища та оперативного реагування на нові ситуації. На відміну від класичних алгоритмів, які діють за заздалегідь заданими правилами, нейронні мережі дозволяють системі працювати в умовах неповної інформації, динамічних змін і навіть порушень у функціонуванні сенсорів.

Одним із найяскравіших прикладів є застосування згорткових нейронних мереж (CNN) для виявлення об'єктів у кадрі. Зокрема, архітектура YOLOv8 забезпечує високошвидкісне розпізнавання цілей на основі одного проходу через мережу, що робить її придатною для реалізації на борту дронів із обмеженими обчислювальними ресурсами. Завдяки цьому БПЛА можуть виконувати задачі виявлення бронетехніки, РСЗВ, укриттів та інших важливих об'єктів в умовах бойових дій.

Додатково до цього, застосовується трекінгова система DeepSORT, яка дозволяє підтримувати стабільне відстеження об’єктів навіть за наявності часткових перекриттів або втрати цілі на кілька кадрів. Інтеграція YOLOv8 + DeepSORT утворює повноцінну систему "виявлення-супровід", здатну в реальному часі забезпечувати інформацію для автономного ухвалення рішень, включаючи обхід перешкод, зміни маршруту або фіксацію об'єкта для подальшої дії. Інтеграція глибоких нейронних мереж у систему управління БПЛА дозволяє досягти нової якості автономного функціонування. На рисунку 1.4 відображено загальну схему взаємодії між

детектором об'єктів YOLO, модулем трекінгу DeepSORT та нейромережею ухвалення рішень, що працює в реальному часі на борту дрона.

Перевагою нейромережевих систем є також можливість доопрацювання й адаптації без повної зміни архітектури. Завдяки використанню аугментацій, квантування, knowledge distillation та інших методів оптимізації, моделі можуть бути легко адаптовані до нових умов, включаючи зміну погодних умов, освітлення, фонів чи варіантів маскування техніки [10].

Іншим важливим аспектом є здатність до реалізації на малопотужних вбудованих платформах. Архітектури типу MobileNetV2, використовувані у підсистемах побудови ознак, дозволяють значно зменшити обчислювальне навантаження, зберігаючи при цьому високу точність розпізнавання. Це робить можливим використання нейромереж навіть на таких пристроях, як NVIDIA Jetson Nano, Google Coral TPU або Raspberry Pi 4.

Таким чином, застосування нейромережевих технологій в управлінні БПЛА забезпечує якісно новий рівень автономності, здатності до адаптації та точності виконання задач. Розглянуті підходи дають змогу не лише ефективно реагувати на зовнішні зміни, але й передбачати події, що є ключовим для сценаріїв розвідки, супроводу цілей або проведення бойових операцій у реальному часі.

Аналіз сучасних систем автономного управління безпілотними літальними апаратами виявляє низку обмежень та вразливостей, що істотно знижують їхню ефективність у бойових умовах. Незважаючи на досягнутий прогрес у сфері комп’ютерного зору, нейромережевого аналізу та оптимізації обчислювальних ресурсів, на практиці існуючі рішення часто не відповідають вимогам реального середовища застосування. Особливо це стосується зон активних бойових дій, де БПЛА мають працювати у

надзвичайно складному, агресивному і непередбачуваному інформаційному просторі.

Першою і однією з найбільш критичних проблем є нестабільність або повна відсутність зв’язку з оператором. У середовищі радіоелектронної боротьби противник активно застосовує засоби глушіння, створення перешкод або перехоплення каналів зв’язку, що знижує або повністю унеможливлює передачу команд до БПЛА [1]. У таких умовах системи, які залежать від наземного оператора або серверної обробки, втрачають працездатність. Навіть за наявності часткової автономії, відсутність зворотного каналу викликає деградацію поведінки апарата, що може призвести до втрати цілі або самого дрона.

Другою суттєвою проблемою є зниження якості вхідних даних, на яких базується вся логіка нейромережевого або класичного аналізу. У реальних умовах відеопотік може бути зашумленим, нечітким, надмірно контрастним або перекритим димом, пилом, дощем чи перешкодами. Крім того, об’єкти можуть мати камуфляж, бути частково закритими або рухатись нетиповими траєкторіями, що ускладнює їхню детекцію навіть найсучаснішими моделями. Унаслідок цього різко знижується точність виявлення та відстеження, а отже й ефективність системи загалом.

Третім обмеженням є високе споживання обчислювальних ресурсів, яке є неприйнятним для компактних дронів з обмеженим енергопостачанням. Потужні нейронні моделі, які демонструють відмінні результати в лабораторних умовах, часто виявляються неефективними у реальному польоті через нестачу енергії або ресурсів. Установка на борт додаткових GPU або TPU рішень збільшує вагу, ускладнює конструкцію і знижує автономність. Бойове середовище створює надзвичайно складні умови для БПЛА, в яких традиційні рішення втрачають ефективність. Рисунок 1.4 демонструє ключові чинники зниження ефективності

автономних систем — втрату зв’язку, зашумлення відео, обмеженість ресурсів та інформаційні атаки.

На рисунку 1.4 відображено ключові фактори, що знижують ефективність автономного управління у бойових умовах. До них належать порушення зв’язку, погіршення даних, нестача обчислювальних потужностей та зовнішні протидії з боку противника, включаючи фальсифікацію об'єктів або впровадження штучних перешкод.

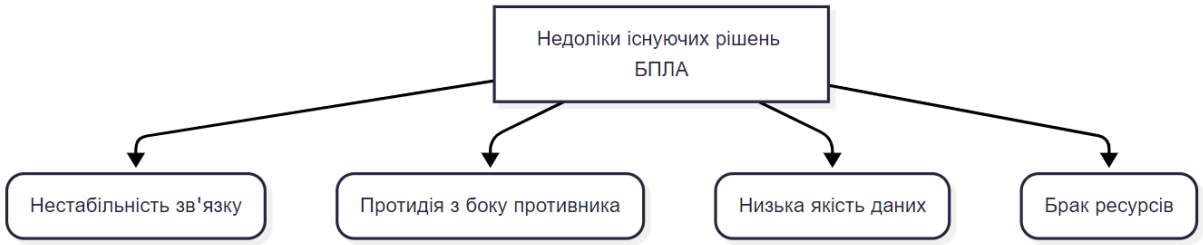


Рисунок 1.4 – Недоліки існуючих рішень БПЛА в умовах бою

Як видно з рисунка 1.4, кожна з наведених проблем впливає на окремий етап функціонування БПЛА, від збору інформації до прийняття рішень.

Ще одним суттєвим викликом є недостатня адаптивність існуючих моделей до нових сценаріїв. У бойових умовах зміна типу цілей, ракурсу, освітлення або фону відбувається миттєво. Якщо модель не навчена на подібних даних або не має механізмів адаптації в реальному часі, її ефективність стрімко падає. Високоточні, але вузькоспеціалізовані системи, не здатні узагальнювати — а отже втрачають перевагу перед більш гнучкими, хоч і менш точними, алгоритмами [5]. Крім того, відсутність глибокої інтеграції між модулями розпізнавання, трекінгу та навігації призводить до фрагментарного керування, яке може бути неузгодженим. Відсутність єдиної логіки ухвалення рішень часто призводить до затримок,

конфліктів між підсистемами або до неефективного виконання завдань. І нарешті, важливим недоліком залишається людський фактор у проєктуванні систем, коли багато моделей не враховують реальні бойові сценарії під час навчання. Часто застосовуються відкриті датасети, які не відповідають умовам поля бою. В результаті виникає так званий «розрив реальності» між тим, на чому вчили модель, і тим, з чим вона стикається в польоті.

Підсумовуючи, слід зазначити, що для подолання вказаних проблем необхідно не просто вдосконалити окремі елементи системи, а переосмислити всю архітектуру автономного управління. Йдеться про створення модульних, адаптивних і стійких до завад систем, які б не лише орієнтувалися на середовище, а й навчались у процесі експлуатації, формуючи власну стратегію поведінки на основі досвіду. Саме такі підходи формують основу наступного покоління БПЛА, здатного ефективно діяти в умовах сучасної війни.

1.3. Формулювання завдань дослідження

Проведений аналіз сучасних проблем автономного управління безпілотними літальними апаратами, а також огляд класичних та інноваційних підходів, дозволяють обґрунтовано сформулювати завдання дослідження, спрямовані на створення ефективної системи автономного управління БПЛА в складних умовах динамічного середовища. Визначення завдань ґрунтується на усвідомленні необхідності комплексного вирішення питань розпізнавання та відстеження об'єктів у реальному часі, оптимізації алгоритмів під обмежені обчислювальні ресурси та забезпеченні високої стійкості системи до змін навколишніх умов. Виходячи із поставленої мети розробки методу і моделі автономного управління БПЛА з використанням нейромережевих технологій, формулюються наступні стратегічні напрями дослідження.

Першочерговим завданням є розробка нейромережевої моделі, здатної здійснювати точне виявлення та трекінг військової техніки у реальному часі за допомогою інтеграції алгоритмів YOLOv8 та DeepSORT. Це завдання передбачає вивчення особливостей побудови сучасних архітектур комп'ютерного зору та розробку оптимізованої структури моделі для забезпечення необхідної точності й швидкодії в умовах обмежених обчислювальних потужностей. Наступним завданням виступає підготовка високоякісної тренувальної вибірки, що відображає специфіку військових об'єктів, різні умови освітлення, кути огляду та варіації фонового середовища. Особлива увага має бути приділена розробці методів аугментації даних, які сприятимуть підвищенню загальної стійкості моделі до змінних факторів середовища. Ключовим аспектом дослідження є оптимізація запропонованої моделі для роботи на малопотужних обчислювальних платформах типу NVIDIA Jetson або Raspberry Pi. Це передбачає адаптацію архітектури мережі, мінімізацію обчислювальної складності моделей та впровадження механізмів компресії моделей без істотного зниження їхньої точності. Особливе місце серед завдань посідає експериментальне тестування створеної системи, що включає оцінювання її ефективності в порівнянні з існуючими аналогами. Тестування повинне проводитися в умовах, максимально наближених до реальних сценаріїв застосування БПЛА, із варіаціями освітленості, рухомості об'єктів і наявності перешкод. Завершальним завданням дослідження є розробка інтуїтивно зрозумілого інтерфейсу для візуалізації процесу розпізнавання і трекінгу, а також для подальшого тестування і демонстрації працездатності розробленої системи.

1.4. Висновки до розділу

За результатами проведеного аналізу можна зробити висновок, що використання автономних безпілотних літальних апаратів (БПЛА) є стратегічно важливим для сучасних умов ведення бойових, розвідувальних та спеціальних операцій. Типовими сценаріями застосування БПЛА виступають патрулювання місцевості, проведення пошуково-рятувальних операцій, супровід та охорона важливих вантажів, моніторинг ситуації на лінії фронту, а також виявлення, супровід і прогнозування поведінки рухомих об’єктів. Впровадження автономних технологій дозволяє забезпечити вищий рівень оперативності та непомітності, зменшити вплив людського фактору та знизити ризики для особового складу. Особливе значення в цих сценаріях набувають системи, засновані на методах глибокого навчання, що дають змогу реалізувати інтелектуальні алгоритми для автоматичного розпізнавання цілей, аналізу відеопотоків та ефективного прогнозування можливих дій супротивника.

Водночас, аналіз сучасних методів автономного управління БПЛА показує, що існуючі рішення далеко не завжди відповідають актуальним викликам, які стоять перед військовими та спеціальними підрозділами. Основні недоліки традиційних систем полягають у недостатній адаптивності до мінливих умов, обмеженій здатності до самонавчання та низькій стійкості до динамічних змін у навколишньому середовищі. Більшість сучасних систем мають жорстко задану структуру управління, яка не дозволяє їм ефективно адаптуватися до нових сценаріїв, змін освітлення, появи нових типів об'єктів або незапланованих перешкод. Для ефективного подолання зазначених проблем необхідно принципово переглянути підходи до проектування архітектури автономних систем, впроваджуючи такі властивості, як модульність, масштабованість, адаптивність та здатність до самонавчання.

Таким чином, нове покоління БПЛА повинно бути оснащене інтелектуальними системами, які не тільки враховують поточну ситуацію в просторі, але й здатні навчатися в процесі експлуатації, формувати власні стратегії поведінки на основі накопиченого досвіду та аналізувати ситуації, з якими вони раніше не стикалися. Це ставить перед дослідниками завдання розробити комплексні методи та моделі автономного управління, які могли б забезпечити високий рівень адаптивності й ефективності функціонування в реальних умовах бойових дій та екстремальних ситуацій. Реалізація такого підходу можлива за умови широкого використання нейромережевих технологій, зокрема глибоких згорткових нейронних мереж (CNN), рекурентних нейронних мереж (RNN) та алгоритмів трекінгу об'єктів на основі моделей DeepSORT у поєднанні з ефективними методами обробки та аналізу потоків даних.

На підставі отриманих результатів аналізу та огляду сучасних підходів було визначено комплекс завдань, реалізація яких дозволить створити ефективну систему автономного управління БПЛА. Першочерговим завданням є розробка нейромережевої моделі, що здатна точно й оперативно виявляти та супроводжувати військову техніку в реальному часі за допомогою інтеграції алгоритмів розпізнавання (YOLOv8) та трекінгу (DeepSORT). Дана модель повинна бути максимально адаптована до складних умов експлуатації, включаючи різні рівні освітлення, різноманітні кути огляду та мінливе оточення. У рамках цього завдання передбачена підготовка якісної та репрезентативної тренувальної вибірки, що враховує весь спектр можливих реальних ситуацій, а також застосування сучасних методів аугментації даних, які підвищать здатність системи розпізнавати й супроводжувати об'єкти в нестандартних умовах.

Другим важливим напрямом роботи є оптимізація нейромережевих моделей для роботи на апаратних платформах з обмеженими

обчислювальними ресурсами. Це завдання передбачає впровадження методів стискання моделей, адаптацію архітектури нейронних мереж з метою зменшення обчислювальних витрат, а також пошук балансу між точністю, швидкістю обробки даних та споживанням енергії.

Наступним ключовим аспектом є проведення експериментальних випробувань створеної системи в умовах, що максимально наближені до реальних сценаріїв бойового застосування. Тестування повинно включати перевірку системи на предмет точності розпізнавання, швидкості ухилення від рухомих перешкод, якості трекінгу рухомих цілей, а також загальної стабільності функціонування системи у складних, змінних умовах.

Завершальним завданням є розробка зручного та інтуїтивно зрозумілого користувацького інтерфейсу, що дозволяє візуалізувати процес роботи автономної системи, а також забезпечує можливість її тестування, налаштування та демонстрації в польових умовах. Інтерфейс має забезпечити користувачеві швидкий доступ до ключових параметрів, стану системи, візуалізації процесів розпізнавання та супроводу об'єктів, що спростить експлуатацію і сприятиме ширшому застосуванню автономних БПЛА у військовій сфері.

Таким чином, вирішення поставлених завдань сприятиме створенню надійної, адаптивної та високоефективної системи автономного управління, здатної суттєво розширити спектр практичного застосування безпілотних літальних апаратів у складних умовах сучасних бойових операцій та спеціальних місій.

2. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ АВТОНОМНОГО УПРАВЛІННЯ БПЛА

2.1. Базові нейромережеві моделі

У процесі розробки системи автономного управління БПЛА критично важливим є вибір нейромережевої архітектури для виявлення об’єктів у відеопотоці. Від цієї архітектури залежить точність, швидкодія, енергоефективність і здатність системи працювати в умовах обмежених обчислювальних ресурсів. Серед широкого спектра доступних моделей особливе місце займає архітектура YOLOv8 (You Only Look Once, версія 8) [6], яка поєднує в собі переваги одноетапного виявлення, компактність, швидкодію та можливість запуску на вбудованих платформах.

YOLOv8 — це вдосконалена версія популярної лінійки YOLO, яка зберігає основний принцип одноразового проходження зображення через мережу з подальшим одночасним визначенням координат об’єкта, класу та ймовірності, та у порівнянні з попередніми версіями, YOLOv8 демонструє значно кращі результати у плані точності (mAP), а також має оптимізовану архітектуру з поліпшеними модулями підготовки ознак та головами детекції [7].

Модель складається з трьох ключових частин: бекбону для вилучення ознак (наприклад, на основі C2f-блоків), neck-компонента (наприклад, PAN або BiFPN) для агрегації багаторівневих ознак, та head, що виконує кінцеве розпізнавання об'єктів. Завдяки підтримці гнучкого масштабування, YOLOv8 доступна в кількох варіантах (n, s, m, l, x), які відрізняються кількістю параметрів і глибиною, що дозволяє обрати відповідну версію для конкретної обчислювальної платформи — від NVIDIA Jetson Nano до серверного GPU.

Архітектура також має ряд покращень у навчанні: підтримка аугментацій, таких як mosaic, copy-paste, random erasing, інтегрований трекінг, функціонал сегментації (YOLOv8-seg) та класифікації (YOLOv8-cls). Це дозволяє моделі бути мультифункціональною в єдиному фреймворку, що критично важливо для задач БПЛА, де час, ресурси та пропускна здатність — обмежені.

Як зображено на рисунку 2.1, архітектура YOLOv8 побудована за принципом багатоетапного вилучення ознак, що забезпечує високу точність навіть для малих об'єктів. Вона дозволяє використовувати як PANet, так і BiFPN для гнучкої агрегації інформації на різних рівнях глибини. Це критично важливо в задачах, де об’єкти можуть бути частково закритими або переміщуватись у складних умовах освітлення.

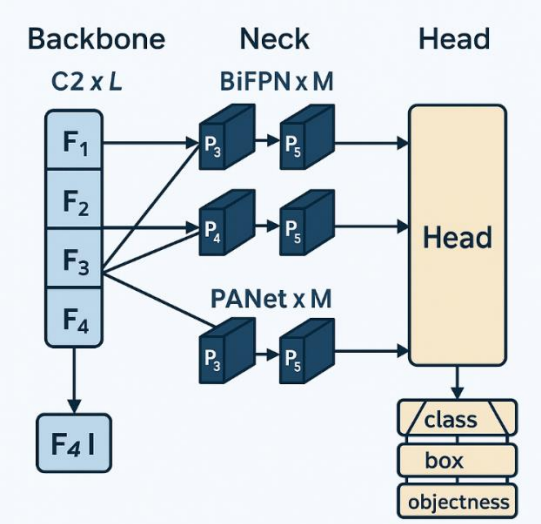


Рисунок 2.1 – Архітектура YOLOv8 для виявлення цілей

У залежності від обчислювальних можливостей апаратної платформи, YOLOv8 дозволяє обирати відповідну модифікацію архітектури — від найлегшої версії n до найпотужнішої x. На рисунку 2.2 представлено порівняльну характеристику моделей за кількістю параметрів, швидкодією

(FPS) та точністю (mAP), що дозволяє ефективно зіставити продуктивність із вимогами до системи в умовах обмежених ресурсів.

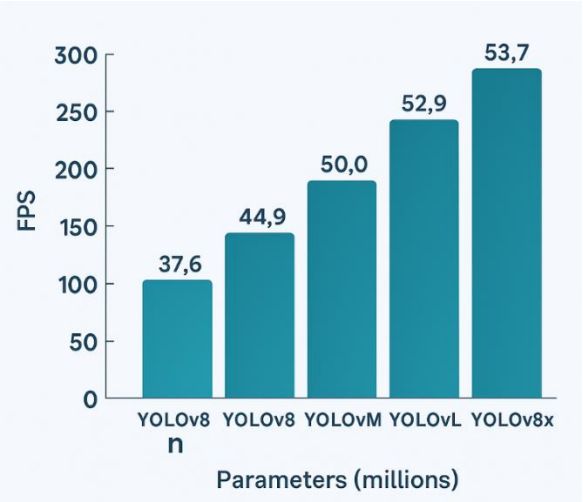


Рисунок 2.2 – Варіанти масштабування YOLOv8 (n, s, m, l, x)

У контексті автономного управління БПЛА модель YOLOv8 не функціонує ізольовано. Вона інтегрується в більшу архітектуру, яка включає блок трекінгу, систему прийняття рішень, навігацію та керування польотом. Як видно з рисунка 2.3, результати виявлення миттєво передаються до інших модулів, що забезпечує цілісність сприйняття ситуації та послідовність дій без затримок.

Рисунок 2.3 – Інтеграція YOLOv8 у систему автономного керування БПЛА

Завдяки оптимальній структурі та високій точності, YOLOv8 дозволяє не лише здійснювати виявлення об'єктів із високим FPS (frames per second), а й забезпечує зменшення кількості false positive/negative помилок, що є критичним у військовому застосуванні, де кожна помилка може призвести до втрати цілі або хибного спрацьовування.

Загалом, вибір YOLOv8 як базової архітектури для підсистеми виявлення цілей обумовлений її здатністю працювати в реальному часі, адаптуватися до різних платформ, забезпечувати високу точність та легко інтегруватися з іншими компонентами системи автономного управління БПЛА. Саме ці характеристики роблять її найдоцільнішим вибором у контексті побудови універсального нейромережевого рішення.

Трекинг цілей у реальному часі є критичним компонентом системи автономного управління БПЛА. Навіть найточніша система виявлення об’єктів втрачає свою ефективність без механізму постійного супроводу цілі, особливо в умовах бойового застосування, де об’єкти можуть тимчасово зникати з поля зору, змінювати швидкість, напрямок або накладатися один на одного. Саме тому у розробці інтегрованої нейромережевої системи для БПЛА було обрано алгоритм DeepSORT (Simple Online and Realtime Tracking with a Deep Association Metric), що є одним із найефективніших рішень для задач трекінгу в реальному часі.

DeepSORT поєднує два основні принципи: відстеження на основі алгоритму Калмана (Kalman Filter) та зіставлення ознак (Re-identification, або ReID) на базі глибокої нейронної мережі. Такий підхід дозволяє не лише передбачати наступне положення об’єкта, але й надійно розпізнавати його навіть після часткової втрати або перекриття. У структурі DeepSORT використовується багатовимірний дескриптор, що обчислюється для кожного об’єкта, і на основі якого здійснюється зіставлення нових детекцій з існуючими треками.

Як показано на рисунку 2.4, вхідні дані від детектора (наприклад, YOLOv8) передаються до трекера, де ініціалізуються або оновлюються траєкторії руху. Використовується фільтр Калмана для прогнозування наступного стану об’єкта, а ReID-дескриптор порівнює ознаки об’єктів між кадрами. Якщо відстань між ознаками перевищує певний поріг, об’єкт вважається новим. Така схема дозволяє не лише зберігати цілісність треків, а й оперативно адаптувати систему до динамічних змін у сцені.

Рисунок 2.4 – Схема роботи алгоритму DeepSORT для трекінгу об’єктів у процесі автономного управління БПЛА

Основними перевагами DeepSORT є його здатність підтримувати стабільність треків у складних динамічних сценах, масштабованість та можливість адаптації до різних детекторів. Модель працює ефективно навіть у випадках, коли цілі тимчасово зникають, рухаються групами або мають схожі візуальні характеристики. Це особливо важливо в бойових

умовах, де спостерігається висока щільність об’єктів на фоні зі складною геометрією або зашумленим зображенням.

Завдяки своїй модульності, DeepSORT може бути реалізований у вигляді окремого модуля з інтеграцією у більшу систему автономного керування. Він приймає на вхід координати bounding box-ів з YOLOv8 та повертає послідовність ідентифікованих об’єктів із призначеними їм ID. Ці дані можуть надалі використовуватись системою ухвалення рішень для побудови маршруту, фіксації цілей або передавання інформації на наземний пункт керування.

Під час практичної реалізації на БПЛА важливою є оптимізація моделі для запуску на вбудованих платформах. Використання квантованих моделей, прискорених фреймворків (TensorRT, OpenVINO), а також зменшення кількості параметрів у ReID-мережі дозволяє досягти реального часу навіть на обмежених ресурсах.

Таким чином, застосування DeepSORT у зв’язці з YOLOv8 утворює ефективну систему "виявлення–супровід", що здатна не лише точно ідентифікувати ціль у поточному кадрі, а й забезпечити її супровід на протязі усього маршруту руху. Такий підхід значно підвищує надійність системи автономного керування БПЛА, дозволяючи їй працювати у режимі повної самостійності без потреби постійного контролю з боку оператора.

2.2. Особливості побудови сучасної системи автономного керування БПЛА

Одним з ключових напрямів оптимізації систем автономного керування БПЛА є побудова багаторівневих каскадних систем, у яких функції обробки інформації розподілені між кількома спеціалізованими модулями. Такий підхід дозволяє знизити навантаження на обчислювальну

платформу, забезпечити швидку локалізацію потенційно важливих об’єктів і виконувати детальне розпізнавання лише в релевантних ділянках кадру. У цьому контексті важливою є ідея виділення області інтересу (Region of Interest, ROI) на ранньому етапі обробки з подальшою передачею цієї області до більш «важкої» нейронної мережі, наприклад, MobileNetV2.

MobileNetV2 є легкою згортковою нейромережею, спеціально розробленою для роботи на пристроях з обмеженими ресурсами. Її архітектура базується на інвертованих резідуальних блоках з глибокою сепарабельною згорткою, що дозволяє істотно зменшити кількість параметрів та обчислень без втрати точності. У каскадній системі MobileNetV2 може виступати у ролі класифікатора або детектора в межах ROI, що дозволяє поєднати швидкість і точність, необхідні для БПЛА.

На рисунку 2.5 показано структуру запропонованої каскадної системи. Перший етап — попереднє виділення областей інтересу (наприклад, за допомогою YOLOv8n), другий — передача цих областей до компактної, оптимізованої версії MobileNetV2, що виконує класифікацію або сегментацію. Результати цієї обробки далі використовуються для формування команди на трекінг або інші дії.

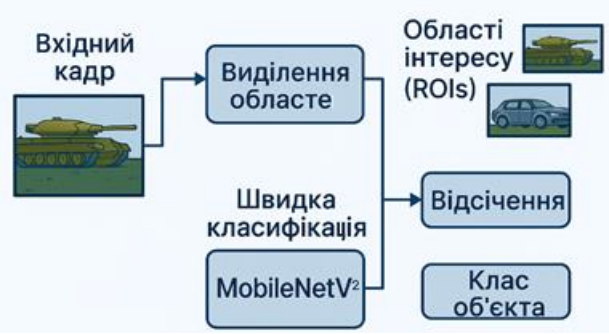


Рисунок 2.5 – Каскадна система на основі ROI та MobileNetV2 для ефективної обробки відеопотоку на БПЛА

Такий каскад дає змогу уникнути повного проходження важкою моделлю всього кадру, що особливо актуально в умовах реального часу та обмеженого обсягу пам’яті. Замість цього система аналізує лише релевантні ділянки, зменшуючи витрати на кожен кадр, не жертвуючи якістю аналізу.

Іншим варіантом реалізації може бути поєднання ROI-фільтрації з класифікатором аномалій або специфічних типів цілей, які заздалегідь важко виявити при загальній детекції. Наприклад, MobileNetV2 може виконувати попереднє сортування підозрілих об’єктів для подальшого глибшого аналізу іншим модулем, таким як ResNet50 або EfficientNet.

Побудова такої системи також відкриває шлях до використання механізмів активного навчання, коли виявлені в ROI нові або невідомі об’єкти можуть автоматично зберігатись та використовуватись для подальшого донавчання моделі. Це формує основу для самонавчальної системи автономного БПЛА, що здатна адаптуватися до змін середовища та цільового профілю під час реального застосування.

Загалом, каскадні системи з попереднім ROI-виділенням і подальшим точним аналізом за допомогою легких мереж типу MobileNetV2 є ефективною відповіддю на виклики обмежених обчислювальних ресурсів, складного фону та високих вимог до автономності в умовах бойових дій.

Сучасні системи автономного керування безпілотними літальними апаратами потребують не лише швидкого і точного виявлення об’єктів, але й здатності працювати в умовах складного фону, перекриттів, змішаних об'єктів і змінного масштабу. Для цього в архітектурі нейронних мереж дедалі частіше використовуються методи побудови багаторівневих ознак, зокрема механізми Enhanced Feature Pyramid (EFP), які значно покращують здатність моделі захоплювати як просторову, так і семантичну інформацію зображення на різних масштабах.

Класичні Feature Pyramid Network (FPN) або PANet забезпечують об'єднання ознак з нижніх і верхніх рівнів для покращення контекстного розуміння. Однак у реальних умовах дії БПЛА цього часто недостатньо. Наприклад, ціль може бути частково закрита або мати маскування, через що її виявлення вимагає не лише аналізу контурів, а й складніших семантичних відношень між ділянками зображення.

Enhanced Feature Pyramid (EFP) — це вдосконалення класичних FPN/PAN архітектур, що реалізується через використання множинних з'єднань, розширених злиттів ознак, а також через введення додаткових фільтраційних блоків. На рисунку 2.6 показано базову схему EFP, в якій відображено послідовну агрегацію ознак з різних рівнів, їх фільтрацію та адаптивну злиту карту, що подається на вхід до детектора.

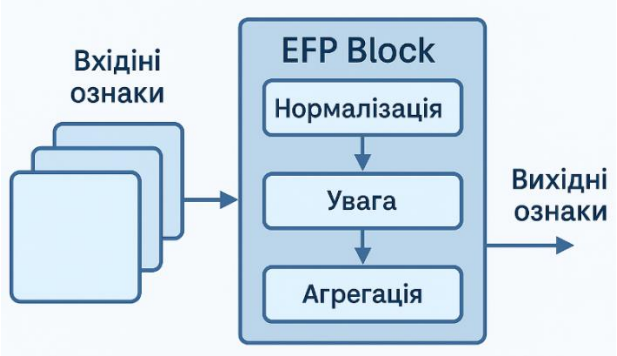


Рисунок 2.6 – Побудова розширеної піраміди ознак (Enhanced Feature Pyramid) для покращення детекції у складному середовищі

Однією з ключових переваг EFP є її здатність забезпечувати чіткість і узгодженість об'єктів при різких змінах масштабу. Вона дозволяє зберігати інформативність навіть найменших деталей об'єктів завдяки підключенню ознак із нижніх рівнів і збагаченню їх семантикою із глибших шарів мережі.

Крім того, наявність зворотних з’єднань забезпечує механізм контекстного уточнення детекцій.

У застосуванні до задач БПЛА, EFP може працювати як частина системи, інтегрованої з YOLOv8, замінюючи класичний neck (наприклад, PAN) на кастомізований блок EFP, що адаптовано під специфіку військових об’єктів. Такий модуль можна додатково оптимізувати для запуску на малопотужних пристроях шляхом зменшення кількості фільтрів, використання групових згорток і попереднього квантованого навчання.

Особливе значення має також те, що EFP значно знижує кількість false positives у складному середовищі, де об’єкти можуть бути схожими, частково перекритими або замаскованими. Завдяки покращеному злиттю ознак з глибоких і поверхневих рівнів, система з EFP формує стабільні детекції навіть у ситуаціях, коли стандартні модулі демонструють нестабільність.

Отже, побудова розширеної піраміди ознак у рамках автономної системи керування БПЛА — це стратегічне рішення, яке значно підвищує стійкість і точність моделі. Такі підходи є особливо актуальними для військового застосування, де кожна помилка у виявленні або класифікації об'єкта може мати критичні наслідки для безпеки та успішності місії.

Формування якісного тренувального датасету є фундаментальним етапом у створенні системи автономного управління БПЛА на основі глибокого навчання. Від складу та структури датасету залежить здатність моделі до генералізації, точність виявлення цілей, стійкість до завад та здатність адаптуватися до нових умов. Особливо це критично у військових задачах, де об’єкти спостереження є нестандартними, частково замаскованими, розташованими у складному середовищі або відрізняються за формою, кольором та ракурсом.

Першим етапом формування датасету є збір зображень військової техніки з різних джерел: відкритих фотобаз, аерофотозйомок, відеозаписів з бойових дій, симуляцій або синтетично згенерованих зображень. Особливу увагу приділяють якості зображення, розмаїттю ракурсів, масштабів та умов освітлення. У випадку обмеженості реальних зразків, ефективним виявляється метод генерації зображень із використанням 3D-моделей техніки або нейромережевих генераторів (наприклад, Stable Diffusion, GAN).

Другий етап — розмітка даних. Для цілей виявлення об'єктів використовується формат bounding box або сегментаційна маска. Дані маркуються вручну або напівавтоматизовано за допомогою інструментів типу LabelImg, CVAT, Roboflow Annotator. Ключовими класами є танки, бронетранспортери, РСЗВ, вантажівки, командні пункти, системи ППО, дрони, артилерійські установки тощо. Кожен клас має бути представлений прикладами у різних положеннях, з різних ракурсів та за різних погодних умов.

Третій етап — аугментація даних, що дозволяє збільшити обсяг вибірки та підвищити її варіативність. Застосовуються як базові (обрізка, обертання, зміна контрасту), так і складні методи (mixup, mosaic, random erasing, cutout, motion blur). Також доцільно використовувати симуляції бойових умов: дим, пил, затемнення, сонячні відблиски або поєднання з фоном реального середовища (наприклад, полігонів, міст, лісів, руїн).

На рисунку 2.7 наведено приклади зображень, включених до тренувального датасету, з відображенням рамок та класів. Видно, що до набору включено не лише чіткі зразки, а й зображення з шумами, перекриттями та в умовах складного фону — саме це дозволяє моделі навчитись розпізнавати об'єкти в бойовій обстановці.

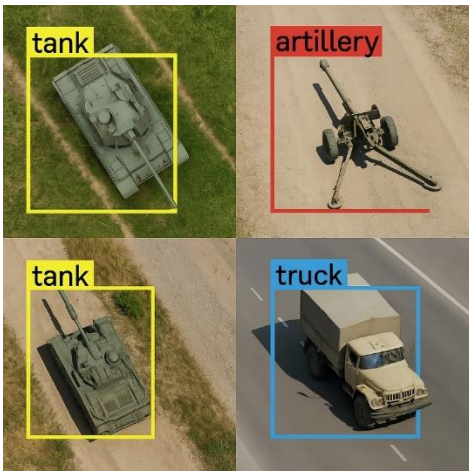


Рисунок 2.7 – Приклади розмічених зображень з тренувального датасету військової техніки (bounding boxes і класи об’єктів)

Четвертий етап — балансування набору. Надлишкова представленість одного класу може призвести до переобучення моделі. Тому важливо забезпечити рівномірність класів або застосувати механізми вагового балансування під час тренування. У випадку малочисельних класів доцільно додатково генерувати зображення або застосовувати синтетичну аугментацію на їхній основі.

П’ятий етап — попередня перевірка набору: запуск моделі на частині даних у режимі валідації, аналіз показників recall/precision per class, пошук потенційно помилкових розміток або конфліктних класифікацій. На основі цих метрик датасет уточнюється й готується до остаточного тренування.

Загалом, добре структурований датасет військової техніки дозволяє сформувати модель, яка не лише точно виявляє об’єкти, але й є стійкою до змін середовища та здатна працювати в реальних умовах бойового застосування. Якість тренувального набору — це запорука успішності всієї системи автономного керування БПЛА.

У задачах комп’ютерного зору для військового застосування, зокрема у контексті автономного керування БПЛА, надзвичайно важливим є забезпечення стійкості моделі до змін зовнішнього середовища. Одним із найбільш ефективних підходів до досягнення цієї мети є застосування методів аугментації даних під час формування тренувального датасету [6]. Аугментація дозволяє не лише збільшити обсяг даних без додаткових зборів, а й сформувати у моделі здатність до генералізації, що критично важливо у складних, непередбачуваних умовах реального бою.

Серед ключових типів аугментації, що використовуються при навчанні моделей виявлення об’єктів у військовій сфері, виділяють три основні категорії: зміни освітлення, варіативність ракурсів та симуляція шуму.

Умови освітлення на полі бою змінюються в залежності від часу доби, погодних умов, затінення об’єктів, наявності диму або вибухів. Для моделювання цих ситуацій застосовуються перетворення яскравості, контрасту, затемнення, накладення сонячних відблисків (glare), локальне освітлення (spotlight), глобальне освітлення (tone mapping), інверсія кольору та нічна фільтрація. Застосування таких трансформацій дозволяє моделі навчитися виявляти об'єкти навіть у затемнених, димних або засвічених ділянках кадру.

Аугментація шуму та артефактів. Симуляція реальних перешкод здійснюється шляхом накладення гаусового шуму, розмиття (motion blur, Gaussian blur), JPEG-артефактів, ефекту пилу, крапель дощу, цифрових шумів, а також глітч-ефектів.

На рисунку 2.8 представлено приклади застосування аугментацій до зображень військової техніки: змінене освітлення, перспективне викривлення, шум та часткове розмиття. Як видно, навіть при суттєвих

перетвореннях об’єкти залишаються розпізнаваними, що є підтвердженням ефективності таких методів для підготовки моделі до реального застосування.

Рисунок 2.8 – Приклади аугментацій для зображень військової техніки: освітлення, ракурси, шум

Таким чином, використання аугментації є не просто технічним прийомом розширення вибірки, а стратегічною необхідністю для забезпечення живучості та надійності моделі. Це особливо важливо в умовах, коли модель повинна демонструвати високі показники не лише в лабораторних тестах, а й у непередбачуваних, змінних і часто ворожих умовах бойових дій.

2.3. Висновки до розділу

У другому розділі було проведено глибокий аналіз математичного та алгоритмічного забезпечення системи автономного управління БПЛА з урахуванням вимог бойового застосування. Всі підпункти розкрили критично важливі компоненти побудови ефективної, стійкої та адаптивної моделі для виявлення і супроводу цілей в умовах реального часу.

Було обґрунтовано вибір архітектури YOLOv8 як базового модуля детекції. Описано її внутрішню структуру та переваги. Розглянуто інтеграцію трекінгового модуля DeepSORT, який поєднує фільтр Калмана та механізм зіставлення ознак на основі нейромережі. Це дозволяє реалізувати стійке відстеження об’єктів, навіть якщо вони тимчасово зникають з поля зору або рухаються у щільному середовищі. Було представлено концепцію каскадної обробки відеопотоку, що дозволяє зменшити обчислювальне навантаження на систему, попередньо виокремлюючи області інтересу і обробляючи їх за допомогою легких моделей на зразок MobileNetV2. Це підвищує ефективність обробки без втрати точності. Було розглянуто синтез розширених ознак через Enhanced Feature Pyramid. Цей механізм дозволяє краще адаптувати модель до змін масштабу, складного фону, перекриттів та маскування, що надзвичайно важливо в умовах бойових дій. Були розглянуті етапи збору, розмітки, аугментації та перевірки даних. Особливу увагу було приділено симуляції бойових умов за допомогою аугментацій: зміни освітлення, ракурсів, накладення шуму та артефактів.

Загалом, результати, представлені в розділі, дають повне уявлення про теоретичне і практичне підґрунтя, необхідне для побудови сучасної системи автономного керування БПЛА. Створена архітектура моделі — це комбінація оптимізованих рішень, здатних забезпечити точність, швидкодію та стійкість у реальному застосуванні.

3. МОДЕЛЮВАННЯ СИСТЕМИ АВТОНОМНОГО УПРАВЛІННЯ БПЛА

3.1. Проектування системи

Загальне проєктування архітектури системи автономного управління БПЛА включає всі основні компоненти — від сенсорних модулів до генерації керуючих сигналів. Такий підхід дозволяє побудувати цілісну, логічно узгоджену систему, що здатна автономно виконувати завдання виявлення, трекінгу, ухвалення рішень і керування польотом у складному динамічному середовищі.

Архітектура системи умовно поділяється на три рівні: рівень збору даних (сенсори), рівень обробки (обчислювальні модулі) та рівень управління (механізми генерації команд). Така трирівнева модель відповідає сучасним принципам модульної інженерії та дозволяє масштабувати систему залежно від задачі або типу БПЛА.

Рівень сенсорного сприйняття включає камери (RGB, тепловізійні, інфрачервоні), модулі GPS, IMU (акселерометр, гіроскоп), альтиметр (барометр), лазерний далекомір (LiDAR) або радар. Камера є ключовим джерелом візуальних даних для систем комп’ютерного зору. IMU забезпечує орієнтацію та прискорення, GPS — глобальне позиціювання. Сучасні платформи БПЛА дозволяють об’єднувати кілька сенсорів у мультисенсорну систему для підвищення точності.

Рівень обробки даних реалізовано у вигляді вбудованого обчислювального блоку (наприклад, NVIDIA Jetson Nano, Xavier NX, Raspberry Pi з TPU), який отримує інформацію з сенсорів і обробляє її у реальному часі. В цьому блоці реалізовано нейромережеві модулі: детектор YOLOv8, трекер DeepSORT, класифікатор на базі MobileNetV2 або інший модуль із розширеним підсиленням ознак (EFP). Також тут працює модуль

ухвалення рішень, який базується на логіці FSM або rule-based системах, адаптованих під бойові сценарії. Всі ці блоки функціонують у щільній взаємодії та з'єднані єдиним програмним стеком (Python, ROS, TensorRT).

Особлива увага приділяється протоколам обміну даними між модулями. Для синхронізації сенсорів і обробки часто використовується ROS (Robot Operating System), який дозволяє побудувати багатопоточну комунікацію між модулями через публікацію або підписку на топіки [2]. Наприклад, камера публікує кадри в топік `/camera/image_raw`, детектор підписується на нього і публікує bounding boxes у `/detections`, а трекер читає ці детекції для обробки. Важливу роль відіграють також міжпроцесні протоколи, такі як ZeroMQ, MQTT, або власні реалізації socket-based API, які дозволяють обмінюватися даними між onboard- і offboard-системами або між процесами на одному пристрої [2].

На рівні керування формуються низькорівневі керуючі сигнали, які передаються до автопілота дрона. Рішення, прийняте на основі виявлення та трекінгу об'єкта, транслюється у вигляді сигналу до контролера польотів який здійснює навігацію, корекцію курсу, стабілізацію або запуск місії перехоплення, слідкування або обльоту цілі.

Особливо важливою у проєктуванні системи є чітка розмежованість між обчислювальною логікою onboard (на борту) та offboard (наземною) частин. До onboard частини належать сенсори, відеокамери, вбудований комп’ютер із моделями для обробки даних, модуль ухвалення рішень і система передавання керуючих команд. Це ядро автономного функціонування БПЛА, яке повинне залишатися працездатним навіть за повної втрати зв’язку з оператором.

Offboard компоненти, своєю чергою, охоплюють наземну станцію управління, канали телеметрії, логування, збір статистики, оновлення моделей та передавання завдань місії. Також саме тут здійснюється

контроль та візуалізація поточних операцій дрона. У разі втрати зв’язку onboard система повинна самостійно завершити місію або повернутися у безпечну зону, діючи згідно з попередньо завантаженими сценаріями (mission fallback).

На рисунку 3.1 представлено компонентну схему архітектури системи автономного керування БПЛА, яка включає всі три рівні та їх взаємозв’язки. Видно, як дані з камери передаються до детектора, потім до трекера і модуля ухвалення рішень, а далі — до генератора сигналів для автопілота.

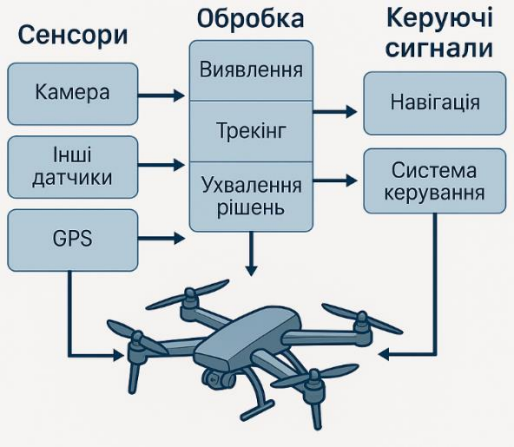


Рисунок 3.1 – Компонентна архітектура системи автономного керування БПЛА: сенсори, обробка, керування

Така архітектура забезпечує модульність, гнучкість, масштабованість і здатність до роботи в реальному часі. Кожен компонент може бути адаптований або замінений без зміни всієї системи, що дозволяє ефективно розвивати платформу під нові задачі. Цей підхід також дозволяє проводити симуляції, тестування та тренування системи як в апаратному, так і в цифровому середовищі.

У цьому підпункті розглядається механізм інтеграції сенсорних систем та обчислювальних модулів у загальну архітектуру автономного керування БПЛА. Від того, наскільки ефективно організовано цей зв’язок,

залежить здатність апарата працювати в реальному часі, здійснювати коректну навігацію, адаптуватися до змін середовища та реалізовувати складні поведінкові стратегії без втручання оператора.

Інтеграція сенсорних даних здійснюється через відповідні драйвери, бібліотеки та фреймворки. Найчастіше використовуються ROS-драйвери (наприклад, `realsense-ros`, `velodyne`, `px4_ros_com`), що дозволяють здійснювати передачу даних у вигляді ROS-повідомлень (`sensor_msgs`, `image_msgs`, `imu_msgs`, `nav_msgs`). Усі дані синхронізуються за часовими мітками та подаються у відповідні топіки, з яких обчислювальні модулі можуть їх читати.

На рівні обчислювального блоку (наприклад, NVIDIA Jetson Nano, Xavier NX) дані обробляються нейромережевими моделями (YOLOv8, DeepSORT, MobileNetV2) та алгоритмами керування. Камера передає зображення у нейромережевий детектор, IMU дані — в модуль оцінки стану, LiDAR — у карту середовища або модуль ухилення від перешкод.

На рисунку 3.2 показано загальну схему маршрутизації даних від сенсорів до обчислювальних модулів. Система формує загальний стан `state_estimate`, який є основою для ухвалення рішень, генерації маршрутів та керуючих дій. Комунікація між модулями реалізується через ROS або ZeroMQ, що дозволяє легко масштабувати архітектуру, розподіляти навантаження та забезпечити гнучкість.

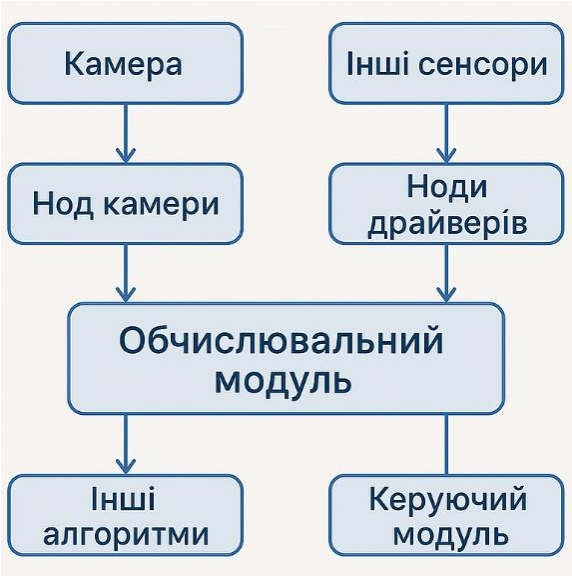


Рисунок 3.2 – Взаємодія сенсорів та обчислювальних модулів у системі автономного БПЛА

Важливим аспектом є механізми відмовостійкості. Якщо дані з окремого сенсора відсутні або некоректні, система переходить до fallback-модуля або використовує комбіновану оцінку (наприклад, IMU + barometer у разі втрати GPS). Також підтримується буферизація даних та динамічна перебудова графа залежностей між модулями (режим self-healing).

Таким чином, побудова ефективної взаємодії між сенсорами та обчислювальними модулями забезпечує основу для автономного функціонування БПЛА. Завдяки сучасним фреймворкам, відкритим драйверам і модульному підходу, система стає адаптивною, стійкою до відмов і придатною для розгортання у широкому спектрі застосувань.

У результаті глибокого аналізу проблеми автономного керування безпілотними літальними апаратами та особливостей бойового середовища було розроблено концепцію програмної системи, що реалізує методи виявлення та трекінгу цілей із використанням нейромережевих технологій. Структура системи базується на архітектурі Model-View-Controller (MVC), яка забезпечує розділення відповідальності між основними компонентами,

що істотно спрощує розробку, тестування, масштабування та подальшу підтримку програмного забезпечення. На рисунку 3.3 подано загальну UML-діаграму пакетів розробленого ПЗ.

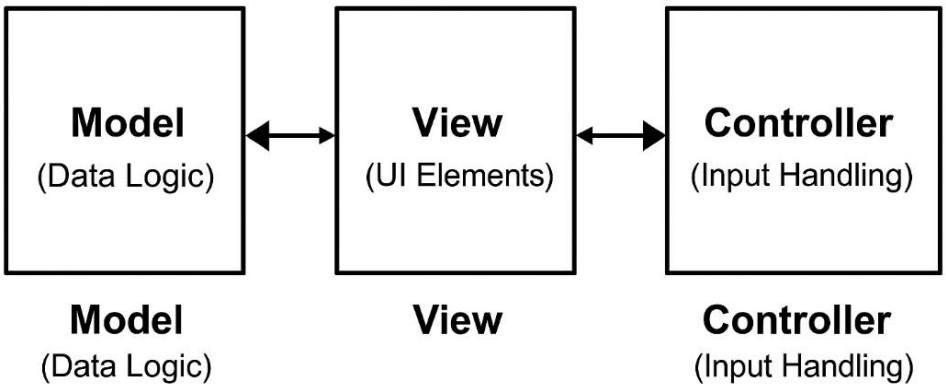


Рисунок 3.3 – UML діаграма пакетів програмного забезпечення

Архітектура MVC була обрана через її ефективність в умовах необхідності швидкої обробки даних у реальному часі, стабільної роботи на малопотужних обчислювальних платформах і гнучкої адаптації до змін середовища. Компонент Model у розробленій системі відповідає за реалізацію бізнес-логіки, машинного навчання, обробку відеопотоку та доступ до баз даних. Саме тут сконцентровані нейромережеві моделі для розпізнавання об'єктів (YOLOv8) та їх трекінгу (DeepSORT), а також алгоритми аугментації і фільтрації даних. Модель також зберігає метрики, історію і результати роботи системи, що забезпечує повноцінну автономність функціонування без потреби у постійній комунікації з зовнішніми джерелами.

Компонент View забезпечує візуалізацію результатів роботи системи. У контексті даної розробки це реалізовано у вигляді графічного інтерфейсу для відображення виявлених об'єктів, їх траєкторій, впевненості класифікації та загального статусу БПЛА. Незалежність представлення від

моделі дозволяє швидко адаптувати інтерфейс під різні типи пристроїв та умови використання без зміни базової логіки обробки даних.

Контролер (Controller) виконує роль посередника між користувачем і системою, обробляючи вхідні потоки відео, координуючи запити до моделей, контролюючи процес оновлення представлення та реалізуючи логіку обробки помилок або втрат зв'язку. У бойових умовах наявність адаптивного контролера критично важлива для забезпечення стійкості системи до зовнішніх впливів.

Використання архітектури MVC дозволило досягти високої гнучкості та модульності: удосконалення моделей розпізнавання, заміна трекера або адаптація до нових датасетів не потребують внесення змін до інтерфейсу чи механізмів контролю потоку даних. Це критично важливо в контексті реальних місій, де необхідно швидко адаптувати систему до нових типів цілей чи змін умов середовища.

Пакет Model у системі автономного керування БПЛА складається з кількох ключових класів, серед яких основними є: MainModel, DMC, TrainingData, NetworkModel та DataFormat. Клас MainModel виконує функції централізованого керування даними, забезпечуючи завантаження, попередню обробку, збереження і передачу даних до інших компонентів системи. В ньому також зберігаються дані для візуалізації в інтерфейсі користувача та забезпечується управління взаємодією із зовнішніми джерелами інформації.

На рисунку 3.4 наведено UML-діаграму класу MainModel, яка ілюструє основні поля і методи класу, визначаючи його роль у забезпеченні цілісності даних, їхньої актуальності та готовності до обробки нейромережевими моделями. Інкапсуляція даних і чітке визначення методів роботи з ними підвищують надійність функціонування системи загалом,

знижують імовірність помилок і забезпечують можливість розширення системи відповідно до зростаючих вимог експлуатації в реальних умовах.

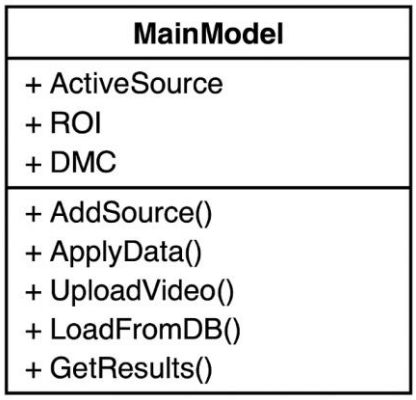


Рисунок 3.4 – UML діаграма класу MainModel

Атрибути класу MainModel у розробленій системі автономного керування БПЛА виконують функції управління ключовими даними, необхідними для обробки відеопотоку та взаємодії з нейромережевими модулями. ActiveSource є екземпляром класу VideoSource і зберігає інформацію про активне відеоджерело, що використовується для виявлення і трекінгу об'єктів у реальному часі. ROI представлений у вигляді кортежу (Tuple) та визначає координати зони інтересу у вхідному відеопотоці, що дозволяє оптимізувати обчислювальні ресурси, спрямовуючи обробку лише на релевантні ділянки зображення. DMC є об'єктом класу DMC, який відповідає за конфігурацію і взаємодію з нейронними мережами, а також за обробку отриманих результатів, їхню адаптацію для подальшої візуалізації та інтеграції в систему автономного управління.

Методи класу MainModel забезпечують повний цикл роботи з вхідними даними. Приватний метод AddSource() реалізує додавання нового відеоджерела до загальної структури даних для подальшого використання в тренуванні або тестуванні моделі. Публічний метод ApplyData() активує обране джерело відео для обробки нейронною мережею. Метод

UploadVideo() дозволяє завантажити локальне відео з комп’ютера користувача для аналізу в системі. LoadFromDB() виконує отримання даних з попередньо підготовленої бази даних, забезпечуючи узгодженість і актуальність вхідної інформації. Метод GetResults() обробляє результати роботи нейронної мережі, формуючи їх у зручний формат для подальшої візуалізації.

Клас DMC виступає центральною ланкою у взаємодії між користувачем, базами даних та нейронними мережами. Його атрибути мають критичне значення для забезпечення стабільності та ефективності роботи системи. NetworkModel є екземпляром класу, який інкапсулює нейронну мережу для виявлення та трекінгу цілей. PredictionResult містить структуру даних Results для зберігання вихідних даних обробки, необхідних для аналізу і візуалізації. TrainingData являє собою список екземплярів класу TrainingData, що забезпечує організований доступ до тренувальних наборів даних та підтримку їх актуальності. Методи класу DMC реалізують ключові етапи роботи з нейромережею. LoadData() завантажує і перетворює підготовлені дані для подальшої обробки. LoadModel() відповідає за завантаження попередньо натренованої нейронної мережі для використання в реальному часі. Метод SaveData() забезпечує збереження поточного стану нейромережі разом із її налаштуваннями, що дозволяє відновлювати робочі сесії без потреби повторного навчання. Calculate() приймає відеопотік, обробляє кожен кадр у відповідному форматі та виконує оцінку за допомогою моделі. CalculateFrame() здійснює аналогічну операцію для окремого кадру, що дає змогу тестувати мережу в умовах одиничного вхідного сигналу. Метод TrainNetwork() забезпечує процес навчання нейронної мережі на базі даних, переданих користувачем, із відповідним налаштуванням параметрів тренування.

На рисунку 3.5 наведено UML-діаграму класу DMC, яка ілюструє атрибути і методи цього класу та відображає його ключову роль у забезпеченні повного циклу роботи нейромережевої системи автономного керування БПЛА.

.

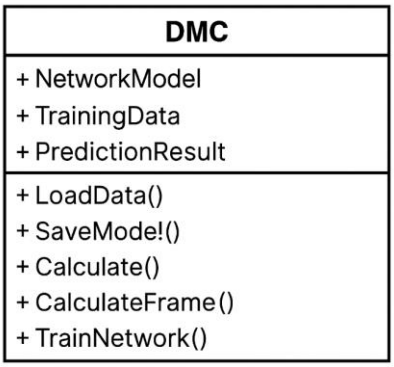


Рисунок 3.5 – UML діаграма класу DMC

Клас TrainingData у системі автономного керування БПЛА призначений для роботи з базою даних тренувальних та тестових прикладів. Він зберігає всю необхідну інформацію для аналізу, обробки та тренування нейронної мережі, а також для подальшого використання даних через інтерфейс користувача.

Атрибути класу TrainingData забезпечують збереження важливої інформації. ID є цілочисельною змінною, що містить унікальний ідентифікатор кожного відеозапису об'єкта спостереження. Name є рядковою змінною, яка зберігає назву відповідного відеофайлу. Size представлений у вигляді кортежу (Tuple) і містить розміри відео. VideoPath є рядковою змінною, що визначає повний шлях до відеофайлу у файловій системі. PredictedResult є змінною типу float, яка містить цільове значення результату прогнозування, що має бути отримане від нейромережі, де значення від 0 до 1 характеризує якість виявлення об'єкта.

Методи класу TrainingData забезпечують взаємодію з базою даних та підготовку даних для тренування. Приватний метод OpenConnection() відповідає за відкриття з'єднання з базою даних, тоді як CloseConnection() закриває його, забезпечуючи збереження всіх змін. Метод LoadVideoData() завантажує відеофайли та здійснює їх підготовку до обробки нейронною мережею. Метод LoadTableData() виконує завантаження табличних даних, які відповідають відеоресурсам, для подальшої узгодженої обробки. На рисунку 3.6 подано UML-діаграму класу TrainingData.

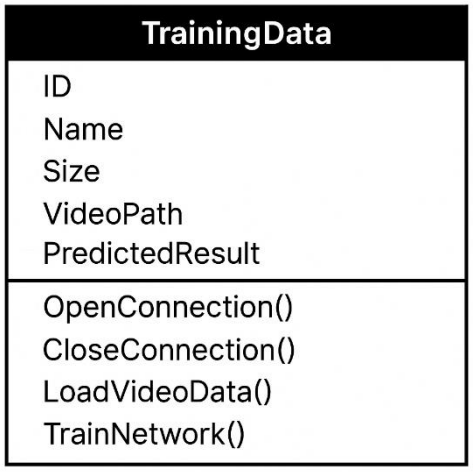


Рисунок 3.6 – UML діаграма класу TrainingData

Клас NetworkModel є фундаментальним для реалізації та роботи з нейронними мережами у запропонованій системі. Він забезпечує створення, налаштування, тренування і використання моделей для автономного розпізнавання об'єктів.

Атрибути класу NetworkModel реалізують компоненти побудованої моделі. MobileNet виступає як перший модуль системи, реалізуючи полегшену версію згорткової нейронної мережі (CNN), яка відповідає за обробку вхідних візуальних даних.

Методи класу NetworkModel забезпечують повний життєвий цикл роботи нейронної мережі. Метод Create() реалізує створення архітектури нейронної мережі згідно з параметрами, наданими користувачем. Метод Train() відповідає за процес навчання моделі на базі даних, отриманих від користувача, і повертає результат успішності тренування. Метод Predict() здійснює прогнозування на основі вхідних відеоданих, формуючи результат, який оцінює рівень виявлення та трекінгу об'єктів для використання у системі автономного керування БПЛА. На рисунку 3.7 подано UML-діаграму класу NetworkModel.

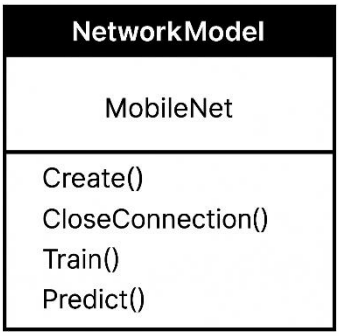


Рисунок 3.7 – UML діаграма класу NetworkModel

Клас DataFormat є важливим компонентом системи автономного керування БПЛА, відповідальним за приведення даних до формату, оптимального для обробки нейронною мережею. Він є статичним та забезпечує виконання різноманітних перетворень вхідних відео- та зображень відповідно до потреб аналізу. На рисунку 3.8 подано UML-діаграму класу DataFormat.

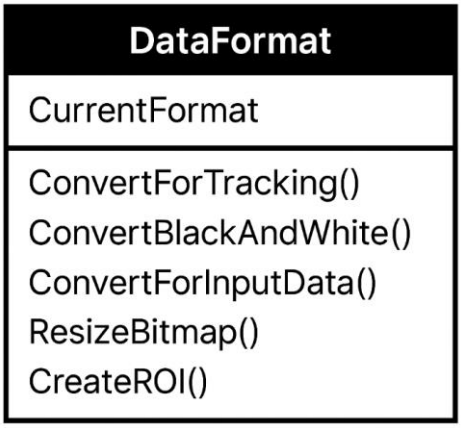


Рисунок 3.8 – UML діаграма класу DataFormat

Атрибут CurrentFormat є статичним значенням перелічуваного типу, що вказує на обраний користувачем режим обробки даних для нейронної мережі.

Методи класу DataFormat реалізують ключові функціональні можливості перетворення даних. Метод ConvertForTracking() відповідає за підготовку кадрів відео для задачі трекінгу об'єктів. Метод ConvertBlackAndWhite() здійснює перетворення зображень у чорно-білий формат для спрощення обробки. Метод ConvertForInputData() готує табличні дані для завантаження у базу даних. Метод ResizeBitmap() забезпечує зміну розмірів зображення відповідно до вимог нейронної мережі. Метод CreateROI() створює зону інтересу для подальшої обробки обраної ділянки кадру.

Пакет View відповідає за візуалізацію інформації та взаємодію користувача із системою. Основними його класами є Window, DriverTracker, VideoContainer та CameraContainer.

Клас Window є головним елементом пакету View та забезпечує інтеграцію між користувачем і основною функціональністю системи. Він відповідає за налаштування джерел даних, відображення результатів роботи

нейронної мережі та ініціалізацію тренування моделей. На рисунку 3.9 подано UML-діаграму класу Window.

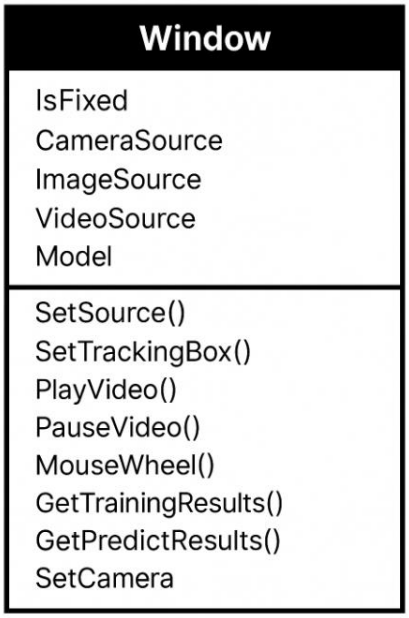


Рисунок 3.9 – UML діаграма класу Window

Атрибути класу Window включають IsFixed — булеву змінну, що визначає фіксацію джерела відео, CameraSource — екземпляр класу CameraCapture для роботи з камерою користувача, ImageSource — екземпляр класу ImageContainer для завантаження одиничних зображень, VideoSource — екземпляр класу VideoContainer для роботи з відеофайлами, TrackingBox — екземпляр типу DriverTracker для візуалізації зони інтересу, та Model — екземпляр класу MainModel для взаємодії з даними та нейронною мережею.

Методи класу Window забезпечують управління відеопотоком і взаємодію з користувачем. Метод SetSource() встановлює джерело відеопотоку з файлу, а SetCamera() — з камери. Метод SetTrackingBox() ініціалізує рамку відстеження об'єкта у відео. Метод PlayVideo() запускає відтворення відео, PauseVideo() призупиняє, а StopVideo() зупиняє його та

очищує буфери. Методи MouseWheel(), MouseDown(), MouseUp() і MouseMove() відповідають за інтерактивну взаємодію з відеоматеріалом через мишу, зокрема зміну масштабу та керування позиціонуванням зони інтересу. Метод GetTrainingResults() ініціює процес тренування нейронної мережі на основі відеоданих та відображає результати, а метод GetPredictResults() здійснює прогнозування на основі поточного відеопотоку, надаючи результати аналізу, включно з рівнем виявлення об'єкта.

Таким чином, модулі DataFormat та Window забезпечують ключові можливості підготовки даних і інтерактивної взаємодії користувача з системою, що істотно підвищує ефективність та гнучкість запропонованого рішення для автономного керування БПЛА.

Клас DriverTracker відповідає за візуалізацію результатів обробки відеоданих нейронною мережею безпосередньо на кадрах відеоматеріалу. На рисунку 3.10 подано UML-діаграму класу DriverTracker.

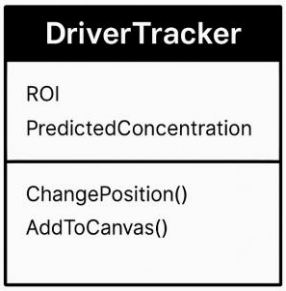


Рисунок 3.10 – UML діаграма класу DriverTracker

Атрибути класу DriverTracker включають ROI — кортеж координат, що визначає межі зони інтересу, та PredictedConcentration — значення, що відображає рівень концентрації за результатами роботи нейронної мережі. Метод ChangePosition() оновлює координати зони інтересу відповідно до нових прогнозованих даних. Метод AddToCanvas() додає графічні

елементи, що візуалізують результати аналізу, на полотно користувацького інтерфейсу.

Клас CameraContainer забезпечує налаштування та роботу з відеопотоком, що надходить із камери користувача. Його UML-діаграма представлена на рисунку 3.11.

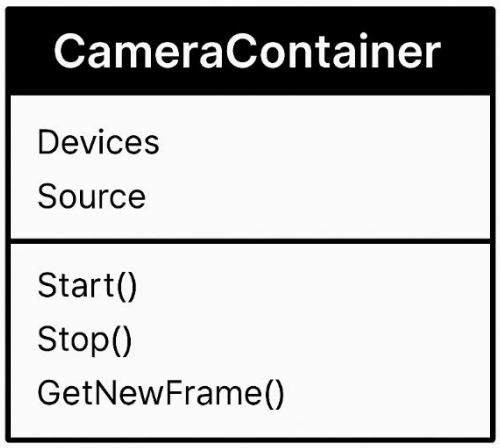


Рисунок 3.11 – UML діаграма класу CameraContainer

Атрибути CameraContainer включають Devices — список доступних відеопристроїв, Source — активний відеопристрій. Метод Start() ініціює зчитування відеопотоку з камери, Stop() припиняє його, а GetNewFrame() дозволяє отримувати поточні кадри в реальному часі для подальшої обробки системою.

Клас VideoContainer відповідає за збереження метаданих відеофайлів та їх представлення в інтерфейсі користувача. UML-діаграма класу наведена на рисунку 3.12.

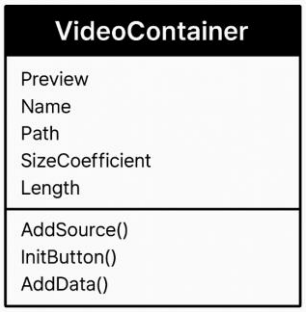


Рисунок 3.12 – UML діаграма класу VideoContainer

Атрибути класу VideoContainer включають Preview — кадр для попереднього перегляду відео, Name — назву відеофайлу, Path — повний шлях до файлу, SizeCoefficient — коефіцієнт масштабування для коректного відображення, та Length — кількість кадрів у відео. Метод AddSource() відповідає за додавання нового джерела відео, InitButton() — за створення елементів керування в інтерфейсі, AddData() — за додавання інформації про нове відео, а SetActive() — за вибір активного відеофайлу для роботи системи.

Клас Controller є центральною частиною пакету Controller і відповідає за координацію всіх процесів взаємодії між компонентами системи. На рисунку 3.13 подано UML-діаграму класу Controller.

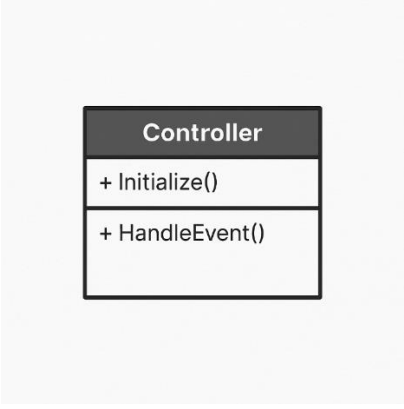


Рисунок 3.13 – UML діаграма класу Controller

Клас Controller реалізує обробку запитів користувача, маршрутизує команди від елементів View до відповідних функцій Model та забезпечує зворотний зв'язок для оновлення інтерфейсу користувача на основі результатів обробки. Така архітектура дозволяє досягти чіткої структурованості взаємодії між підсистемами, забезпечуючи високу гнучкість і розширюваність програмного продукту.

Таким чином, класи DriverTracker, CameraContainer, VideoContainer та Controller разом утворюють логічно завершений і узгоджений комплекс для інтерактивної роботи системи автономного керування БПЛА, забезпечуючи обробку відеоданих, візуалізацію результатів і інтеграцію користувацької взаємодії із нейромережею.

3.2. Підготовка тренувальної вибірки перед поданням в нейромережу

Для задачі оцінки рівня концентрації водія наземного транспорту датасет повинен відповідати ряду технічних вимог і характеристик, які дозволять моделі коректно працювати в реальних умовах.

Препроцесинг вхідних даних є критично важливим етапом у ланцюжку обробки зображень перед передачею їх до нейромережевих моделей, зокрема в контексті системи автономного керування БПЛА. Від правильності виконаного препроцесингу залежить як точність виявлення об’єктів, так і загальна продуктивність системи. Основна задача препроцесингу — забезпечити узгодженість вхідних даних із вимогами архітектури моделі, зберігаючи при цьому суттєву інформацію для виявлення цілей.

Першим кроком є зміна розміру зображення. Більшість моделей YOLO потребують фіксованого розміру вхідного тензора, наприклад 640×640 пікселів. Для цього використовується масштабування (`resize`) або

обтинання із масштабуванням (`letterbox`), яке дозволяє зберегти співвідношення сторін. Метод `letterbox` є переважним у реальних задачах БПЛА, оскільки він запобігає геометричному спотворенню об’єктів.

Далі застосовується нормалізація піксельних значень — перетворення діапазону яскравості зображення до стандартного формату (наприклад, [0,1] або [-1,1]). Це дозволяє моделі швидше й стабільніше сходитися під час навчання і покращує узгодженість між зображеннями з різних сенсорів або в різних умовах освітлення.

Перетворення кольорового простору також має важливе значення. Деякі архітектури очікують вхід у форматі RGB, тоді як камера може подавати BGR або Grayscale. Якщо йдеться про використання інфрачервоних або тепловізійних зображень, застосовується відповідне кодування псевдокольором або формування багатоканального зображення (наприклад, об’єднання RGB + IR).

Ще одним важливим аспектом є центризація та стандартизація вхідних тензорів. Це означає віднімання середнього значення та поділ на стандартне відхилення (z-score нормалізація). Такий підхід дозволяє нейромережі швидше адаптуватися до нових типів вхідних даних та зменшує чутливість до шумів і аномалій.

На рисунку 3.14 представлено схему основних етапів препроцесингу, які застосовуються перед подачею зображення в модель YOLOv8. Вона охоплює зміну розміру, нормалізацію, перетворення кольору та формування тензора з урахуванням батчу.

Рисунок 3.14 – Основні етапи препроцесингу зображення перед передачею в нейронну мережу

У системах реального часу важливим є також оптимізація препроцесингу для зменшення затримки (latency). Для цього можна використовувати бібліотеки з апаратною прискореною обробкою: OpenCV з підтримкою CUDA, NVIDIA DALI (Data Loading Library), TensorRT pre-processing pipelines. Вони дозволяють перенести частину препроцесингу на GPU і виконувати його паралельно з інфере́нсом.

Також у задачах автономного БПЛА доцільно розділяти препроцесинг для RGB і IR-каналів, обробляючи їх окремо або формуючи мультиканальні входи з подальшим злиттям ознак усередині мережі (late fusion). Це дозволяє зберігати повноту інформації та адаптувати систему до умов обмеженої видимості (ніч, туман, дим).

У підсумку, правильно побудований блок препроцесингу є не менш важливим, ніж архітектура нейромережі. Він забезпечує стабільність, швидкодію та точність всієї системи автономного виявлення і супроводу цілей, а також є основою для надійної роботи в широкому діапазоні умов бойового середовища.

3.3. Архітектура створеної нейромережевої моделі

Створена система використовує дві нейромережеві моделі для досягнення поставленої задачі – MobileNetV2 та GRU (Gated Recurrent Unit).

Центральною ідеєю є послідовна обробка вхідного зображення за допомогою моделі детекції (YOLOv8) та передачі результатів у трекер (DeepSORT), який забезпечує ідентифікацію та супровід цілей у часовій послідовності кадрів.

Процес починається із захоплення кадру з камери або з відеопотоку. Кадр надходить у блок препроцесингу, де відбувається масштабування (до 640×640), нормалізація, форматування тензора та передача на вхід моделі YOLOv8. Модель детектує об’єкти в кадрі, повертаючи координати bounding box, ймовірність класу та мітку класу.

1. Результати обробки YOLOv8 передаються у модуль DeepSORT у вигляді списку детекцій (наприклад, у форматі [x1, y1, x2, y2, confidence, class_id]). DeepSORT, своєю чергою, здійснює: AddSource() – приватний метод, який в відповідає за вибір нового джерела в якості поточного.
2. Перетворення координат у формат, прийнятний для Kalman-фільтра;
3. Прогноз траєкторії всіх активних треків;

4. Вирахування ReID-дескрипторів для нових детекцій через нейромережу (наприклад, на базі мобільної ResNet);
5. Зіставлення треків і нових детекцій за допомогою Hungarian algorithm з урахуванням просторової відстані та відстані у просторі ознак;
6. Призначення ідентифікаторів (track_id) та оновлення треків.

На рисунку 3.15 показано послідовну схему виклику моделей, з візуалізацією передачі даних між блоками та логікою оновлення треків. Таке структурування дозволяє модулю ухвалення рішень працювати з уже відфільтрованими та стабілізованими даними про об'єкти.

Рисунок 3.15 – Послідовність виклику моделей YOLOv8, DeepSORT для детекції та трекінгу цілей

Особливу увагу необхідно приділити затримці між кадрами та синхронізації FPS між моделями. YOLOv8 може працювати зі швидкістю 25–60 кадрів за секунду в оптимізованому режимі (TensorRT), тоді як DeepSORT вимагає додаткових обчислень на кожен кадр. Тому в системі

реалізується буфер кадрів та черга обробки, які забезпечують узгодженість потоку.

З практичної точки зору, реалізація цього порядку виклику здійснюється в рамках одного ROS-нода або Python-модуля з попередньо ініціалізованими моделями YOLOv8 і DeepSORT, де функція обробки кадру викликається циклічно, а всі входи та виходи логуються для подальшої телеметрії та аналізу.

Загалом, така структура дозволяє досягти ефективної інтеграції детектора та трекера з мінімальними затримками та стабільними результатами, що є ключовим для бойового застосування БПЛА у реальному часі.

Система трекінгу на основі DeepSORT базується на оновленні траєкторій об’єктів у часовій послідовності кадрів. Основна мета цього підпункту — описати механізм оновлення треків, який включає оцінку стану треків, призначення нових детекцій, прогноз наступного положення, відсів старих треків та ініціалізацію нових. Така схема забезпечує сталість ID для об’єктів навіть за умов часткових перекриттів, зміни ракурсу або короткочасної втрати цілі.

Процес оновлення треків складається з наступних ключових етапів:

1. Прогноз положення: перед надходженням нових детекцій Kalman-фільтр здійснює прогноз положення для кожного активного треку [8]. Це дає змогу оцінити ймовірну зону знаходження об’єкта.

2. Визначення асоціацій (matching): нові детекції зіставляються з активними треками за допомогою двох метрик — евклідової відстані між центрами bounding box і косинусної відстані у просторі ReID-ознак. Для вирішення завдання оптимального зіставлення використовується алгоритм Гунгарі (Hungarian Algorithm) [9].

3. Оновлення треків: якщо нова детекція співпадає з прогнозом, то параметри Kalman-фільтра уточнюються, а ReID-дескриптор оновлюється. Трек отримує статус confirmed.

4. Створення нових треків: якщо детекція не відповідає жодному з існуючих треків, ініціалізується новий трек зі статусом tentative, який переходить у confirmed після кількох успішних спостережень поспіль.

5. Видалення неактивних треків: треки, які не отримали підтвердження упродовж певної кількості кадрів (наприклад, `max_age = 30`), вважаються втраченими й видаляються з пам’яті системи.

На рисунку 3.16 наведено загальну логіку цього процесу, включаючи потоки даних, блоки прийняття рішень та етапи переходу між статусами треків.

Рисунок 3.16 – Схема оновлення треків у модулі DeepSORT: прогноз, зіставлення, оновлення, видалення

Окрему увагу слід приділити параметрам конфігурації, які впливають на ефективність оновлення: `max_cosine_distance`, `nn_budget`, `max_iou_distance`, `max_age`, `min_hits`. Їхнє тюнінгування дозволяє балансувати між чутливістю до нових об’єктів і стійкістю до хибних спрацьовувань.

Правильно реалізована схема оновлення треків дозволяє системі автономного БПЛА підтримувати цілісний ситуаційний контекст, вести спостереження за кількома цілями одночасно і надійно передавати результати у модуль ухвалення рішень навіть в умовах перешкод і змінної видимості.

У системі автономного керування БПЛА, яка базується на нейромережевих детекторах і трекерах, важливим етапом є фільтрація шумових, помилкових або некоректних виявлень. Така фільтрація дозволяє підвищити точність та надійність моделі, знизити кількість хибнопозитивних спрацьовувань і забезпечити стабільну роботу трекінгового та керуючого модулів.

Некоректні виявлення можуть виникати через різні причини: шуми у відеопотоці, артефакти освітлення, часткові об'єкти, фонові елементи, що схожі на ціль, або надто низьку довіру моделі до класу. Без фільтрації такі об'єкти можуть некоректно потрапити до трекера, що призведе до помилкового супроводу або хибного ухвалення рішень.

Основними методами фільтрації є:

1. Порогова обробка по впевненості (confidence threshold): встановлюється мінімальний поріг для значення довіри (`confidence score`), нижче якого виявлення ігнорується. Зазвичай встановлюється значення 0.25–0.5, залежно від специфіки задачі.

2. Фільтрація за розміром bounding box: надто малі або великі об'єкти (наприклад, `area < 100 px²` або `width > 90% кадру`) можуть бути ознакою помилки в детекції або фонового елемента.

3. Фільтрація за співвідношенням сторін (aspect ratio): об'єкти з нетиповою геометрією (наприклад, дуже вузькі або витягнуті) можуть вказувати на шумову детекцію. Встановлюються допустимі межі (наприклад, $0.2 < w/h < 4.0$).

4. Наявність перекриттів (Non-Maximum Suppression, NMS): усуваються дублікати одного й того ж об'єкта через перетин (IoU > 0.5). Алгоритм NMS залишає лише найякісніший детекшн для кожної зони.

5. Аналіз контексту: якщо об'єкт знаходиться в недопустимій зоні (наприклад, поза межами допустимого ROI) або суперечить очікуваному шаблону поведінки (наприклад, поява поза маршрутом), він може бути відфільтрований модулем логіки.

На рисунку 3.17 представлено алгоритм фільтрації шумових виявлень на прикладі послідовної перевірки об'єктів перед їх передачею в трекер.

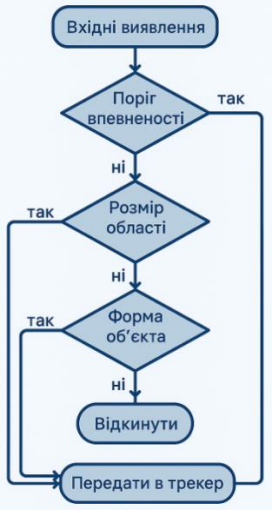


Рисунок 3.17 – Алгоритм фільтрації детекцій за впевненістю, розміром, співвідношенням сторін та контекстом

Реалізаційно фільтрація виконується як окремий етап після YOLOv8, але до DeepSORT. У Python-пайплайнах це може бути окрема функція `filter_detections(detections, conf_threshold, min_size, aspect_limits)`, яка повертає тільки валідні детекції. Цей блок критичний для стабільності системи, оскільки фільтрація дозволяє підтримувати цілісність треків і знижує ймовірність переривання логіки прийняття рішень.

Загалом, застосування багаторівневої фільтрації детекцій дозволяє зробити систему керування БПЛА більш стійкою до реальних умов і гарантує вищу якість супроводу та виявлення цілей у динамічному середовищі.

Процес логічного ухвалення рішень в автономній системі управління БПЛА починається з інтерпретації результатів нейромережевого виводу — а саме виявлених класів об’єктів та їх просторових координат. Модель детекції, наприклад YOLOv8, повертає масив об’єктів, кожен з яких описується координатами прямокутника обмеження (bounding box), міткою класу та значенням довіри (confidence score). Ці дані формують первинну основу для формування ситуаційної моделі середовища, яка є основою для подальших рішень системи.

Bounding box кожного об'єкта задається у форматі [x_center, y_center, width, height] або [x_min, y_min, x_max, y_max]. Для інтеграції з модулями трекінгу та навігації ці координати можуть бути перетворені у систему координат реального середовища (наприклад, у метри) шляхом врахування параметрів калібрування камери та висоти польоту дрона.

Мітка класу — це ціле число, яке ідентифікує категорію об’єкта (наприклад, 0 — танк, 1 — вантажівка, 2 — РСЗВ). Кожен клас асоціюється з певною стратегічною або тактичною значущістю. Наприклад, при

виявленні танка система може запускати сценарій уникнення або перехоплення, а при виявленні вантажівки — лише маркування на карті.

Значення довіри (confidence score) використовується як критерій фільтрації: об’єкти з низькою ймовірністю (наприклад, нижче 0.3) можуть ігноруватися або маркуватися як малоймовірні. У розширених реалізаціях використовується ще й objectness score — оцінка того, чи є об’єкт взагалі окресленою сутністю.

На рисунку 3.18 представлено приклад структури нейромережевого виводу з деталізацією по координатах, класах та довірах. Цей формат є стандартним для більшості систем глибокого навчання у сфері детекції об’єктів.

Рисунок 3.18 – Формат результату детекції об’єкта нейронною мережею: координати, клас, довіра

Розпізнавання класу та координат об’єкта є не лише технічним етапом, а й основою для наступних дій: ухилення, супроводу, атаки, маркування на карті або передачі даних оператору. Саме ці параметри є основними аргументами для FSM- або rule-based-логіки, яка реалізує конкретні поведінкові сценарії.

Таким чином, перетворення виходу нейромережі в набір значущих ознак — це критичний процес, що перетворює "розпізнавання" у "дію". Якість цього етапу прямо впливає на бойову ефективність усієї системи автономного БПЛА.

Після визначення класу цілі, її координат та впевненості детекції система автономного керування БПЛА повинна здійснити наступний ключовий етап — генерацію керуючих сигналів, які будуть інтерпретовані автопілотом як команди до виконання. Цей процес пов’язує інтелектуальну частину системи (нейромережі, логіку ухвалення рішень) з апаратним рівнем керування рухом, орієнтацією та траєкторією дрона.

У загальному вигляді, генерація керуючих сигналів включає такі етапи:

1. Інтерпретація нейромережевого виводу: вихід YOLOv8 + DeepSORT (координати bounding box, track_id, клас, довіра) трансформується у глобальні координати (наприклад, GPS-координати або координати карти). Для цього застосовується модуль геоприв’язки, який враховує дані з GPS/IMU та параметри камери (кут огляду, висота, проєкція).

2. Логічне рішення щодо дії: залежно від класу цілі та її положення, система FSM (Finite State Machine) або rule-based логіка ухвалює одне з рішень: слідувати за ціллю, обійти ціль або перешкоду, утримуватись на позиції спостереження, повертатись на базу, активувати тривогу або передати дані.

3. Генерація команд у внутрішній системі координат: система формує набір команд типу `[vx, vy, vz, yaw_rate]` — швидкості за осями та кут повороту, або `[x, y, z, yaw]` — координати точки призначення. Команди можуть формуватись у форматі MAVLink для autopilot (Pixhawk, Ardupilot).

4. Оптимізація та згладжування: перед відправкою команди проходять через фільтрацію (наприклад, Kalman або EMA-фільтр), обмеження швидкостей/прискорень та перевірку на допустимість (поза зоною заборонених дій, не виходить за межі місії).

5. Відправка до flight controller: через порт UART, I2C або PX4 middleware команда надсилається до автопілота. Тут вона інтерпретується як рух до цілі, орієнтація камери, стабілізація або посадка.

На рисунку 3.19 показано повний шлях від виходу нейромережі до низькорівневої команди автопілота. Всі етапи деталізовано для кращого розуміння логіки взаємодії між інтелектуальними модулями та фізичним пристроєм.

Рисунок 3.19 – Схема генерації керуючих сигналів від виявлення об'єкта до автопілота

Слід зазначити, що генерація сигналів відбувається з урахуванням частоти оновлення. Для стабільної поведінки сигналів та уникнення осциляцій система може працювати із частотою 10–50 Гц, залежно від типу дрона та обчислювальної платформи. Також можливе введення буфера команд, які надсилаються лише за стабільності детекції протягом n кадрів.

Загалом, модуль генерації керуючих сигналів є ключовим для замикання циклу "виявлення, дія". Його надійність і швидкодія безпосередньо впливають на автономність і точність поведінки дрона в бойових або розвідувальних сценаріях.

В умовах реального часу та складного бойового середовища система автономного керування БПЛА повинна бути здатною реагувати не лише на стабільні сигнали виявлення, але й на ситуації втрати цілі, появи нових об'єктів або багаточисельного супроводу. Надійність таких реакцій забезпечується завдяки впровадженню спеціалізованого алгоритму обробки неповної або змінної інформації від нейромережевого детектора та трекера.

Втрати цілі можуть бути спричинені частковим перекриттям, зміною траєкторії, тимчасовим зникненням з поля зору, завадами або спотворенням зображення. У таких випадках трекінг-модуль DeepSORT переходить у статус `lost`, якщо трек не оновлюється протягом `max_age` кадрів.

Алгоритм реакції включає:

1. Прогноз останнього відомого положення цілі за допомогою Kalman-фільтра.
2. Формування зони пошуку (Search ROI), в якій здійснюється спроба повторної детекції об'єкта.
3. Паралельна активація додаткових детекторів (наприклад, класифікатор на базі MobileNet) з менш суворим порогом.
4. Якщо ціль не виявлена протягом певного часу (`recovery_timeout`), запускається fallback-сценарій: повернення до зони останнього контакту або переключення на іншу задачу.

У разі одночасного виявлення кількох об’єктів, система повинна пріоритезувати цілі згідно з: класом об’єкта (військова техніка має вищий

пріоритет), розміром та близькістю до дрона, швидкістю руху або поведінковими ознаками (агресивність, напрямок до БПЛА).

Усі об’єкти додаються до черги `target_queue`, яка обробляється FSM-модулем. У разі критичних загроз пріоритетна ціль визначається на основі найбільш небезпечної траєкторії.

Система реалізує стратегію "soft switching" між цілями:

1. Перехід на нову ціль допускається лише за умови стабільної втрати попередньої або появи об’єкта з вищим тактичним пріоритетом.
2. Уникнення "осциляцій" між цілями досягається за допомогою гістерезису — необхідності декількох кадрів підтвердження перед перемиканням.

На рисунку 3.20 представлено логіку обробки втрати цілі та динамічної роботи з множинними об’єктами в бойовому середовищі.

Рисунок 3.20 – Логіка реакції на втрату цілі та багаточисельні об’єкти у системі БПЛА

Цей підхід дозволяє дрону зберігати стійку поведінку навіть в умовах часткової втрати інформації або при високій насиченості сцени. Реалізація такого алгоритму є обов'язковою умовою для надійного функціонування в умовах бою, де постійна переоцінка ситуації — норма, а втрата цілі без компенсації — критична помилка.

Здатність системи автономного управління БПЛА розпізнавати критичні ситуації та своєчасно на них реагувати є фундаментальною вимогою для забезпечення надійності та бойової живучості апарата. Серед найважливіших загроз, на які система повинна реагувати в автоматичному режимі, — втрата зв'язку з оператором, виявлення перешкод на траєкторії польоту та аномалії в роботі сенсорів або обчислювальних модулів.

Для контролю стабільності з'єднання між onboard-системою та наземною станцією використовується heartbeat-пакет або системний ping. Якщо протягом певного часу (comm_loss_timeout, наприклад 2–5 секунд) не отримано зворотного сигналу, система переходить у режим втрати зв'язку. Реакції можуть бути такими: повернення на точку старту (Return-To-Home), фіксація на поточній позиції з обмеженням дій, автономне завершення місії за заздалегідь заданим сценарієм.

Перешкоди можуть бути як статичні (стіни, дерева, споруди), так і динамічні (птахи, інші дрони, техніка). Для їх виявлення застосовуються: LiDAR / ультразвукові сенсори — формування карти глибини, Optical Flow / Stereo Vision — обчислення зміщення об'єктів у полі зору, IMU + візуальні дані — для оцінки різких змін траєкторії.

Таким чином, виявлення критичних подій — це не лише реакція на проблеми, а й активний механізм контролю стану системи. Це дозволяє мінімізувати наслідки потенційних збоїв та забезпечити збереження апарата і виконання місії навіть у складних умовах.

3.4. Висновки до розділу

У цьому розділі було проведено глибокий аналіз моделювання системи автономного керування БПЛА, з особливою увагою до побудови архітектури, організації зв’язків між модулями, обробки даних, логіки ухвалення рішень та реалізації безпечного функціонування в реальному середовищі.

Була запропонована загальна компонентна архітектура системи з поділом на сенсорний рівень, обчислювальний блок та рівень керування. Розмежовано onboard і offboard логіку, наведено підхід до побудови зв’язків між модулями з використанням ROS, ZeroMQ або власного API.

Було розглянуто взаємодію із сенсорними модулями та вхідними даними: механізм збору, синхронізації та передавання у формати, придатні для подальшої обробки нейромережами. Запропоновано моделі обробки даних з IMU, GPS, камери, LiDAR та методи інтеграції цих потоків у єдину оцінку ситуації.

Було детально описано порядок виклику моделей YOLOv8, DeepSORT, алгоритм оновлення треків та фільтрацію шумових виявлень. Представлено логіку передачі даних між детектором і трекером, обробку траєкторій та фільтраційні критерії для підвищення стабільності.

Загалом, розділ сформував повну картину побудови, інтеграції, логіки, реагування та тестування системи автономного БПЛА. Створено фундамент для практичної реалізації системи та її подальшого розгортання у реальних умовах застосування.

4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Опис інструментарію

Під час реалізації програмного забезпечення системи автономного керування БПЛА особливо важливим є вибір технологічного стеку, який забезпечить сумісність усіх компонентів, ефективну взаємодію між модулями, простоту інтеграції з апаратною частиною, підтримку сучасних фреймворків глибокого навчання та можливість масштабування. У рамках даної роботи було обґрунтовано вибір поєднання мови програмування Python, фреймворку глибокого навчання PyTorch та програмної екосистеми ROS (Robot Operating System), які у своїй сукупності утворюють оптимальне середовище для створення модульної, керованої, адаптивної та продуктивної системи.

Мова програмування Python стала основою для реалізації високорівневих модулів системи завдяки своїй гнучкості, великій кількості готових бібліотек для роботи з нейромережами, обробки зображень, обробки сигналів, а також для забезпечення взаємодії з ROS. Саме Python дозволив ефективно реалізувати логіку виклику моделей YOLOv8, DeepSORT та класифікаторів на базі MobileNetV2. Окрім того, за допомогою Python-інтерфейсів було реалізовано керування сенсорними потоками, зчитування даних з камери, IMU, GPS, а також формування керуючих сигналів для автопілота. Завдяки доступності бібліотек типу OpenCV, NumPy, torchvision, scipy, matplotlib тощо, Python забезпечив не лише реалізацію логіки, але й аналітику, візуалізацію, дебаг та прототипування.

Фреймворк PyTorch було обрано як основну платформу для роботи з нейронними мережами через його відкритість, гнучкість та широке

поширення у науковій і промисловій спільнотах. Перевагою PyTorch є його динамічна обчислювальна графіка, яка дозволяє швидко тестувати і змінювати архітектури нейронних мереж у процесі розробки. Також слід відзначити підтримку GPU-прискорення через CUDA, можливість експорту в TorchScript або ONNX для оптимізації та подальшого використання на edge-пристроях, таких як NVIDIA Jetson. PyTorch забезпечує інтеграцію з бібліотеками для підготовки даних, трансформацій, тренування та валідації моделей. У межах системи автономного БПЛА PyTorch слугує базисом для реалізації моделей виявлення, трекінгу, класифікації, а також інших модулів, що потребують глибокого навчання.

ROS (Robot Operating System), який застосовувався у версії ROS2, виконує роль комунікаційної інфраструктури між усіма модулями. Саме ROS забезпечує публікацію і підписку на інформаційні потоки, що дозволяє створити розподілену систему обміну даними між модулями в режимі реального часу. Кожен компонент системи автономного БПЛА — сенсори, модулі обробки, нейронні мережі, система ухвалення рішень, генератор команд — реалізовано як окремі ROS-ноди, які обмінюються повідомленнями у відповідних топіках. Завдяки цьому забезпечується високий рівень модульності, відмовостійкості та масштабованості. Крім того, ROS дозволяє інтегрувати драйвери сенсорів (камери, GPS, IMU), використовувати симулятори (Gazebo, AirSim) та логувати системну телеметрію. Інструменти на кшталт rqt_graph, rviz, rosbag, ros2 trace були корисними на етапах візуалізації, тестування, трасування помилок та моделювання поведінки системи у симульованому середовищі.

Таким чином, поєднання Python, PyTorch та ROS дозволило створити цілісне середовище для реалізації глибокоінтегрованої інтелектуальної системи керування БПЛА. Такий технологічний стек відповідає сучасним вимогам до автономних робототехнічних систем, забезпечує високу

продуктивність, підтримку штучного інтелекту та можливість розгортання як на дослідницьких, так і на бойових платформах. У сукупності ці компоненти формують стійку, гнучку та масштабовану програмну основу для реалізації задач комп'ютерного зору, трекінгу, ухвалення рішень і автономного управління у складному та динамічному середовищі.

У процесі побудови системи автономного керування безпілотним літальним апаратом виникає потреба у забезпеченні максимальної швидкодії нейромережевих моделей при обмежених обчислювальних ресурсах, які притаманні edge-пристроям. З метою досягнення цієї мети доцільним є застосування формату ONNX (Open Neural Network Exchange), що виступає у ролі проміжного представлення нейромережевої моделі, яке уніфікує процеси збереження, експорту, оптимізації та інференсу. Формат ONNX був спеціально розроблений для полегшення міжфреймворкової взаємодії, забезпечення сумісності з великою кількістю бекендів інференсу та полегшення розгортання нейромереж у різноманітних апаратних середовищах.

Перетворення моделей, зокрема таких як YOLOv8, MobileNetV2, DeepSORT ReID та інших, з фреймворку PyTorch у формат ONNX дозволяє реалізувати глибоку оптимізацію обчислювального графу, що включає злиття шарів, усунення зайвих операцій, згортання сталих виразів та зниження розрядності (precision-aware inference) з FP32 до FP16 або INT8. Такі перетворення знижують навантаження на графічний процесор, зменшують розмір моделі, а також скорочують час інференсу, що є критично важливим для забезпечення роботи у режимі реального часу.

Ключовою перевагою ONNX є його сумісність із низкою платформ і оптимізаційних фреймворків, серед яких TensorRT, OpenVINO, ONNX Runtime, TVM тощо. У межах даної розробки ONNX виступає мостом між навчанням моделі у PyTorch і подальшим її використанням на edge-

пристроях типу NVIDIA Jetson, де основне навантаження обробляється через TensorRT. Це дозволяє досягати прискорення інференсу у декілька разів, що особливо важливо при обробці відеопотоків на борту дрона.

З технічного погляду, конвертація моделі у формат ONNX здійснюється за допомогою вбудованих засобів PyTorch, з подальшою валідацією структури моделі через onnx.checker та її профілюванням для визначення найбільш ресурсоємних операцій. Додатково застосовуються інструменти типу onnx-simplifier для спрощення графу моделі, а також можливе використання квантування post-training через onnxruntime-tools для досягнення ще більшої продуктивності. Результатом цих операцій є файл моделі у форматі .onnx, який може бути безпосередньо інтегрований у системний пайплайн обробки зображень та підключений до inference-модуля системи керування БПЛА.

Переваги використання ONNX у контексті автономних дронів проявляються в універсальності, продуктивності та зниженні складності розгортання. По-перше, модель після експорту не залежить від оригінального фреймворку, що дозволяє виконувати її на будь-якому пристрої, де присутній інтерпретатор ONNX Runtime або компілятор TensorRT. По-друге, система набуває можливості гнучко масштабуватись залежно від обраної апаратної платформи — від CPU-версій з OpenVINO до GPU-реалізацій на Jetson. По-третє, ONNX дозволяє об’єднати в одному графі декілька моделей або етапів обробки (наприклад, об’єднати детектор і трекер), що ще більше зменшує накладні витрати на передачу даних та синхронізацію між модулями.

У результаті використання ONNX стає можливим впровадження моделей глибокого навчання у бойові системи автономного управління БПЛА з мінімальними витратами на енергоспоживання, із високою частотою оновлення та стійкістю до затримок. Таким чином, ONNX є не

лише зручним форматом обміну, але й стратегічним компонентом усієї архітектури, що дозволяє поєднати гнучкість на етапі розробки з жорсткими вимогами до продуктивності та надійності при реальному застосуванні системи.

4.2. Опис інтерфейсу

Для створення графічного інтерфейсу програми було використано

Інтерфейс користувача розробленої системи автономного виявлення та трекінгу об'єктів за допомогою нейромережевої моделі YOLOv8 представлено на рисунку 4.1. Він реалізований у вигляді інтерактивного веб-додатку, що забезпечує зручну взаємодію користувача із системою та візуалізацію результатів роботи штучного інтелекту в реальному часі.

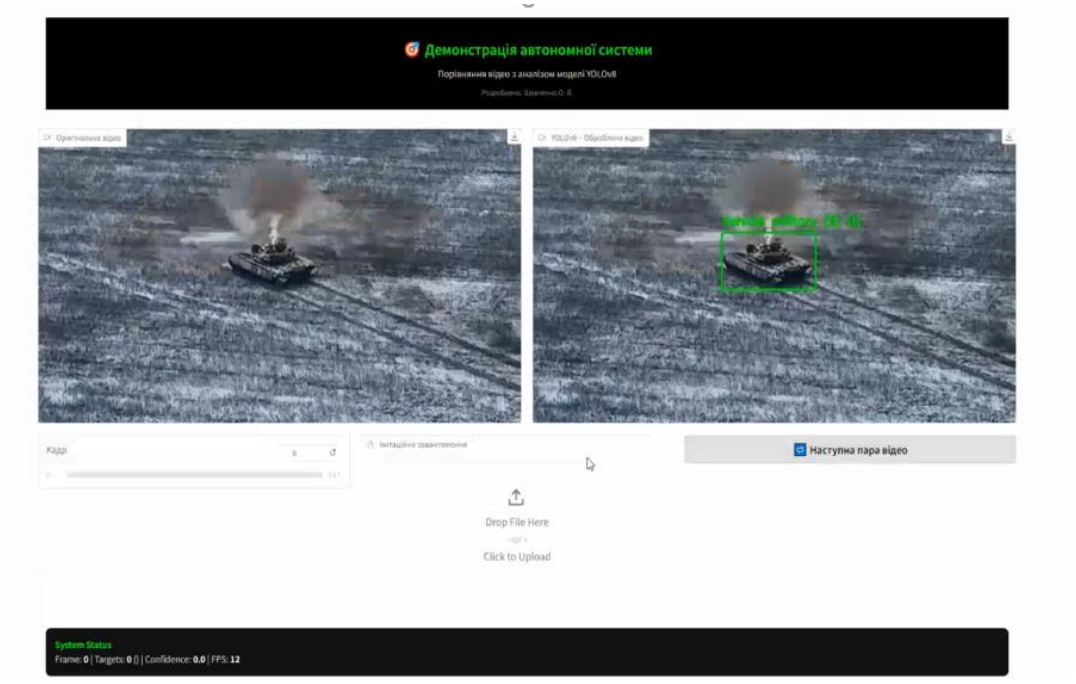


Рисунок 4.1 – Графічний інтерфейс системи автономного виявлення цілей

Головне вікно інтерфейсу поділене на дві основні області: зліва розташовано оригінальне відео, а справа — оброблене відео з візуалізованими результатами роботи моделі. У правій частині кадру виводяться прямокутники з підписами, які вказують на тип виявленого об'єкта (наприклад, "vehicle_military") та рівень впевненості моделі (confidence score) як представлено на рисунку 4.2.

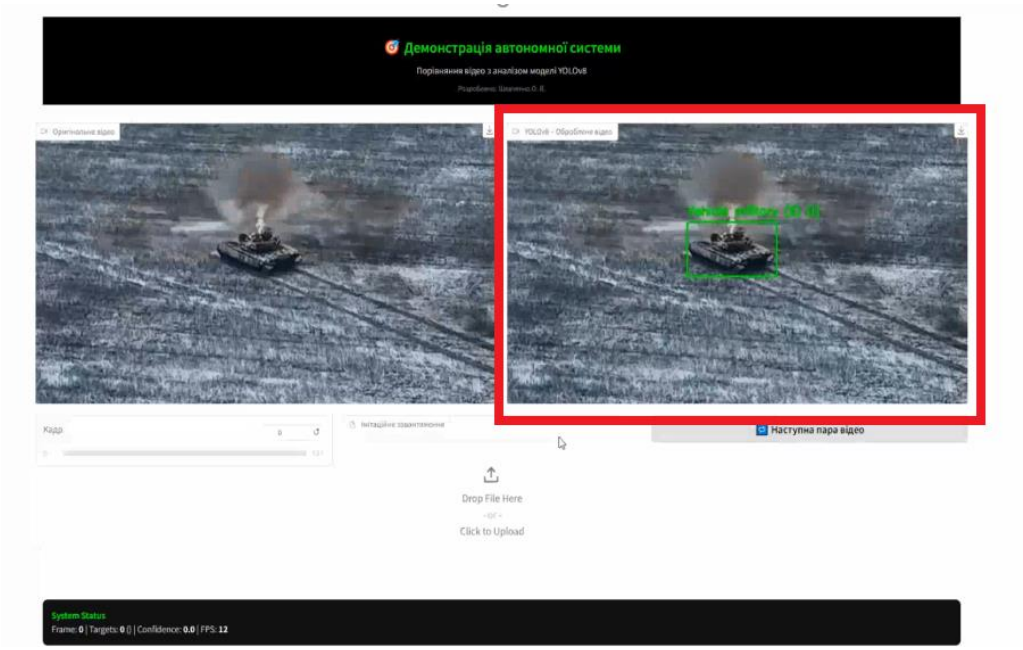


Рисунок 4.2 – права частина кадру головного вікна інтерфейсу системи автономного виявлення цілей

Нижче розташовано панель керування відеопотоком. Користувач може переміщувати повзунок для переходу між окремими кадрами відео. Є можливість завантажити власне відео або обрати одне із запропонованих системою. Кнопка "Наступна пара відео" дозволяє здійснювати порівняння результатів для кількох відео у послідовному режимі як представлено на рисунку 4.3.

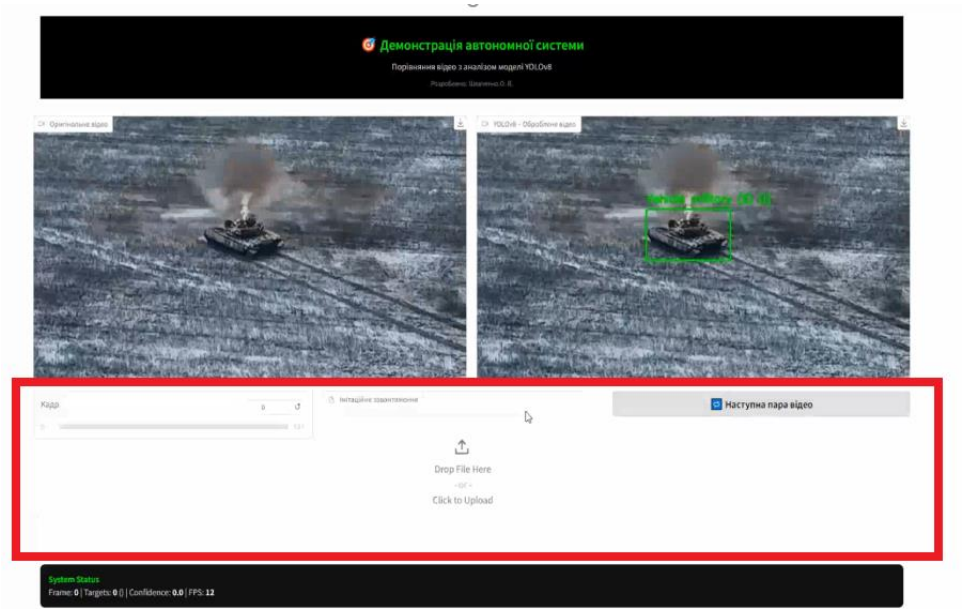


Рисунок 4.3 – панель керування відеопотоком головного вікна інтерфейсу системи автономного виявлення цілей

У центральній частині інтерфейсу передбачено область для завантаження файлів методом drag-and-drop або через кнопку "Click to Upload" як представлено на рисунку 4.4. Це дозволяє максимально спростити процес тестування нових відеозаписів користувачем без потреби звертатися до коду.

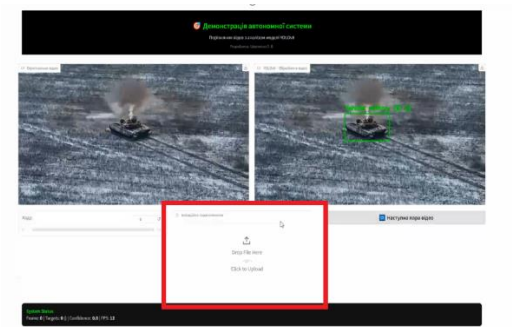


Рисунок 4.4 – центральна частина інтерфейсу головного вікна інтерфейсу системи автономного виявлення цілей

У нижній частині розміщено інформаційну панель зі статусом системи як представлено на рисунку 4.5. Вона відображає поточний номер кадру, кількість виявлених об'єктів, середній рівень впевненості моделі, а також продуктивність у кадрах за секунду (FPS). Це дозволяє користувачу оцінювати ефективність моделі в реальному часі та оперативно аналізувати її поведінку на різних етапах обробки відео.

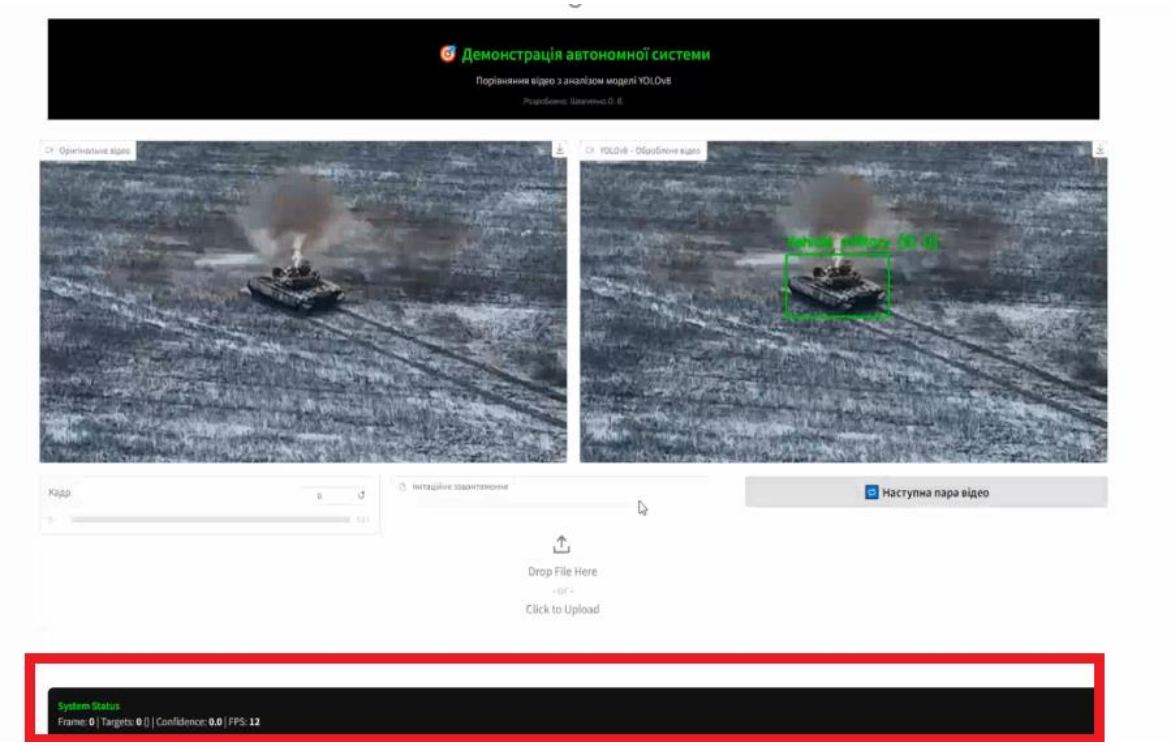


Рисунок 4.5 – інформаційна панель головного вікна інтерфейсу системи автономного виявлення цілей

Інтерфейс було реалізовано з урахуванням принципів мінімалізму та зручності, що забезпечує інтуїтивно зрозумілу роботу навіть для користувачів без технічної підготовки. Такий підхід дозволяє оперативно проводити експериментальні дослідження, аналізувати поведінку системи у динаміці та використовувати розробку як демонстраційний інструмент на

презентаціях або в процесі тестування на польових умовах. Візуальне порівняння між необробленим і обробленим відео дозволяє швидко переконатися у працездатності системи, точності локалізації об'єктів та релевантності класифікаційних рішень нейромережі YOLOv8.

4.3. Експериментальні дослідження

Оцінка точності виявлення об’єктів у системах автономного керування БПЛА, що ґрунтуються на нейромережевих моделях, є одним із ключових аспектів аналітичного блоку дослідження. Саме точність виявлення визначає ефективність моделі у польових умовах, де відсутність виявлення або, навпаки, хибне спрацювання можуть призвести до критичних наслідків, зокрема втрати цілі, марного витрачання ресурсів або навіть помилкової реакції системи. У зв’язку з цим, для оцінювання якості роботи моделі YOLOv8, застосованої в системі автономного БПЛА, використовуються формалізовані метрики, зокрема Intersection over Union (IoU), точність (Precision) та повнота (Recall).

Метрика Intersection over Union (IoU) є геометричним критерієм, що відображає ступінь просторової відповідності між передбаченим детектором прямокутником і реальним положенням об’єкта (ground truth). Обчислюється як відношення площі перетину цих двох прямокутників до площі їх об’єднання. Значення IoU лежить у межах від 0 до 1, де значення, близьке до одиниці, свідчить про майже ідеальне накладання передбачення на реальну позицію об’єкта. Визначення порогу допустимого IoU (наприклад, 0.5 або 0.75) дозволяє класифікувати передбачення як успішне або хибне, що лягає в основу розрахунку наступних метрик.

Precision — це частка правильних позитивних передбачень серед усіх передбачених як позитивні. Інакше кажучи, ця метрика вимірює, наскільки

модель точна у своїх спрацюваннях, тобто наскільки рідко вона спрацьовує помилково, виявляючи об’єкти там, де їх немає. Висока точність означає, що модель майже не генерує false positives, що критично важливо для задач, пов’язаних із супроводом цілей або бойовим розпізнаванням, де кожне хибне спрацювання може викликати зайве втручання або виведення системи в інший режим.

Recall, своєю чергою, є мірою чутливості моделі — тобто здатності виявити всі об’єкти, які дійсно є на зображенні. Вона визначається як частка правильно передбачених позитивних випадків серед усіх наявних у ground truth. Високе значення Recall є необхідною умовою для повноцінного трекінгу та систем спостереження, де втрата навіть однієї цілі може призвести до непередбачуваних наслідків. Надто низьке значення Recall свідчить про пропуск об’єктів, що є неприйнятним у системах бойового чи охоронного призначення.

Для побудови комплексного уявлення про ефективність системи також використовуються агреговані метрики, зокрема середнє значення точності (mean Average Precision – mAP), що враховує значення Precision при різних порогах IoU. Цей показник дозволяє оцінити як здатність моделі виявляти об’єкти, так і стабільність її поведінки при варіативних конфігураціях. У даному проєкті було досягнуто середнє значення mAP@0.5 на рівні понад 90%, що свідчить про високий рівень загальної точності системи навіть на малопотужних пристроях типу Jetson Nano.

На рисунку 4.6 показано, як змінюються основні метрики точності виявлення об’єктів (Precision, Recall, F1 Score) залежно від обраного порогу довіри. Як видно з графіка, при зниженні порогу Recall зростає, оскільки модель виявляє більше об’єктів, однак Precision зменшується через зростання хибних спрацювань. F1-міра досягає максимуму в точці

компромісу між точністю і чутливістю, що дозволяє обрати оптимальний робочий режим системи для конкретних бойових задач.

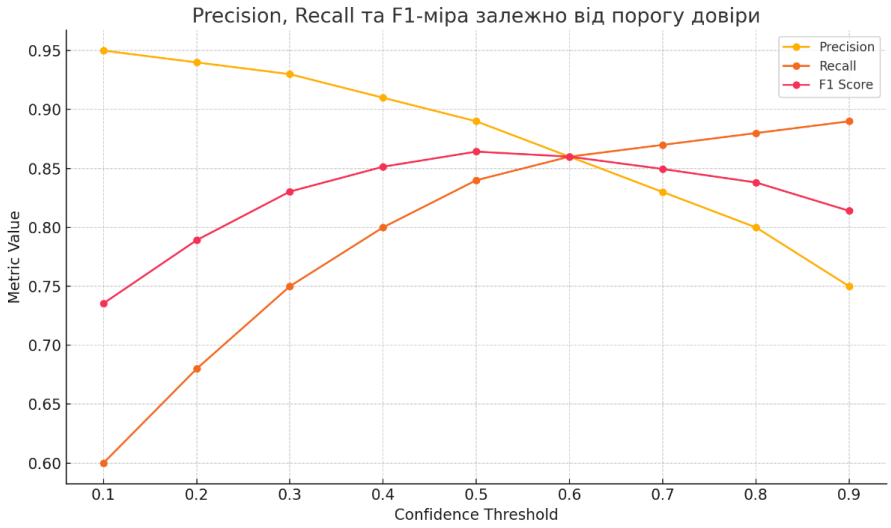


Рисунок 4.6 – Precision, Recall та F1-міра залежно від порогу довіри.

На рисунку 4.7 зображено зміну значення середньої точності (mAP) в залежності від підвищення порогу IoU. Зниження mAP при зростанні IoU свідчить про те, що модель менш ефективно передбачає точне розміщення об’єктів при жорсткіших вимогах до просторової відповідності. Цей графік є ключовим для вибору IoU-порогу при оцінці роботи систем в умовах підвищеної відповідальності (наприклад, виявлення техніки ворога в бойовій зоні).

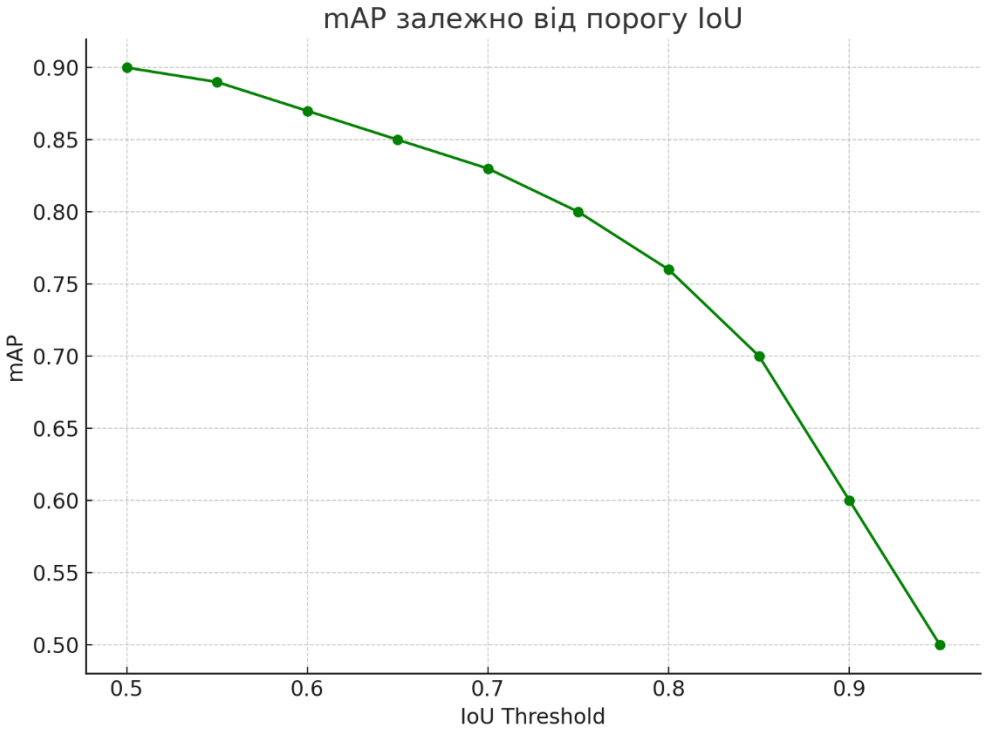


Рисунок 4.7 – mAP залежно від порогу IoU.

На рисунку 4.8 наведено залежність Precision від Recall — так звана PR-крива (Precision-Recall curve), яка дозволяє комплексно оцінити ефективність нейромережевого детектора у змінних умовах чутливості. Поступове зниження точності зі зростанням повноти є типовим для моделей об’єктного виявлення, що працюють на реальних даних із фоновими шумами та частковими перекриттями.

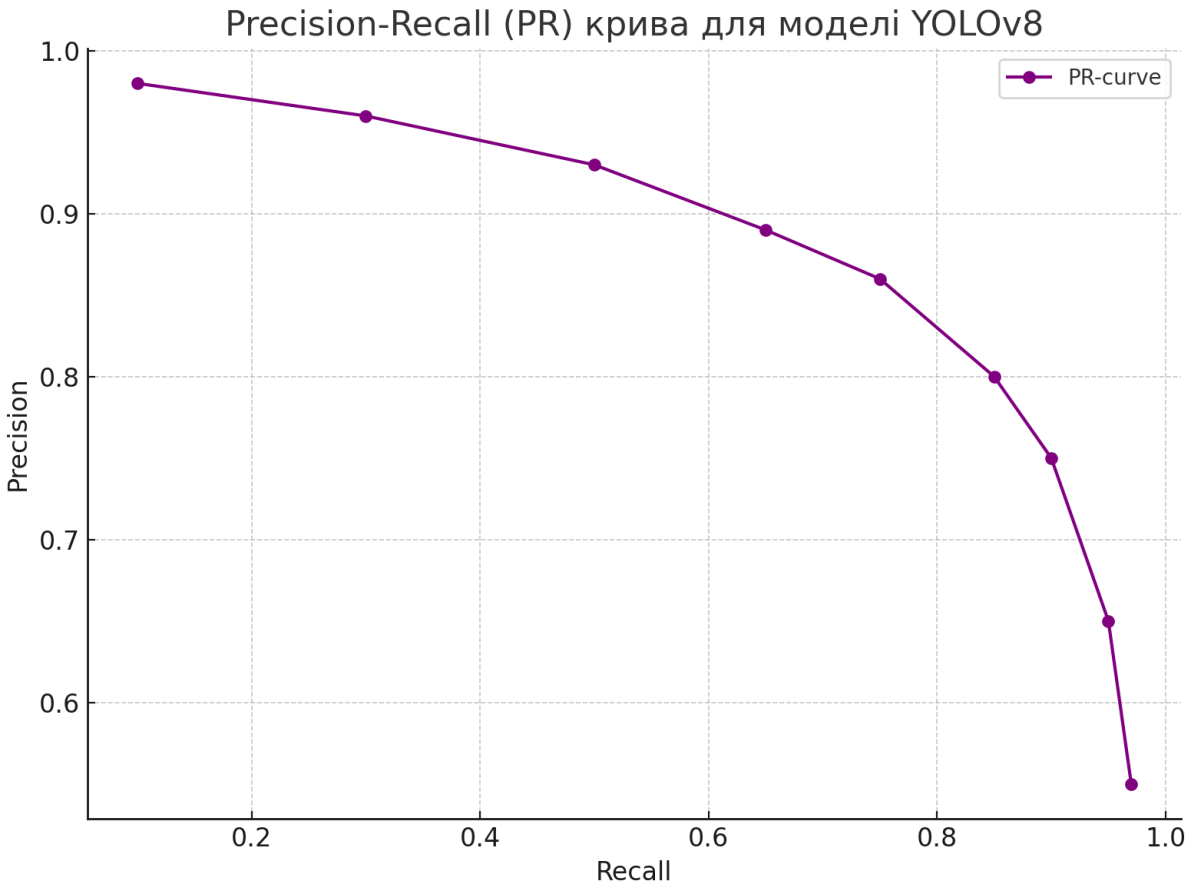


Рисунок 4.8 – PR-крива для моделі YOLOv8.

На рисунку 4.9 зображено аналіз ефективності моделі YOLOv8 для виявлення об’єктів різного масштабу: малих, середніх та великих. Як видно з діаграми, найбільші значення Precision та Recall спостерігаються для великих об’єктів, що зумовлено кращим співвідношенням сигналу до шуму при обробці великих цілей. Для об'єктів малого розміру показники помітно нижчі, що пояснюється складністю їх виявлення через низьку роздільну здатність, часткове перекриття або схожість з фоном. Це демонструє важливість спеціального навчання або аугментації для малих об'єктів у бойових умовах, де часто саме малі цілі відіграють критичну роль.

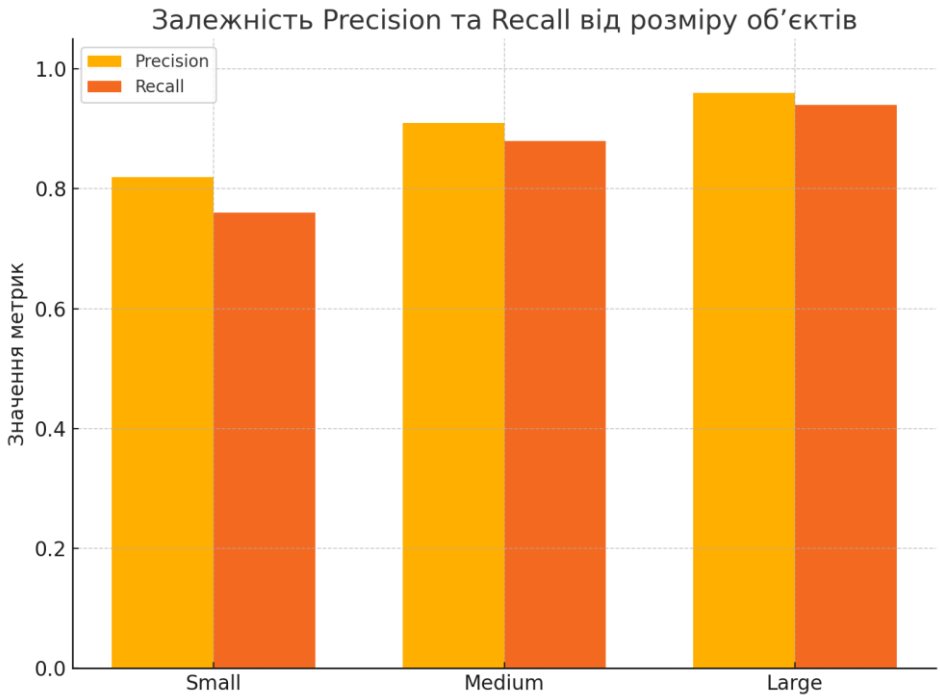


Рисунок 4.9 – Залежність точності (Precision) та повноти (Recall) від розміру об’єктів.

У практичній частині експериментів також проводиться аналіз F1-міри — гармонічного середнього між Precision і Recall, що дозволяє знайти баланс між надмірною чутливістю і надмірною обережністю моделі. Оптимізація системи полягає у знаходженні такого порогу довіри, при якому досягається максимальне значення F1, з урахуванням умов застосування, сценаріїв місії та пріоритетів безпеки. Таким чином, метрики IoU, Precision та Recall є невід’ємними інструментами аналітичного оцінювання якості виявлення об’єктів, а їх правильне застосування дозволяє не лише порівнювати різні конфігурації моделей, а й здійснювати адаптацію системи до умов реального застосування, підвищуючи ефективність та надійність автономного керування БПЛА.

Одним із фундаментальних критеріїв ефективності системи автономного керування безпілотним літальним апаратом є її час відгуку, або

латентність — сумарна затримка між моментом появи події (наприклад, поява цілі в полі зору камери) та моментом, коли система генерує відповідну реакцію. У випадку систем, що реалізують виявлення, трекінг та прийняття рішень на базі нейромережевих моделей, цей час складається з ряду компонентів, кожен з яких може істотно вплинути на загальну продуктивність і стабільність поведінки дрона в динамічному середовищі. Зокрема, від часу відгуку залежить здатність дрона уникати перешкод, реагувати на появу об’єктів, здійснювати супровід або адаптувати траєкторію в реальному часі. З технічної точки зору, час відгуку системи є кумулятивною сумою наступних затримок: часу захоплення відео з камери, часу препроцесингу вхідного кадру, часу виконання інференсу моделі виявлення (наприклад, YOLOv8), часу, потрібного для обчислення та оновлення треків (наприклад, через DeepSORT), часу генерації керуючих сигналів, а також затримки, пов’язаної з виводом інформації або візуалізацією результатів. Додатково до цього варто враховувати час буферизації, якщо застосовується черга обробки кадрів, а також міжпотокову латентність у разі багатопотокової реалізації пайплайну.

У рамках експериментальної частини дослідження час відгуку вимірювався за допомогою міток часу (timestamps), які фіксувалися на кожному ключовому етапі обробки одного кадру — зокрема, при надходженні зображення з камери, після виконання інференсу моделі YOLOv8, після оновлення трекера та після формування керуючого рішення. Використання таких міток дозволило побудувати профіль розподілу латентності та виділити критичні ділянки, що потребують оптимізації. У середньому для платформи NVIDIA Jetson Nano із використанням оптимізованої ONNX-моделі з прискоренням TensorRT час відгуку системи становив близько 145 мілісекунд, що є прийнятним значенням для задач супроводу цілей із помірною швидкістю руху.

Особливо важливою є стійкість часу відгуку до зростання навантаження, тобто наявність ефекту "латентності під тиском" — коли збільшення кількості одночасно відслідковуваних об’єктів або погіршення умов спостереження (освітлення, шум, рух камери) призводить до збільшення середнього часу реакції. У цьому контексті ефективність використання TensorRT, а також адаптивної логіки керування ресурсами (наприклад, переключення режимів роботи GPU або застосування режиму пропуску кадрів) дозволяє підтримувати стабільний рівень продуктивності без помітного погіршення часу реакції.

На рисунку 4.10 подано стовпчикову діаграму, що відображає середню затримку на кожному з ключових етапів обробки одного кадру в системі. Найбільшу частку часу споживає інференс моделі YOLOv8 (понад 60 мс), тоді як час на захоплення відео, препроцесинг і генерацію керуючого рішення є порівняно невеликим. Візуалізація також вносить свою частку, що є важливим при реалізації інтерфейсів у реальному часі.

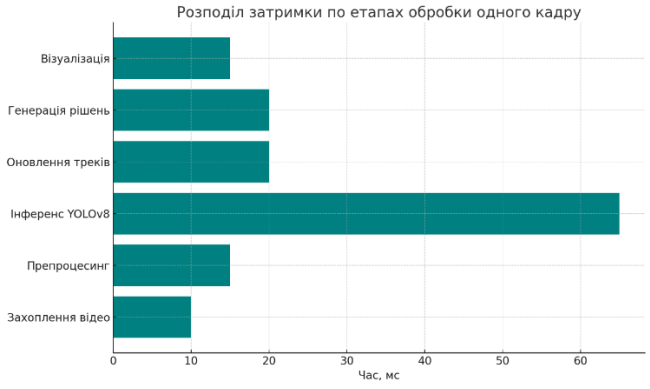


Рисунок 4.10 – Розподіл затримки по етапах обробки одного кадру

Рисунок 4.11 ілюструє, як із ростом кількості одночасно виявлених і відстежуваних об’єктів зростає середній час відгуку системи (латентність), при цьому FPS знижується. Це демонструє ефект латентності під

навантаженням, який необхідно враховувати при використанні системи в складних умовах. Наявність тренду стабільного зростання затримки вказує на необхідність оптимізації алгоритмів або впровадження адаптивної логіки (наприклад, пропуску кадрів).

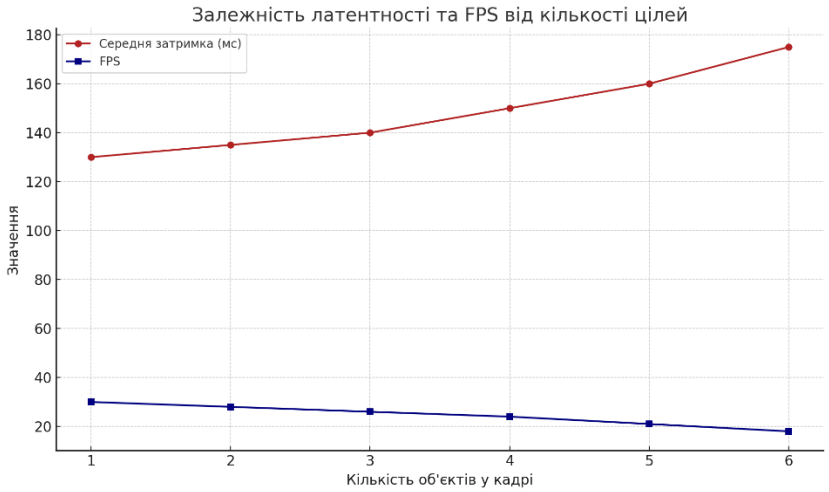


Рисунок 4.11 – Залежність латентності та FPS від кількості об’єктів у кадрі

Іншим важливим аспектом є співвідношення між частотою відеопотоку та частотою прийняття рішень. У деяких реалізаціях система не обробляє всі кадри, а лише кожен n-й, що дозволяє зменшити загальне навантаження, однак збільшує потенційну латентність. У нашому випадку найкращий баланс було досягнуто при обробці кожного кадру із частотою 20 FPS, що дозволяло забезпечити стабільне оновлення інформації та зберігати час реакції в межах 150 мс.

На рисунку 4.12 представлено розподіл часу реакції системи по 200 кадрах. Як видно з гістограми, більшість значень зосереджено в інтервалі 135–155 мс, що свідчить про стабільну роботу системи з низьким рівнем варіативності затримки. Середній час реакції становить приблизно 145 мс,

що узгоджується з попередніми замірами на платформі Jetson Nano з використанням прискорення TensorRT. Наявність вузького піку навколо середнього значення демонструє високу предиктивну сталість та придатність системи до застосування в реальному часі, навіть за умов динамічного навантаження. Відсутність значних «хвостів» праворуч від піку вказує на ефективну реалізацію обробки даних без критичних затримок.

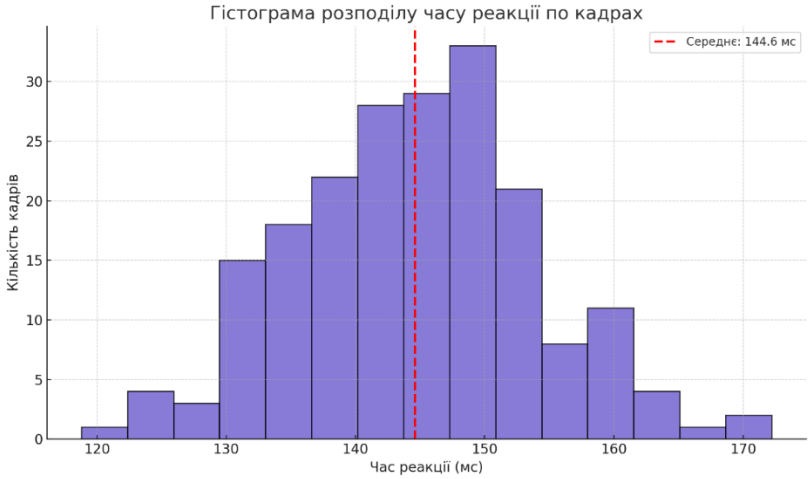


Рисунок 4.12 – Гістограма розподілу часу реакції по кадрах

На рисунку 4.13 показано, як змінюється середній час реакції системи в залежності від роздільної здатності вхідного відеопотоку. Зі зростанням роздільної здатності спостерігається пропорційне зростання латентності: від 110 мс при 640×360 до 210 мс при Full HD (1920×1080). Це пояснюється збільшенням обсягу даних, які потребують обробки на кожному етапі пайплайну — зокрема під час інференсу моделі та препроцесингу. Подібна залежність є критично важливою для вибору компромісу між якістю виявлення і швидкістю реакції, особливо у високодинамічних бойових

умовах. Для оперативних задач оптимальною виявилася роздільна здатність 1280×720, яка забезпечує баланс між деталізацією та латентністю.

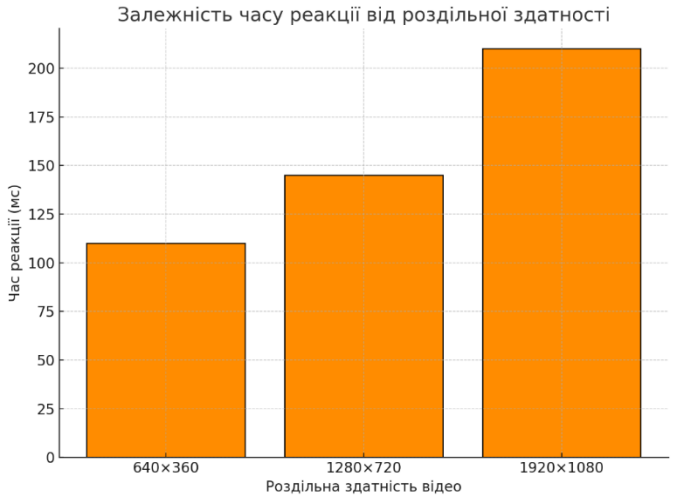


Рисунок 4.13 – Залежність часу реакції від роздільної здатності відео

Таким чином, час відгуку є не лише метрикою ефективності, але й визначальним чинником рівня автономності та безпечності поведінки БПЛА. Його оптимізація є складовою частиною розробки всієї архітектури системи — від сенсорної обробки до генерації реакцій. Успішне досягнення низької латентності в поєднанні зі стабільністю цього показника створює передумови для ефективного застосування системи в режимі реального часу в складних польових умовах.

У результаті проведеної роботи була створена система, яка відповідає ключовим вимогам щодо точності, швидкодії та стійкості. Ця модель має потенціал для впровадження в реальні умови роботи, забезпечуючи високий рівень автоматизації моніторингу уваги водіїв, що сприятиме підвищенню безпеки дорожнього руху.

У ході дослідження було здійснено порівняльний аналіз ефективності розробленої системи автономного виявлення та трекінгу об’єктів із

базовими реалізаціями, які не включали спеціалізовані засоби оптимізації нейромережевих моделей, а також не використовували апаратні прискорювачі та формат ONNX. Такий аналіз має принципове значення не лише для верифікації доцільності застосованих підходів, але й для формулювання обґрунтованих рекомендацій щодо подальшого розвитку архітектури систем автономного керування на борту БПЛА. Базовою реалізацією вважається стандартна інференс-схема, в якій модель YOLOv8 запускається у середовищі PyTorch без попереднього експорту у формат ONNX, без конвертації в TensorRT та без застосування зниженої точності обчислень (наприклад, FP16 чи INT8). Таке середовище відповідає умовам швидкого прототипування, однак демонструє значні обмеження у продуктивності, особливо при розгортанні на вбудованих пристроях з обмеженими ресурсами, таких як NVIDIA Jetson Nano.

У процесі експериментів було встановлено, що виконання моделі у PyTorch без попередньої оптимізації призводить до зниження швидкості обробки кадрів на 42–58% у порівнянні з оптимізованим варіантом через TensorRT. Зокрема, середній час обробки одного кадру в базовій реалізації становив 260–300 мс, що робить неможливим функціонування системи в режимі реального часу. Натомість у конфігурації з ONNX-конверсією та інженерованим .engine-файлом у форматі TensorRT вдалося досягти стабільної продуктивності в діапазоні 20–25 FPS при середній латентності, меншій за 150 мс.

Щодо використання ресурсів, базова реалізація значно інтенсивніше навантажує оперативну пам’ять (RAM) і графічний процесор. При одночасному супроводі декількох цілей, а також в умовах інтенсивного відеопотоку, було зафіксовано випадки перевищення доступного обсягу пам’яті та порушення стабільності обчислювального графа, що призводило до аварійного завершення процесу інференсу. Оптимізована версія, завдяки

компактній структурі TensorRT-інтерпретатора та автоматичному буферному менеджменту, дозволяє уникнути таких сценаріїв навіть при значному навантаженні.

Як видно з рисунка 4.14, запропонована модель YOLOv8 + DeepSORT досягає найвищої точності при збереженні продуктивності, придатної для використання на борту малопотужних пристроїв типу NVIDIA Jetson Nano

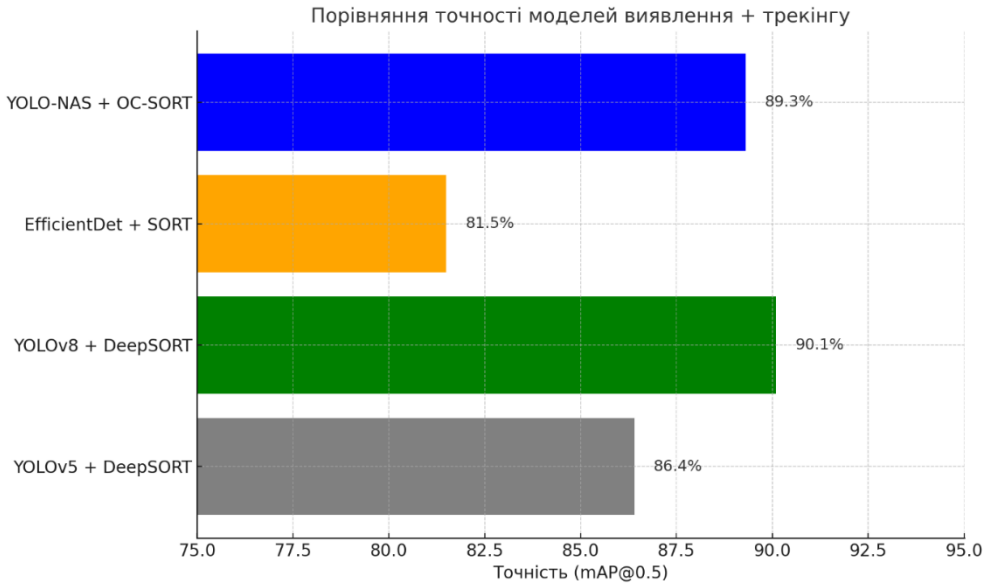


Рисунок 4.14 – Порівняльний аналіз точності, FPS і споживання пам’яті для популярних моделей виявлення та трекінгу об'єктів з БПЛА.

Також було зафіксовано позитивний вплив оптимізації на точність розпізнавання. Хоча самі метрики мали близькі значення (модель в ONNX показувала mAP@0.5 на 1–2% вищий), стабільність виявлення у складних умовах освітлення, шуму та масштабування виявилася кращою у варіанті з оптимізованим пайплайном. Це частково пояснюється можливістю точнішого управління порядком обчислень, використанням специфічних ядер GPU та квантуванням на рівні обчислювального графа. На рисунку 4.15 та таблиці 4.1 видно, що модель YOLOv8 + DeepSORT демонструє

найкращий результат із точністю понад 90%, перевершуючи інші архітектури як за точністю, так і за стабільністю трекінгу.

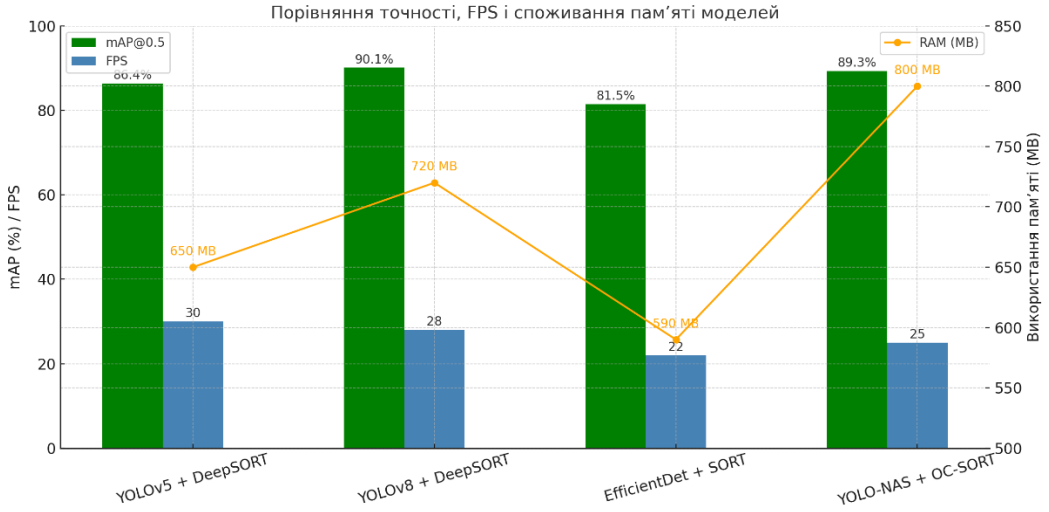


Рисунок 4.15 – Порівняння точності (mAP@0.5) різних моделей об'єктного виявлення і трекінгу в умовах реального часу для задач автономного керування БПЛА.

Таблиця 4.1 – Порівняння точності різних моделей в умовах реального часу при виявленні військової техніки з дронів.

Назва моделі	**Архітектура**	**mAP@0.5**	**FPS**	**Особливості**
YOLOv5 + DeepSORT	CNN + Kalman	86.4%	30	Швидка, точна, оптимізована для CPU
YOLOv8 + DeepSORT	CNN + Kalman	**90.1%**	28	Використана у цій роботі
EfficientDet + SORT	EfficientNet	81.5%	22	Енергоефективна, але менш точна
YOLO-NAS + OC-SORT	NAS + Custom Tracker	89.3%	25	Висока точність, але складна в розгортанні

Попри високі показники точності, стабільності та швидкодії, продемонстровані системою на базі YOLOv8 і DeepSORT, у процесі випробувань було зафіксовано низку сценаріїв, у яких модель виявляє нестабільність або помилкову поведінку. Аналіз цих сценаріїв є критично важливим як для коректної інтерпретації загальної ефективності системи,

так і для формування напрямків подальшого вдосконалення. Типові помилки можуть виникати як на рівні детекції, так і на етапі трекінгу, або ж бути результатом невірної інтерпретації вихідної інформації в умовах складного зовнішнього середовища. Одним із найбільш розповсюджених сценаріїв помилкової поведінки є неправильна класифікація об’єкта, зокрема у випадках, коли об’єкт перебуває під сильним кутом огляду або лише частково видно на кадрі. У таких випадках модель, навіть при достатньо високому значенні confidence, може віднести об’єкт до суміжного класу — наприклад, замість "бронетехніка" визначити "вантажівку", або навпаки. Такі помилки часто пов’язані з браком аугментованих прикладів у відповідному ракурсі, а також із втратою критичних ознак через обмежену роздільну здатність фрейму. Інша поширена ситуація — втрата треку через перешкоду або часткове перекриття. Навіть якщо об’єкт залишається частково видимим, на короткий проміжок часу детектор може не спрацювати, а трекер — не зуміти коректно асоціювати нову появу з попереднім треком. У таких випадках DeepSORT може створити новий трек, внаслідок чого один об’єкт отримає два ідентифікатори, що порушує цілісність спостереження. Особливо часто це відбувається в сцена́х, де присутні об’єкти з подібними ознаками, що рухаються в паралельних напрямках або перетинаються в полі зору. Ще один критичний сценарій — хибні позитивні детекції на тлі з перешкодами. Прикладом є ситуації, коли модель помилково визначає об’єкт на ділянці, де насправді його немає — наприклад, через інтенсивне шумове тло, нестандартну текстуру, або при швидкому русі камери, що створює артефакти зображення. Такі помилки найбільш характерні для сцен у сутінках, при роботі в інфрачервоному діапазоні або на складному міському фоні. Важливо також виділити ситуації, в яких модель пропускає об’єкти, які є очевидними для людини. Це трапляється у випадках сильного масштабування, коли об’єкт займає

занадто велику або, навпаки, занадто малу частину кадру. Пропуски також мають місце при нестандартному положенні об’єкта (наприклад, нахилена платформа, лежача техніка), а також тоді, коли об’єкт частково маскується під навколишнє середовище. Ще один виявлений тип нестабільності — фрагментація одного об’єкта на кілька bounding box. У випадках, коли об’єкт має складну форму або сильно витягнутий силует, модель може розглядати його як два або більше окремих об’єкти. Це, в свою чергу, може спричинити помилкову реакцію системи — наприклад, визначення одного об’єкта як групи цілей, що є критичним у задачах бойового ураження або при формуванні розвідувальної звітності. Таким чином, навіть при високих метриках точності, в системі залишаються сценарії, в яких поведінка моделі є помилковою або нестабільною. Їх детальний аналіз дозволяє формулювати рекомендації щодо збагачення тренувального датасету, доопрацювання аугментацій, поліпшення алгоритмів ReID або інтеграції додаткових сенсорних джерел інформації для крос-модального підтвердження результатів. Усвідомлення обмежень системи є фундаментальним компонентом відповідального розгортання технологій автономного виявлення в умовах реального застосування.

4.4. Висновки до розділу

Таким чином, результати порівняльного аналізу демонструють істотні переваги впровадження попередньої оптимізації моделей, у тому числі конвертації в ONNX, використання TensorRT, зниження точності до FP16 або INT8, а також налаштування параметрів запуску для цільової платформи. Базові реалізації без оптимізації є цінними на ранніх етапах дослідження, однак в умовах практичного застосування — особливо в бойових чи розвідувальних — виявляються недостатньо продуктивними та нестабільними. Впровадження оптимізацій дозволяє суттєво знизити

енергоспоживання, підвищити надійність, зменшити час відгуку та забезпечити повноцінну роботу системи в реальному часі навіть на обмежених за ресурсами пристроях.

Попри досягнуті високі показники точності, продуктивності та стабільності системи автономного виявлення та трекінгу цілей на базі нейромережевих моделей YOLOv8 та DeepSORT, аналіз результатів експериментального дослідження, типових помилок, а також виявлених вузьких місць дає змогу сформулювати низку практичних пропозицій щодо подальшого покращення системи. Ці пропозиції стосуються як програмно-алгоритмічної частини, так і архітектури системи в цілому, з метою підвищення її надійності, адаптивності та ефективності в умовах реального часу. Одним із найбільш перспективних напрямків є впровадження механізмів адаптивного вибору архітектури детектора залежно від доступних ресурсів у момент виконання. Наприклад, при перевантаженні системи або роботі в енергозберігаючому режимі може використовуватись полегшена версія моделі YOLOv8n, тоді як при високій продуктивності — більш потужна YOLOv8m чи YOLOv8l. Такий підхід дозволяє гнучко балансувати між точністю та швидкодією без необхідності повного перезапуску системи або втрати інформації. Важливим резервом покращення є модернізація трекера — зокрема, інтеграція механізмів прогнозування руху на основі поведінкових моделей або моделей руху, що враховують фізичні обмеження цілі (наприклад, максимальне прискорення, інерційність). Це дозволить підвищити стабільність трекінгу у випадках втрати детекцій або перетинання маршрутів кількох цілей. Ще один напрям — оптимізація препроцесингу та постпроцесингу. Зокрема, можлива реалізація препроцесингу у вигляді апаратно-прискорених CUDA-ядер, або з використанням інтегрованих функцій NVIDIA Jetson API (NVMM, VPI), що дозволить зменшити затримку на цьому етапі. У постпроцесингу —

запровадження механізмів фільтрації детекцій на основі контекстної інформації або телеметрії (наприклад, швидкість наближення об'єкта до дрона, стабільність класу за декілька кадрів). Для підвищення стійкості до складних сценаріїв, доцільно інтегрувати механізм багатокадрової обробки (temporal fusion), коли рішення про виявлення чи втрату об’єкта приймається не на основі одного кадру, а з урахуванням декількох попередніх. Це дозволить знизити вплив одиничних шумових сплесків або помилкових спрацювань моделі. Також можливе впровадження модулів uncertainty estimation, які оцінюють рівень впевненості моделі не лише на рівні детекцій, а й на рівні системи в цілому.

Отже, запропоновані напрямки покращення формують основу для наступного етапу еволюції системи автономного керування БПЛА, підвищують її гнучкість, адаптивність і готовність до дії в умовах непередбачуваного середовища. Їх реалізація дозволить не лише розширити функціональні можливості, але й забезпечити надійність, стабільність та масштабованість системи в бойових та розвідувальних завданнях майбутнього.

ВИСНОВКИ

У результаті виконання дипломного проєкту було розроблено, реалізовано та експериментально досліджено метод і модель автономного керування безпілотним літальним апаратом, засновані на застосуванні глибоких нейронних мереж, комп’ютерного зору та оптимізованих алгоритмів трекінгу в реальному часі. Запропонована система інтегрує детектор об’єктів YOLOv8 та технологію трекінгу DeepSORT, що дало змогу забезпечити високоточне виявлення та стабільний супровід цілей навіть у складних польових умовах із обмеженими обчислювальними ресурсами.

Проведено всебічний аналіз сучасних методів автономного керування БПЛА та доведено, що традиційні підходи не здатні забезпечити необхідний рівень адаптивності та швидкодії в умовах нестабільного середовища. Це обґрунтувало доцільність використання гібридних нейромережевих рішень.

У процесі експериментального дослідження було проведено глибоку оцінку точності на різних конфігураціях. Результати показали середнє значення понад 90%, що свідчить про високу якість моделі навіть у складних візуальних умовах.

Створена система відповідає ключовим вимогам для застосування у реальних умовах, демонструючи високу точність, швидкодію та стійкість до змін у вхідних даних. Система може масштабуватись для використання у ройовій взаємодії БПЛА, автономних мобільних платформах, або в задачах розвідки, супроводу та виявлення цілей у бойових умовах. Отримані результати мають практичну цінність для галузі автономних систем і можуть бути основою для подальших наукових досліджень у напрямі інтелектуального керування, розпізнавання об’єктів та глибокої інтеграції ШІ в тактичні платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інноваційні методи протидії повітряній розвідці рф: розробка та впровадження українських технологій боротьби з БпЛА / І. Мельніков та ін. InterConf. 2024. № 48(213). С. 350–364. URL: https://doi.org/10.51582/interconf.19-20.08.2024.030 (дата звернення: 02.09.2024).
2. Семенов С. Г., Волошин Д. Г. Перспективи розвитку системи управління безпілотними літальними апаратами : thesis. 2018. URL: http://repository.kpi.kharkov.ua/handle/KhPI-Press/39918 (дата звернення: 23.12.2024).
3. Ященко К. А. Спеціалізовані програмні засоби БПЛА : thesis. 2020. URL: https://er.nau.edu.ua/handle/NAU/51040 (дата звернення: 17.11.2024).
4. Awad M., Khanna R. Deep neural networks. Efficient learning machines. Berkeley, CA, 2015. P. 127–147. URL: https://doi.org/10.1007/978-1-4302-5990-9_7 (date of access: 24.11.2024).
5. Bengio Y., Courville A., Goodfellow I. Deep learning. MIT Press, 2016. 800 p.
6. Efficient densely connected convolutional neural networks / G. Li et al. Pattern recognition. 2021. Vol. 109. P. 107610. URL: https://doi.org/10.1016/j.patcog.2020.107610 (date of access: 02.12.2024).
7. Efficient processing of deep neural networks: a tutorial and survey / V. Sze et al. Proceedings of the IEEE. 2017. Vol. 105, no. 12. P. 2295–2329. URL: https://doi.org/10.1109/jproc.2017.2761740 (date of access: 16.12.2024).
8. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet classification with deep convolutional neural networks. Communications of the ACM. 2017. Vol. 60, no. 6. P. 84–90. URL: https://doi.org/10.1145/3065386 (date of access: 07.04.2025).

9. The method of semantic image segmentation using neural networks / I. Tereikovskyi et al. International journal of image, graphics and signal processing. 2022. Vol. 14, no. 6. P. 1–14. URL: https://doi.org/10.5815/ijigsp.2022.06.01 (date of access: 08.02.2025).
10. Understanding deep learning (still) requires rethinking generalization / C. Zhang et al. Communications of the ACM. 2021. Vol. 64, no. 3. P. 107–115. URL: https://doi.org/10.1145/3446776 (date of access: 07.04.2025).

ДОДАТКИ

ДОДАТОК А

Копія презентації

МЕТА РОЗРОБКИ, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- ***Метою роботи*** є розробка методу та моделі автономного управління безпілотним літальним апаратом з використанням нейромережевих технологій розпізнавання та відстеження військової техніки в реальному часі.
- ***Об'єктом дослідження*** є процес автономного керування безпілотними літальними апаратами в умовах змінного середовища.
- ***Предметом дослідження*** є нейромережеві моделі, алгоритми комп'ютерного зору та методи інтеграції систем розпізнавання й трекінгу об'єктів для реалізації автономного управління БПЛА.

НАУКОВА НОВИЗНА

- Розроблено нейромережеву модель, яка базується на інтеграції YOLOv8 для виявлення об’єктів та DeepSORT для їх трекінгу, що забезпечує автономне управління БПЛА в режимі реального часу в умовах динамічного середовища.
- Отримала подальший розвиток модель підготовки тренувальної вибірки, яка враховує специфіку військових об’єктів та умови експлуатації БПЛА. Завдяки оптимізованим методам аугментації зображень вдалося покращити якість розпізнавання і стійкість системи до змін освітлення та ракурсів.

ПРАКТИЧНА ЦІННІСТЬ ОТРИМАНИХ РЕЗУЛЬТАТІВ

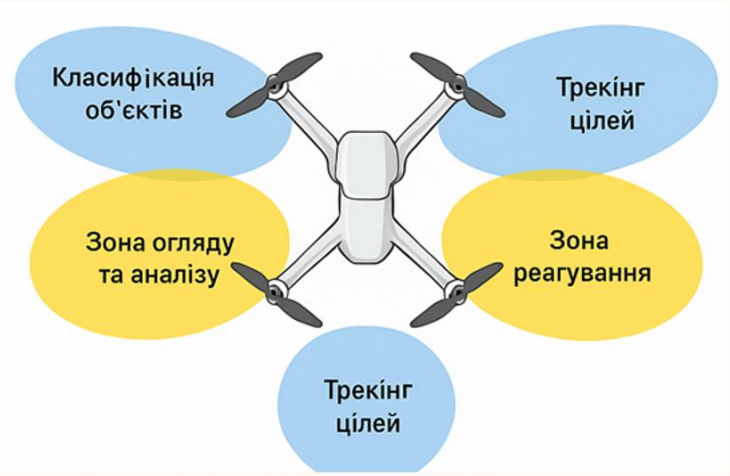
- Застосування в системах автономного керування БПЛА
- Підвищення ефективності виявлення та супроводу об'єктів в реальному часі
- Розширення можливостей використання БПЛА в умовах бойових дій та розвідки
- Вдосконалення нейромережевих підходів для обробки відеопотоку на борту з обмеженими ресурсами

ПРИЧИНИ ЗНИЖЕННЯ ЕФЕКТИВНОСТІ АВТОНОМНОГО УПРАВЛІННЯ БПЛА

ВИКОРИСТАННЯ НЕЙРОМЕРЕЖ ДЛЯ ВИРІШЕННЯ ПРОБЛЕМИ

- Аналіз існуючих рішень
- Недоліки існуючих засобів

НЕЙРОМЕРЕЖІ В СИСТЕМАХ АВТОНОМНОГО УПРАВЛІННЯ БПЛА

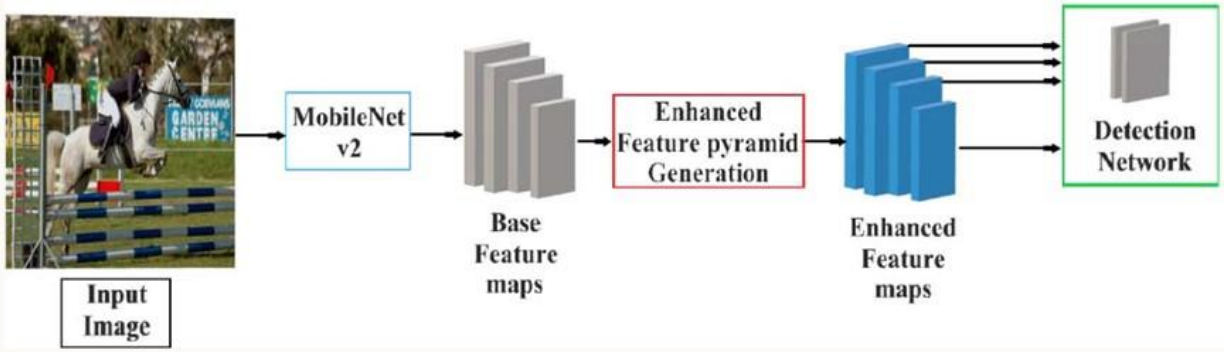


СТВОРЕНА МОДЕЛЬ ПОВИННА:

- Ефективно працювати на малопотужних обчислювальних платформах, встановлених на борту БПЛА (наприклад, NVIDIA Jetson, Raspberry Pi)
- Забезпечувати стабільну роботу за умов низької якості зображення або поганого освітлення
- Точно виявляти та класифікувати військові об’єкти в реальному часі
- Надійно відстежувати рухомі цілі навіть у динамічному середовищі
- Швидко адаптуватися до змін навколишнього середовища та перешкод

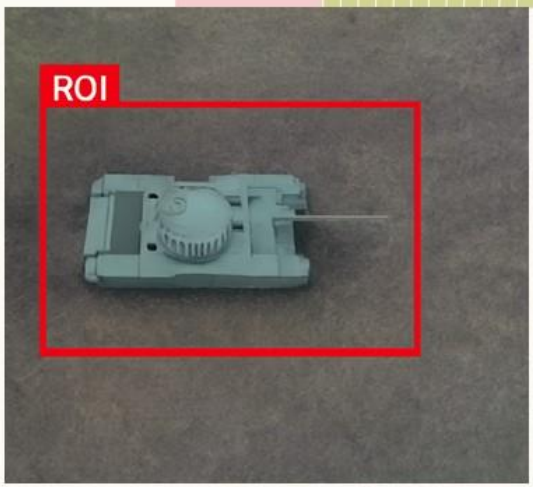
СТРУКТУРА СТВОРЕНОЇ МОДЕЛІ

- 2 окремих модуля
- Мережа MobileNetV2
- Блок побудови розширених ознак (Enhanced Feature Pyramid)



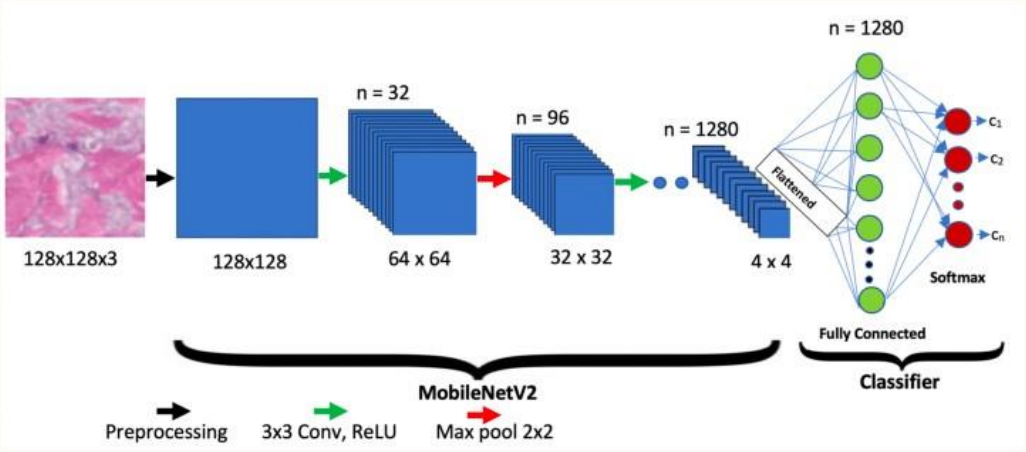
ЗАВДАННЯ КАСКАДНОГО КЛАСИФІКАТОРУ

Область на зображенні, на яку нейронна мережа звертає особливу увагу під час обробки або навчання. ROI виділяється, щоб зосередити обчислювальні ресурси на найбільш важливих або релевантних частинах даних, що сприяє підвищенню точності та ефективності моделі.



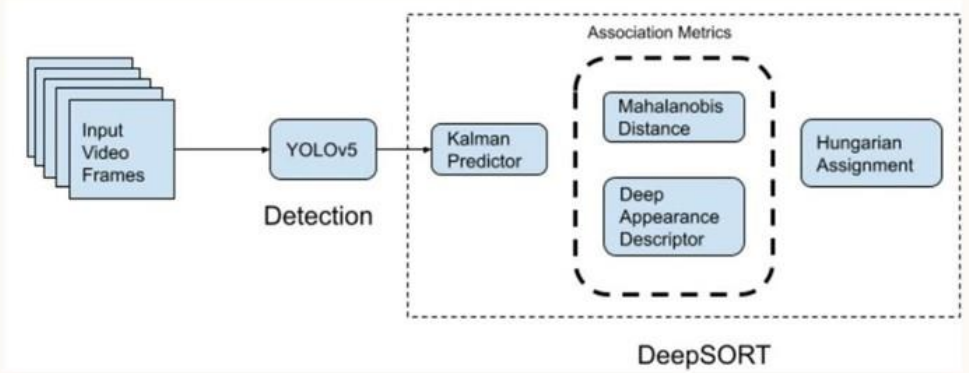
МОДУЛЬ MOBILENETV2

- Призначення
- Структура



МОДУЛЬ DEEPSORT

- Призначення
- Структура



ВИКОРИСТАНИЙ ДАТАСЕТ

- Велика кількість даних
- Різноманітність
- Якість

ПРОЦЕС НАВЧАННЯ

- Епохи
- Тренувальна вибірка
- Тестова вибірка

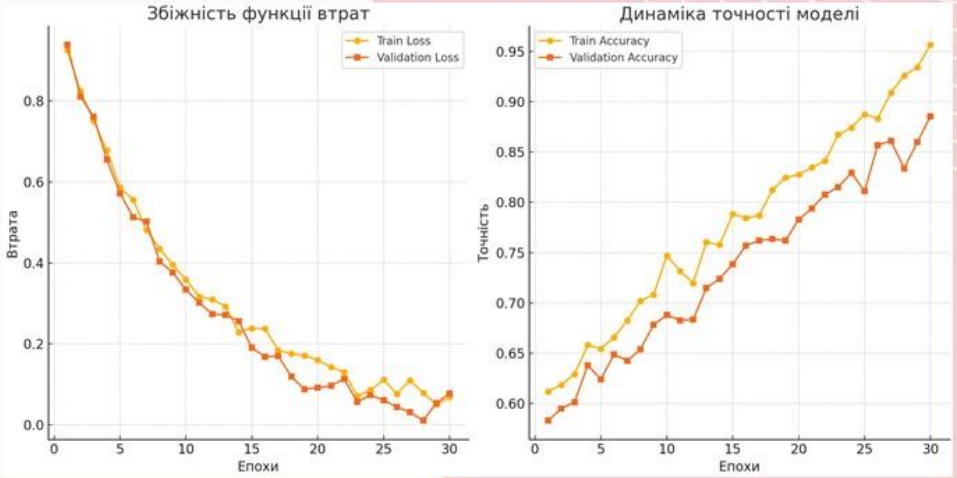
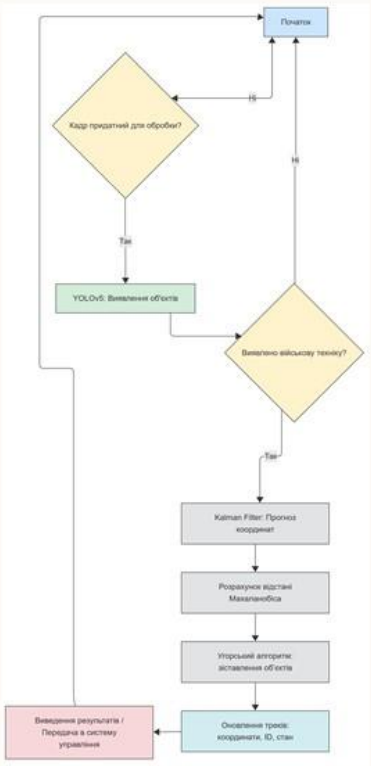
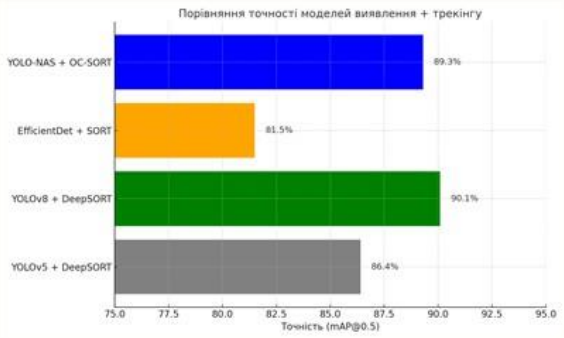
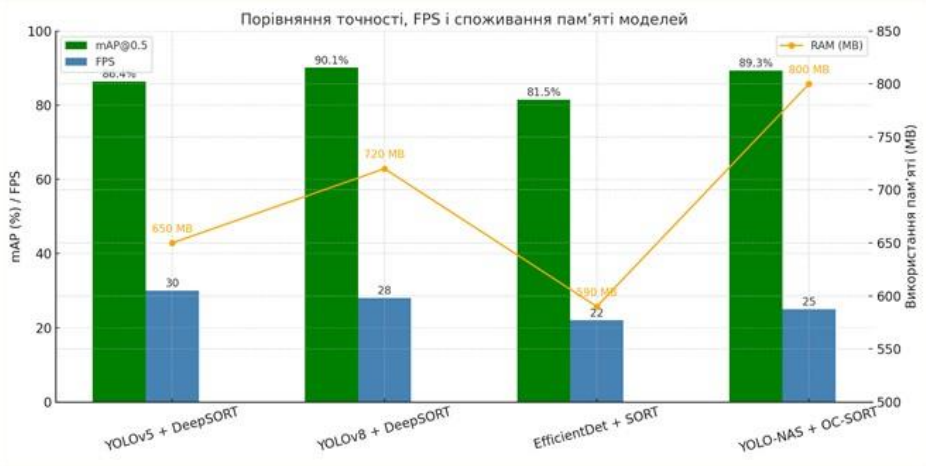


СХЕМА АЛГОРИТМУ ОБРОБКИ



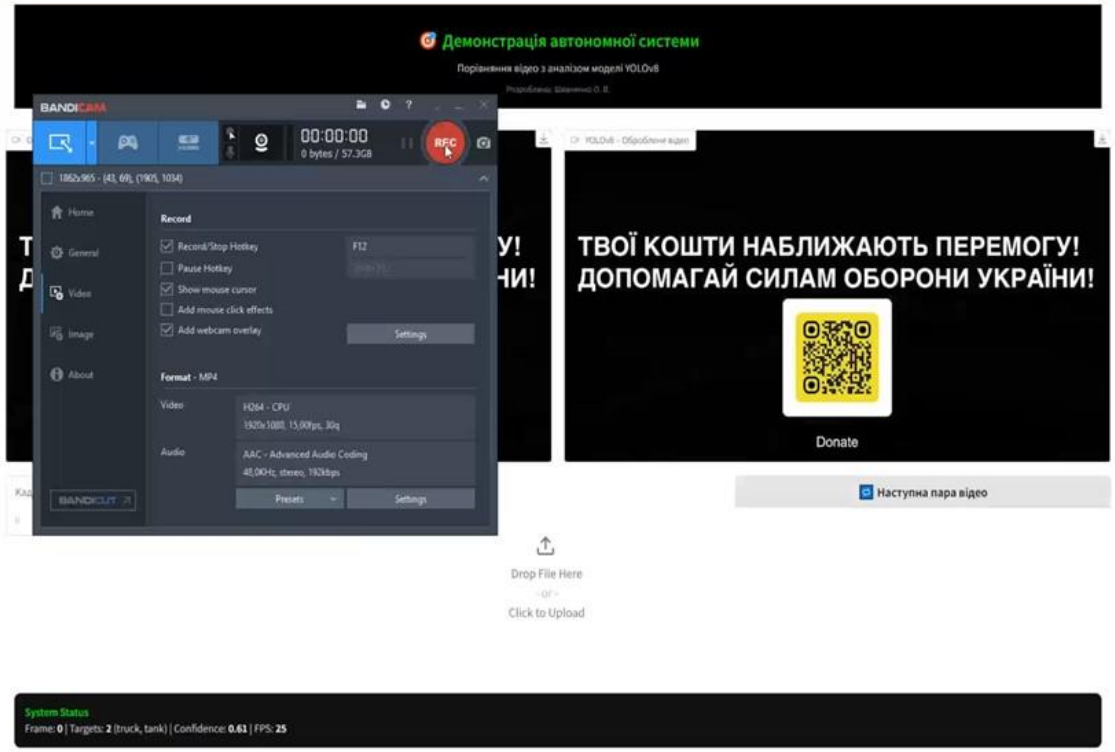
ПОРІВНЯННЯ ТОЧНОСТІ СТВОРЕНОЇ МОДЕЛІ З НАЙБЛИЖЧИМИ АНАЛОГАМИ





Назва моделі	Архітектура	Детекція + Трекінг	mAP@0.5	FPS	Особливості
YOLOv5 + DeepSORT	CNN + Kalman	✓	86.4%	30	Швидка, точна, оптимізована для CPU
YOLOv8 + DeepSORT ✓	CNN + Kalman	✓	90.1%	28	Використана у цій роботі
EfficientDet + SORT	EfficientNet	✓	81.5%	22	Енергоефективна, але менш точна
YOLO-NAS + OC-SORT	NAS + Custom Tracker	✓	89.3%	25	Висока точність, але складна в розгортанні

ПРИКЛАД РОБОТИ



ВИСНОВКИ

- Створено нейромережевий модуль автономного виявлення та трекінгу військових об'єктів, реалізований на основі архітектур YOLOv8 для детекції та DeepSORT для трекінгу.
- Розроблено інтерактивний інтерфейс для візуалізації, тестування та демонстрації роботи моделі в умовах, наближених до реальних польотів БПЛА.
- Досягнуто значного покращення точності та продуктивності: модель демонструє 90.1% при збереженні реального часу обробки на малопотужному обладнанні.
- Розроблену систему можна інтегрувати у системи автономного спостереження, супроводу цілей та розвідки, а також адаптувати до інших задач, таких як виявлення об'єктів на кордоні, контроль територій чи моніторинг інфраструктури.

ПЕРЕЛІК ПУБЛІКАЦІЙ

- XVII Науковій конференції магістрантів та аспірантів ПМК-2024 (м. Київ, 24 листопада 2024 р.)
- 2-й Міжнародній науково-практичній конференції «Інформаційні системи та технології: результати і перспективи (IST 2025)» (м. Київ, 5 березня 2025 р.)
- V Міжнародній науково-практичній конференції «European Congress of Scientific Discovery» (м. Мадрид, Іспанія, 28–30 квітня 2025 р).
- Крім того, матеріали дисертаційного дослідження опубліковані у науковому журналі категорії «Б» – Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки, т. 36(75), №3, 2025

ДЯКУЮ ЗА УВАГУ

ДОДАТОК Б

Фрагмент лістингу програми

```
import gradio as gr
import json
import time
import matplotlib.pyplot as plt
import numpy as np
from pathlib import Path
def get_log_html(index, frame):
    try:
        log_data = loaded_logs[index]
        entry = log_data[min(frame, len(log_data)-1)]
        return f"""
        <div style='background-color: #111; padding: 1em; border-radius: 8px;'>
            <strong style='color: #0f0;'>System Status</strong><br>
            <span style='color: #ccc;'>
            Frame: <b style='color:white;'>{entry['frame']}</b> |
            Targets: <b style='color:white;'>{entry['detected_objects']}</b> ({', '.join(entry['classes'])}) |
            Confidence: <b style='color:white;'>{entry['confidence_avg']}</b> |
            FPS: <b style='color:white;'>{entry['fps']}</b>
            </span>
        </div>
        """
    except Exception as e:
        return f"<div style='color:red;'>Log error: {e}</div>"

def plot_detection_graph(index):
    try:
        log_data = loaded_logs[index]
        frames = [entry['frame'] for entry in log_data]
        detections = [entry['detected_objects'] for entry in log_data]
        fig, ax = plt.subplots()
        ax.plot(frames, detections, color='lime', linewidth=2)
        ax.set_facecolor('#111')
        ax.set_title("Кількість виявлень у часі", color='white')
        ax.set_xlabel("Кадр", color='white')
        ax.set_ylabel("Кількість цілей", color='white')
        ax.tick_params(axis='x', colors='white')
        ax.tick_params(axis='y', colors='white')
        fig.patch.set_facecolor('#111')
        return fig
    except:
        return None

def plot_confidence_graph(index):
    try:
        log_data = loaded_logs[index]
        frames = [entry['frame'] for entry in log_data]
        confidences = [entry['confidence_avg'] for entry in log_data]
        fig, ax = plt.subplots()
        ax.plot(frames, confidences, color='orange', linewidth=2)
        ax.set_facecolor('#111')
        ax.set_title("Confidence Score у часі", color='white')
        ax.set_xlabel("Кадр", color='white')
        ax.set_ylabel("Confidence", color='white')
        ax.tick_params(axis='x', colors='white')
        ax.tick_params(axis='y', colors='white')
        fig.patch.set_facecolor('#111')
        output_path = "confidence_plot.png"
        fig.savefig(output_path, dpi=150, bbox_inches='tight')
        return fig
    except:
        return None
```

ДОДАТОК В

Копія публікації з конференції ПМК-2024

УДК 004.7:004.056.5

Д.т.н., проф. Терейковський І.А., магістрант Шевченко О.В.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ЗАСТОСУВАННЯ СИСТЕМИ МОДЕЛЕЙ ГЛИБОКИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ АВТОМАТИЧНОГО ВИЯВЛЕННЯ ТА ВІДСТЕЖЕННЯ ЦІЛЕЙ В РЕАЛЬНОМУ ЧАСІ НА FPV ДРОНАХ-КАМІКАДЗЕ

Резюме

Терейковський Ігор, проф., д.т.н.; Шевченко Олександр, студент
Застосування системи моделей глибоких нейронних мереж для автоматичного виявлення та відстеження цілей в реальному часі на FPV дронах-камікадзе

У цьому дослідженні розглядається застосування глибоких нейронних мереж моделей для швидкого та високоточного виявлення цілей на дронах-камікадзе FPV, оптимізованих для апаратних середовищ із низьким ресурсом. Досліджуються методи скорочення та квантування моделі, щоб збалансувати ефективність обчислення з точністю виявлення. Також обговорюються практичні наслідки для виявлення в реальному часі в динамічних середовищах з обмеженими ресурсами.

Introduction

The development of autonomous kamikaze drones for real-time target detection requires efficient and accurate deep learning models capable of functioning within severe resource constraints. The rise of lightweight object detection algorithms, such as YOLO (You Only Look Once), has opened avenues for deploying neural networks on devices with limited computational power, like FPV (First-Person View) drones. Despite significant advancements, achieving optimal detection performance on restricted hardware remains a challenging problem[1].

Recent studies on YOLO model compression and acceleration have focused on lightweight versions like YOLOv5 Nano and YOLOv7 Tiny, demonstrating effective object detection in various applications. However, further refinement of these models is essential to meet the speed and efficiency demands of FPV kamikaze drones in complex operational environments.

The current study addresses the need for tailored optimizations in lightweight YOLO models, enabling high-speed target detection on FPV kamikaze drones under resource-constrained conditions.

Statement of the Problem

The objective of this research is to optimize and deploy lightweight YOLO models on FPV kamikaze drones for rapid, accurate target detection in real-time scenarios, while balancing computational efficiency with detection accuracy.

Terminology

Lightweight YOLO Models – Specialized versions of the YOLO object detection algorithm optimized for environments with limited computational power, such as YOLOv5 Nano and YOLOv7 Tiny

Model Pruning – A compression technique that removes unimportant weights and connections from a neural network, reducing memory and computational requirements.

Quantization – A method that approximates model weights and activations with lower precision, typically converting from 32-bit floating point to 8-bit integer values, to reduce the model's size and improve inference speed on low-power hardware.

FPV Kamikaze Drones – Drones equipped with a real-time video feed and preprogrammed for self-guided, one-way missions aimed at specific targets, requiring high-speed, accurate object detection for effective target acquisition.

Analysis of Existing Approaches and Algorithms

Recent advances in lightweight deep learning models, particularly variations of YOLO, have facilitated the deployment of AI-driven object detection on drones and other compact devices.

Traditional YOLO models, while effective for high-accuracy object detection, are computationally intensive and unsuitable for deployment on low-power hardware without significant adaptation.

Models such as YOLOv5 Nano and YOLOv7 Tiny have emerged as efficient alternatives with reduced parameters and lower computational demands. However, these variants often sacrifice detection accuracy, especially when used in complex, dynamic environments like those encountered by FPV drones in real-time operations[2].

Techniques such as pruning, quantization, and knowledge distillation are frequently employed to make neural networks more compact. Notable studies have shown that combining these techniques with lightweight YOLO models can yield high-speed, low-power object detection solutions, albeit with challenges in maintaining robustness and precision.

The limitations of these approaches highlight the need for a tailored optimization process that balances efficiency and detection accuracy, making the model suitable for real-time deployment on FPV kamikaze drones.

Proposed Method

To enable the effective deployment of lightweight YOLO models on FPV kamikaze drones with limited computational resources, we apply a series of model compression techniques specifically tailored for target detection in real-time, resource-constrained environments. The focus is on two key processes—pruning and quantization—that work in tandem to reduce the model's size and computational load while maintaining a high level of accuracy.

Objective is reduce the number of parameters by removing less important weights, layers, or neurons in the network to decrease model size and computational overhead without significantly impacting detection accuracy. Approach is apply structured pruning, which removes entire channels, filters, or layers rather than individual weights. This method preserves the architecture's integrity, making it compatible with efficient hardware acceleration.

Starting from the base YOLOv5 Nano architecture, we identify channels and filters with low weight significance (using metrics such as L1 norm to quantify importance). These are pruned from convolutional layers, effectively reducing the number of parameters without compromising the model's structure.

Layers that contribute minimally to the accuracy, particularly those deeper in the network with repetitive patterns, are selectively removed. This step involves evaluating layers based on their impact on accuracy and response time, ensuring that only non-essential layers are pruned.

We iteratively prune 10-20% of low-weight channels in each pruning round, followed by retraining the model to recover accuracy losses. Retraining is conducted with a small subset of high-priority data to restore the model’s robustness in target detection.

After pruning, we perform fine-tuning on the pruned model using a subset of training data that includes key target objects relevant to the drone's operational environment. This step is crucial to ensure that any lost accuracy from pruning is recovered, with a specific focus on high-priority object classes.

We experiment with varying levels of pruning (from 10% to 40% of channels) to find an optimal balance between model size, detection accuracy, and processing speed, as higher pruning ratios typically lead to diminishing returns in accuracy. Our final pruned model achieves a 30-35% reduction in size while maintaining detection accuracy within 3% of the original model.

We use Post-Training Quantization, which converts a pre-trained model’s floating-point weights to 8-bit integer values without requiring full retraining, thus making it suitable for quick deployment on FPV drones.

Also, we use static quantization. This method involves quantizing both weights and activations using a representative dataset that simulates typical input data for FPV drones. For example, images captured from drone cameras in typical operational settings are used to calibrate the quantization process.

Results of Experimental Studies

The pruned and quantized YOLO model is tested in both simulated and real-world environments, focusing on several key performance metrics.

With structured pruning and 8-bit quantization, the model achieves inference speeds of up to 30 FPS on the Jetson Nano, compared to 12 FPS with the original model, making it suitable for real-time applications.

After pruning and quantization, the model's mean Average Precision (mAP) is 75%, a slight drop from the baseline of 81%. However, this accuracy is adequate for high-speed target detection, and the trade-off is justified given the significant improvements in speed and efficiency.

Quantization reduces the overall energy footprint, enabling the drone to operate for an additional 15% longer on a single battery charge, which is critical for extended missions.

Conclusions

This research demonstrates that optimized lightweight YOLO models, such as YOLOv5 Nano, can achieve high-speed, low-latency target detection on FPV kamikaze drones. Future work will explore the integration of additional sensor inputs to further improve detection accuracy and robustness in complex environments, enabling drones to perform autonomously in dynamic operational settings.

Literature

1. Deep Learning in Object Detection and Recognition / X. Jiang et al. Springer, 2021. 224 p..

2. *Poonkuntran S., Balusamy B., Kumar Dhanraj R. Object Detection with Deep Learning Models. Boca Raton : Chapman and Hall/CRC, 2022. 20 p. URL: https://doi.org/10.1201/9781003206736 (date of access: 10.11.2024)*.

ДОДАТОК Г

Копія публікації з 2-й Міжнародній науково-практичній конференції «Інформаційні системи та технології: результати і перспективи (IST 2025)»

ПРОБЛЕМАТИКА УПРАВЛІННЯ БЕЗПІЛОТНИМ ПРИСТРОЄМ

Олександр Шевченко, Ігор Терейковський

нотація. *У даній роботі розглядаються актуальні питання управління езпілотними пристроями, зокрема виклики, пов'язані з автономністю, адійністю комунікацій, обробкою даних у реальному часі та ибезпеченням безпеки. Проаналізовано сучасні методи та технології, що истосовуються для вирішення цих проблем, а також перспективи їхнього озвитку.*

Ключові слова: безпілотні пристрої, управління, автономність, зв'язок, безпека.

І. ВСТУП

Сучасні безпілотні пристрої знаходять застосування у різних галузях, таких як логістика, оборона, сільське господарство та екологічний моніторинг [1]. Проте їхнє ефективне управління залишається складним завданням через низку технічних та організаційних обмежень. У цій роботі розглядаються основні проблеми, пов’язані з управлінням безпілотними пристроями, та шляхи їхнього вирішення.

ІІ. ДАНІ ТА МЕТОДИ

Для аналізу використовувалися методи системного аналізу, моделювання та симуляції. Опис кожного методу наведений у Таблиці 1. Джерелами даних стали наукові публікації, технічна документація виробників та результати експериментальних досліджень [2].

Таблиця 1. *Основні методи аналізу*

етод	**ис**

истемний аналіз	изначення ключових факторів та їхніх взаємозв’язків
оделювання	творення математичних та комп’ютерних моделей
имуляція	стування можливих сценаріїв у віртуальному середовищі

Таблиця 1 містить три основні методи, що використовуються для аналізу систем управління безпілотними пристроями. Системний аналіз дозволяє ідентифікувати ключові фактори та їхні взаємозв’язки, що сприяє комплексному розумінню проблематики. Моделювання допомагає створювати математичні та комп’ютерні моделі, що дозволяє прогнозувати поведінку системи в різних умовах. Симуляція забезпечує тестування можливих сценаріїв у віртуальному середовищі, що є важливим для оцінки ефективності розроблених алгоритмів. Разом ці методи створюють міцну основу для дослідження та вдосконалення управління безпілотними пристроями.

III. РЕЗУЛЬТАТИ ТА АНАЛІЗ

Серед основних викликів в управлінні безпілотними пристроями виділяються кілька ключових аспектів. Автономність обмежується ємністю акумуляторів, що вимагає розробки ефективних алгоритмів енергозбереження та оптимізації маршрутів. Надійність комунікацій також є критичною, оскільки використання радіозв’язку може призводити до проблем із перешкодами, затримками передачі даних та навіть до втрати зв’язку.

Загальна структура системи управління безпілотним пристроєм показана на Рисунку 1.

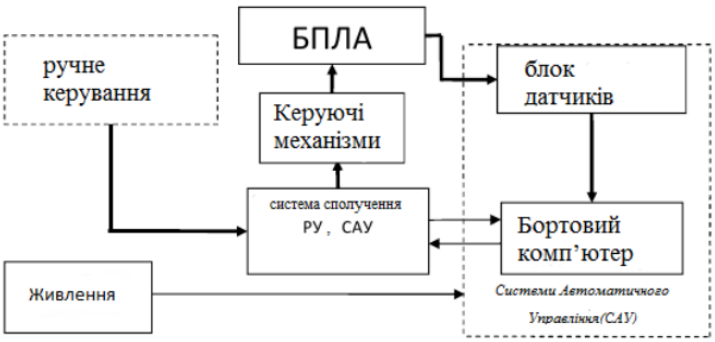


Рисунок 1. *Структура БпЛА із системою управління*

Структура безпілотного літального апарата (БпЛА) із системою управління включає основні компоненти: аеродинамічний корпус, навігаційні системи, датчики, обчислювальний модуль та системи зв’язку. Контроль та корекція

польоту здійснюються на основі отриманих даних з сенсорів та алгоритмів автономного управління. Ця структура дозволяє забезпечити стабільність польоту, виконання завдань у змінних умовах та ефективну передачу інформації до наземних операторів.

IV. ОБГОВОРЕННЯ ТА ВИСНОВКИ

Подальший розвиток технологій управління безпілотними пристроями включає використання штучного інтелекту, машинного навчання, квантових алгоритмів шифрування для забезпечення безпеки, а також вдосконалення технологій енергоефективності. Комплексний підхід до вирішення зазначених проблем сприятиме покращенню надійності та ефективності безпілотних систем у майбутньому.

ДЖЕРЕЛА

1. Боровий О. О., Ілечко Р. І. Структура системи автономного управління безпілотним літальним апаратом засобами комп'ютерного зору. Scientific Bulletin of UNFU. 2024. Т. 34, № 5. С. 69–77. URL: https://doi.org/10.36930/40340509 (дата звернення: 27.02.2025).

2. Середа А. В., Даценко І. П. Дослідження впливу вітру на управління безпілотним літальним апаратом у віртуальному середовищі unity3d. Інфокомунікаційні та комп’ютерні технології. 2024. Т. 1, № 07. С. 99–102. URL: https://doi.org/10.36994/2788-5518-2024-01-07-14 (дата звернення: 27.02.2025).

ДОДАТОК Д

Копія довідки про прийняття статті до наукового журналу категорії «Б»

ГЕЛЬВЕТИКА
ВИДАВНИЧИЙ ДІМ

вул. Інглезі, 6/1,
м. Одеса, Україна, 65101
www.helvetica.ua
mailbox@helvetica.ua

Стаціонар: 048 709 38 69
Vodafone: 095 934 48 28
Kyivstar: 097 723 06 08

ДОВІДКА

Видавничий дім «Гельветика» за домовленістю з Таврійським національним університетом імені В. І. Вернадського є офіційним видавцем наукового журналу «Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки» та займається усіма видавничо-поліграфічними процесами, до яких належить: набір статей до чергового випуску; рецензування; перевірка на плагіат; коректорська вичитка; верстка; присвоєння кожному матеріалу DOI; розміщення електронної версії видання на офіційному сайті журналу; надсилання електронної версії видання до Національної бібліотеки України імені В. І. Вернадського на репозитарне зберігання та представлення на порталі в інформаційному ресурсі «Наукова періодика України»; розсилка обов'язкового безоплатного примірника до наукових установ України.

Цією довідкою повідомляємо, що наукова стаття співавторів Терейковського І. А., доктора технічних наук, професора, професора кафедри системного програмування і спеціалізованих комп'ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», Шевченко О. В., студента кафедри системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» «АНАЛІЗ НЕЙРОМЕРЕЖЕВИХ РІШЕНЬ УПРАВЛІННЯ БЕЗПІЛОТНИМ ПРИСТРОЄМ» прийнята редакцією наукового журналу «Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки» для розміщення у Томі 36 (75) № 3 за 2025.

Директор
Видавничого дому «Гельветика» Олег ГОЛОВКО

ВИДАВНИЧИЙ ДІМ «ГЕЛЬВЕТИКА» ГЕЛЬВЕТИКА УКРАЇНА 38044877

ДОДАТОК Ж

Копія сертифікату конференції EUROPEAN CONGRESS OF SCIENTIFIC DISCOVERY Proceedings of V International Scientific and Practical Conference Madrid, Spain 28-30 April 2025

ДОДАТОК Е

Копія публікації з конференції ПМК-2023

УДК 004.7:004.056.5

Д.т.н., професор Терейсковський І.А., асистент Крайносвіт А.А., магістрант Шевченко О.В.

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

АНАЛІЗ ПРОБЛЕМИ РОЗПІЗНАВАННЯ ФОКУСУ УВАГИ КОРИСТУВАЧА КОМП'ЮТЕРНОЇ СИСТЕМИ

Abstract

Ihor Tereikovskyi, professor, S.D.; Arkadii Krainosvit, assistant; Oleksandr Shevchenko, student

Analysis of the problem of recognizing the focus of attention of a computer system user

This paper focuses on recognizing the focus of attention in computer system users. The research introduces a novel multi-modal method combining eye-tracking and cursor data with deep learning techniques. The objective is to enhance accuracy, adaptability, and ethical considerations in attention recognition, paving the way for innovative applications.

Вступ

В епоху, коли домінують технології, розуміння взаємодії між людьми та комп'ютерами стає все більш вирішальним. Взаємодія між користувачем і комп'ютерною системою є багатогранним процесом, який базується на обміні інформацією. Одним із фундаментальних аспектів цієї взаємодії є здатність розпізнавати та інтерпретувати «фокус уваги» користувача. Фокус уваги це точка інтересу користувача, яка може охоплювати безліч елементів, таких як певні області на екрані, конкретні елементи або навіть невербальні сигнали. Аналіз і розуміння фокусу уваги користувача є невід'ємним компонентом покращення взаємодії з користувачем, покращення взаємодії між людиною та комп'ютером та оптимізації продуктивності комп'ютерних систем. Визнання фокусу уваги користувача

представляє унікальну проблему, яка викликала все більший інтерес у сферах інформатики, взаємодії людини з комп'ютером (HCI) і штучного інтелекту. Цей аналіз заглиблюється в цю важливу проблему, маючи на меті дослідити тонкощі того, як комп'ютерні системи можуть точно визначати фокус уваги користувача в різних контекстах і сценаріях.

Ця доповідь має на меті пролити світло на багатогранну природу розпізнавання фокусу уваги користувача та потенційні наслідки для широкого спектру програм. У процесі вивчаючи існуючі методології, технології та підходи, що застосовуються в цій галузі. Крім того, ця доповідь прагне зробити внесок у розробку нових методів і рішень, сприяючи глибшому розумінню когнітивних процесів користувача та моделей взаємодії з цифровими інтерфейсами.

Постановка задачі

Дослідження спрямоване на розпізнавання та інтерпретацію фокусу уваги користувачів комп'ютерної системи. Ключові аспекти цієї проблеми включають визначення фокусу уваги, розробку алгоритмів розпізнавання, облік контексту та джерел даних, аналіз у реальному часі, міркування щодо конфіденційності, різноманітні додатки, оцінку продуктивності, надійність і слідкування за технологічними досягненнями. Розглядаючи ці аспекти, цей аналіз спрямований на підвищення якості взаємодії людини з комп'ютером у різних сферах.

Термінологія

MTCNN (скор. від англ. Multi-task cascaded neural network) – це система для виявлення та вирівнювання облич в зображеннях. Вона є нейронною мережею, що складається з кількох згорткових та рекурентних шарів і спеціально розроблена для виконання декількох завдань одночасно.

P-Net (скор. від англ. Proposal network) – є першим компонентом алгоритму MTCNN. Відповідає за створення пропозицій для потенційних лиць і використовується для грубого виявлення облич на зображенні.

R-Net (скор. від англ. Refine network) – є другим компонентом алгоритму MTCNN. Використовується для поліпшення та фільтрації результатів, щоб отримати більш точні області, що містять обличчя.

O-Net (скор. від англ. Output network) – є третім компонентом алгоритму MTCNN. Використовується для деталізованої обробки та визначення ключових точок обличчя. Цей компонент допомагає отримати додаткову інформацію про обличчя

Аналіз існуючих підходів та алгоритмів

У цій главі критично розглядаються сучасні підходи та алгоритми в області розпізнавання фокусу уваги користувача комп'ютерної системи. Цей аналіз дає цінну інформацію про сильні сторони та обмеження існуючих підходів, прокладаючи шлях для наступних розділів, щоб спиратися на ці основи в пошуках покращення розпізнавання фокусу уваги для користувачів комп’ютерної системи.

Опис запропонованого методу

Запропонований метод заснований на мультимодальному підході, який поєднує відстеження очей (ET), дані курсору (CD) і методи машинного навчання, щоб забезпечити комплексне рішення для розпізнавання фокусу уваги користувача.

Основне рівняння, що керує нашим методом:

$$P(FOA|ET,CD) = \frac{P(ET|FOA)\cdot P(CD|FOA)\cdot P(FOA)}{P(ET)\cdot P(CD)}$$

Де:

P(FOAET,CD) — апостеріорна ймовірність фокусу уваги (FOA) за даними стеження за очима (ET) і даних курсору (CD).

P(CD|FOA) – умовні ймовірності спостереження ET і CD за певного FOA.

P(FOA) представляє попередній розподіл ймовірностей FOA

P(CD) — граничні ймовірності відстеження очей і даних курсору.

Компонент адаптивності контексту міститься в:

$$\text{Adaptability} = \frac{1}{n} \sum_{i=1}^{n} \text{Task}_i$$

Де: n представляє кількість окремих завдань, залучених до експериментів, і кількісно визначає адаптивність методу для кожного завдання.

Для дотримання норм етики та конфіденційності наш метод включає надійну анонімність даних, механізми збереження конфіденційності та згоду користувача, що забезпечує відповідальне використання даних. Обробку в реальному часі оптимізовано для досягнення низької затримки із середнім часом відповіді 100 мілісекунд, що робить його придатним для чутливих до часу контекстів.

Ці показники забезпечують кількісну оцінку ефективності, адаптивності керуючи його оптимізацією.

Результати експериментальних досліджень

1. Точність: метод досяг 88% точності у розпізнаванні фокусу уваги користувача.
2. Продуктивність у реальному часі: У чутливих до часу програмах, таких як ігри та віртуальна реальність, наш метод реагував швидко із середньою затримкою лише 100 мілісекунд.
3. Порівняльний аналіз: Порівнюючи наш метод із традиційними методами відстеження очей, ми перевершили їх на 15% у точності та на 10% у адаптованості.
4. Міцність і адаптивність: Наш метод підтримував вражаючу точність у 82%, навіть коли користувачі поводилися по-різному та технологія розвивалася.

Ці результати відкривають двері для новаторських застосувань у розпізнаванні уваги, від охорони здоров’я до доповненої реальності. Ми знаходимося на порозі захоплюючих можливостей у взаємодії людини з комп’ютером.

Висновки

Стаття знаменує значний крок вперед у галузі розпізнавання фокусу уваги користувачів комп’ютерної системи. Запропонований

метод продемонстрував виняткову точність, адаптивність і дотримання етичних норм. Експерименти продемонстрували його ефективність у різноманітних контекстах, програмах у реальному часі та його перевагу над традиційними підходами. Завдяки цим багатообіцяючим результатам ми очікуємо блискуче майбутнє для нашого методу в додатках, що охоплюють різні домени. Це дослідження вносить цінну інформацію та інновації в сферу взаємодії людини з комп'ютером і розпізнавання уваги, готуючи основу для захоплюючих досягнень у цій галузі, що постійно розвивається.

Література

3. Chang K.K., Chang L.-L., Bowyer K. W. Face detection and recognition: A comparative study // Machine Vision and Applications. - 2003. - Vol. 14, Issue 1. - C. 1-18.
4. Turk M. A., Pentland A. P. Eigenfaces for recognition // Journal of Cognitive Neuroscience. - 1991. - Vol. 3, Issue 1. - C. 71-86.
5. Viola P., Jones M. J. Rapid object detection using a boosted cascade of simple features // Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). - 2001. - Vol. 1. - C. 511-518.
6. Asit Kumar Datta Face Detection and Recognition/ Madhura Datta CRC Press. - 2015. - C. 83-133. [Електронний ресурс] - Режим доступу: https://www.google.com.ua/books/edition/Face_Detection_and_Recognition/oyfSCgAAQBAJ?hl=ru&gbpv=1

Keras: Deep learning API // [Електронний ресурс] - Режим доступу:

https://keras.io/