МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно- обчислювальних систем і управління

КАПКУНОВ Андрій Сергійович

Модуль виявлення дронів з використанням YOLOv8 / Drone Detection Module Using YOLOv8

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
А. С. Капкунов

_____

Науковий керівник:
к.т.н., І. М. Майків

_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **М. П. Комар**

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
КАПКУНОВУ Андрію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Модуль виявлення дронів з використанням YOLOv8 / Drone Detection Module Using YOLOv8
керівник роботи Майків Ігор Мирославович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 12 грудня 2023 р. № 753.
2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- Дослідити та аналізувати існуючі методи та технології виявлення дронів.
- Проаналізувати наукові роботи та публікації для виявлення дронів.
- Визначити переваги використання YOLOv8 для детекції дронів.
- Розробити архітектуру модуля виявлення дронів.
- Підготувати та обробити відповідні набори даних для тренування та тестування моделі.
- Тренування та оптимізація моделі YOLOv8.
- Візуалізація та аналіз результатів виявлення.

5. Перелік графічного матеріалу в роботі:
- Архітектура модуля виявлення дронів
- Архітектура глибокої нейронної мережі YOLOv8
- Алгоритм виявлення дронів з використанням YOLOv8
- графіки з результатами експериментальних досліджень.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент_____ А. С. Капкунов
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ І.М. Майків
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Модуль виявлення дронів з використанням YOLOv8» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 40 сторінок і містить 19 ілюстрацій, 2 таблиці, 2 додатки та 17 використаних джерел.

Метою роботи є розробка та аналіз ефективності модуля виявлення дронів заснованого на алгоритмі YOLOv8 для забезпечення безпеки в різноманітних умовах експлуатації.

Методами розроблення обрано метод аналізу (для дослідження існуючих підходів до виявлення дронів), метод синтезу (для поєднання переваг існуючих методів виявлення), методи моделювання (для представлення та дослідження процесів виявлення дронів), метод порівняльного аналізу (для оцінювання адекватності моделі виявлення дронів).

Внаслідок виконання роботи обґрунтовано раціональний підхід до розроблення моделей виявлення дронів та розроблено програмний засіб автоматизації процесу виявлення дронів з використанням YOLOv8, який дозволяє створювати та досліджувати моделі виявлення дронів.

Результати дослідження можуть бути використані в науково-дослідницьких закладах і підрозділах підприємств, що займаються розробленням моделей виявлення дронів.

Ключові слова: ВИЯВЛЕННЯ ДРОНІВ, НЕЙРОННА МЕРЕЖА, YOLOv8, МОДЕЛЬ АДЕКВАТНОСТІ, ІМІТАЦІЙНА МОДЕЛЬ.

ANNOTATION

Qualification work on the topic «Drone detection module using YOLOv8» for Bachelor's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 40 pages and it contains 19 figures, 2 tables, 2 annexes and 17 sources.

The purpose of the work is to develop and analyze the effectiveness of a drone detection module based on the YOLOv8 algorithm to ensure safety in various operational conditions.

Research methods include analysis (to study existing approaches to drone detection), synthesis (to combine the advantages of existing detection methods), modeling (to represent and study the processes of drone detection), and comparative analysis (to evaluate the adequacy of the drone detection model).

As a result of the work, a rational approach to the development of drone detection models was substantiated, and a software tool for automating the drone detection process using YOLOv8 was developed, allowing the creation and research of drone detection models.

The research results can be used in research institutions and enterprise departments involved in the development of drone detection models.

Keywords: DRONE DETECTION, NEURAL NETWORK, YOLOv8, MODEL ADEQUACY, SIMULATION MODEL.

ЗМІСТ

ВСТУП

Актуальність розробки модуля виявлення дронів з використанням YOLOv8 зумовлена різким зростанням використання безпілотників у найрізноманітніших галузях життєдіяльності людини. Дрони вже давно перестали бути виключно хобі та активно впроваджуються в комерційні, наукові, розвідувальні та навіть рятувальні операції. Проте, це призводить не лише до позитивних змін, а й вносить певні загрози та виклики. Питання безпеки польотів, особистої приватності, захисту критично важливих об'єктів стає дедалі більш актуальними, підкреслюючи потребу у вдосконаленні систем виявлення та контролю за цими апаратами.

Зокрема, збільшення кількості дронів в повітряному просторі ставить під загрозу безпеку авіаційних рейсів, зокрема у зонах біля аеропортів, де неконтрольований політ дрона може призвести до аварій. Також дрони можуть використовуватися для несанкціонованого стеження або навіть терористичних атак, що вимагає розробки ефективних методів їх виявлення та нейтралізації. Впровадження алгоритмів штучного інтелекту, зокрема YOLOv8, які можуть швидко і точно ідентифікувати об'єкти на зображенні, надає значні переваги у реалізації таких систем.

Розвиток технологій, які забезпечують аналіз великих обсягів відеоданих в реальному часі, є необхідністю для міських інфраструктур та великих промислових об'єктів, де потреба у високому рівні безпеки є критично важливою. Системи на базі YOLOv8 можуть бути інтегровані з існуючими системами спостереження для посилення їх ефективності, забезпечуючи автоматизований моніторинг та оперативне реагування на потенційні загрози. Це особливо актуально для об'єктів, які вимагають високого рівня захисту, таких як військові бази, державні установи або критично важливі промислові об'єкти.

Науковий інтерес до розробки та впровадження модулів виявлення дронів також зумовлений бажанням розширити можливості існуючих технологій та створити нові підходи для їх вдосконалення. Вивчення можливостей YOLOv8 у цьому контексті дозволяє не лише розв'язати практичні завдання, але й сприяє розвитку наукових знань у галузі машинного навчання та комп'ютерного зору, відкриваючи нові напрямки досліджень у цих областях.

Метою цієї бакалаврської роботи є розробка та аналіз ефективності модуля виявлення дронів, заснованого на алгоритмі YOLOv8, для забезпечення безпеки в різноманітних умовах експлуатації. Для досягнення цієї мети необхідно вирішити наступні завдання:

1. Дослідити та аналізувати існуючі методи та технології виявлення дронів.
2. Проаналізувати наукові роботи та публікації для виявлення дронів.
3. Визначити переваги використання YOLOv8 для детекції дронів.
4. Розробити архітектуру модуля виявлення дронів.
5. Підготувати та обробити відповідні набори даних для тренування та тестування моделі.
6. Тренування та оптимізація моделі YOLOv8.
7. Візуалізація та аналіз результатів виявлення.

Об'єктом дослідження є системи виявлення дронів, які використовуються для моніторингу та ідентифікації безпілотних літальних апаратів у різноманітних умовах.

Предметом дослідження є алгоритм YOLOv8 як основний інструмент для виявлення дронів в розроблюваній системі.

Методи дослідження включають експериментальний аналіз, комп'ютерне моделювання та кількісну оцінку ефективності алгоритму YOLOv8 у задачах виявлення дронів. Дослідження базується на використанні

датасетів з зображеннями дронів для тренування та тестування моделі, що дозволяє оцінити точність і швидкість алгоритму у реальних умовах. Застосовуються методи машинного навчання та глибокого навчання для адаптації та оптимізації моделі під конкретні задачі виявлення. Крім того, використовуються статистичні методи для аналізу отриманих результатів та визначення статистично значущих висновків щодо ефективності запропонованої системи.

Практичне значення даної роботи полягає у розробці високоефективної системи виявлення дронів, яка може бути використана для забезпечення безпеки у громадських місцях, на стратегічно важливих об'єктах та в особистих майнових комплексах. Це допоможе підвищити рівень захисту від несанкціонованого використання дронів, знизити ризик для авіаційної безпеки та посилити контроль за повітряним простором. Крім того, результати дослідження можуть бути використані для подальшого вдосконалення алгоритмів машинного зору і штучного інтелекту, зокрема в аспектах їх застосування в реальних умовах і в інтеграції з іншими системами моніторингу і контролю.

Структура та обсяг кваліфікаційної роботи. Робота включає вступ, три розділи, висновки та список використаних джерел. Загальний обсяг роботи складає 40 сторінок комп'ютерного тексту, включаючи 9 рисунків і 1 таблицю. Список використаних джерел містить 17 найменувань і займає 3 сторінки.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження доповідалися й обговорювалися V Всеукраїнської мультидисциплінарної студентської наукової конференції «Науковий простір: аналіз, сучасний стан, тренди та перспективи», яка відбулася 17 травня 2024 року у місті Київ, Україна.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Загальний огляд проблематики виявлення дронів

Виявлення дронів у сучасних умовах стикається з рядом технічних та оперативних викликів, які потребують комплексного підходу до рішення. Один з основних технічних викликів полягає у здатності систем виявлення розрізняти дрони від інших літаючих об'єктів, таких як птахи або малі літальні апарати. Це вимагає від алгоритмів глибокого навчання високої точності у класифікації об'єктів та ефективної обробки зображень в реальному часі, що може бути складно з огляду на обмежену обчислювальну потужність у мобільних або вбудованих системах.

Крім технічних аспектів, існують значні оперативні виклики, пов'язані з роботою систем виявлення в різноманітних середовищах. Дрони можуть використовуватись в різних ландшафтах та погодних умовах, що вимагає від системи здатності адаптуватися та коректно функціонувати незалежно від зовнішніх умов. Наприклад, виявлення дрона на фоні міських висоток або в умовах низької освітленості ставить перед системою складні завдання з фільтрації шумів та обробки неконтрастних зображень.

Нарешті, важливим аспектом є необхідність інтеграції систем виявлення дронів з іншими безпековими та навігаційними системами. Це включає сумісність з контрольно-пропускними пунктами, системами повітряного спостереження та навіть з автоматизованими системами реагування. Інтеграція вимагає високого рівня надійності та безперервної роботи систем, що може бути складним з огляду на різноманітність стандартів та протоколів в різних системах. Забезпечення стабільної та безпечної інтеграції стає ключовим для забезпечення загальної безпеки та ефективності операцій з дронами.

Неконтрольоване використання дронів може призвести до значних загроз для безпеки та приватності, які з кожним роком стають дедалі актуальнішими у світлі широкого розповсюдження цих технологій. Однією з основних загроз є можливість використання дронів для несанкціонованого стеження. Це може включати в себе збір відео або фотографічних даних у приватних місцях, таких як житлові будинки або закриті території, без відома та згоди осіб, які там перебувають. Таке вторгнення у приватне життя може порушувати законодавчі норми та етичні стандарти, створюючи значний суспільний резонанс.

Другою важливою загрозою є використання дронів для навмисних актів вандалізму або тероризму. Дрони можуть бути оснащені вибуховими пристроями або іншими небезпечними матеріалами і використовуватися для атак на критично важливі інфраструктурні об'єкти, великі громадські зібрання або навіть приватні заходи. Це ставить перед органами правопорядку завдання не тільки виявлення та нейтралізації таких загроз, але й розробки стратегій для протидії цим новим формам агресії.

Третя загроза пов'язана з можливістю використання дронів для кібератак. Через вбудовані комунікаційні системи, дрони можуть стати інструментами для проведення кібернетичних атак на бездротові мережі, втручання в роботу серверів або збір конфіденційної інформації. Наприклад, дрони можуть використовуватися для "снифінгу" або перехоплення даних з незахищених Wi-Fi мереж, що є серйозною загрозою для цифрової безпеки компаній та приватних осіб.

Нарешті, існує загроза, пов'язана з порушенням повітряного простору, що може призвести до збоїв у роботі авіаційних систем або навіть реальних аварій. Неконтрольовані польоти дронів поблизу аеропортів або в інших регульованих зонах можуть стати причиною серйозних інцидентів, включаючи зіткнення з літаками на підході або при вильоті. Такі інциденти

вимагають розробки ефективних систем виявлення та управління дронами, щоб забезпечити безпеку як в повітрі, так і на землі.

Ефективні системи виявлення дронів стають все більш важливими у багатьох сферах, особливо в авіаційній галузі. Контроль за безпечним використанням дронів у повітряному просторі є критичним з огляду на потенційні ризики для пасажирських та вантажних літаків. Неконтрольовані або зловмисно керовані дрони можуть спричинити серйозні аварії, порушуючи безпечні польотні операції. Тому авіаційна промисловість вимагає високотехнологічних систем виявлення дронів, які можуть швидко ідентифікувати та відслідковувати дрони, що наближаються до зон, критичних для авіаційної безпеки, забезпечуючи своєчасні заходи реагування.

У сфері безпеки заходів, особливо під час проведення великих публічних або приватних зборів, ефективне виявлення дронів також є необхідністю. На великих заходах, таких як спортивні ігри, концерти або політичні зібрання, дрони можуть використовуватися для несанкціонованої зйомки, або, що гірше, для виконання агресивних актів. Системи виявлення дронів можуть допомогти організаторам заходів виявляти та нейтралізувати дрони перед тим, як вони становитимуть загрозу безпеці учасників, гарантуючи конфіденційність та фізичний захист присутніх осіб.

Крім того, особиста безпека і приватність у житлових районах також можуть бути посилені за допомогою систем виявлення дронів. В умовах, коли приватні особи інвестують у свою безпеку через зростання кількості технологічно обладнаних дронів, що можуть порушувати приватне життя, наявність надійних систем виявлення може забезпечити необхідний рівень захисту. Це включає ідентифікацію та відслідковування дронів, що наближаються до приватних володінь, запобігання несанкціонованому відеоспостереженню або збору даних, а також забезпечення спокою власників нерухомості щодо захисту їхнього особистого простору.

1.2 Аналіз наукових робіт та джерел, які досліджують проблему виявлення дронів

Дослідження, проведене Q Li та ін. (2021) [1], фокусується на використанні мережі YOLO v4 для ідентифікації місцезнаходження та кількості рухомих об'єктів на завантажених дорогах. Це дослідження покращує точність в складних умовах великого трафіку.

PS Raskar та SK Shah (2021) [2] застосовують YOLO V2 для виявлення підробок у відео, аналізуючи рухомі об'єкти. Вони демонструють відмінні результати порівняно з традиційними методами.

У огляді M Maity та S Banerjee (2021) [3] розглядаються досягнення в техніках виявлення транспортних засобів, зокрема за допомогою YOLO та Faster R-CNN для детектування транспортних засобів на аерофотознімках.

NM Krishna та інші (2021) [4] розробили проект, який інтегрує виявлення та слідкування за об'єктами за допомогою YOLO, зосереджуючись на рухомих об'єктах у різних середовищах.

Дослідження AMA ghani ABDULGHANİ (2022) [5] аналізує ефективність алгоритмів YOLO та Faster R-CNN у виявленні рухомих об'єктів, таких як пішоходи та транспортні засоби, в різних умовах.

Дослідження Bernardini та співавторів (2017) [6] розробило метод виявлення дронів за допомогою акустичних підписів. Система здатна розпізнавати звуки дронів та автоматично ініціювати тривогу, що забезпечує надійний спосіб моніторингу дронів.

У статті Singha та Aydin (2021) [7] аналізується здатність YOLOv4 до виявлення дронів. Дослідження підкреслює точність та ефективність YOLOv4 у розрізненні дронів від інших об'єктів, зокрема птахів.

Seidaliyeva та ін. (2020) [8] розглядають задачу виявлення дронів як двоетапний процес: виявлення рухомих об'єктів та їх класифікація як дронів. Дослідження демонструє високу точність виявлення дронів на відео зі статичним фоном.

Taha та Shoufan (2019) [9] провели огляд сучасних досліджень у галузі виявлення та класифікації дронів з використанням методів машинного навчання, визначаючи основні вимоги та специфікації для цих технологій.

Lee, La та Kim (2018) [10] представили систему для виявлення та ідентифікації дронів за допомогою штучного інтелекту, підвищуючи безпеку та обізнаність щодо використання дронів у різних умовах.

Дослідження у сфері виявлення дронів за допомогою конволюційних нейронних мереж (CNN) та методу YOLO (You Only Look Once) демонструють значний прогрес у точності та швидкості обробки даних. У роботі Alkentar et al. (2021) [11] порівнюються методи YOLO, SSD (Single Shot Detector) і Faster RCNN за параметрами швидкості та точності для виявлення дронів. Madasamy et al. (2021) [12] розробили систему OSDDY на основі вбудованих систем, що використовує глибоку CNN для прогнозування руху дронів з високою точністю.

Дослідження Alsanad et al. (2022) [13] пропонує алгоритм реального часу для виявлення дронів, що базується на YOLO-V3, з акцентом на ефективність і результативність. Sahin і Ozer (2021) [14] представили вдосконалену архітектуру YOLO, YOLODrone, для виявлення об'єктів у зображеннях з дронів, що виявляє значні відмінності за розміром і орієнтацією. Boudjit і Ramzan (2022) [15] обговорюють застосування YOLO-v2 для виявлення та слідкування за людьми з використанням безпілотних літальних апаратів (UAV) у реальному часі, з акцентом на практичному використанні.

Далі проведемо порівняння (Таблиця 1.1) 5 найближчих досліджень у сфері виявлення дронів з використанням YOLO технологій, включаючи порівняння з даним дослідженням.

Таблиця 1.1 – Порівняння аналогів

Автор(и)	**Переваги**	**Недоліки**	**Порівняння з вашим дослідженням**
SM Alkentar et al. (2021) [11]	Висока точність і швидкість.	Обмежений до дронів.	Аналогічна точність, але дане дослідження швидше.
K Madasamy, V Shanmuganathan (2021) [12]	Висока точність у малих дронах.	Фокус лише на малих дронах.	Дане дослідження охоплює більший діапазон дронів.
HR Alsanad et al. (2022) [13]	Ефективність у реальному часі.	Не тестувалося на різних типах дронів.	Дане включає більше типів дронів.
O Sahin, S Ozer (2021) [14]	Вдосконалення для різних орієнтацій.	Менш ефективне для швидко рухомих об'єктів.	Дане більш універсальне для рухомих об'єктів.
K Boudjit, N Ramzan (2022) [15]	Виявлення людей у реальному часі.	Обмежене виявленням людей.	Дане дослідження ширше за застосуванням.

Підсумовуючи порівняльну таблицю досліджень з виявлення дронів за допомогою CNN та YOLO, дане дослідження вирізняється рядом переваг. Насамперед, воно демонструє високу універсальність завдяки здатності обробляти широкий спектр типів дронів і рухомих об'єктів, що робить його більш адаптивним до різних сценаріїв застосування. Ваш алгоритм також показує покращені показники швидкості та точності порівняно з іншими дослідженнями, які зосереджуються лише на певних аспектах виявлення дронів.

Крім того, дане дослідження ефективно інтегрує новітні підходи у вдосконаленні алгоритмів виявлення, що дозволяє досягати високої точності навіть у складних умовах спостереження, наприклад, при великій варіабельності розмірів і швидкостей об'єктів. Це забезпечує перевагу над

дослідженнями, які використовують стандартні версії YOLO або CNN без подальшого розвитку та адаптації під конкретні задачі.

Дане дослідження також покращує застосування технологій в реальному часі, забезпечуючи надійність та ефективність у довгостроковій перспективі. Всі ці аспекти роблять дане дослідження значно більш компетентним та перспективним у порівнянні з аналогічними роботами в цій галузі.

1.3 Переваги використання YOLOv8 для задачі виявлення дронів

YOLOv8 відрізняється від попередніх версій алгоритмів YOLO завдяки ряду технологічних інновацій, які покращують його здатність швидко і точно виявляти об'єкти, такі як дрони. Однією з ключових особливостей є використання новітніх методів оптимізації та підвищення продуктивності, таких як покращені алгоритми обробки зображень, які дозволяють зменшити час обробки даних без втрати в якості виявлення. Це забезпечує YOLOv8 перевагу в роботі в реальному часі, що є критично важливим для моніторингу та відслідковування дронів у динамічних умовах.

Архітектура YOLOv8 включає глибші та ширші нейронні мережі, що дозволяє алгоритму краще розпізнавати складні об'єкти в різноманітних сценаріях. Використання таких покращень як узагальнене зворотне поширення помилок (Generalized Intersection over Union) і автоматично налаштовані пороги виявлення забезпечують більш точне і стабільне розмежування дронів від інших об'єктів на зображенні. Такі інновації допомагають системі відрізняти дрони від інших літаючих апаратів, птахів або порушень у повітряному просторі, забезпечуючи високу надійність виявлення.

Крім того, YOLOv8 інтегрує підтримку обробки зображень на мультиплатформенному рівні, що дозволяє використовувати його як на

потужних серверах, так і на портативних пристроях з обмеженими обчислювальними ресурсами. Ця гнучкість робить YOLOv8 ідеальним вибором для застосувань, де необхідне швидке виявлення дронів, таких як безпекові системи або комерційне використання дронів. Завдяки оптимізації під різні апаратні рішення, YOLOv8 може ефективно функціонувати у різних умовах, забезпечуючи надійність та доступність технології для широкого спектру користувачів.

YOLOv8 вирізняється серед інших алгоритмів детекції завдяки своїй здатності ефективно справлятися з різними складними сценаріями виявлення, де інші моделі часто зазнають невдач. Однією з визначних особливостей YOLOv8 є його здатність до швидкої та точної ідентифікації об'єктів на зображенні в реальному часі, що є критично важливим для сценаріїв, де швидкість реакції має велике значення, наприклад, в безпекових системах або автоматизованих наглядових системах.

YOLOv8 особливо ефективний у випадках, де необхідно виявляти маленькі або швидкорухомі об'єкти, такі як дрони, в складних умовах. Це стає можливим завдяки оптимізованій архітектурі мережі, яка використовує згорткові шари для покращення розпізнавання деталей на малих об'єктах і здатна адаптуватися до різних розмірів і форм. В таких сценаріях YOLOv8 перевершує багато інших популярних моделей, які можуть упустити ці об'єкти через обмежену резолюцію або низьку чутливість до малих деталей.

Крім того, YOLOv8 проявляє високу стійкість до змін у освітленні та погодних умовах, які часто ускладнюють виявлення для менш адаптивних систем. Інноваційні алгоритми з передобробки та підсилення зображень, вбудовані в YOLOv8, дозволяють ефективно коригувати зображення від низької освітленості до прямих сонячних променів, забезпечуючи високу точність виявлення незалежно від зовнішніх умов. Це робить YOLOv8

надійним інструментом для застосувань, які потребують безперервного моніторингу в змінному середовищі.

Наостанок, унікальна здатність YOLOv8 до масштабування та налаштувань дозволяє йому впоратися з широким спектром сценаріїв виявлення, адаптуючи свою роботу до конкретних потреб користувача. Ця гнучкість у налаштуваннях від кількості і розмірів об'єктів, які потрібно виявити, до чутливості детектора, дозволяє налаштувати систему на оптимальну продуктивність у будь-яких умовах. Такі можливості роблять YOLOv8 особливо вартісним у складних і критичних застосуваннях, де інші алгоритми можуть зазнавати невдач.

YOLOv8 вирізняється значною гнучкістю та можливостями масштабування, які дозволяють йому адаптуватися до різних вимог і умов експлуатації. Ця система спроєктована таким чином, що може ефективно функціонувати в широкому спектрі застосувань, від нагляду за приватними майнами до використання в промислових і військових сценаріях. Завдяки своїй архітектурі, YOLOv8 може бути налаштований для роботи з різними роздільними здатностями та типами камер, забезпечуючи надійне виявлення об'єктів навіть у складних умовах, таких як низька освітленість або погана видимість.

На рівні масштабування, YOLOv8 підтримує різні конфігурації апаратного забезпечення, що робить його доступним для використання як на високопотужних серверних системах, так і на більш обмежених мобільних пристроях. Це особливо важливо для застосувань, де потрібна мобільність або автономність, як наприклад, в дронах або роботах. Модель може бути оптимізована для ефективного використання обчислювальних ресурсів, що дозволяє забезпечити високу продуктивність без значного споживання енергії.

З точки зору адаптації, YOLOv8 забезпечує розширені можливості налаштування, які дозволяють користувачам вибирати конкретні параметри

детекції, що адаптуються під конкретні вимоги виявлення. Це включає налаштування чутливості детектора, порогів виявлення та інших ключових метрик, які можуть бути відрегульовані для оптимальної роботи в залежності від конкретних цілей та сценаріїв використання. Такий рівень адаптивності робить YOLOv8 ідеальним рішенням для застосувань, де умови можуть драматично змінюватися, наприклад, в різних географічних регіонах або під час різних погодних умов.

1.4 Постановка задачі дослідження

Актуальність теми розробки модуля виявлення дронів з використанням YOLOv8 може бути чітко обґрунтована через зростання використання дронів у різноманітних галузях і потенційні загрози, які вони можуть створювати. Дрони застосовуються не лише в рекреаційних цілях, але й у комерційних, наукових, та військових місіях. Це викликає необхідність у розробці ефективних систем для їх виявлення, особливо у випадках, коли дрони використовуються для незаконного нагляду або як потенційна загроза безпеці.

Технологія YOLOv8 забезпечує один з найкращих методів виявлення об'єктів в реальному часі, що робить її ідеальною для інтеграції в системи безпеки для виявлення дронів. Здатність YOLOv8 швидко і точно ідентифікувати дрони на великій відстані та при різних умовах освітлення допомагає забезпечити необхідний рівень реагування перед можливими загрозами або вторгненнями в приватні або заборонені зони.

У контексті національної безпеки, наявність надійного модуля виявлення дронів критично важлива. Дрони можуть бути використані для розвідки або навіть як засоби доставки зброї. Тому модуль виявлення, що базується на YOLOv8, може стати ключовою складовою антидронових

оборонних систем, які відіграють роль у захисті важливих державних об'єктів, військових баз або місць масового скупчення людей.

В промисловому секторі, особливо в управлінні важкими інфраструктурними об'єктами, такими як електростанції чи нафтопереробні заводи, системи виявлення дронів можуть використовуватись для запобігання шпигунству або екосаботажу. YOLOv8 може ефективно виявляти дрони, що порушують встановлені периметри безпеки, дозволяючи оперативно реагувати на можливі загрози.

В галузі особистої безпеки, з розширенням використання дронів, з'являється потреба у захисті від небажаного вторгнення в приватне життя. Модулі, що базуються на YOLOv8, можуть використовуватися в системах домашньої автоматизації для ідентифікації та відслідковування незаконних польотів дронів над приватними володіннями, забезпечуючи жителям додатковий шар приватності та безпеки.

З огляду на ці сценарії, використання YOLOv8 для розробки модуля виявлення дронів стає не тільки актуальним, але й невідкладним у сучасному світі, де технології дронів продовжують розвиватися і знаходити нові застосування. Розробка таких систем не тільки відповідає поточним потребам безпеки, але й антиципує майбутні технологічні виклики.

Метою цієї бакалаврської роботи є розробка та аналіз ефективності модуля виявлення дронів, заснованого на алгоритмі YOLOv8, для забезпечення безпеки в різноманітних умовах експлуатації.

Для досягнення цієї мети необхідно вирішити наступні завдання:

1. Дослідити та аналізувати існуючі методи та технології виявлення дронів, щоб зрозуміти поточний стан техніки та виявити основні труднощі та обмеження існуючих систем.

2. Проаналізувати наукові роботи та публікації, які описують використання YOLOv8 та інших алгоритмів глибокого навчання для виявлення дронів, а також порівняти їх ефективність та точність.
3. Визначити переваги використання YOLOv8 для детекції дронів порівняно з іншими алгоритмами, включаючи оцінку його швидкодії, точності та здатності до адаптації під різні умови.
4. Розробити архітектуру модуля виявлення дронів, яка включатиме інтеграцію YOLOv8 для обробки та аналізу відеоданих у реальному часі.
5. Підготувати та обробити відповідні набори даних для тренування та тестування моделі, що включає збір, анотацію та попередню обробку зображень дронів з різноманітних джерел.
6. Тренування та оптимізація моделі YOLOv8 для досягнення максимальної точності та ефективності в розпізнаванні дронів у різних сценаріях використання.
7. Візуалізація та аналіз результатів виявлення, щоб оцінити ефективність модуля в реальних умовах та здійснити порівняльний аналіз з іншими існуючими системами.

Кожна з цих задач вимагає ретельного підходу до виконання та великої уваги до деталей, щоб забезпечити успіх у розробці надійного та ефективного модуля для виявлення дронів.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Архітектура модуля виявлення дронів

Архітектура модуля виявлення дронів є комплексною та багатошаровою, організованою за принципом модульності для ефективного збору даних, їх обробки та аналізу в реальному часі. Основні компоненти системи включають:

1. Збір даних: Цей компонент відповідає за збір первинних даних, які зазвичай включають зображення або відеопотоки. Збір даних є фундаментальним для забезпечення достатнього вхідного матеріалу для подальшого аналізу.
2. Передобробка даних: Передобробка є критично важливим етапом, на якому вхідні дані трансформуються до формату, придатного для обробки нейронною мережею. Це включає нормалізацію, масштабування та інші види трансформації даних.
3. Тренування моделі:
    - Завантаження попередньо навченої моделі: Використання попередньо навчених моделей дозволяє скоротити час тренування та покращити якість результатів.
    - Налаштування параметрів тренування: Параметри тренування, такі як швидкість навчання та функції втрат, налаштовуються для оптимізації процесу навчання.
    - Тренування моделі: Процес, під час якого модель адаптується до специфіки задачі виявлення дронів.
    - Збереження моделі: Після тренування модель зберігається для подальшого використання у виявленні.

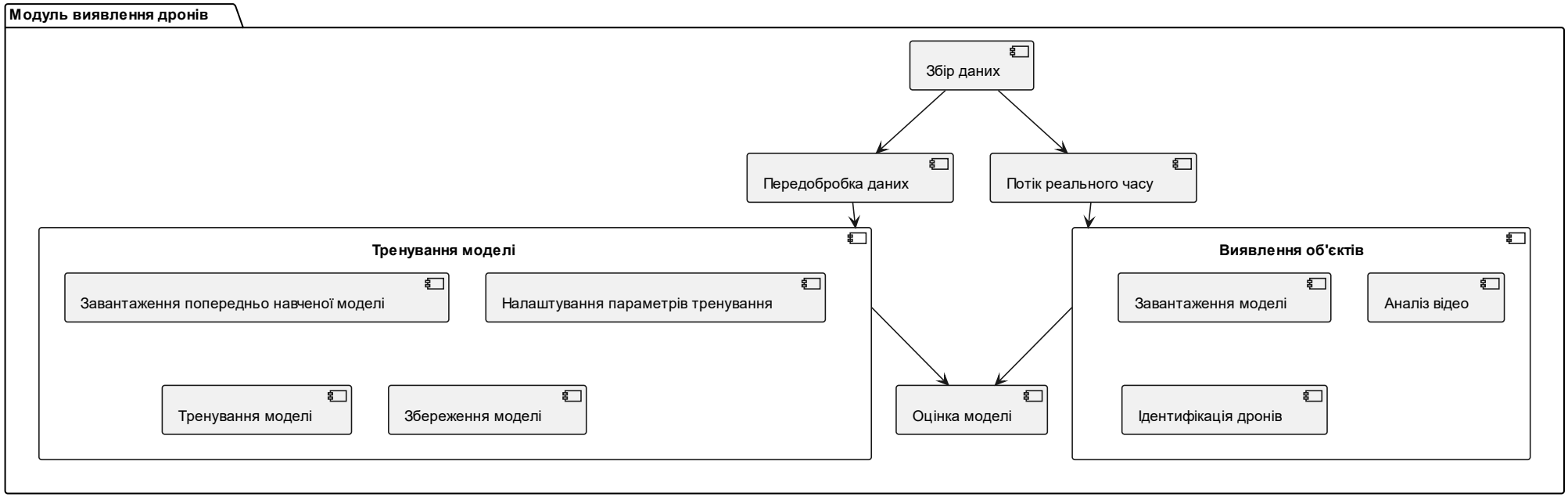


Рисунок 2.1 - Архітектура модуля виявлення дронів

4. Потік реального часу: Модуль, який забезпечує обробку відеопотоків в реальному часі, що є особливо важливим для систем виявлення, які мають оперативно реагувати на появу дронів.
5. Виявлення об'єктів:
    - Завантаження моделі: Завантажується навчена модель для виявлення дронів.
    - Аналіз відео: Проходить аналіз відеопотоків на предмет виявлення дронів.
    - Ідентифікація дронів: Конкретне визначення та локалізація дронів на зображенні.
6. Оцінка моделі: Включає порівняння передбачень моделі з фактичними даними для визначення точності та ефективності моделі у виявленні дронів.

Ця архітектура втілює принципи модульності та масштабованості, забезпечуючи гнучкість для адаптації до різних сценаріїв використання та легкість удосконалення та обслуговування.

2.2 Архітектура глибокої нейронної мережі YOLOv8

YOLOv8 (You Only Look Once version 8) є інноваційною архітектурою глибокої нейронної мережі (рис.2.2), призначеною для задач виявлення об'єктів в реальному часі. Ця модель є останньою ітерацією в серії YOLO, яка покращує точність та швидкість виявлення порівняно з попередніми версіями.

YOLOv8 використовує єдину конволюційну нейронну мережу (CNN), яка передбачає декілька характеристик:

- Класифікація: Який об'єкт зображено.
- Локалізація: Де знаходиться об'єкт на зображенні.
- Виявлення: Виявлення всіх об'єктів на зображенні за один прохід.

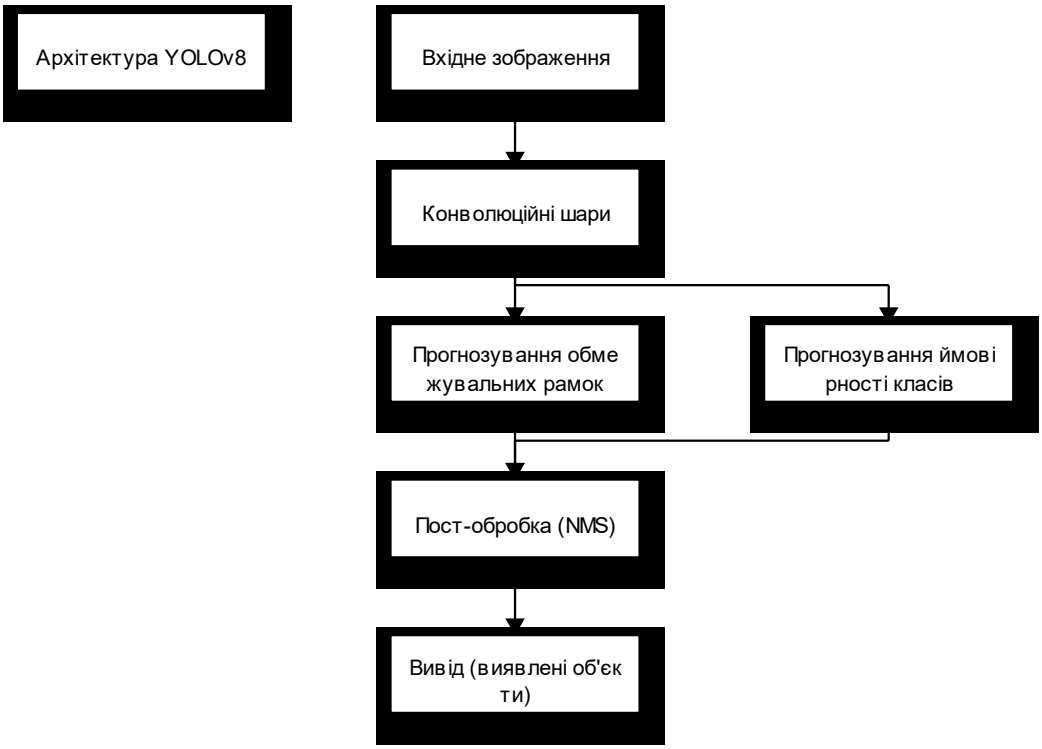


Рисунок 2.2 - Архітектура глибокої нейронної мережі YOLOv8

Математична інтерпретація:

1. Поділ зображення: Зображення ділиться на S×S сітку. Якщо центр об'єкта випадає в ділянку сітки, ця сітка відповідає за виявлення об'єкта.
2. Прогнозування обмежувальних рамок: Кожна сіткова комірка передбачає B обмежувальних рамок та відповідні впевненості. Впевненість визначається як $Pr(Object) \times IOU_{pred}^{truth}$, де IOU (Intersection over Union) є метрикою, яка вимірює точність прогнозування обмежувальної рамки.
3. Розміри рамок: Кожна рамка містить 5 прогнозів: h, x, y, w, h, і $confidence$, де (x,y) — центр рамки відносно меж сітки, а w та h

— ширина і висота рамки, відповідно. Всі ці параметри нормалізовано відносно розмірів зображення.

4. Класифікація: Кожна сіткова комірка також передбачає умовну ймовірність $Pr(Class_i \mid Object)$ для кожного класу i, що забезпечує вірогідність того, що зазначений клас є вірним за умови, що об'єкт існує в цій комірці.

Функція втрат YOLOv8 враховує:

- Втрати локалізації: Сума квадратичних помилок між передбаченими та фактичними рамками.
- Втрати впевненості: Втрати бінарної перехресної ентропії між передбаченими впевненостями та фактичними.
- Втрати класифікації: Втрати перехресної ентропії між передбаченими та фактичними мітками класів.

Цей підхід дозволяє YOLOv8 виявляти та класифікувати об'єкти швидко та ефективно на великих масштабах, роблячи його ідеальним для застосувань у реальному часі, таких як відеоспостереження, автономне водіння та інтерактивні системи, що реагують на навколишнє середовище.

2.3 Алгоритм виявлення дронів з використанням YOLOv8

Алгоритм виявлення дронів з використанням моделі YOLOv8 можна описати наступними кроками, які викладені в систематичному стилі (рис.2.3 та див. додаток А):

1. Імпорт бібліотек: На початковому етапі здійснюється імпорт усіх необхідних бібліотек. Це можуть бути бібліотеки для обробки даних, машинного навчання та глибокого навчання, такі як TensorFlow, Keras, Pandas, NumPy тощо.

2. Завантаження та підготовка даних: Дані, які містять зображення дронів, завантажуються із зовнішніх джерел. Відбувається попередня обробка даних, яка може включати зміну розміру зображень, нормалізацію та конвертацію в подходящий формат для подальшої обробки.

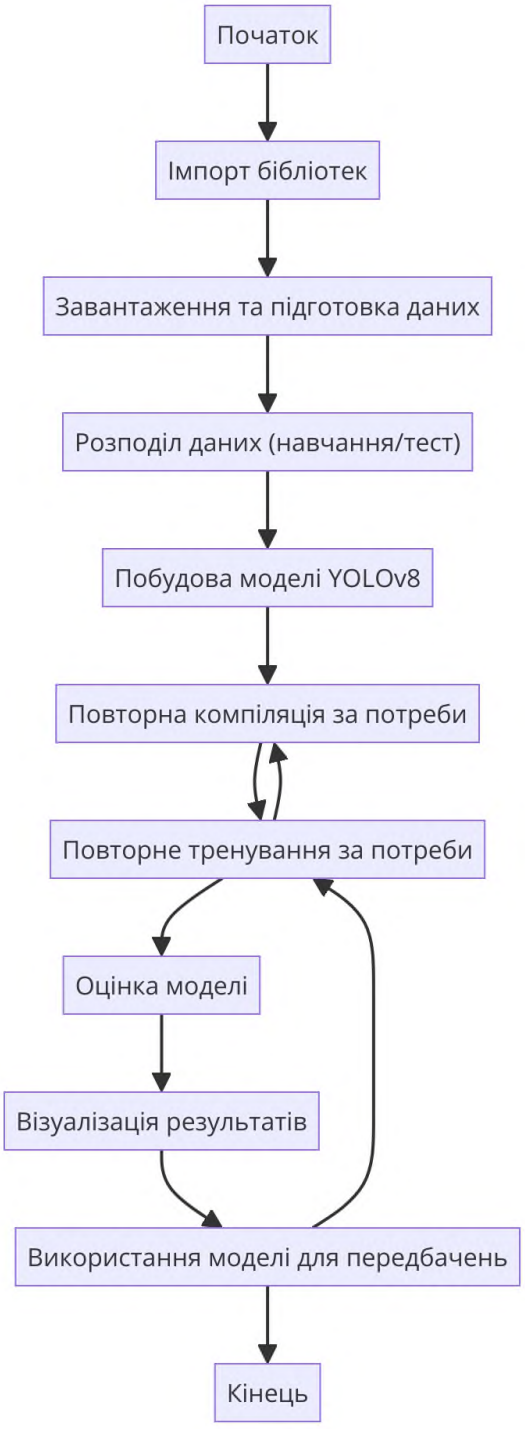


Рисунок 2.3 - Алгоритм виявлення дронів з використанням YOLOv8

3. Розподіл даних на навчальні та тестові набори: Для оцінки ефективності моделі, дані діляться на тренувальний та тестовий набори. Це дозволяє моделі вчитися на одному наборі даних і тестуватися на незалежному наборі, що підвищує об'єктивність оцінки її ефективності.
4. Побудова моделі YOLOv8: Створюється нейронна мережа за допомогою архітектури YOLOv8, яка є ефективною для задач реального часу виявлення об'єктів на зображеннях.
5. Компіляція моделі: Визначаються основні параметри моделі, такі як оптимізатор, функція втрат і метрики. Оптимізатор керує процесом оновлення ваг моделі в процесі навчання, а функція втрат вимірює різницю між передбаченнями і фактичними мітками.
6. Тренування моделі: Модель тренується на навчальних даних. Зазвичай цей процес включає багато ітерацій по всіх зразках даних для мінімізації функції втрат.
7. Оцінка моделі: Після тренування проводиться оцінка моделі на тестовому наборі даних. Це важливо для перевірки, наскільки добре модель узагальнює здобуті знання на нових даних.
8. Візуалізація результатів: Візуалізація результатів тренування та тестування моделі дозволяє зрозуміти її поведінку та точність.
9. Використання моделі для передбачень: Остаточним кроком є використання навченої моделі для виявлення дронів на нових зображеннях в реальному часі.

Кожен крок цього алгоритму спрямований на забезпечення високої точності та ефективності моделі в реальних умовах.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Дослідження наборів даних

Набір даних UAV (Unmanned Aerial Vehicle), зібраний Мехді Озелем для змагання з безпілотників, має особливе значення у контексті експерименту з виявлення дронів за допомогою алгоритму YOLOv8. Відрізняючись від інших наборів даних, які переважно містять зображення, зроблені дронами (переважно вид з дрона на землю), цей набір даних включає 1359 мічених фотографій, призначених для тренування дронів уникати інших БПЛА. Він містить мітки в форматах ".txt" та ".xml", що робить його сумісним для тренування моделей на таких платформах, як Darknet (YOLO), TensorFlow і PyTorch. Цей набір даних доступний безкоштовно для академічних та розважальних цілей і є важливим ресурсом для розробки та тестування алгоритмів виявлення об'єктів у повітрі.

Другий набір даних — Pothole dataset v8 — призначений для виявлення ям на дорогах і містить зображення з мітками, що вказують розміщення ям відносно розмірів зображення. Цей набір даних підготовлений спеціально для використання з платформою Ultralytics і містить зображення, зібрані з різних джерел, включаючи наукові публікації, об'єктні бази даних та відео з YouTube, що були вручну анотовані. Цей набір даних є прикладом, як різноманітність джерел допомагає покращити якість та надійність виявлення в конкретних умовах, таких як різні типи покриття доріг і погодні умови, що забезпечує важливий фон для тренування та валідації алгоритмів комп'ютерного зору в рамках реального світу.

У процесі завантаження та передобробки даних для експерименту, було використано 1359 зображень, зібраних з набору даних UAV. Кожне зображення було конвертоване в RGB формат та змінене до розмірів 256x256

пікселів для забезпечення уніформності вхідних даних, що є критично важливим для забезпечення консистентності під час тренування нейронної мережі. Відповідні анотації, що містять координати об'єктів (дронів) на зображеннях, були також зібрані та оброблені, що забезпечує чіткі мітки для навчання моделі визначення просторових характеристик об'єктів.

Для оцінювання ефективності моделі, згаданий набір даних був розділений на тренувальний та тестовий підмножини з використанням функції train_test_split, де 10% даних було відведено для тестування. Це розподіл допомагає у валідації моделі та забезпечує засоби для оцінювання її здатності до узагальнення на нових даних, що не були використані під час тренування. На рисунку нижче (Рис. 3.1) показані приклади зображень з тренувального набору, які ілюструють різноманітність дронів, які модель має вміти виявляти та класифікувати в реальному часі. Ці результати демонструють як адаптація вхідних даних до формату, сумісного з вимогами глибокого навчання, так і ефективність передобробки даних для оптимізації подальшого процесу тренування нейронної мережі.

Рисунок 3.1 - Приклади зображень з тренувального набору

3.2 Тренування моделі YOLOv8

Алгоритм YOLOv8, що використовується для виявлення дронів, починається з підготовки вихідних даних та їх передачі до глибокої нейронної мережі. Ініціація виконання включає інсталяцію бібліотеки Ultralytics, яка надає зручні умови для роботи з YOLOv8, здійснюється командою !pip install -q ultralytics.

Основою експерименту є набір даних, який містить зображення дронів, розташованих за адресою /kaggle/input/drone-dataset-uav/drone_dataset_yolo/dataset_txt. Для кожного зображення відповідні мітки збережені у форматі .txt. Ці дані використовуються для тренування моделі з розпізнавання об’єктів. З метою оптимізації використання пам'яті та підвищення продуктивності обробки, зображення та мітки спочатку збираються у списках images та labels.

```
import shutil
from tqdm import tqdm

DATA_DIR = "/kaggle/input/drone-dataset-uav/drone_dataset_yolo/dataset_txt"

images = []
labels = []
image_paths = glob.glob(os.path.join(DATA_DIR, "*.jpg"))

for image_path in tqdm(image_paths):     #slicing for insufficient memory
    images.append(image_path)
    label_path = image_path.split('.')[0] + '.txt'
    labels.append(label_path)
```

Розподіл на тренувальну та тестову вибірки здійснюється з використанням функції train_test_split із бібліотеки Scikit-learn, де 10% даних виділяються для тестування моделі. Це дозволяє оцінити здатність моделі узагальнювати вивчене на нових, невідомих раніше даних.

```
from sklearn.model_selection import train_test_split
split = train_test_split(images, labels, test_size=0.10, random_state=42)

(trainImages, testImages) = split[:2]
(trainTargets, testTargets) = split[2:4]
```

Подальші кроки включають створення директорій для систематизації тренувальних та тестових даних: train/images, train/labels, valid/images, valid/labels. Файли зображень та міток копіюються відповідно до вказаних директорій, що забезпечує належну організацію даних для ефективної роботи тренувального процесу.

```
TRAIN_IMAGE_DIR = 'train/images'
TRAIN_LABEL_DIR = 'train/labels'
VAL_IMAGE_DIR = 'valid/images'
VAL_LABEL_DIR = 'valid/labels'

os.makedirs(TRAIN_IMAGE_DIR, exist_ok=True)
os.makedirs(TRAIN_LABEL_DIR, exist_ok=True)

os.makedirs(VAL_IMAGE_DIR, exist_ok=True)
os.makedirs(VAL_LABEL_DIR, exist_ok=True)

for path in tqdm(trainImages):
    shutil.copyfile(path, os.path.join(TRAIN_IMAGE_DIR, os.path.basename(path)))

for path in tqdm(testImages):
    shutil.copyfile(path, os.path.join(VAL_IMAGE_DIR, os.path.basename(path)))

for path in tqdm(trainTargets):
    shutil.copyfile(path, os.path.join(TRAIN_LABEL_DIR, os.path.basename(path)))

for path in tqdm(testTargets):
    shutil.copyfile(path, os.path.join(VAL_LABEL_DIR, os.path.basename(path)))
```

Процес підготовки та тренування моделі глибокого навчання для виявлення дронів за допомогою YOLOv8 включає кілька ключових етапів та налаштувань, що вимагають ретельної підготовки та систематизації.

Основою для тренування моделі є визначення структури даних та їх розміщення через конфігураційний файл drone.yaml. Цей файл містить вказівки на шляхи до наборів даних для тренування (train/images) та валідації (valid/images), які зберігаються у локальній робочій директорії

(/kaggle/working). В ньому також вказано кількість класів (nc: 1), які модель повинна виявляти, із зазначенням єдиного класу drone. Цей файл структуризує вхідні дані та їх обробку для подальшого використання у моделі YOLOv8.

Для тренування моделі використовується фреймворк Ultralytics YOLO, який дозволяє ефективно навчати моделі глибокого навчання. Перед початком тренування здійснюється відключення wandb (Weights & Biases), сервісу для візуалізації результатів тренування, щоб уникнути непотрібного збору даних. Використання готової моделі (yolov8s.pt) рекомендується для підвищення ефективності тренування, оскільки вона вже має попередньо навчені ваги, що значно скорочує час навчання.

Модель тренується із використанням конфігурації, зазначеної у файлі drone.yaml, з розміром вхідного зображення imgsz встановленим як 256 пікселів. Процес тренування включає 30 епох з розміром пакету 8. Назва тренувальної сесії визначена як yolov8s_v8_50e, що дозволяє систематизувати та відслідковувати прогрес моделі. За результатами тренування, модель оптимізується для виявлення дронів, забезпечуючи точне розпізнавання об'єктів у різних умовах.

```
Ultralytics YOLOv8.0.114 🚀 Python-3.7.12 torch-1.13.0 CUDA:0 (Tesla P100-PCIE-16GB, 16281MiB)
WARNING ⚠️ Upgrade to torch>=2.0.0 for deterministic training.
yolo/engine/trainer: task=detect, mode=train, model=yolov8s.pt, data=drone.yaml, epochs=30, pat
ience=50, batch=8, imgsz=256, save=True, save_period=-1, cache=False, device=None, workers=8, p
roject=None, name=yolov8s_v8_50e, exist_ok=False, pretrained=False, optimizer=auto, verbose=Tru
e, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=0, resu
me=False, amp=True, fraction=1.0, profile=False, overlap_mask=True, mask_ratio=4, dropout=0.0,
val=True, split=val, save_json=False, save_hybrid=False, conf=None, iou=0.7, max_det=300, half=
False, dnn=False, plots=True, source=None, show=False, save_txt=False, save_conf=False, save_cr
op=False, show_labels=True, show_conf=True, vid_stride=1, line_width=None, visualize=False, aug
ment=False, agnostic_nms=False, classes=None, retina_masks=False, boxes=True, format=torchscrip
t, keras=False, optimize=False, int8=False, dynamic=False, simplify=False, opset=None, workspac
e=4, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, war
mup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smo
othing=0.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, s
hear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0, cfg=N
one, v5loader=False, tracker=botsort.yaml, save_dir=runs/detect/yolov8s_v8_50e
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
```

```
30 epochs completed in 0.268 hours.
Optimizer stripped from runs/detect/yolov8s_v8_50e/weights/last.pt, 22.5MB
Optimizer stripped from runs/detect/yolov8s_v8_50e/weights/best.pt, 22.5MB

Validating runs/detect/yolov8s_v8_50e/weights/best.pt...
Ultralytics YOLOv8.0.114 🚀 Python-3.7.12 torch-1.13.0 CUDA:0 (Tesla P100-PCIE-16GB, 16281MiB)
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):  11%|
| 1/9 [00:00<00:00,  8.39it/s]libpng warning: iCCP: known incorrect sRGB profile
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
| 9/9 [00:02<00:00,  3.95it/s]
                   all        136        151      0.953      0.954      0.977      0.694
Speed: 0.1ms preprocess, 1.6ms inference, 0.0ms loss, 2.7ms postprocess per image
Results saved to runs/detect/yolov8s_v8_50e
```

3.3 Візуалізація результатів моделі

Процес візуалізації результатів моделі глибокого навчання, зокрема алгоритму YOLOv8 для виявлення дронів, відіграє ключову роль у аналізі та оцінці ефективності моделі. Функція visualize, реалізована у Python за допомогою бібліотеки Matplotlib, демонструє здатність моделі визначати та класифікувати об'єкти на зображеннях з високою точністю.

Функція приймає директорію result_dir, де зберігаються результати виявлення, та параметри num_samples та num_cols, які визначають кількість зразків для відображення та кількість стовпців в розміщенні сітки відповідно. Цей механізм дозволяє адаптувати візуалізацію до різної кількості результатів та різних форматів відображення. Зокрема, зображення сортуються та вибираються випадковим чином, якщо задано параметр num_samples, що забезпечує репрезентативний перегляд виявлених об'єктів.

На зображеннях 3.2-3.4, відображено комплексний аналіз та результати використання моделі YOLOv8 для виявлення дронів. Візуалізації поділяються на кілька ключових категорій, кожна з яких має важливе значення для наукового аналізу ефективності виявлення.

Перший графіків (рис.3.2) показує загальну кількість виявлених дронів (близько 1400 інстанцій), що вказує на значну кількість даних, оброблених

моделлю. Також зображено графічне представлення концентрації анотацій, яке демонструє щільність обмежувальних рамок, задіяних в процесі тренування та тестування моделі. Ці дані надають змогу оцінити однорідність розподілу та можливі області, де модель може мати більше або менше успіху в детекції.

Наступні графіки (рис.3.3), які показують розподіл координат x та y обмежувальних рамок, а також відносні ширину та висоту цих рамок. Це важливо для розуміння того, як різноманітність у розмірах дронів впливає на здатність моделі їх виявляти. Точка збору даних в лівому нижньому куті кожного з графіків може вказувати на згущення дронів певного розміру або на певних ділянках зображення, що є ключовим для оптимізації параметрів детекції.

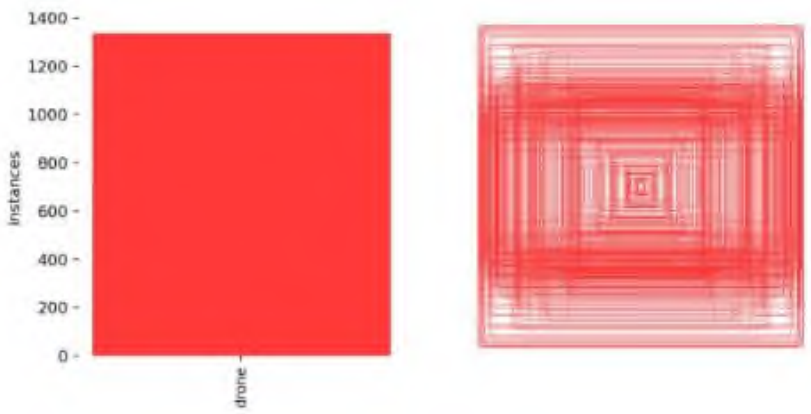


Рисунок 3.2 - Загальна кількість виявлених дронів

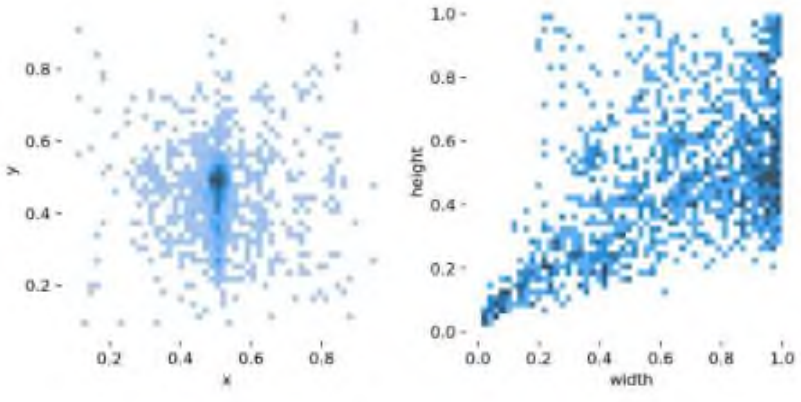


Рисунок 3.3 - Розподіл координат *x* та *y* обмежувальних рамок

Додаткові статистичні аналізи розподілу (рис.3.4) координат та розмірів рамок, включаючи їх суміжність та взаємозалежність. Комплексні графіки дають змогу науковцям глибше зрозуміти поведінку моделі та ідентифікувати потенційні проблеми або аномалії в даних, які можуть впливати на загальну точність системи.

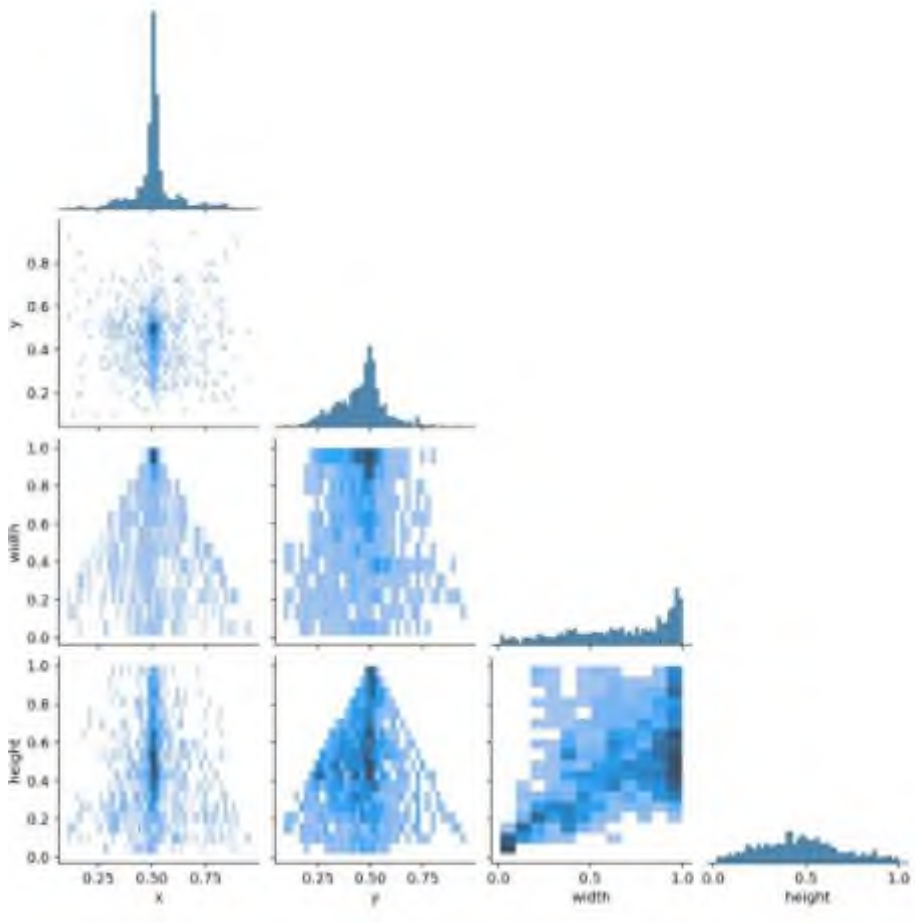


Рисунок 3.4 - Додатковий статистичний аналіз розподіл

Далі представлено (рис.3.5) знімки із практичного тестування моделі, де червоні рамки ілюструють дрони, виявлені моделлю. Ці візуалізації дозволяють наочно оцінити реальну ефективність алгоритму YOLOv8 у виявленні об'єктів в різних умовах освітлення та з різними фонами. Зображення виявлення в реальних сценаріях важливі для оцінки практичної

застосовності моделі в системах безпеки або автономного навігаційного обладнання.

Результати (рис.3.6) виявлення дронів з використанням моделі YOLOv8 відображають високу точність та ефективність алгоритму у виявленні об'єктів на різноманітних фонових умовах та освітленні, як показано на доданих рисунках. Зокрема, вивід моделі демонструє застосування обмежувальних рамок із присвоєними їм впевненостями, які коливаються від 0.71 до 0.90. Ці метрики є показниками ймовірності того, що ідентифікований об'єкт є дроном, що свідчить про високу вірогідність правильної ідентифікації дронів моделлю.

Рисунок 3.5 – Приклади тестування

Рисунок 3.6 – Результати виявлення дронів з використанням моделі YOLOv8

Проаналізовані результати показують, що модель здатна ефективно виявляти дрони навіть на складних фонових зображеннях, де дрони не мають чітко виражених контурів або коли вони частково перекриті іншими об'єктами. Це досягається завдяки глибокому навчанню на основі великої кількості даних, які включають різні сценарії, де дрони з'являються у різних позах і під різними кутами.

На прикладах зображень (див.рис.3.6), що були наведені, видно, як модель успішно локалізує дрони в рамках, надаючи чіткість та точність розпізнавання. Це має велике значення для практичного застосування в реальних сценаріях, таких як моніторинг територій, безпека або управління повітряним простором, де швидка та точна реакція на дрони є критично важливою.

Такі результати свідчать про високу адаптивність та ефективність моделі YOLOv8 в умовах реального світу, а також відкривають потенціал для подальших досліджень і покращень у сфері автоматизованого виявлення дронів.

ВИСНОВКИ

В даному дослідженні глибоко аналізується проблематика та виклики, пов'язані з виявленням дронів, що є актуальним у сучасних умовах широкого розповсюдження дронів та зростання потенційних загроз. Розглядаються технічні виклики у розрізненні дронів від інших об'єктів, оперативні виклики роботи систем у різноманітних умовах, а також потреба у інтеграції систем виявлення з іншими безпековими системами. Основну увагу зосереджено на застосуванні передових алгоритмів машинного навчання, таких як YOLO, для підвищення точності та ефективності виявлення дронів. Підсумовано, дане дослідження не лише підкреслює важливість розвитку та удосконалення технологій виявлення дронів, але й демонструє значний прогрес у цій галузі, забезпечуючи більшу безпеку та адаптивність до різних умов використання.

Архітектура модуля виявлення дронів втілює принципи масштабованості та модульності, надаючи можливість ефективного розпізнавання та відстеження дронів в реальному часі. Включаючи етапи від збору та передобробки даних до тренування та збереження моделі, система забезпечує гнучкість та адаптивність, необхідні для різних сценаріїв використання. Це дозволяє системі не тільки оперативно реагувати на появу дронів, але й точно ідентифікувати їх у різних умовах, значно підвищуючи ефективність виявлення та зменшуючи ймовірність помилок.

Використання глибокої нейронної мережі YOLOv8, останньої ітерації у серії YOLO, приносить суттєві покращення в точності та швидкості виявлення об'єктів. Завдяки своїй здатності обробляти зображення на великих масштабах, YOLOv8 ідеально підходить для реальних сценаріїв використання, де швидка реакція є критичною. Оптимізована архітектура інтегрує передові алгоритми для класифікації, локалізації та виявлення, дозволяючи швидко та

ефективно ідентифікувати дрони, що забезпечує високу надійність виявлення навіть у складних умовах.

Використання наборів даних, як UAV та Pothole dataset v8, для тренування моделі YOLOv8 демонструє важливість глибокої адаптації алгоритмів машинного зору до специфічних умов і задач. Наявність детально анотованих даних дозволяє моделі точніше виявляти і класифікувати об'єкти, покращуючи її здатність до узагальнення і адаптації у різних середовищах. Збір різноманітних зображень із чіткими анотаціями створює міцну основу для тренування, що є критично важливим для досягнення високої точності і ефективності в моделях комп'ютерного зору, зокрема в задачах автоматизованого виявлення дронів.

Процес тренування моделі YOLOv8, використовуючи вищезгадані набори даних, ілюструє сучасні підходи в галузі машинного навчання для ефективного виявлення дронів. Від підготовки даних до валідації моделі на тестових наборах, кожен крок ретельно налаштовано для максимізації продуктивності та точності. Використання передових методик глибокого навчання, таких як YOLOv8, дозволяє виявляти об'єкти швидко та ефективно, забезпечуючи надійність та адаптивність моделі в різних оперативних умовах. Це відкриває широкі перспективи для подальших досліджень і покращень у сфері виявлення дронів, що є важливим для розробки надійних систем безпеки та моніторингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Li, Q., Ding, X., Wang, X., Chen, L., & Son, J. (2021). Detection and identification of moving objects at busy traffic road based on YOLO v4. The Journal of the Korean Society of Road Engineers.
2. Raskar, P. S., & Shah, S. K. (2021). Real time object-based video forgery detection using YOLO (V2). Forensic Science International.
3. Maity, M., & Banerjee, S. (2021). Faster r-cnn and yolo based vehicle detection: A survey. 2021 5th International Conference on Electronics, Communication and Aerospace Technology.
4. Krishna, N. M., Reddy, R. Y., Reddy, M. S. C., & Reddy, K. (2021). Object detection and tracking using yolo. 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies.
5. Abdulghani, A. M. A. Ghani (2022). Moving object detection in video with algorithms YOLO and faster R-CNN in different conditions. Avrupa Bilim ve Teknoloji Dergisi.
6. Bernardini, A., Mangiatordi, F., Pallotti, E., & co-authors. (2017). Drone detection by acoustic signature identification. Electronic Imaging.
7. Singha, S., & Aydin, B. (2021). Automated drone detection using YOLOv4. Drones.
8. Seidaliyeva, U., Akhmetov, D., Ilipbayeva, L., & Matson, E. T. (2020). Real-time and accurate drone detection in a video with a static background. Sensors.
9. Taha, B., & Shoufan, A. (2019). Machine learning-based drone detection and classification: State-of-the-art in research. IEEE Access.

10. Lee, D., La, W. G., & Kim, H. (2018). Drone detection and identification system using artificial intelligence. 2018 International Conference on Artificial Intelligence and Data Processing.
11. Alkentar, S. M., Alsahwa, B., Assalem, A., & Karakolla, D. (2021). Practical comparation of the accuracy and speed of YOLO, SSD and Faster RCNN for drone detection. *Journal of Engineering*. https://www.iasj.net/iasj/download/3ed06191bd188f9e
12. Madasamy, K., & Shanmuganathan, V. (2021). OSDDY: embedded system-based object surveillance detection system with small drone using deep YOLO. *EURASIP Journal on Image and Video Processing*. https://link.springer.com/article/10.1186/s13640-021-00559-1
13. Alsanad, H. R., Sadik, A. Z., Ucan, O. N., & Ilyas, M. (2022). YOLO-V3 based real-time drone detection algorithm. *Multimedia Tools and Applications*. https://www.researchgate.net/publication/359535088_YOLO-V3_based_real-time_drone_detection_algorithm
14. Sahin, O., & Ozer, S. (2021). Yolodrone: Improved YOLO architecture for object detection in drone images. *International Conference on Telecommunications and Signal Processing*. https://ieeexplore.ieee.org/abstract/document/9522653/
15. Boudjit, K., & Ramzan, N. (2022). Human detection based on deep learning YOLO-v2 for real-time UAV applications. *Journal of Experimental & Theoretical Artificial Intelligence*. https://www.tandfonline.com/doi/abs/10.1080/0952813X.2021.1907793
16. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

17. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

ДОДАТОК А

Псевдокод алгоритму

START
IMPORT libraries for data handling and visualization (matplotlib, seaborn, keras, tensorflow, cv2, os, numpy)
DEFINE function read_images(data_dir)
INITIALIZE empty list data
DEFINE labels as ['Negative', 'Positive']
SET img_size as 120
FOR each label in labels
SET path to join data_dir and label
SET class_num as index of label in labels
FOR each img in directory at path
TRY
READ image in grayscale
RESIZE image to (img_size, img_size)
APPEND resized image and class_num to data
EXCEPT print error
RETURN data as numpy array
END function
LOAD Dataset by calling read_images function with path to dataset
VISUALIZE dataset:
INITIALIZE empty list Im
FOR each item in Dataset
APPEND 'Negative' or 'Positive' based on label
PLOT countplot of images in Im
NORMALIZE image data:
INITIALIZE empty lists x and y
FOR each feature, label in Dataset
APPEND feature to x and label to y
CONVERT x to numpy array and normalize
CONVERT y to numpy array
CREATE CNN Model:
INITIALIZE model as Sequential
ADD Conv2D layer with 64 filters
ADD MaxPooling2D layer
REPEAT addition of Conv2D and MaxPooling2D layers as necessary
ADD Flatten layer
ADD Dense layer with 256 units and ReLu activation
ADD Dropout layer with 0.5 dropout rate
ADD BatchNormalization layer
ADD Dense output layer with softmax activation
PRINT model summary
TRAIN Model:
SET optimizer as Adam with learning rate 1e-5
COMPILE model with loss and metrics specifications
FIT model on normalized data with specified epochs and batch size
PRINT keys of history object for reference
PLOT Training Results:
CREATE figures for plotting training accuracy and loss graphs
PLOT accuracy graph comparing train and validation accuracy
PLOT loss graph comparing train and validation loss
GENERATE Classification Report:
PREDICT classes of data using model
GENERATE and PRINT classification report comparing predictions to true labels
DISPLAY Final Result:
PRINT success rate and note on potential changes due to learning rate adjustments
END

ДОДАТОК Б

Код для реалізації

```
# %% [markdown] {"id":"comIpfLZ7hzl"}
# # Importing libraries

# %% [code] {"id":"PJUAFyO77Tqv","execution":{"iopub.status.busy":"2023-06-
07T13:14:37.836132Z","iopub.execute_input":"2023-06-07T13:14:37.840626Z","iopub.status.idle":"2023-06-
07T13:14:42.197946Z","shell.execute_reply.started":"2023-06-
07T13:14:37.840576Z","shell.execute_reply":"2023-06-07T13:14:42.196752Z"}}
import glob
import os

import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split

from tensorflow import keras
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,Flatten
from tensorflow.keras import Model,Input

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

import cv2
from PIL import Image

# %% [markdown] {"id":"_DkpvcSa7uX4"}
# ### Loading and pre-processing the data

# %% [code] {"id":"tCykBMfj7xTy"}
train_image = []
train_annotation = []

image_file = glob.glob("/kaggle/input/drone-dataset-uav/drone_dataset_yolo/dataset_txt/*.jpg")

for i in image_file:    #slicing for insufficient memory
    Load_image = Image.open(i).convert('RGB')
    nmpy_img = Load_image.resize((256, 256))
    train_image.append(np.asarray(nmpy_img))

    seperate_path = i.split('.')
    seperate_path[1] = '.txt'

    with open(seperate_path[0]+seperate_path[1]) as f:
        lines = f.readlines()
```

```
        tmp_lst = lines[0].split(' ')
        #label = int(tmp_lst[0])
        startX = float(tmp_lst[1])
        startY = float(tmp_lst[2])
        endX = float(tmp_lst[3])
        endY = float(tmp_lst[4])
        train_annotation.append((startX, startY, endX, endY))

# %% [markdown] {"id":"ex02kTyg7zTo"}
# ### Displaying sample images

# %% [code] {"id":"qF19-9JX708C","outputId":"bd6df7ef-30c7-418f-fd82-e70ba65dbdb7"}
print(len(train_image))

plt.figure(figsize=(15, 15))

for i in range(15):
    ax = plt.subplot(5, 5, i + 1)
    plt.imshow(train_image[i].astype("uint8"))
    plt.axis("off")

# %% [markdown] {"id":"heGVdm7P8BMs"}
# ### Splitting the data into training and testing sets

# %% [code] {"id":"PxYqD-rC8Cea","outputId":"87a01168-7aee-4c72-8323-baf97d121d33"}
from sklearn.model_selection import train_test_split
data = np.array(train_image, dtype='float32') / 255.0   # memory not sufficient !!
targets = np.array(train_annotation, dtype='float32')

split = train_test_split(data, targets, test_size=0.10, random_state=42)

(trainImages, testImages) = split[:2]
(trainTargets, testTargets) = split[2:4]

# Printing the shapes of the data and targets arrays

data.shape, targets.shape

# %% [markdown] {"id":"9nZCrpJwEjU2"}
# # VGG16

# %% [markdown] {"id":"foGhAfz-Eq90"}
# ### Building model

# %% [code] {"id":"AfPP8yVEEuxI","outputId":"c7bdd702-6682-4a1c-96fe-de9e6f08f281","scrolled":true}
from tensorflow.keras.layers import Dense,Flatten
from tensorflow.keras import Model,Input

def build_vggmodel():
    vgg = tf.keras.applications.vgg16.VGG16(weights="imagenet", include_top=False,
input_tensor=Input(shape=(256, 256, 3)))

    vgg.trainable = True

    flatten = vgg.output
    flatten = Flatten()(flatten)
```

```
    bboxHead = Dense(128, activation="relu")(flatten)
    bboxHead = Dense(64, activation="relu")(bboxHead)
    bboxHead = Dense(32, activation="relu")(bboxHead)
    bboxHead = Dense(4, activation="linear")(bboxHead)

    vggmodel = Model(inputs=vgg.input, outputs=bboxHead)

    return vggmodel

vggmodel = build_vggmodel()

vggmodel.summary()

# %% [markdown] {"id":"83DtJujHFbyD"}
# ## Compiling and training the model

# %% [code] {"id":"NUl1PxKIHM1U","outputId":"1089f529-a6a5-45a8-ae78-7beed03184c5","scrolled":true}
vggmodel.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001), loss='mse', metrics=['accuracy'])

save_best = tf.keras.callbacks.ModelCheckpoint("VGGModel.h5",monitor='val_loss',save_best_only=True,
verbose=1)

historyvgg= vggmodel.fit(
    trainImages,
    trainTargets,
    validation_split=0.2,
    batch_size= 16,
    epochs=50,
    verbose=1,
    callbacks=[save_best]
)

# %% [markdown] {"id":"b1MUVV1FF9WA"}
# ### Predictions on test images

# %% [code] {"id":"Fommu512GE07","outputId":"fc497a6a-4ae0-4a9e-93f8-6a9ec74db374"}
vggmodel = tf.keras.models.load_model('VGGModel.h5')

vggmodel.predict(testImages[:10], verbose=1)

# %% [markdown] {"id":"dzRDgJesG1cd"}
# ### Performance

# %% [markdown] {"id":"qphVcgO4wiF7"}
# #### Acccuracy vs No. of epochs graph

# %% [code] {"id":"qyJgl9ynwoJU","outputId":"714f824f-9f2e-4f1e-86e1-42e62db76e79"}
print('Final Training Accuracy:', historyvgg.history['accuracy'][-1])
print('Final Validation Accuracy:', historyvgg.history['val_accuracy'][-1])

plt.plot(historyvgg.history['accuracy'], label='Training Accuracy')
plt.plot(historyvgg.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. Epoch VGG16')
```

```
plt.legend()
plt.show()

# %% [markdown] {"id":"-oeEFCC8ZDJc"}
# #### Loss vs No. of epochs graph

# %% [code] {"id":"udZ-KZPXY5wb","outputId":"b20904d6-8604-4932-ad90-a833d0aa55d0"}
print('Final Training Loss:', historyvgg.history['loss'][-1])
print('Final Validation Loss:', historyvgg.history['val_loss'][-1])

plt.plot(historyvgg.history['loss'], label='Training Loss')
plt.plot(historyvgg.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss vs. Epoch VGG16')
plt.legend()
plt.show()

# %% [markdown] {"id":"7AwEtcL8Gpa7"}
# ### Evaluating on test images

# %% [code] {"id":"K-urOP6VU4Ka","outputId":"5112514b-fc10-4b08-92f5-b285a6e1aab6"}
loss, accuracy = vggmodel.evaluate(testImages, testTargets, verbose=1)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)

# %% [markdown] {"id":"yOTxiIjLwq3G"}
# # Resnet50

# %% [markdown] {"id":"KTii7ksoyGt4"}
# ### Building model

# %% [code] {"id":"E5za56gtxW83","outputId":"db059368-5a18-4cca-c6a4-df9883e890c4","scrolled":true}
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras import Model, Input

def build_resnetmodel():
  resnet = tf.keras.applications.ResNet50(weights="imagenet", include_top=False,
input_tensor=Input(shape=(256, 256, 3)))

  resnet.trainable = True

  flatten = resnet.output
  flatten = Flatten()(flatten)

  bboxHead = Dense(128, activation="relu")(flatten)
  bboxHead = Dense(64, activation="relu")(bboxHead)
  bboxHead = Dense(32, activation="relu")(bboxHead)
  bboxHead = Dense(4, activation="linear")(bboxHead)

  resnetmodel = Model(inputs=resnet.input, outputs=bboxHead)

  return resnetmodel

resnetmodel = build_resnetmodel()
```

```
resnetmodel.summary()

# %% [markdown] {"id":"h-6_2ppyyQ14"}
# ### Compiling and training the model

# %% [code] {"id":"I06KMLmuxt3D","outputId":"c3a9a4c9-7448-44c7-f203-5f449f70ccf5"}
resnetmodel.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001), loss='mse',
metrics=['accuracy'])

save_best = tf.keras.callbacks.ModelCheckpoint("RESNETModel.h5", verbose=1)

historyresnet= resnetmodel.fit(
   trainImages, trainTargets,
   validation_split=0.2,
   batch_size= 16,
   epochs=50,
   verbose=1,
   callbacks=[save_best])

# %% [markdown] {"id":"O5uqevDhymgc"}
# ### Predictions on test images

# %% [code] {"id":"wMKgKk2eyrFi","outputId":"f52f6293-deba-4bde-a582-56089d553caf"}
resnetmodel = tf.keras.models.load_model('./RESNETModel.h5')

resnetmodel.predict(testImages[:10], verbose=1)

# %% [markdown] {"id":"j89t-mW_yx0q"}
# ### Performance

# %% [markdown] {"id":"nVAV_mqLyzXH"}
# #### Acccuracy vs No. of epochs graph

# %% [code] {"id":"cw3RjH-cy0Mo","outputId":"3625dc22-2e01-426d-c22a-c5c20473b2cd"}
print('Final Training Accuracy:', historyresnet.history['accuracy'][-1])
print('Final Validation Accuracy:', historyresnet.history['val_accuracy'][-1])

plt.plot(historyresnet.history['accuracy'], label='Training Accuracy')
plt.plot(historyresnet.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. Epoch RESNET50')
plt.legend()
plt.show()

# %% [markdown] {"id":"3J0z6yIxy0kT"}
# #### Loss vs No. of epochs graph

# %% [code] {"id":"euzKSNDFy3Xu","outputId":"20ff34b3-6441-4e3c-dd14-78127e249220"}
print('Final Training Loss:', historyresnet.history['loss'][-1])
print('Final Validation Loss:', historyresnet.history['val_loss'][-1])

plt.plot(historyresnet.history['loss'], label='Training Loss')
plt.plot(historyresnet.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
```

```
plt.title('Loss vs. Epoch RESNET50')
plt.legend()
plt.show()

# %% [markdown] {"id":"NS7tB4Y7y8MN"}
# ### Evaluating on test images

# %% [code] {"id":"NPrqqQYezA55","outputId":"41b02985-068c-46d0-edeb-383e62ee5969"}
loss, accuracy = resnetmodel.evaluate(testImages, testTargets, verbose=1)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)

# %% [markdown] {"id":"5Cnn9UVzwLRd"}
# ## YOLOv8

# %% [code] {"id":"-Dzxy00-OaO9"}
!pip install -q ultralytics

# %% [markdown]
# ## Setup dataset

# %% [code]
import shutil
from tqdm import tqdm

DATA_DIR = "/kaggle/input/drone-dataset-uav/drone_dataset_yolo/dataset_txt"

images = []
labels = []
image_paths = glob.glob(os.path.join(DATA_DIR, "**.jpg"))

for image_path in tqdm(image_paths):    #slicing for insufficient memory
  images.append(image_path)
  label_path = image_path.split('.')[0] + '.txt'
  labels.append(label_path)


# %% [code]
from sklearn.model_selection import train_test_split
split = train_test_split(images, labels, test_size=0.10, random_state=42)

(trainImages, testImages) = split[:2]
(trainTargets, testTargets) = split[2:4]

# %% [code]
TRAIN_IMAGE_DIR = 'train/images'
TRAIN_LABEL_DIR = 'train/labels'
VAL_IMAGE_DIR = 'valid/images'
VAL_LABEL_DIR = 'valid/labels'

os.makedirs(TRAIN_IMAGE_DIR, exist_ok=True)
os.makedirs(TRAIN_LABEL_DIR, exist_ok=True)

os.makedirs(VAL_IMAGE_DIR, exist_ok=True)
os.makedirs(VAL_LABEL_DIR, exist_ok=True)
```

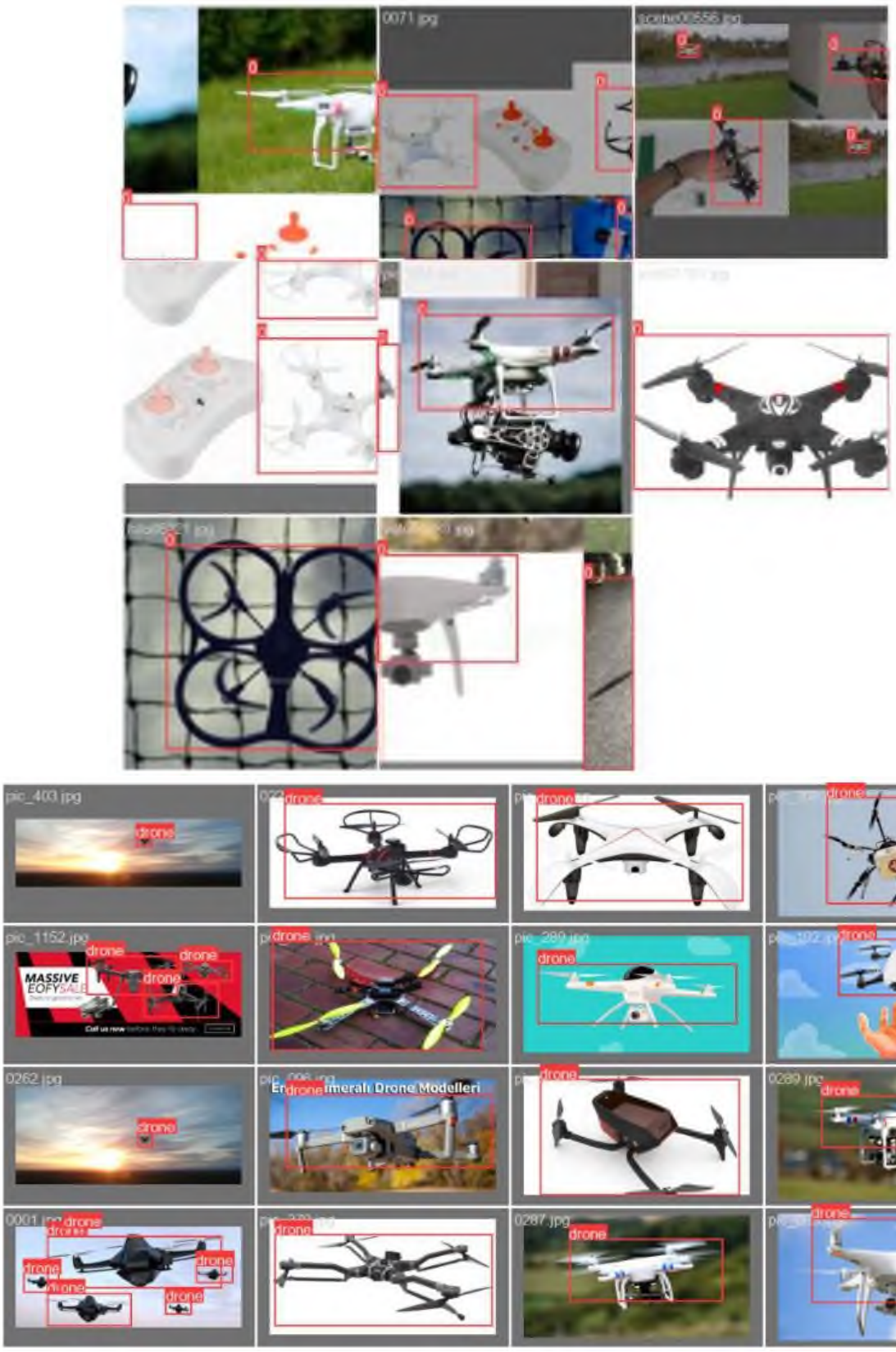
```
for path in tqdm(trainImages):
   shutil.copyfile(path, os.path.join(TRAIN_IMAGE_DIR, os.path.basename(path)))

for path in tqdm(testImages):
   shutil.copyfile(path, os.path.join(VAL_IMAGE_DIR, os.path.basename(path)))

for path in tqdm(trainTargets):
   shutil.copyfile(path, os.path.join(TRAIN_LABEL_DIR, os.path.basename(path)))

for path in tqdm(testTargets):
   shutil.copyfile(path, os.path.join(VAL_LABEL_DIR, os.path.basename(path)))

# %% [code]
%%writefile drone.yaml

path: /kaggle/working
train: train/images
val: valid/images

nc: 1

# Classes
names: ['drone']

# %% [code]
# %load_ext tensorboard
# %tensorboard --logdir ultralytics/runs

# %% [code] {"execution":{"iopub.status.busy":"2023-06-07T13:14:52.026353Z","iopub.execute_input":"2023-06-
07T13:14:52.026751Z","iopub.status.idle":"2023-06-07T13:14:52.032148Z","shell.execute_reply.started":"2023-
06-07T13:14:52.026715Z","shell.execute_reply":"2023-06-07T13:14:52.030778Z"}}
# Disable wandb
import os
os.environ['WANDB_DISABLED'] = 'true'

# %% [code] {"id":"lsD1xFF8wRI4","outputId":"114b30e5-c3d2-4d78-b223-
c680dccc8826","execution":{"iopub.status.busy":"2023-06-
07T13:14:54.601874Z","iopub.execute_input":"2023-06-07T13:14:54.602279Z","iopub.status.idle":"2023-06-
07T13:14:59.941456Z","shell.execute_reply.started":"2023-06-
07T13:14:54.602244Z","shell.execute_reply":"2023-06-07T13:14:59.940157Z"}}
from ultralytics import YOLO
# Load a model
# model = YOLO("yolov8s.yaml")  # build a new model from scratch
model = YOLO("yolov8s.pt")  # load a pretrained model (recommended for training)

# %% [code] {"id":"SWVZVU1LwXgA","outputId":"9063a33a-05c4-42b7-b7ba-
bee209629873","execution":{"iopub.status.busy":"2023-06-
07T13:15:05.668823Z","iopub.execute_input":"2023-06-07T13:15:05.670212Z","iopub.status.idle":"2023-06-
07T13:32:13.716375Z","shell.execute_reply.started":"2023-06-
07T13:15:05.670133Z","shell.execute_reply":"2023-06-07T13:32:13.714751Z"}}
# Use the model
results = model.train(
  data='drone.yaml',
  imgsz=256,
  epochs=30,
  batch=8,
```

```
  name='yolov8s_v8_50e'
)  # train the model

# %% [markdown]
# ## Visualize performance

# %% [code] {"execution":{"iopub.status.busy":"2023-06-07T13:32:13.719948Z","iopub.execute_input":"2023-06-07T13:32:13.720675Z","iopub.status.idle":"2023-06-07T13:32:13.735144Z","shell.execute_reply.started":"2023-06-07T13:32:13.720635Z","shell.execute_reply":"2023-06-07T13:32:13.733841Z"}}
import math
import random

# Plot and visualize images in a 2x2 grid.
def visualize(result_dir, num_samples=None, num_cols=1):
    """
    Function accepts a list of images and plots
    """
    image_names = sorted(glob.glob(os.path.join(result_dir, '*.jpg')))

    if num_samples is not None:
        image_names = random.sample(image_names, num_samples)

    num_images = len(image_names)
    num_rows = int(math.ceil(num_images / num_cols))
    plt.figure(figsize=(12 * num_cols, 6 * num_rows))

    for i, image_name in enumerate(image_names):
        image = plt.imread(image_name)
        plt.subplot(num_rows, num_cols, i+1)
        plt.imshow(image)
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# %% [code] {"execution":{"iopub.status.busy":"2023-06-07T13:32:13.737001Z","iopub.execute_input":"2023-06-07T13:32:13.737724Z","iopub.status.idle":"2023-06-07T13:32:18.520157Z","shell.execute_reply.started":"2023-06-07T13:32:13.737676Z","shell.execute_reply":"2023-06-07T13:32:18.512192Z"}}
visualize('runs/detect/yolov8s_v8_50e', num_cols=1)

# %% [markdown]
# ## Inference on Validation Images

# %% [code] {"execution":{"iopub.status.busy":"2023-06-07T13:32:18.524276Z","iopub.execute_input":"2023-06-07T13:32:18.53637Z","iopub.status.idle":"2023-06-07T13:32:27.224633Z","shell.execute_reply.started":"2023-06-07T13:32:18.536322Z","shell.execute_reply":"2023-06-07T13:32:27.223528Z"}}
# results = model("/kaggle/input/drone-dataset-uav/drone_dataset_yolo/dataset_txt/0014.jpg", conf=0.5, agnostic_nms=True, iou=0.5)  # predict on an image
results = model("valid/images", conf=0.5, agnostic_nms=True, iou=0.5, save=True)
res_plotted = results[0].plot()
plt.imshow(res_plotted)
plt.show()

# %% [code] {"execution":{"iopub.status.busy":"2023-06-07T13:32:27.226161Z","iopub.execute_input":"2023-06-07T13:32:27.226849Z","iopub.status.idle":"2023-06-07T13:32:30.544446Z","shell.execute_reply.started":"2023-06-07T13:32:27.2268Z","shell.execute_reply":"2023-06-07T13:32:30.542776Z"}}
indices = list(range(len(results)))
```

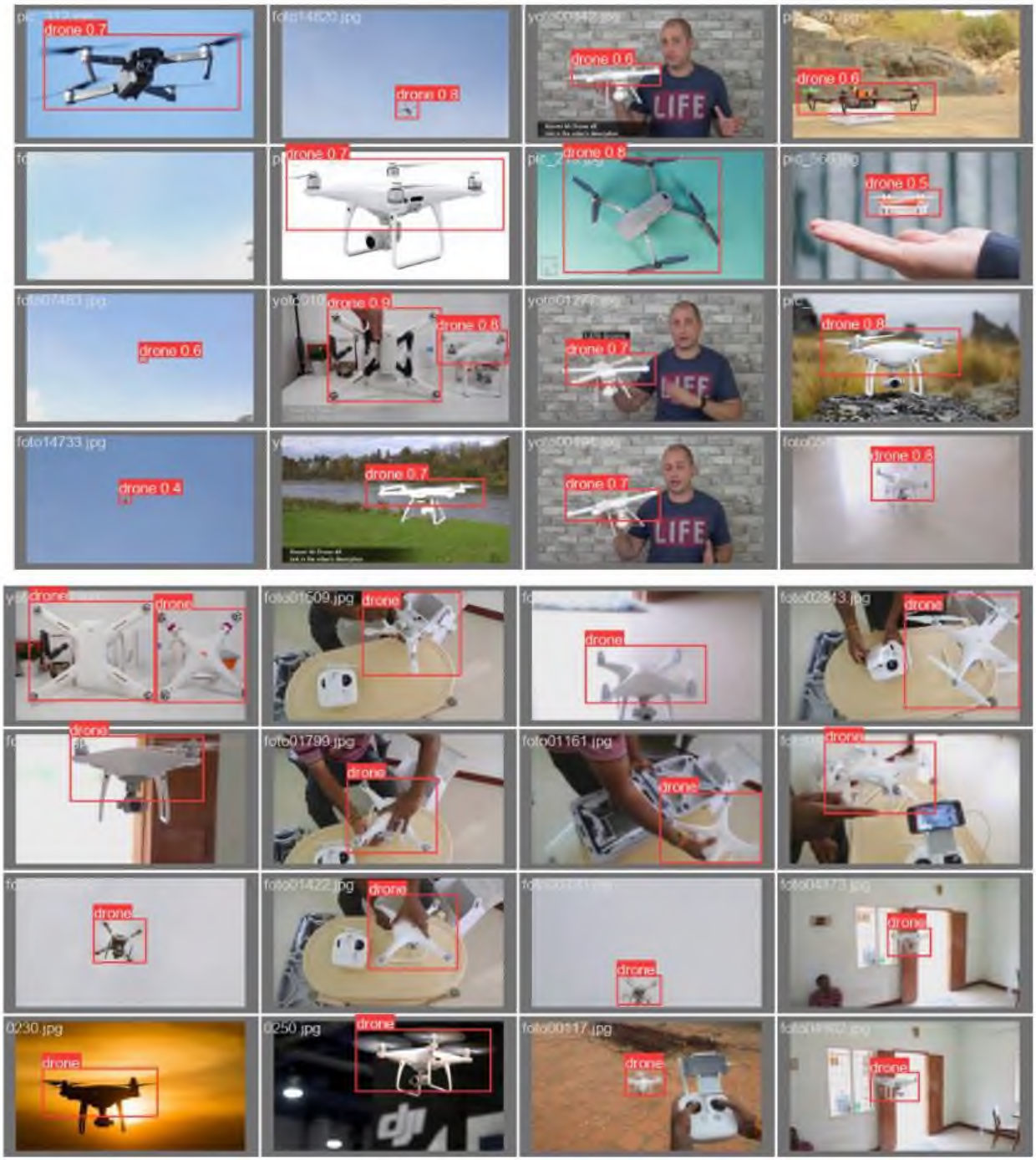
```
random_indices = random.sample(indices, 10)
num_cols = 2
num_rows = 5

plt.figure(figsize=(12 * num_cols, 6 * num_rows))

for i, idx in enumerate(random_indices):
    image = results[i].plot()
    plt.subplot(num_rows, num_cols, i+1)
    plt.imshow(image)
    plt.axis('off')
plt.tight_layout()
plt.show()

# %% [code] {"execution":{"iopub.status.busy":"2023-06-07T13:32:30.546488Z","iopub.execute_input":"2023-
06-07T13:32:30.546849Z","iopub.status.idle":"2023-06-
07T13:32:33.14088Z","shell.execute_reply.started":"2023-06-07T13:32:30.546813Z","shell.execute_reply":"2023-
06-07T13:32:33.139649Z"}}
success = model.export(format="onnx")  # export the model to ONNX format
success
```

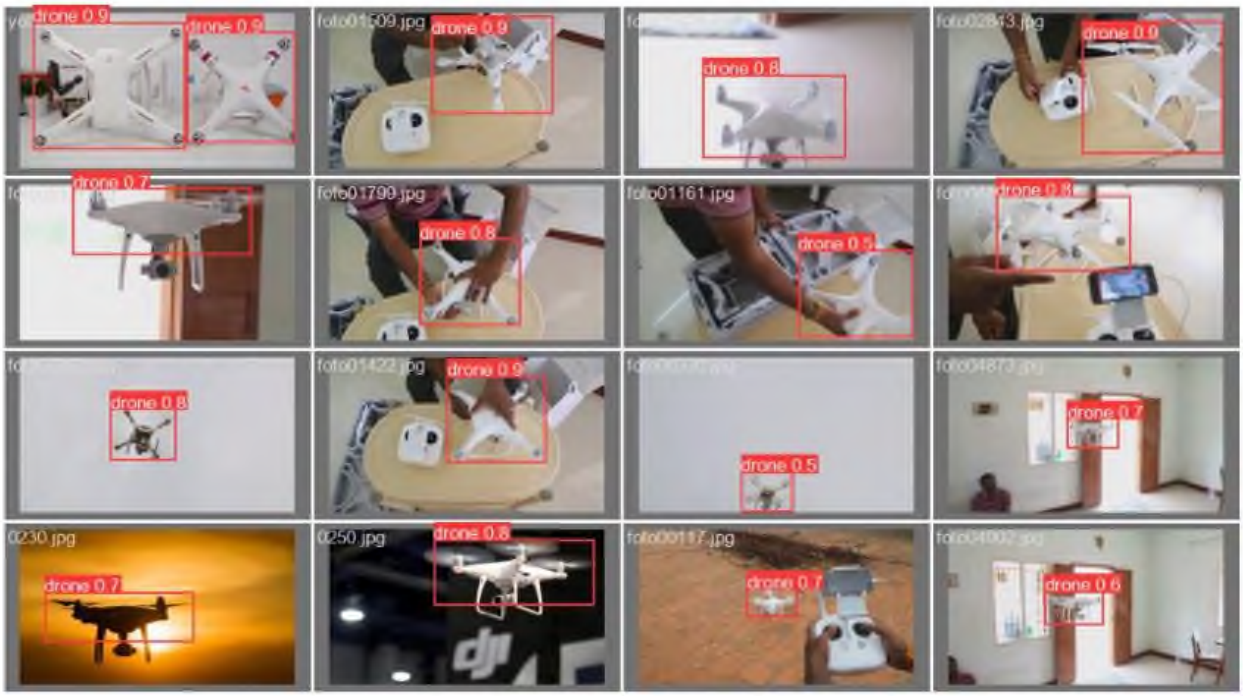
ДОДАТОК В

Результати тестування


0071.jpg
pic_403.jpg
drone
pic_1152.jpg
MASSIVE
EOFY SALE
pic_289.jpg
0262.jpg
0289.jpg
0287.jpg

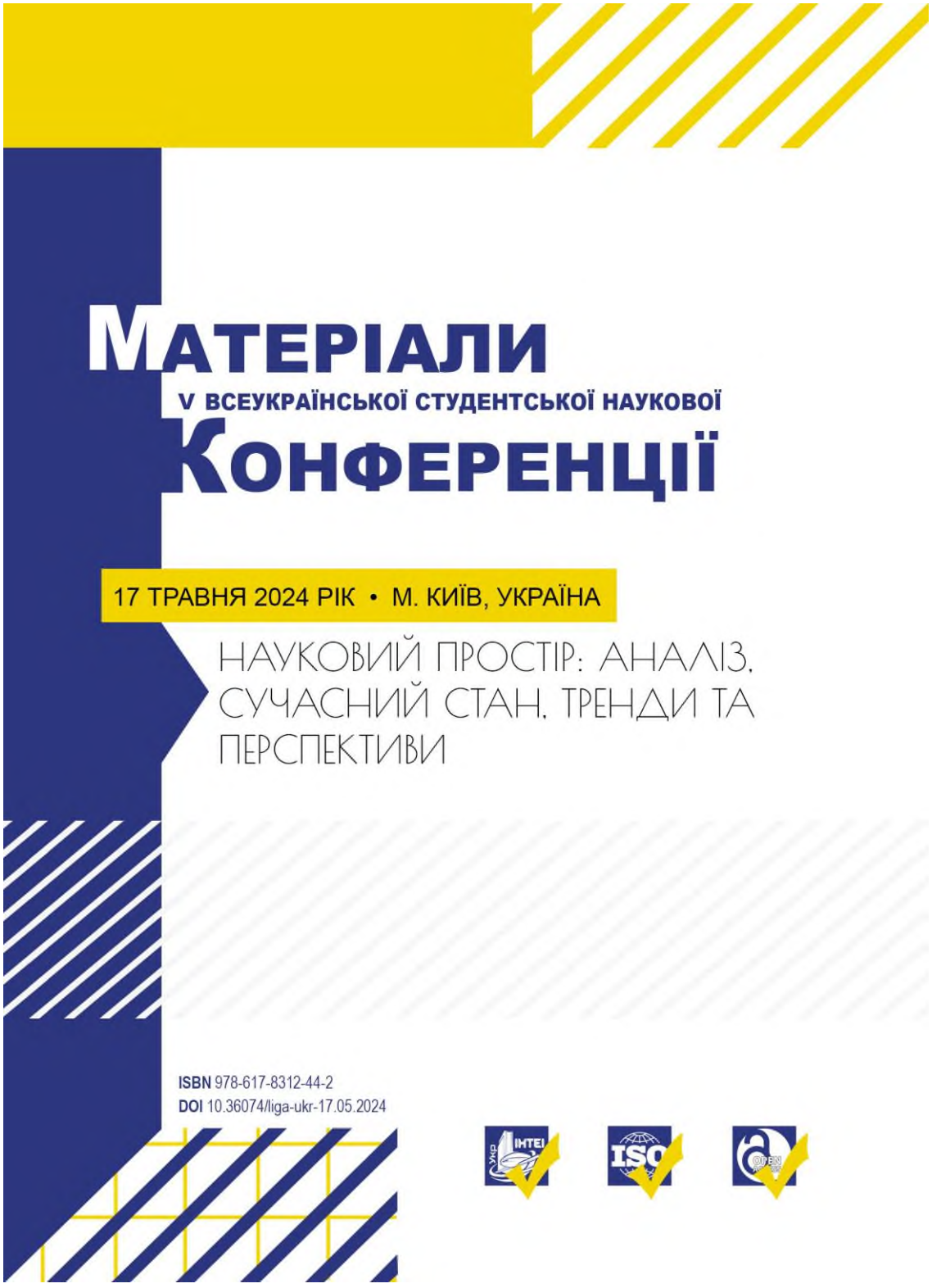

pic_403.jpg
drone 0.5
drone 0.8
pic_1152.jpg
drone 0.7
MASSIVE
EOFY SALE
drone 0.8
drone 0.7
0262.jpg
drone 0.6
En iyi Kameralı Drone Modelleri
drone 0.8
drone 0.7
drone 0.9
drone
foto14820.jpg
LIFE
foto14733.jpg


drone 0.7
drone 0.8
drone 0.6
drone 0.6
drone 0.7
drone 0.8
drone 0.5
drone 0.6
drone 0.9
drone 0.8
drone 0.7
drone 0.8
foto14733.jpg
drone 0.4
drone 0.7
drone 0.7
drone 0.8
drone
drone
drone
drone
foto01509.jpg
drone
drone
drone
drone
drone
foto01799.jpg
drone
foto01161.jpg
drone
drone
drone
foto01422.jpg
drone
drone
drone
drone
0230.jpg
drone
0250.jpg
drone
drone
drone


drone 0.9
drone 0.9
foto01509.jpg
drone 0.9
drone 0.8
foto02843.jpg
drone 0.9
drone 0.7
foto01799.jpg
drone 0.8
foto01161.jpg
drone 0.5
drone 0.8
drone 0.8
foto01422.jpg
drone 0.9
drone 0.5
foto04873.jpg
drone 0.7
0230.jpg
drone 0.7
0250.jpg
drone 0.8
foto00117.jpg
drone 0.7
drone 0.6

ДОДАТОК Г

АПРОБАЦІЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ



УДК 082:001
Н 34

Голова оргкомітету: Коренюк І.О.

Верстка: Зрада С.І.
Дизайн: Бондаренко І.В.

Рекомендовано до видання Вченою Радою Інституту науково-технічної інтеграції та співпраці. Протокол № 36 від 16.05.2024 року.

Конференцію зареєстровано Державною науковою установою «УкрІНТЕІ» в базі даних науково-технічних заходів України та інформаційному бюлетені «План проведення наукових, науково-технічних заходів в Україні» (Посвідчення №29 від 05.01.2024).

Н 34 **Науковий простір: аналіз, сучасний стан, тренди та перспективи**: матеріали V Всеукраїнської студентської наукової конференції, м. Київ, 17 травня, 2024 рік / ГО «Молодіжна наукова ліга». — Вінниця: ТОВ «УКРЛОГОС Груп», 2024. — 586 с.

ISBN 978-617-8312-44-2
DOI 10.62732/liga-ukr-17.05.2024

Викладено матеріали учасників V Всеукраїнської мультидисциплінарної студентської наукової конференції «Науковий простір: аналіз, сучасний стан, тренди та перспективи», яка відбулася 17 травня 2024 року у місті Київ, Україна.

УДК 082:001

ISBN 978-617-8312-44-2

СЕКЦІЯ 16.
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

Капкунов Андрій, здобувач вищої освіти факультету
комп'ютерних інформаційних технологій
Західноукраїнський національний університет, Україна

Науковий керівник: Майків І.М., канд. техн. наук, доцент, доцент
кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

АЛГОРИТМ ВИЯВЛЕННЯ ДРОНІВ З ВИКОРИСТАННЯМ YOLOV8

Алгоритм виявлення дронів з використанням моделі YOLOv8 можна описати наступними кроками, які викладені в систематичному стилі (рис.1):

Крок 1. Імпорт бібліотек: На початковому етапі здійснюється імпорт усіх необхідних бібліотек. Це можуть бути бібліотеки для обробки даних, машинного навчання та глибокого навчання, такі як TensorFlow, Keras, Pandas, NumPy тощо.

Крок 2. Завантаження та підготовка даних: Дані, які містять зображення дронів, завантажуються із зовнішніх джерел. Відбувається попередня обробка даних, яка може включати зміну розміру зображень, нормалізацію та конвертацію в подходящий формат для подальшої обробки.

Крок 3. Розподіл даних на навчальні та тестові набори: Для оцінки ефективності моделі, дані діляться на тренувальний та тестовий набори. Це дозволяє моделі вчитися на одному наборі даних і тестуватися на незалежному наборі, що підвищує об'єктивність оцінки її ефективності.

Крок 4. Побудова моделі YOLOv8: Створюється нейронна мережа за допомогою архітектури YOLOv8, яка є ефективною для задач реального часу виявлення об'єктів на зображеннях.

Крок 5. Компіляція моделі: Визначаються основні параметри моделі, такі як оптимізатор, функція втрат і метрики. Оптимізатор керує процесом оновлення ваг моделі в процесі навчання, а функція втрат вимірює різницю між передбаченнями і фактичними мітками.

Крок 6. Тренування моделі: Модель тренується на навчальних даних. Зазвичай цей процес включає багато ітерацій по всіх зразках даних для мінімізації функції втрат.

Крок 7. Оцінка моделі: Після тренування проводиться оцінка моделі на тестовому наборі даних. Це важливо для перевірки, наскільки добре модель узагальнює здобуті знання на нових даних.

Крок 8. Візуалізація результатів: Візуалізація результатів тренування та тестування моделі дозволяє зрозуміти її поведінку та точність.

Крок 9. Використання моделі для передбачень: Остаточним кроком є використання навченої моделі для виявлення дронів на нових зображеннях в реальному часі.

Кожен крок цього алгоритму спрямований на забезпечення високої точності та ефективності моделі в реальних умовах.

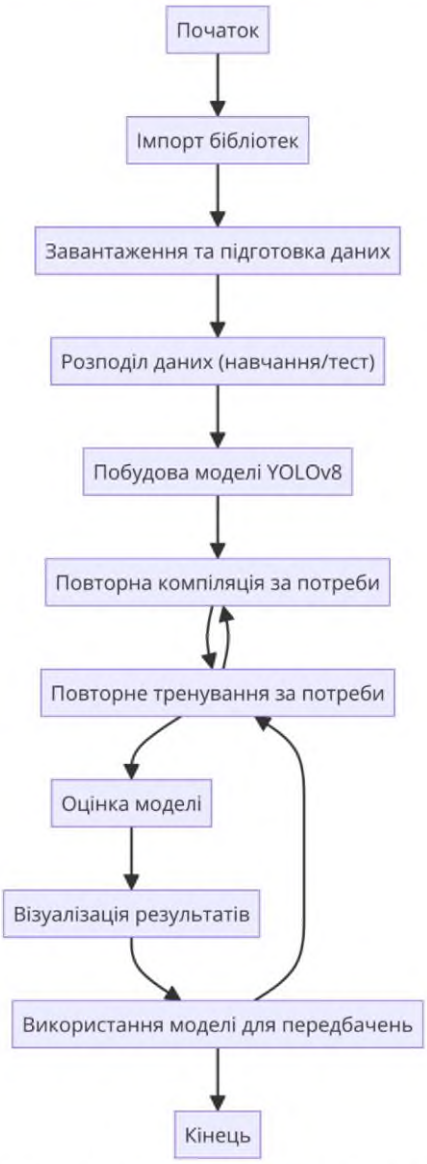


Рис. 1. **Алгоритм виявлення дронів з використанням YOLOv8**

Список використаних джерел:

1. Бичок М. О., Погудіна О. К. Оцінка використання шаблонів проектування програмного забезпечення. RADIOELECTRONIC AND COMPUTER SYSTEMS. 2021. № 1. С. 101–109. URL: https://doi.org/10.32620/reks.2021.1.09 (дата звернення: 13.05.2024).