Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження різних архітектур YOLO на ефективність виявлення обєктів за допомогою БПЛА
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-1

Паріяр Д.Ч
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Керівник проф. Білоус Н.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Паріяру Деннісу Четовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження різних архітектур YOLO на ефективність виявлення обєктів за допомогою БПЛА»
Затверджена наказом по університету від 29.03. 2024р. № 250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19.06.2024

3. Вихідні дані до роботи теоретичні відомості щодо моделей YOLO, їх метрики та точність, вибір кращої для роботи з БПЛА, розробка клієнт-серверного додатку для виявлення військової техніки та людських ресурсів на фото і відео віддалено, мова програмування Python ,бібліотеки Pytorch, ultralitycs, matlolib, LabelImg, Django, середовище розробки Pycharm.

4. Перелік питань, що потрібно опрацювати в роботі аналіз та порівняння існуючих нейронних мереж для розпізнавання об'єктів за допомогою БПЛА, вибір підходящої нейронної мережі YOLO, та вибір версії, тренування нейронної мережі, підготовка та розмічування власного набору даних, проведення експериментів та аналіз отриманих результатів, створення програмного додатку.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Видача завдання	06.01.2024	виконано
2	Аналіз предметної галузі	12.01.2024	виконано
3	Аналіз та вибір API для дослідження	04.02.2024	виконано
4	Аналіз та моделювання предметної області	15.02.2024	виконано
5	Планування експериментів	24.02.2024	виконано
6	Експериментальні дослідження	03.03.2024	виконано
7	Аналіз результатів експериментальних досліджень та розробка рекомендацій	17.03.2024	виконано
8	Написання та оформлення статті та тез доповіді	21.03.2024	виконано
9	Підготовка пояснювальної записки	12.04.2024	виконано
10	Підготовка презентації та доповіді	22.05.2024	виконано
11	Нормоконтроль	23.05.2024	виконано
12	Рецензування	24.05.2024	виконано
13	Занесення диплома в електронний архів	19.06.2024	виконано
14	Попередній захист	19.06.2024	виконано
15	Допуск до захисту у зав. кафедри	20.06.2024	виконано

Дата видачі завдання «6» січня 2024 р.

Студент _____ Паріяр Д.Ч.

(підпис)

Керівник роботи_____ проф. Білоус Н.В.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить ст.69, рис.44, 24 джерел.

БЕЗПІЛОТНІ ЛІТАЛЬНІ АПАРАТИ, ГЛИБОКЕ НАВЧАННЯ, YOLO, ВИЯВЛЕННЯ ОБ’ЄКТІВ, ВІЙСЬКОВА ТЕХНІКА, PYTHON.

Об'єкт дослідження - архітектури YOLO для виявлення об'єктів на зображеннях, отриманих від безпілотних літальних апаратів (БПЛА).

Мета роботи - проведення комплексного аналізу та порівняння різних архітектур YOLO для визначення їх ефективності у завданні виявлення об'єктів на зображеннях, отриманих від БПЛА.

Метод рішення - для досягнення поставленої мети використовується метод глибокого навчання, зокрема архітектури YOLO.

Результати дослідження - результати дослідження нададуть глибокий інсайт у ефективність різних архітектур YOLO у завданні виявлення об'єктів на зображеннях, отриманих від БПЛА.

UAVS, DEEP LEARNING, YOLO, OBJECT DETECTION, MILITARY VIHICLE, PYTHON.

The object of research is YOLO architectures for detection in images obtained from unmanned aerial vehicles (UAVs).

The purpose of the work is to carry out a comprehensive analysis and comparison of different YOLO architectures to determine their effectiveness in the task of detecting objects on images obtained from UAVs.

The solution method - to achieve the goal, the deep learning method, in particular the YOLO architecture.

The results of the study- are the results of the study will provide a deep insight into the effectiveness of different YOLO architectures in the task of detecting objects in images obtained from UAVs.

Я, *Паріяр Денніс Четович* , студент гр. *ІПЗм-22-1*, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота, що буде представлена для публічного захисту, виконана самостійно, в ній не містятьсяелементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмовидо допуску курсової роботи до захисту та застосування дисциплінарних заходів.

Робота виконана відповідно до вимог академічної доброчесності та пройшла перевірку на плагіат, який складає – 0.7 %.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

YOLO - You Only Look Once

SSD – Single Shot Multibox Detector

FPS – frames per second

mAP – mean average precision

R-CNN – Region-based Convolutional Neural Network

БПЛА – Безпілотні літальні апарати

ВСТУП

У сучасному світі безпілотні літальні апарати (БПЛА) стали ключовим елементом в різноманітних сферах, включаючи аерозйомку, дослідження, військове застосування та навіть розваги. Однак зростання використання БПЛА викликає серйозні виклики у сфері виявлення об'єктів на землі та в повітрі.

Однією з ключових проблем є потреба в точних та ефективних системах виявлення об'єктів для забезпечення безпеки, контролю та оптимального використання безпілотних апаратів. Задача виявлення об'єктів включає в себе розпізнавання різних об'єктів на зображеннях, отриманих від вбудованих камер БПЛА. Однією з основних труднощів є забезпечення високої точності виявлення при різних умовах освітлення, погодних умовах та різноманітних ландшафтах. Великий обсяг даних, отриманих від БПЛА, також створює виклики у сфері обробки та аналізу цих даних.

Постійний розвиток технологій глибокого навчання та алгоритмів обробки зображень відкриває нові можливості для оптимізації систем виявлення об'єктів на зображеннях, отриманих від БПЛА. Однак, разом з цим, постають нові виклики щодо ресурсоємності та інтеграції цих технологій у реальних умовах експлуатації.

Загальний огляд проблематики виявлення об'єктів з використанням БПЛА визначає актуальність і необхідність подальших досліджень та вдосконалення технологій для забезпечення ефективності та надійності в цьому важливому напрямку використання безпілотних літальних апаратів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної області

Розпізнавання зображень широко використовується у різних сферах, таких як безпека та відеоспостереження , транспортні технології та автоматизація руху транспортних засобів, машинобудування, медицина та реклама, сегментація зображень, спорт положення тіла людини у різних випадках. Проте, однією з проблем, які виникають у систем розпізнавання зображень, є необхідність оптимізації для забезпечення реального часу виконання .

Для досягнення розпізнавання зображень в реальному часі, потрібно використовувати оптимізаційні алгоритми обробки зображень. Це може включати швидкі алгоритмічні методи та оптимізацію обчислювальних операцій. Крім того, використання спеціалізованих апаратних рішень, таких як графічні процесори (GPU) або процесори для машинного навчання, може значно прискорити обробку зображень.

Ще одним ефективним підходом до оптимізації розпізнавання зображень в реальному часі є паралельна обробка даних. Це означає розподіл обчислювальних завдань між різними обчислювальними ресурсами, що сприяє прискоренню обробки зображень та надає миттєве розпізнавання.

Однак важливо знайти золоту середину між точністю розпізнавання та швидкістю обробки. У деяких сценаріях може бути необхідно зменшити складність алгоритмів або використовувати менш точні методи розпізнавання для досягнення реального часу, але це може впливати на точність результатів розпізнавання. Таким чином, розробники повинні виважено аналізувати вимоги та контекст застосування системи розпізнавання зображень і приймати відповідні рішення, здійснюючи компроміс між точністю та швидкістю.

Оптимізація процесу розпізнавання зображень у реальному часі знаходить застосування в різних сферах. В транспорті, розпізнавання зображень може

допомагати виявляти номерні знаки автомобілів для забезпечення безпеки та контролю. У сфері безпеки де швидке реагування на загрозу може бути вирішальним. У медицині та спорті це може використовуватися для автоматичного виявлення патологій на медичних зображеннях, або положення суглобів при виконанні вправ відносно осі тіла, що значно допомагає пацієнтам при відновленні ,або запобігає надмірному навантаженню на суглоби для професійних спортсменів .Таким чином ми бачимо, що в роботі було розроблено метод ідентифікації та порівняння поз і вправ, які виконує людина, та матимуть низьку чутливість до помилок[1].

Розпізнавання зображень у реальному часі продовжує розвиватися, і нові підходи та технології з'являються для підвищення ефективності та швидкості цього процесу. Одним із таких напрямків є використання глибокого навчання та нейромереж.

За допомогою навчання нейронних мереж ми можемо бачити за допомогою компьютерного зору не тільки положення голови людини, а навіть розпізнавати емоції на обличчі , це дає нам змогу більш точно та швидко діагностувати захворювання, або зрозуміти емоції людини в режими реального часу . Також слід зазначити що в роботі [2], експериментальним шляхом було доведено що спеціалізований набір даних на якому натренована нейронна мережа має менший відсоток хибних спрацювань, навіть при умові меншої кількості зображень, саме тому в моїй роботі використовується кастомний датасет. Ми можемо зробити висновок на основі статті [3], що розпізнавання емоцій є потужною та дуже корисною технікою для оцінки емоційного стану людини та прогнозування її поведінки, що може бути використано у сфері маркетингу чи реклами, крім того оцінка емоцій корисні в розробці систем взаємодії людини і машини.

Глибоке навчання, шляхом тренування глибоких нейронних мереж на обширних наборах даних, забезпечує високу точність розпізнавання зображень. Однак існючі стандартні архітектури глибоких мереж, такі як згорткові нейронні мережі (CNN), можуть виявитися обчислювально витратними, обмежуючи їх швидкість розпізнавання. Тому проводяться дослідження у напрямку оптимізації

та стиснення глибоких моделей, використовуючи методи, такі як прунінг, квантизація та компресія моделей.

Виникають нові архітектури нейронних мереж, спеціально розроблені для розпізнавання зображень у реальному часі. Наприклад, YOLO, MobileNet та EfficientNet є прикладами таких архітектур, які досягають високої точності при низькому обчислювальному завантаженні. Вони використовують легковагові фільтри та оптимізовану структуру мережі, що дозволяє ефективно виконувати операції розпізнавання на ресурсно обмежених пристроях, таких як мобільні телефони або вбудовані системи.

Розподілені обчислення та використання хмарних платформ також сприяють оптимізації розпізнавання зображень у реальному часі. Обчислення можуть бути розподілені між кількома обчислювальними вузлами або виконуватися на потужних серверах у хмарному середовищі. Це дозволяє використовувати більші обчислювальні ресурси та прискорювати обробку зображень.

Хмарні платформи надають можливість користувачам отримувати доступ до високопродуктивних обчислювальних ресурсів, графічних процесорів і спеціалізованих пристроїв для машинного навчання. Це дозволяє здійснювати обробку зображень в реальному часі навіть на пристроях з обмеженими обчислювальними можливостями, використовуючи віддалені сервери для виконання обчислень.

Додатково, оптимізація розпізнавання зображень в реальному масштабі часу може включати використання спеціалізованих апаратних пристроїв, таких як Tensor Processing Units (TPU) або Field-Programmable Gate Arrays (FPGA). Ці пристрої прискорюють обчислення зображень, виконуючи паралельні обчислення та оптимізуючи апаратну архітектуру спеціально для розпізнавання зображень.

Використання оптимізації розпізнавання зображень в реальному часі розповсюджується на різні галузі. В медицині це може використовуватися для автоматичного виявлення ознак раку на медичних зображеннях або аналізу рентгенівських знімків. У транспорті розпізнавання зображень може

застосовуватися для систем автоматичного виявлення номерних знаків або виявлення воді.

Оптимізація процесу розпізнавання зображень у реальному часі вимагає використання передових технологій. Розробники повинні застосовувати сучасні методи машинного навчання, штучного інтелекту і новітні апаратні рішення. Постійне оновлення технологій виявляється ключовим, оскільки це дозволяє досягати найкращих результатів і використовувати потенціал нових інновацій.

Узагальнено, оптимізація розпізнавання зображень в реальному часі є надзвичайно важливим завданням, оскільки вона забезпечує ефективність та швидкість розпізнавання в різних галузях. Ця технологія знаходить застосування там, де потрібне швидке та точне визначення об'єктів. Завдяки своїй потужності та потенціалу, оптимізація розпізнавання зображень буде продовжувати розвиватися, змінюючи наш спосіб сприйняття світу та розширюючи можливості наших технологій. Дослідження та розвиток в цій області активно просуваються за допомогою оптимізаційних алгоритмів, спеціалізованих апаратних рішень і передових технологій [6].

1.2 Аналіз методів для визначення об'єктів на зображеннях в реальному часі.

Була проведена аналіз та ретельний розгляд проблеми розпізнавання зображень у реальному часі. Під час оцінки різних підходів до цього завдання виявлено кілька основних методів , які застосовуються для досягнення цієї мети. Аналіз дозволив відокремити найбільш поширені та результативні стратегії, спрямовані на ефективне та швидке розпізнавання об'єктів у режимі реального часу. Глибоке вивчення цих підходів допомагає зрозуміти основні принципи їх функціонування та визначити найбільш відповідні та перспективні методи для подальших досліджень у рамках даної роботи.

1.2.1 Глибокі нейронні мережі

У контексті розпізнавання зображень у реальному часі глибокі нейронні мережі виявляються надзвичайно потужними інструментами. Основними видами

глибоких нейронних мереж, які використовуються в цій області, є конволюційні нейронні мережі (Convolutional Neural Networks - CNN), рекурентні нейронні мережі (Recurrent Neural Networks - RNN), та мережі довготривалої короткочасної пам'яті (Long Short-Term Memory - LSTM) [7].

Конволюційні нейронні мережі виявляються особливо ефективними для розпізнавання об'єктів на зображеннях. Їх ефективність полягає в застосуванні операцій згортки, які автоматично виявляють та виділяють локальні залежності та важливі ознаки у зображенні. Цей процес дозволяє CNN самостійно визначати різноманітні аспекти, такі як контури, текстури та більш складні структури об'єктів, що допомагає впевненіше розпізнавати їх.

Таким чином, глибокі нейронні мережі, зокрема конволюційні нейронні мережі, відіграють важливу роль у вдосконаленні систем розпізнавання зображень у реальному часі, забезпечуючи високий рівень автоматизації та точності у виявленні об'єктів.

Рекурентні нейронні мережі та мережі довготривалої короткочасної пам'яті дозволяють моделювати залежності в часових послідовностях, що є особливо корисним для розпізнавання об'єктів у відеопотоці, де зображення представлені у вигляді послідовних кадрів. Рекурентні нейронні мережі використовують попередні вихідні дані для передачі контексту між кадрами, а мережі LSTM забезпечують збереження та використання довготривалих залежностей, що дозволяє більш ефективно відстежувати та розпізнавати об'єкти в часових послідовностях [8].

Використання глибоких нейронних мереж для розпізнавання зображень у реальному часі включає кілька етапів. Спочатку глибока нейронна мережа піддається навчанню на обширному наборі зображень, де кожен зразок має відповідну мітку класу. Процес навчання включає в себе згорткові шари для автоматичного виявлення ознак на різних рівнях абстракції, пулінгові шари для зменшення розмірності зображення та повнозв'язані шари для класифікації об'єктів (див. рис. 1.1).

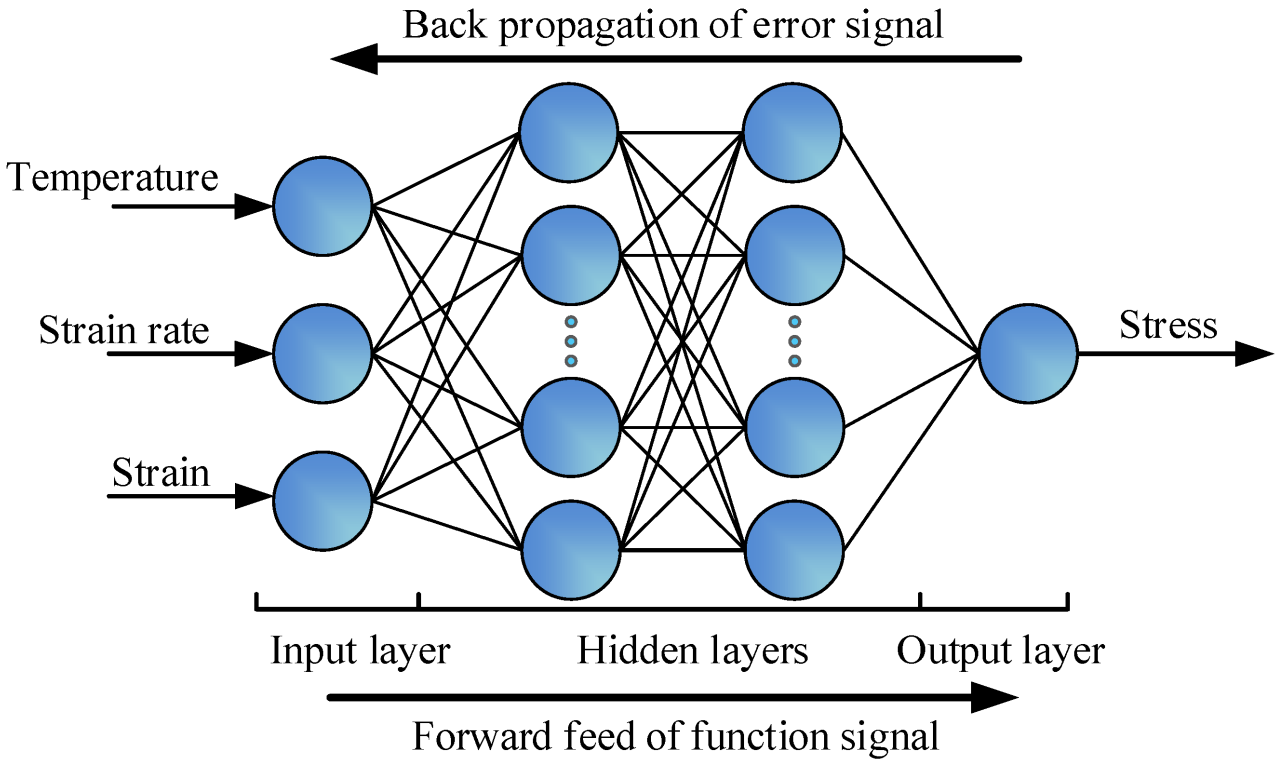

Рисунок 1.1 – Схема роботи глибоких нейронних мереж (за даними [9])

Після завершення навчання глибока нейронна мережа стає здатною до застосування для розпізнавання об'єктів у реальному часі. Зображення можуть надходити з відеопотоку чи інших джерел та подаються на обробку через мережу. Цей процес включає проходження зображення через мережу, аналіз його ознак та класифікацію об'єктів на основі навчених моделей.

Глибокі нейронні мережі відзначаються високою швидкістю та точністю у розпізнаванні зображень у реальному часі. Вони автоматично виявляють складні ознаки та структури об'єктів, що сприяє впевненій класифікації у реальному часі. Використання рекурентних нейронних мереж та мереж довготривалої короткочасної пам'яті дозволяє зберігати та використовувати контекстуальну інформацію, що покращує процес розпізнавання об'єктів у відеопотоці.

1.2.2 Використання попередньо навчених моделей

Використання попередньо навчених моделей стало ключовим підходом у виявленні об'єктів на зображеннях в реальному часі, забезпечуючи ефективність та

точність процесу. Цей метод використовує заздалегідь навчені нейронні мережі, які вже засвоїли властивості та характеристики різних класів об'єктів.

Основні характеристики використання попередньо навчених моделей:

Типи Моделей:

- загальні моделі деякі попередньо навчені моделі розроблені для різноманітних завдань та можуть включати в себе виявлення облич, класифікацію об'єктів, або сегментацію зображень;
- спеціалізовані моделі існують моделі, які спеціалізуються на конкретних типах об'єктів або сценаріях, таких як виявлення транспортних засобів, рослин чи людей;
- файн-тюнінг: деякі розробники використовують процес файн-тюнінгу, де попередньо навчену модель доопрацьовують або адаптовують до конкретного завдання, підвищуючи її точність у конкретному контексті.

Використання попередньо навчених моделей дозволяє значно зменшити час, потрібний для навчання, оскільки модель вже опанувала загальні характеристики зображень.Завдяки оптимізації та готовності попередньо навчених моделей, вони ідеально підходять для використання в реальному часі, де потрібно швидке та точне виявлення об'єктів. Цей метод може бути використаний в різних галузях, включаючи медицину, безпеку, транспорт, та інші, де потрібно надійне та оперативне виявлення об'єктів. Використання попередньо навчених моделей може стикатися з викликами адаптації до конкретних умов або класів об'єктів, тому деякі моделі можуть вимагати додаткового налаштування.

Використання попередньо навчених моделей у виявленні об'єктів на зображеннях в реальному часі відкриває широкі можливості для розробників, спрощуючи процес та забезпечуючи високу точність у різноманітних сценаріях використання.

2 ПОСТАВКА ЗАДАЧІ

Мета даного дослідження полягає в аналізі та порівнянні різних архітектур YOLO (You Only Look Once) з метою покращення ефективності виявлення технічних об’єктів за допомогою безпілотних літальних апаратів, а також розробці веб додатку з використанням найкращої моделі. Дослідження спрямоване на визначення оптимальних архітектур YOLO для застосування в умовах виявлення військової техніки та людськи ресурсів ворога з повітряного простору.

Ключові етапи дослідження включають:

- аналіз архітектур нейронних мереж: провести докладний аналіз різних архітектур нейронних мереж RetinaNet, SSD, YOLO з визначенням їхніх характеристик, переваг та обмежень у контексті виявлення об'єктів за допомогою БПЛА;
- оцінка продуктивності: порівняти продуктивність різних архітектур нейронних мереж в умовах застосування БПЛА. Врахувати час реакції, точність та швидкодію алгоритмів виявлення об'єктів;
- вибір версії YOLO: : порівняти продуктивність різних архітектур YOLO в умовах застосування БПЛА. Врахувати час реакції, точність та швидкодію алгоритмів виявлення об'єктів. Підібрати версії обраної архітектури YOLO для використання з БПЛА;
- вибір набору даних для тренування YOLO: так як вибір набору даних є критично важливим для створення точної та надійної моделі , навчання нейронної мережі важливо врахувати декілька факторів ,специфіка об’єктів ,які необхідно виявляти, умови зйомки, кути нахилу та різноманітність освітлення;
- аналіз проведених есперименів на навчених моделях: проведено аналіз 4 версій YOLO починаючи з yolov5 – yolov8, виходячи з метриків навчання та тестування. Провести серію експериментів для валідації оптимізованих архітектур YOLO на реальних даних, отриманих від БПЛА. Оцінити їх продуктивність у варіативних умовах зйомки та вимірювань;

- створення власного набору даних: для навчання будь якої нейронної мережі необхідно використовувати набор даних, так як ключові об'єкти пошуку в нас військова техніка та люди, створений та розмічений власний додатковий набор даних за допомогою бібліотеки LabelImg на основі власних та зображень з мережі інтернет. Використаний метод файн-тюнинг донавчання моделі на власному датасеті для отримання більш точної та стабільної роботи моделі;
- розробка програмного забезпечення: розробка влючає в себе клієнт серверний додаток, взаэмодія здійснюється за допомогою методу "post" отримання зображення від користувача, зберігає його в таблиці БД, зчитує дане зображення з БД та надсилає його до моделі YOLO, зберігає результат детекції та надсилає клієнту на веб сторінку додатку. Завершення дослідження передбачає розробку комплексного підходу до вдосконалення систем, та створенням веб додатку для виявлення об'єктів у реальному часі за допомогою безпілотних літальних апаратів. Отримані результати дослідження не лише покладають основи для вибору оптимальних архітектур YOLO, але і розширюють можливості використання цих систем у практичних завданнях, таких як моніторинг, рятувальні операції та інші сфери застосування.

Підсумувати результати дослідження, надати висновки щодо ефективності різних архітектур YOLO у виявленні об'єктів за допомогою БПЛА. Запропонувати рекомендації для використання найбільш оптимальних архітектур у конкретних сценаріях.

Додатково, розглядаючи визначені аспекти ефективності та точності, наша робота спрямована на створення оптимізованих методів виявлення об'єктів, які враховують особливості безпілотних систем. Отримані оптимізації сприятимуть швидшому та надійнішому виявленню об'єктів у реальному часі, покращуючи при цьому загальну продуктивність та ефективність використання безпілотних літальних апаратів в завданнях розпізнавання об'єктів.

3 МЕТОДИ ВИРІШЕННЯ ПРОБЛЕМИ

3.1 Критерії вибору моделі за допомогою метрик

Існує кілька критеріїв, які можна використовувати для вибору моделі. Основними серед них є такі показники як FPS і точність, які вказують на ефективність моделі. Щодо визначення інших характеристик, які відображають вимоги до обчислювальної системи, використовуються параметри, такі як кількість операцій з плаваючою комою в секунду (FLOPs) і кількість параметрів моделі.

Кількість параметрів у моделі вказує на обсяг інформації, який необхідно опрацювати під час навчання моделі. Ці параметри можуть включати ваги і зсуви в нейронних мережах. Кількість параметрів слугує показником складності моделі і може бути використана для порівняння обсягу пам'яті, необхідного для зберігання моделі. Чим більше параметрів, тим більше пам'яті потрібно для зберігання та навчання моделі.

FLOPs вимірює обчислювальну складність моделі, вказуючи на кількість операцій з плаваючою точкою, які модель виконує за секунду. FLOPs служить показником обчислювальної складності моделі і може бути використаний для порівняння ефективності різних моделей. Чим більше значення FLOPs, тим більше ресурсів потрібно для виконання моделі.

Значення кадрів на секунду (FPS) показує, скільки кадрів обробляє модель на визначення об'єктів у них. У випадку визначення автомобілів, важливо стежити за цим параметром. Для стабільної роботи системи оптимальне значення FPS зазвичай коливається між 15 і 30, що забезпечує достатню плавність відео. Хоча більш високе значення було б привітним, для цього потрібна оптимізація моделі та використання потужнішого обчислювального пристрою. Для перевірки точності розпізнавання моделі використовується середня точність моделі (mAP). mAP представляє середнє значення AP для всіх класів, які розпізнає модель. AP - це середня точність розпізнавання класу в моделі. Формула для розрахунку mAP виглядає так [9]:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i, \quad (3.1)$$

де N – загальна кількість класів, які розпізнаються моделлю;

AP_i – показник AP для класу i.

Для розрахунку AP потрібні попередньо визначені метрики:

– iou;

– точність (precision);

– повнота (recall);

– pr-крива (precision-recall curve).

IoU - це метрика, яка визначає відношення перетину областей, які є істинними та передбаченими, до їх об'єднання (див. рис. 3.1).

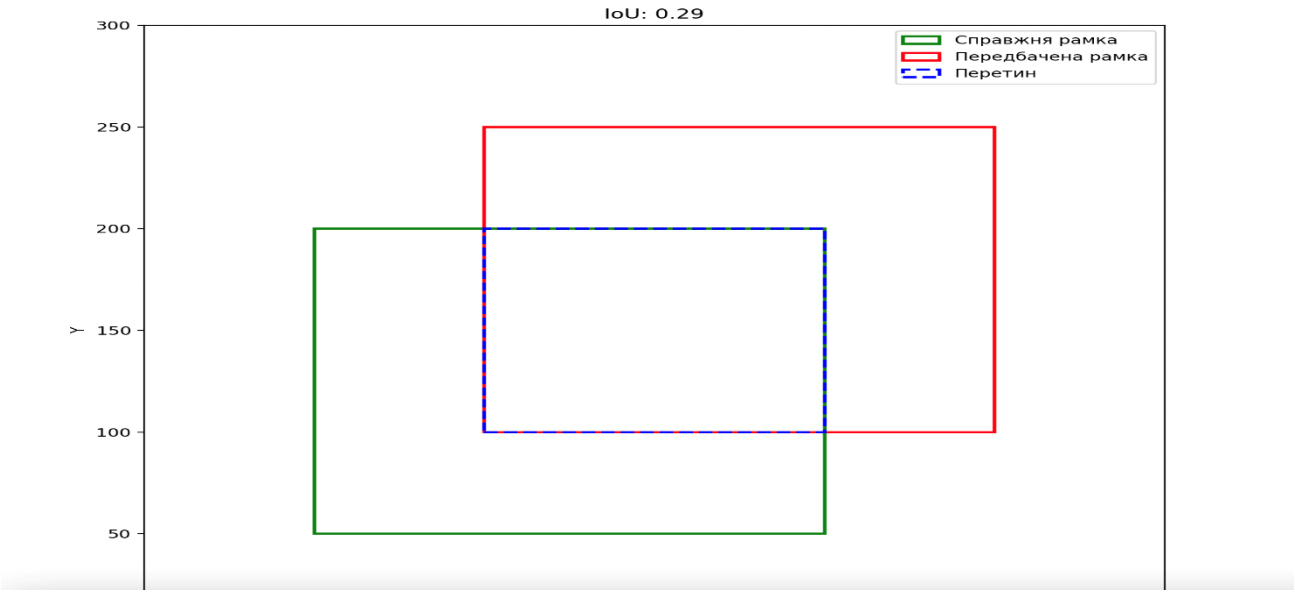

Рисунок 3.1 – Визначення метрики IoU[9]

Параметр IoU приймає значення α від 0 до 1. При α = 0 немає розпізнавання, а при α = 1 прогнозована область істинно збігається. Для розрахунку інших метрик зазвичай використовується поріг α = 0.5. Якщо α вище порогу точності, це вважається правильним позитивним результатом (TP). Якщо α нижче порогу або прогноз невірний, це буде хибним позитивним (FP). Якщо немає жодного прогнозу для об'єкта, це буде хибним негативним (FN). Значення правильно

негативного (TN) не використовується, оскільки ми не цікавимося точністю визначення відсутності об'єкта .

TP, FP, та FN використовуються для визначення таких метрик, як точність та повнота. Точність обчислюється як відношення правильно визначених позитивних прикладів до загальної кількості об’єктів, визначених як позитивні (TP/(TP+FP)). Повнота вимірюється як відношення правильно визначених позитивних прикладів до загальної кількості позитивних прикладів (TP/(TP+FN)). З точки зору точності можна зробити висновок про те, наскільки точними є прогнози щодо виявлених об’єктів, тоді як повнота вказує на те, наскільки ефективно було виявлено всі позитивні випадки.

Точність та повнота використовуються для побудови кривої точності-повноти (PR-кривої), яка демонструє, наскільки ефективно модель може виявити позитивні класи, зберігаючи при цьому високий рівень точності. Для обчислення площі під PR-кривою (AP) використовується метод, який застосовується у метриці PASCAL VOC 2010. Існують різні підходи до цього, що включають у себе різні пороги та методи обчислення AP. Ці підходи також мають свої власні набори даних для перевірки якості моделі, часто включаючи базові класи, такі як тварини, автомобілі та люди. Розглянемо набори даних, які використовувалися та використовуються для перевірки якості роботи моделі:

- PASCAL VOC 2005: цей метод використовував криву ROC замість прямої PR;
- PASCAL VOC 2007: ця методика полягає у використанні одинадцяти точкової інтерполяції для зменшення впливу змінливості PR-кривої на результат.

Повнота поділяється на 11 рівнів. Для кожного рівня повноти обчислюється максимальна точність серед усіх порогових значень, які перевищують або дорівнюють поточному рівню повноти (див. рис. 3.3). Тоді AP обчислюється як середнє значення одинадцяти інтерпольованих значень [10].

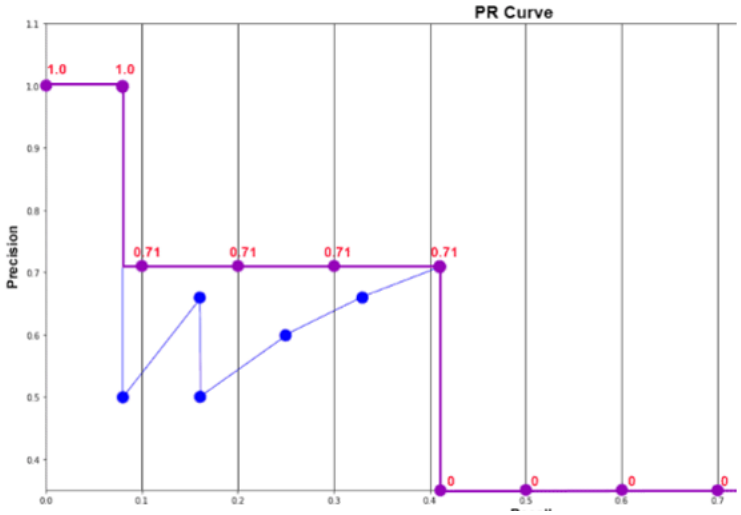

Рисунок 3.3 – Метод точкової інтерполяції [9]

В даний час MS COCO mAP є однією з найпоширеніших метрик для оцінки ефективності моделей. Цей підхід використовує 101-бальну інтерполяційну версію AP. Натомість оцінки при одному пороговому значенні для IoU замінено на 10 порогових значень, які охоплюють діапазон від 0.5 до 0.95 з кроком 0.05. Середнє значення mAP з отриманих значень AP використовується для порівняння моделей [9].

3.2 Можливості у моделях визначення

В залежності від завдання розпізнавання існують різні підходи та алгоритми для виконання кожного з них. Основні кроки виявлення об'єктів на зображенні або відеокадрі включають наступне:

– виявлення наявних об'єктів на зображенні;
– класифікація кожного виявленого об'єкта та оцінка його розміру за допомогою обмежувальної рамки.

Залежно від вимог програми, ці кроки можуть виконуватися одночасно на одному етапі або в окремі два етапи. Таким чином, алгоритми поділяються на одноетапні та двоетапні [4].

У двоетапних моделях спочатку визначаються області інтересу (RoI) на зображенні за допомогою методів комп'ютерного зору або глибинних мереж. Після

цього, наступний прохід використовує ці області для виділення ознак, які служать для класифікації об'єктів та поліпшення визначених областей. Порівняно з одноетапними моделями, двоетапні досягають більшої точності завдяки кращій локалізації об'єктів на зображенні, що також полегшує розпізнавання малих об'єктів або груп. Однак через більшу обчислювальну складність швидкість обробки кадрів на секунду (FPS) у таких моделях менша, ніж у одноетапних. Програми, які використовують двоетапні моделі, базуються на обласних згорткових нейронних мережах (R-CNN) та їх вдосконалених версіях (Fast-CNN, Faster-CNN, G-RCNN) [5].

У відміну від моделей з двома етапами, одноетапні моделі безпосередньо встановлюють рамки обмеження об'єкта на зображенні, не пропонуючи перед цим області на зображенні. Цей метод надає перевагу у швидкості виявлення та класифікації, оскільки він потребує менше часу для виконання меншої кількості операцій. Через це одноетапні моделі є оптимальним вибором для застосунків, які працюють в реальному часі. Проте вони менше справляються з локалізацією об'єктів порівняно з двоетапними моделями, і не так ефективно впораються з розпізнаванням об'єктів неправильної форми чи групи невеликих об'єктів. До найпопулярніших моделей, які використовуються для розпізнавання в реальному часі [11], входять:

- RetinaNet;
- SSD (Single Shot Detector);
- YOLO (You only look once).

3.3 Вибір нейронної мережі для реалізації розпізнавання в реальному часі

Так як основною метою цього дослідження є реалізація розпізнавання в реальному часі, ми порівняємо різні алгоритми одноетапного визначення, оскільки вони є швидшими за двоетапні моделі. Ми проведемо аналіз результатів одноетапних моделей.

RetinaNet, одна з провідних моделей для виявлення об'єктів унікальна своєю одноетапною структурою. Вона спроектована для оптимізації та балансування

роботи детекторів, щоб досягти більш ефективних результатів. Ця архітектура базується на заміні перехресних ентропійних втрат, характерних для попередніх моделей, на фокусні втрати. Крім цього, вона використовує мережі ResNet та Feature Pyramid Network (FPN).

FPN має структуру з верху донизу і горизонтальні зв'язки, які додаються до кожного рівня. Це дозволяє поєднувати інформативні деталі з низькою роздільною здатністю та менш інформативні деталі з високою роздільною здатністю. Кожен рівень FPN має свою повністю згорткову мережу, що включає дві підмережі: одну для класифікації та іншу для регресії. Класифікаційна мережа використовується для багатокласового прогнозування, тоді як регресійна мережа служить для передбачення відповідних обмежувальних рамок для класифікованих об'єктів.

Основною інновацією у RetinaNet є використання фокусних втрат (Focal Loss) у складі підмережі для класифікації. Оскільки знаходження об'єктів у одноетапних алгоритмах часто стикається з нерівномірністю кількості позитивних та негативних прикладів, це може впливати на процес навчання моделі. Фокусні втрати надають різну вагу складним прикладам, що ефективно розв'язує проблему дисбалансу класів у моделі виявлення [12-13].

Для прискорення обробки зображень модель SSD відмовляється від використання мережі пропозицій областей, замість цього використовуючи багатомасштабні ознаки та області за замовчуванням. Роботу SSD можна розділити на три основні етапи. Перший етап полягає у виділенні важливих карт ознак під час витягування ознак, використовуючи лише повністю згорткові шари. Після виділення ознак наступним етапом є ідентифікація та локалізація об'єктів, використовуючи також лише повністю згорткову нейромережу. Основною метою другого етапу є створення найбільш відповідних карт меж для всіх карт ознак, що надають інформацію про об'єкт у вигляді обмежувальних рамок. Після пройдених двох етапів третій відповідає за фільтрацію зайвих рамок. Остаточні результати виявлення генеруються за допомогою алгоритму немаксимального пригнічення (NMS), який виявляє дублікати рамок та зберігає тільки найбільш впевнені

прогнози. Використання NMS дозволяє досягти як швидкості, так і точності виявлення [12-13].

Архітектура YOLO ґрунтується на трьох основних методах, які дозволяють ефективно визначати об'єкти та роблять її однією з найбільш поширених моделей серед програмного забезпечення. Перший з цих методів - використання залишкових блоків. Кожна комірка в цій мережі виступає як центральна точка, для якої робиться окремий прогноз.

Другий метод включає врахування кожної з центральних точок при створенні обмежувальних рамок для конкретного прогнозу. Хоча класифікація відбувається ефективно для кожної комірки мережі, визначення обмежувальних рамок для кожного прогнозу є більш складним завданням.

Останнім методом є використання метрики IoU, яка допомагає обчислити найкращі обмежувальні рамки для виконання задачі виявлення об'єктів [13]. З результатів досліджень, представлених на рисунках 3.4 та 3.5, проведених для виявлення фармацевтичних препаратів та на основі датасету COCO, можна зробити висновок, що кожна з моделей може бути спеціалізована в певній сфері та у певних умовах.

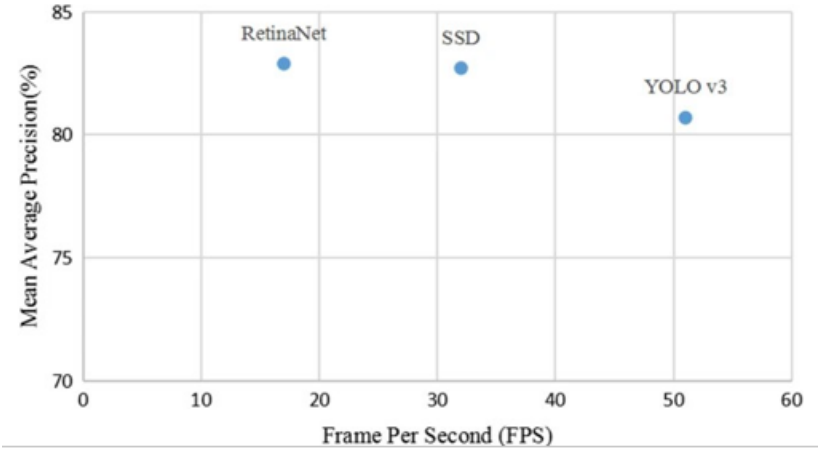

Рисунок 3.4 – Порівняння одноетапних моделей за FPS та mAP [12]

RetinaNet забезпечує високу точність з вимогою до більшого часу для обробки кожного зображення. Це особливо корисно, якщо ми не дуже залежимо від

високої швидкості кадрів на секунду, але важливо забезпечити високу точність у виявленні об'єктів.

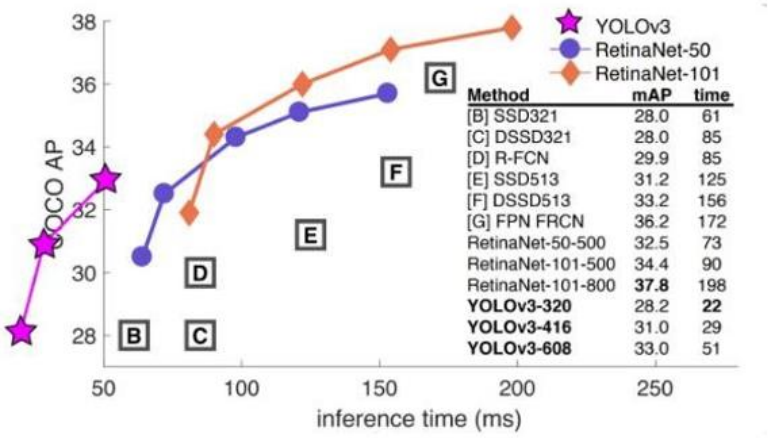

Рисунок 3.5 – Порівняння моделей за часом опрацювання та mAP [14]

SSD лежить посередині між YOLO та RetinaNet щодо якості та швидкості. Для створення програми розпізнавання в реальному часі, яка працюватиме на простій апаратурі чи мобільних пристроях, був розроблений MobileNetSSD. Ця модель SSD використовує архітектуру MobileNet як базу, що дозволяє забезпечити високу швидкість кадрів на простій апаратурі, але знижує рівень точності в порівнянні зі звичайним SSD чи іншими моделями.

Модель YOLO відзначається найвищими показниками швидкості, хоча за точністю вона поступається іншим моделям. Це досягається за рахунок модифікації архітектури для підвищення швидкості обробки зображень. Незважаючи на те, що модель YOLO є відносно новою, завдяки широкому використанню та активній спільноті розробників, нові версії моделей з різними змінами випускаються досить часто. YOLO пропонує широкий вибір моделей для кожної версії, що дозволяє використовувати цю архітектуру на різних типах обчислювальних пристроїв, вибираючи між найвищою швидкістю або точністю в залежності від потреби.

Після аналізу різних одноетапних моделей було вирішено використовувати модель YOLO для розробки програмного забезпечення розпізнавання в режимі реального часу. Її висока швидкість обробки дозволяє працювати в реальному часі, а можливість вибору моделі в залежності від потужності обчислювального

пристрою, а також простота налаштування та тренування, завдяки великому співтовариству розробників, зробили YOLO оптимальним вибором.

4 РЕАЛІЗАЦІЯ НАВЧАННЯ НЕРОННОЇ МЕРЕЖІ YOLO

4.1 Поняття архітектури Yolo

YOLO (You Only Look Once) - це інноваційна архітектура глибоких нейронних мереж, призначена для виявлення та класифікації об'єктів на зображеннях. Відмінністю YOLO є те, що вона здатна здійснювати виявлення об'єктів у реальному часі, а це досягається завдяки одноразовому погляду на зображення, а не розділенню його на різні регіони чи області.

Основні характеристики YOLO включають глобальний підхід до виявлення, використання гридової структури для поділу зображення, інтеграцію інформації з різних частин зображення та високу ефективність у реальному часі.

4.1.1 Вибір моделі YOLO

Сімейство моделей YOLO складається з численних варіантів, які були розроблені в різні періоди часу (див. рис. 4.1). Оскільки YOLO має відкритий вихідний код, різні розробники вносили свої вдосконалення до алгоритмів і використовували різні підходи для створення моделей.

Рисунок 4.1 – Історія розвитку YOLO [10]

Кожна ітерація мала свої варіанти моделей з різними розмірами, які можна було використовувати залежно від характеристик обчислювальної техніки. Щоб обрати найвідповіднішу модель, розглянемо порівняння потужності та швидкості останніх ітерацій YOLO, представлене на рисунку 4.2.

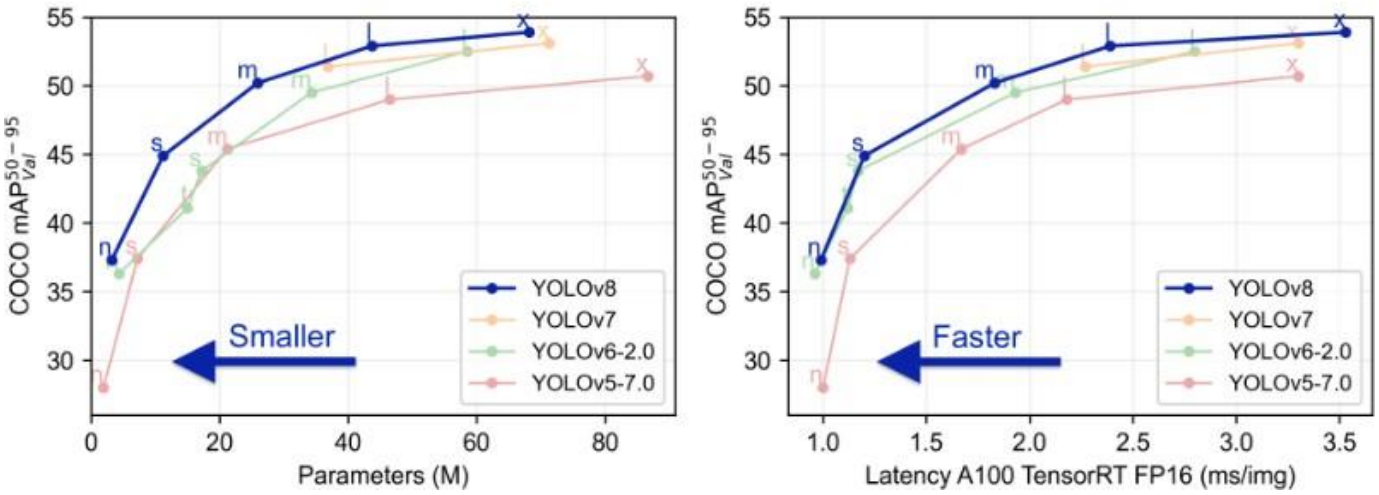

Рисунок 4.2 – Порівняння найновіших варіантів моделей YOLO [10]

Після аналізу різних моделей можна зазначити, що YOLOv5 має найменший обсяг та найнижчу точність та швидкість порівняно з іншими моделями. У той же час YOLOv8 показує кращі або подібні результати порівняно з іншими моделями. Тому для розробки системи розпізнавання ми виберемо модель YOLOv8. Розглянемо варіанти YOLOv8, що включають Nano, Small, Medium, Large, і Extra Large, зазначені на рисунку 4.3.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)
YOLOv8n	640	37.3	80.4	0.99	3.2
YOLOv8s	640	44.9	128.4	1.20	11.2
YOLOv8m	640	50.2	234.7	1.83	25.9
YOLOv8l	640	52.9	375.2	2.39	43.7
YOLOv8x	640	53.9	479.1	3.53	68.2

Рисунок 4.3 – Порівняння моделей YOLOv8 [10]

Результати показують, що кожна з моделей може бути застосована в різних областях. Наявність GPU в обчислювальному пристрої значно підвищує обчислювальну потужність. Завдяки можливості GPU виконувати паралельні

обчислення, більшу пам'ять та спеціалізовану роботу з векторами та матрицями, параметри мережі обчислюються швидше.

YOLOv8n має найнижчу точність серед усіх інших моделей, але через свій розмір та кількість параметрів вона може навчатися швидше на слабкому комп'ютері. Цю модель можна використовувати на мобільних пристроях або слабких системах, забезпечуючи максимальну кількість кадрів на секунду (FPS) у порівнянні з іншими моделями.

YOLOv8s має вже вищу точність, але її розмір вже значно більший. Це означає, що модель потребує більше часу на обробку кадру, особливо на CPU. Рекомендується використовувати цю модель у поєднанні з потужним процесором або слабкою GPU.

YOLOv8m має більшу точність, але значно повільніше обробляє кадри, ніж легші версії. Рекомендується використовувати цю модель, якщо потрібно ще більше підвищити точність для розпізнавання більшої кількості деталей та об’єктів на кадрі. Для роботи в реальному часі рекомендується використовувати середню за потужністю GPU.

YOLOv8l та YOLOv8x використовуються, коли потрібна максимальна точність розпізнавання, особливо якщо об'єктів та деталей дуже багато на кадрі і вони можуть бути дрібними. Для роботи в реальному часі потрібно використовувати потужну GPU.

У нашому проекті точність розпізнавання має бути достатньою, оскільки деталі на зображеннях є маленькими. Планується використовувати систему для розпізнавання військової техніки та людських ресурсів з GPU з середнім обсягом відеопам'яті, тому ми вирішили обрати модель YOLO medium, яка має найкраще співвідношення характеристик .

4.1.2 Архітектура моделі

Архітектура YOLOv8 складається з трьох компонентів: хребет (backbone), шия (neck) та голова (head). На рисунку 4.4 представлена структура мережі, де

голова та шия зображені як один модуль. Зауважимо, що ця архітектура відноситься до одноетапного виявлення.

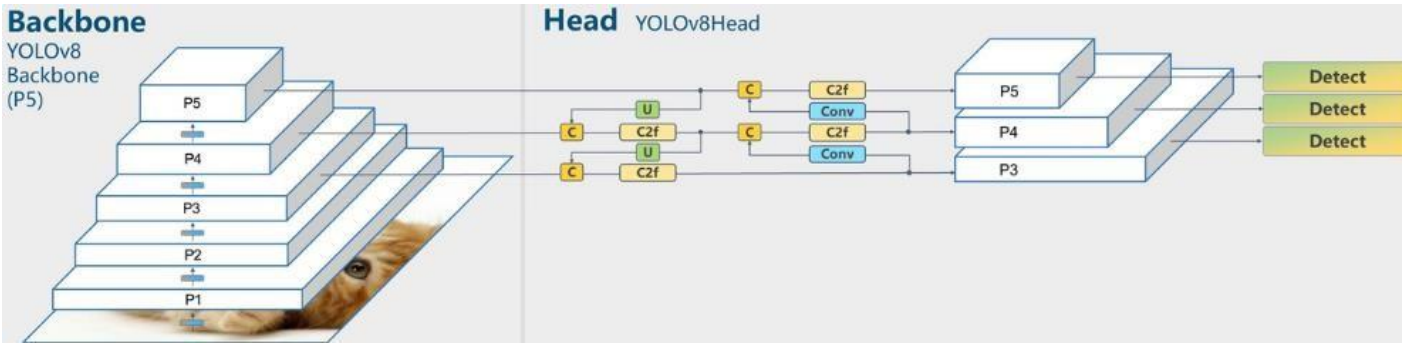

Рисунок 4.4 – Структура YOLOv8 Network [15]

Модель YOLOv8 має одну особливість, яка відрізняє її від інших моделей сімейства: вона працює без якорів. Замість того, щоб користуватися заздалегідь визначеними рамками різних масштабів для виявлення об'єктів, які називаються якорями, YOLOv8 безпосередньо прогнозує центр об'єкта. Під час навчання моделі кожній області у навчальних даних призначалася певна область з якорями, яка мала найбільше співвідношення перетину до об'єкта (IoU). Тоді модель навчилася прогнозувати зміщення та зміну масштабу від цієї області до базової області, замість прогнозування якоря безпосередньо.

Перевага безякірного виявлення полягає в тому, що воно більш гнучке і ефективне. В якірному виявленні потрібно ручно визначати якірні рамки, що може бути недосконалим, якщо якорі не відповідають розподілу форм об'єктів у даних. Виявлення об'єктів без використання якорів допомагає зменшити кількість пропозицій для виявлення, що полегшує виконання NMS [16].

Сама модель YOLO має наступні складові, такі як хребет або головний канал - це блок, який використовується для виділення ознак. Він відповідає за створення набору високорівневих ознак в різних масштабах та з різними роздільними здатностями, які потім використовуються іншими частинами мережі. Цей підхід подібний до того, що було розглянуто в RetinaNet застосовуючи FPN.

Для цього зображення його обробляють кількома шарами згортки для отримання карт ознак. Шари організовані в ієрархічному порядку, де кожен з них фіксує особливості на різних рівнях. Ці шари виконують базові операції, такі як

згортка, пакетна нормалізація та активаційна функція. У цій моделі ці шари об'єднані в етап Conv. Також у цій моделі як функція активації використовується SiLU [17], яка записується так:

$$f(x) = x\sigma(x), \tag{4.1}$$

де σ – сигмоїда.

Давайте порівняємо графіки функцій SiLU і ReLU, зображені на діаграмі 4.5.

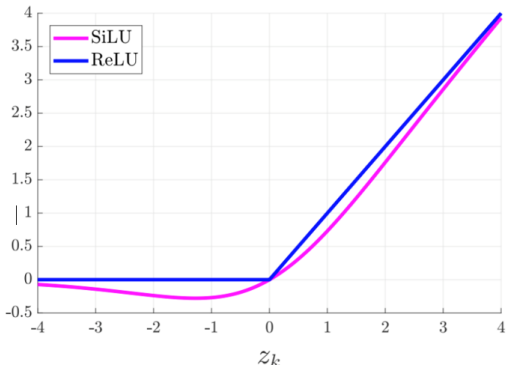

Рисунок 4.5 – Функція активації SiLU порівняно з ReLU [17]

Можна помітити, що SiLU має перевагу над ReLU тим, що вона забезпечує більш плавний градієнт, що дозволяє зафіксувати більше нелінійних особливостей. Крім того, у функції SiLU немає проблеми "вмираючого" ReLU, оскільки вона, так само як Leaky ReLU, активується на від'ємному діапазоні. Основою для розвитку є модифікація моделі CSPDarknet53, архітектура якої зображена на рисунку 4.6. Однією з особливостей модифікованої мережі є використання модуля C2f. Завдання цього модуля полягає в тому, що він розділяє вхідні картки ознак на дві частини: одна з них переходить до наступного блоку згорткових шарів, тоді як друга частина проходить через декілька згорткових шарів [16].

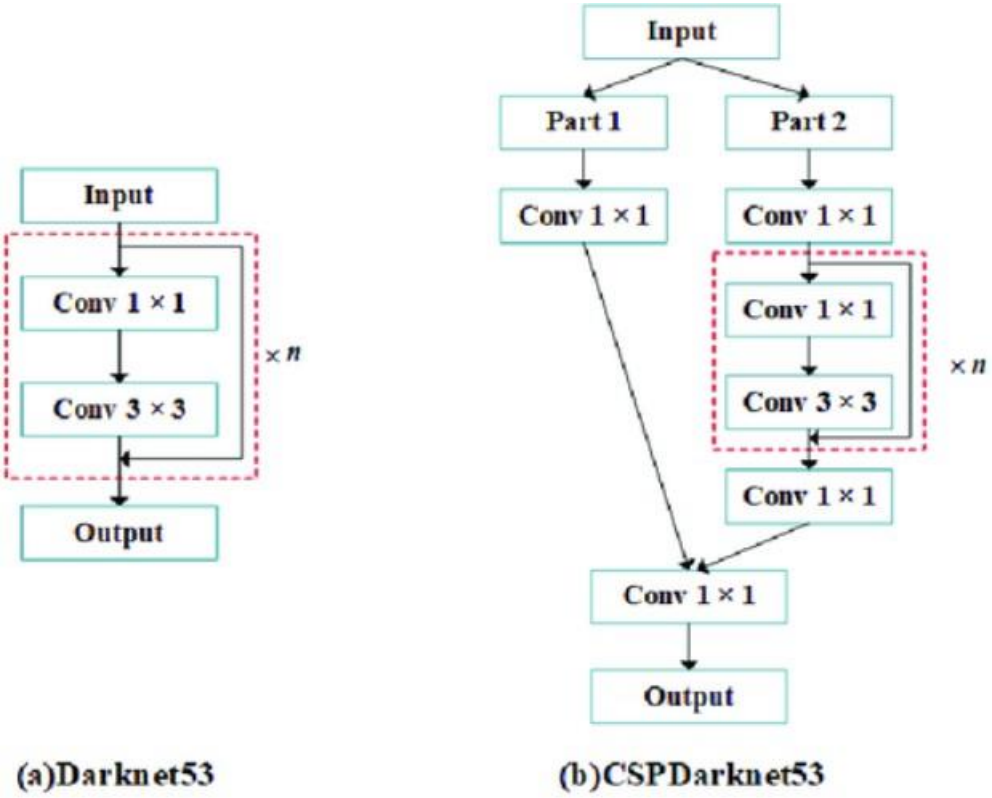

Рисунок 4.6 – Архітектура CSPDarknet53 [18]

Модулі C2F в YOLOv8 використовують частково поперечний підхід (Cross Stage Partial, CSP), що був натхненний DenseNet, з таких причин [19]:

– зменшення проблеми зникаючого градієнта: з ростом глибини мережі стає складніше передавати сигнал помилки від виходу до ранніх шарів під час зворотного поширення. CSP допомагає вирішити цю проблему, підсилюючи потік градієнтів через мережу;
– сприяння розповсюдженню особливостей: архітектура CSP полегшує зв'язок між шарами в мережі, що дозволяє більш ефективно передавати особливості від ранніх шарів до пізніх;
– підтримка повторного використання функцій: за допомогою з'єднання шарів та обміну інформацією між ними CSP стимулює мережу використовувати та повторно використовувати функції на різних рівнях, що підвищує ефективність використання функцій;
– зменшення кількості параметрів: концепція CSP є параметрично-ефективною, оскільки вона забезпечує прямі зв'язки між рівнями, що

допомагає зменшити кількість параметрів у мережі і створити більш легку та ефективну модель.

Отже, впровадження модуля CSP (рис. 4.7) як складової YOLOv8 сприяє підвищенню ефективності моделі, її продуктивності та загальної ефективності виявлення об'єктів [20].

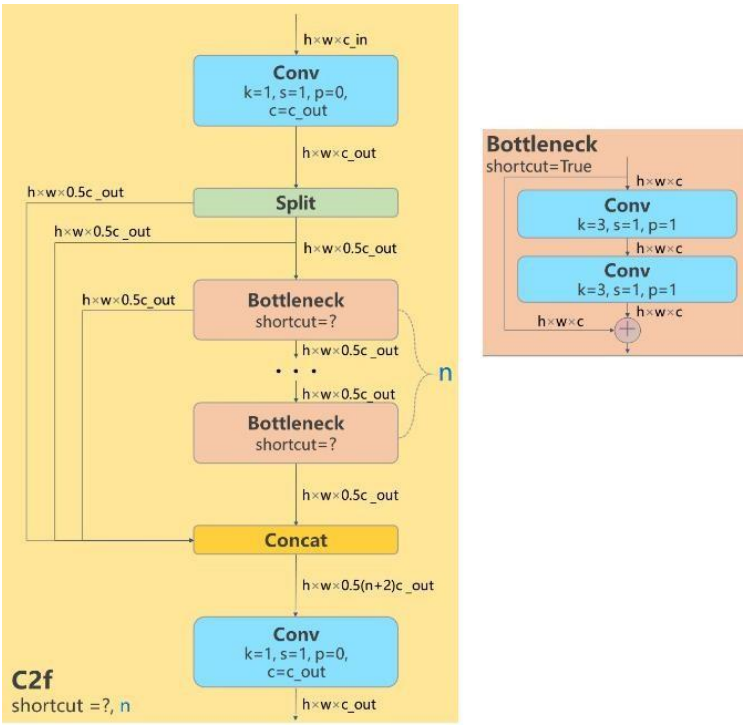

Рисунок 4.7 – Структура модулю C2f в YOLOv8 [15]

Число n визначає кількість разів, які модуль Bottleneck застосовується на кожному рівні CSP. Це число може змінюватися в залежності від конкретного рівня в архітектурі та типу моделі YOLOv8.

Модуль Bottleneck допомагає зменшити розмірність карт ознак, що знижує обчислювальну складність мережі і сприяє навчанню вищих рівнів ознак. Модуль C2f агрегує карти ознак об'єктів різних масштабів для виявлення об'єктів різних розмірів. Він об'єднує об'єкти з високою роздільною здатністю з попереднього етапу і семантично багаті об'єкти з низькою роздільною здатністю з наступного етапу вздовж глибинного виміру, створюючи нову карту об'єктів із детальною просторовою та глибокою семантичною інформацією [18].

Заключним етапом маршруту є модуль швидкого об'єднання просторових пірамід SPPF (рис. 4.8).

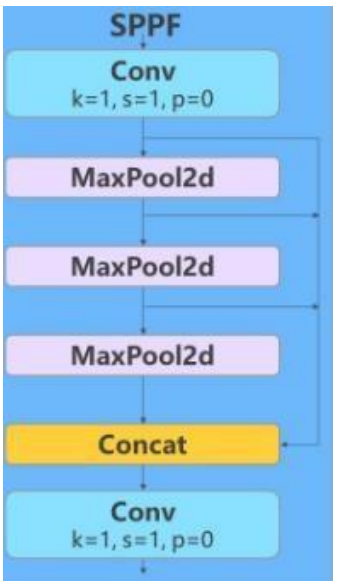

Рисунок 4.8 – Структура модулю SPPF [14]

Основна мета цього модуля полягає в тому, щоб об'єднати вхідні дані і створити вихід з фіксованим обсягом. Він переважно вигідний тим, що може значно розширити допустиме поле та виокремити найважливіші контекстуальні характеристики, при цьому не погіршуючи продуктивність мережі [18].

Після пройдення головної магістралі, перед переходом до головного блоку моделі, отримані ознаки проходять через шийний вузол. Основна мета шийного вузла полягає у змішуванні та комбінуванні карт ознак, які створені основною частиною моделі, щоб підготувати їх до подальшої обробки в головному блоку. Для цього шийний вузол також використовує модулі C2f, Conv, а також шари злиття та збільшення роздільної здатності зображення [19].

Голова отримує інформацію від шиї і відповідає за остаточне розпізнавання об’єктів. Вона генерує три види виходів, щоб прогнозувати на різних масштабах. Перший вид виходу служить для виявлення великих об'єктів і формується на основі загальних ознак, хоча з меншою деталізацією. Другий вид виходу відповідає за розпізнавання об'єктів середнього розміру і базується на карті з середньою розшифровкою. Третій вид виходу визначає менші об'єкти та використовує деталізовані картини з високою роздільною здатністю.

Модуль визначення в голові використовує як згорткові, так і лінійні шари (див. рис. 4.9). Карти ознак проходять через послідовність згорткових і лінійних

шарів, результати яких використовуються для визначення функцій втрати для меж та класів об’єктів.

Рисунок 4.9 – Модуль виявлення [14]

Для оцінки втрат обмежувальних рамок використовуються два підходи: повна втрата CIoU та розподільна фокусна втрата DFL. Перше забезпечує точніше прогнозування розміру та положення об'єктів, враховуючи не лише їх перекриття, а й відстань між центрами та співвідношення сторін. Це поліпшує точність моделі та збігається швидше, ніж інші підходи IoU. Другий підхід, DFL, спрямований на зменшення дисбалансу між завданнями класифікації та локалізації, фокусуючись під час навчання на складних прикладах. Він використовує прогнозоване значення IoU, що дозволяє зробити обидва завдання більш узгодженими [21].

YOLOv8 використовує бінарну перехресну ентропію (BCE) для визначення втрат у класифікації. Цей метод застосовується в ситуаціях, коли потрібно визначити, чи належить елемент певному класу. Втрати визначаються за різницею між реальним класом і передбаченою ймовірністю цього класу. Якщо прогноз точний і впевнений, втрати мінімальні; якщо прогноз неправильний або неоднозначний, втрати зростають. У випадку невизначеного прогнозу втрати знаходяться між цими екстремами. Для багатокласової класифікації підрахунок втрат розглядається як кілька окремих завдань BCE [22]

4.1.3 Тренування моделі

Для навчання моделі YOLOv8 необхідно створити набор даних, або використати існуючий який більш за все підходить для виконання поставлених задач. Попередньо модель було навчено на наборі даних VisDrone2019, також було підготовлено підбірку з 2903 зображень . Основна ідея YOLO полягає в тому, щоб

розділити вхідне зображення на сітку та в одному проході здійснити визначення та класифікацію об'єктів.

Після чого ми поділили набір зображень на 3 частини (папки) “train”, “valid” та “test”.

Тренування (train): ця частина даних використовується для навчання моделі. Модель в процесі тренування «вивчає» зображення та відповідні мітки (класи та координати об'єктів), щоб вона могла ефективно визначати об'єкти.

Валідація (valid): валідаційний набір використовується для налаштування гіперпараметрів моделі та контролю її навчання. Модель не бачить дані з валідаційного набору під час тренування, але результати її передбачення на цих даних використовуються для визначення, наскільки добре модель вчиться та чи відбувається перенавчання (overfitting).

Тестування (test): тестовий набір використовується для оцінки загальної ефективності моделі після завершення тренування. Це незалежні дані, які модель не бачила під час навчання або валідації. Тестування дозволяє оцінити, наскільки добре модель узагальнює свої знання на нові дані.

Цей підхід допомагає уникнути перенавчання, де модель вчиться дуже добре на тренувальних даних, але не може ефективно визначати нові дані. Також, він дозволяє оцінити реальну продуктивність моделі на нових, реальних даних, що не використовувалися під час тренування [23].

Далі за допомогою LabelImg обвели та позначили об’єкти які в подальшому повинні розпізнаватись.

LabelImg де є всі необхідні інструменти для цього процесу. LabelImg - це безкоштовний інструмент для розмічення об'єктів на зображеннях. Цей інструмент дозволяє користувачам створювати мітки (анотації) для об'єктів, таких як обличчя, автомобілі, тварини тощо, на фотографіях чи у відео (див. рис. 4.10).

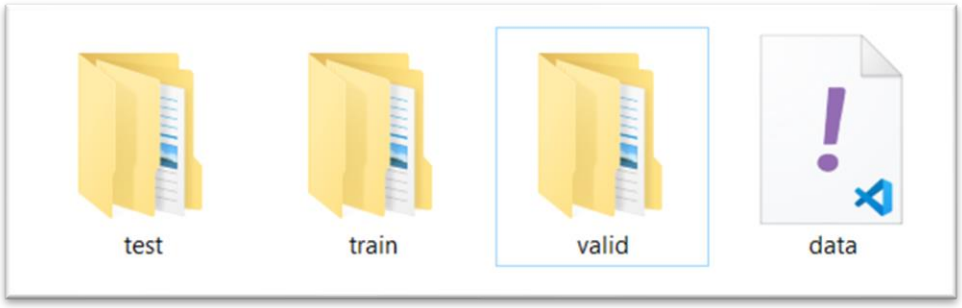

Рисунок 4.10 – Приклад поділу набору даних[власний рисунок].

На зображенні виділяються об'єкти за допомогою контурів, кожен з яких відповідає своєму класу. Основні функції LabelImg включають в себе:

- ручне розмічування: ми маємо можливість виділяти області на зображенні, розміри, положення;
- множинне розмічення: ми маємо можливість на одному зображенні визначати багато об’єктів та призначати їм класи і мітки;
- багатоформатність: LabelImg підтримує різні формати для аннотацій ,а саме XML для Pascal VOC та YOLO;
- кросплатформеність: цей інструмент доступний для Windows,Linux, macOS.

Цей інструмент дуже часто застовоюється для підготовки даних для навчання нейронних мереж .

Для YOLOv8 дані подаються у наступному форматі (рис. 4.11): індекс класу, координати центру об'єкту, ширина та висота об'єкту.

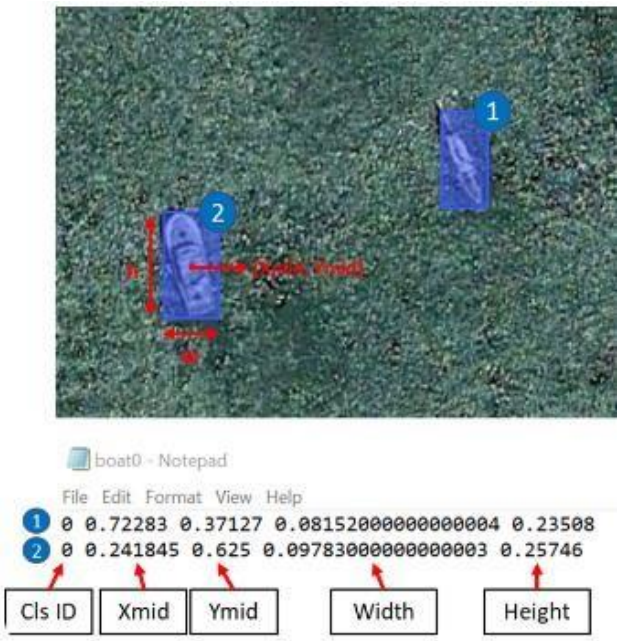

Рисунок 4.11 – Зразок анотованого зображення з інформаційним файлом [24]

Це дозволить створити розширений набір зображень для навчання, включаючи зображення з шумами, розмиттям та зміненим положенням. Такий підхід дозволяє створювати модель, яка краще пристосована до різноманітних умов і шумів.

Після створення датасету його можна використовувати з різними моделями навчання. У нашому випадку ми зацікавлені в YOLOv8, тому імпортуємо дані у відповідному форматі.

Адже нам потрібно використовувати конфігураційний файл data.yaml.

У контексті YOLO, data.yaml містить інформацію про ваш набір даних та конфігурацію моделі. Зазвичай цей файл включає такі елементи:

- шлях до зображень: прямий шлях або шлях до викачування набору даних на тренування, валідації та тестування;
- шлях до файлів анотацій: прямий шлях до файлів анотації;
- кількість класів: кількість класів або об'єктів, які будуть шукати модель.

Як вже зазначено вище, ми плануємо використовувати модель YOLOv8 у нашій роботі, оскільки нам не відомо, наскільки потужні обчислювальні ресурси будуть доступні для використання.

Для навчання ми встановимо такі параметри:

– mode = train;
– task = detect;
– epochs = 40;
– batch = -1;
– imgsz = 640;
– optimizer = Adam;
– lr0 = 0.0001;
– momentum = 0.9;
– model = yolov8m.pt.

Розглянемо встановлені параметри. Режим (Mode) визначає, як Yolo працює: у режимі тренування, валідації або передбачення. Задача (Task) визначає тип завдання моделі YOLO, такий як виявлення, сегментація або класифікація. Оскільки ми плануємо виявляти об'єкти в реальному часі, ми обираємо опцію "detect".

Epochs визначає кількість епох, протягом яких тренується модель. З більшою кількістю епох модель може краще навчитися.

Параметр Batch відповідає за кількість зображень у пакеті, який надсилається моделі для обробки. Більша кількість зображень у пакеті сприяє швидшому навчанню, але вимагає більше відеопам'яті. Значення -1 дозволяє системі автоматично вибирати оптимальну кількість зображень у пакеті.

Imgsz визначає розмір вхідного зображення до моделі. Більший розмір сприяє точнішому розпізнаванню, але тренування займатиме більше часу і ресурсів.

Optimizer дозволяє вибрати оптимізатор для моделі. Можливі опції включають SGD, Adam, RMSProp. З розглянутих варіантів оптимізаторів ми обираємо Adam.

Замість того, щоб навчати модель спочатку, можна використовувати попередньо навчену модель на датасеті COCO. Це прискорить та спростить процес навчання, оскільки будуть задані початкові параметри вагів.

Lr0 представляє собою початкову швидкість навчання моделі, рекомендовану для оптимізатора Adam як 0.0001. У YOLO, моментум представлений параметром beta1, який рекомендується встановлювати на рівні 0.9.

Після тренування моделі ми отримаємо характеристики моделі, які показані на рисунку 4.13.

```
                 from  n    params  module                                       arguments
  0                -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
  1                -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                -1  1     29056  ultralytics.nn.modules.block.C2f             [64, 64, 1, True]
  3                -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  4                -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  5                -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  6                -1  2    788480  ultralytics.nn.modules.block.C2f             [256, 256, 2, True]
  7                -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                -1  1   1838080  ultralytics.nn.modules.block.C2f             [512, 512, 1, True]
  9                -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11            [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                -1  1    591360  ultralytics.nn.modules.block.C2f             [768, 256, 1]
 13                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14            [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 16                -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 17           [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 19                -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 20            [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                -1  1   1969152  ultralytics.nn.modules.block.C2f             [768, 512, 1]
 22      [15, 18, 21]  1   2117983  ultralytics.nn.modules.head.Detect           [5, [128, 256, 512]]
Model summary: 225 layers, 11137535 parameters, 11137519 gradients

Model summary (fused): 168 layers, 11127519 parameters, 0 gradients
```

Рисунок 4.13 –YOLOv8 перед навчанням та після нього[власний рисунок]

Після закінчення навчання ми отримали графіки метрик для оцінки якості навченої моделі. Давайте проаналізуємо ці результати.

На діаграмі 4.14 показані графіки функцій втрат, точності та повноти, а також метрики mAP для IoU 50, а також для діапазону значень IoU від 50 до 95.

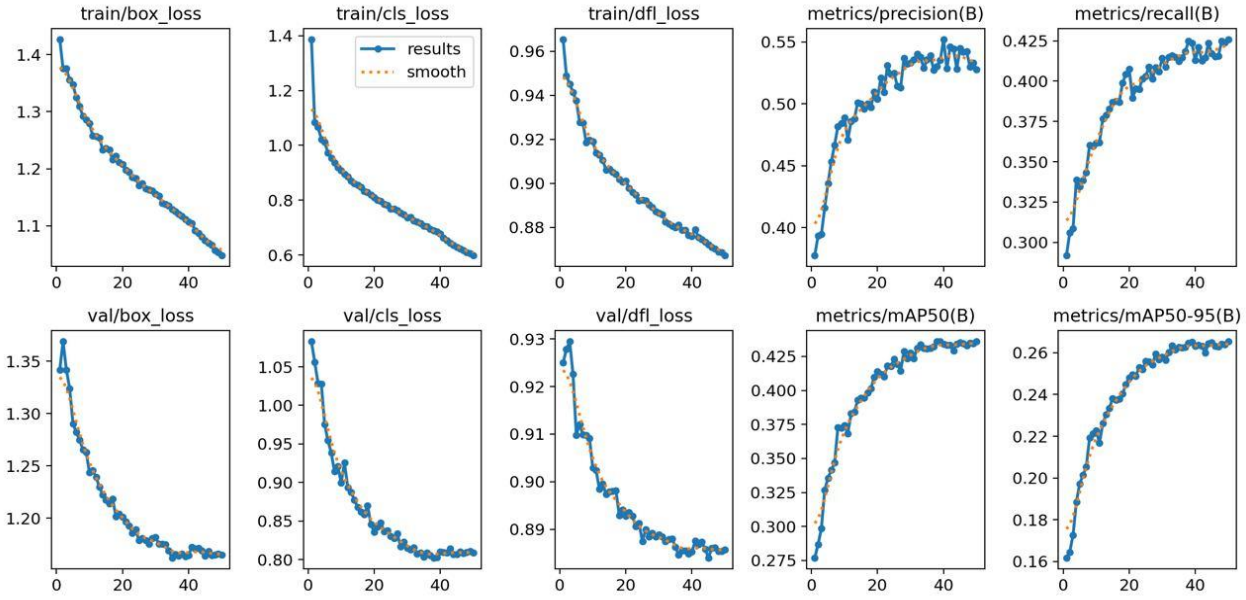

Рисунок 4.14 – Результати втратних функцій[власний рисунок]

На графіках функцій втрат представлені результати тренування та валідації для кожного типу втрат. Середні втрати під час тренування майже вдвічі менші, ніж під час валідації, що може свідчити про можливе перенавчання моделі на тренувальних даних. Однак показники втрат під час валідації також демонструють високу ефективність моделі.

Результати по повноті та точності свідчать про те, що модель точно класифікує об'єкти на зображеннях. Метрика mAP також підтверджує високу точність у виділенні та класифікації об'єктів. Хоча точність знижується при збільшенні порогу IoU, вона залишається на високому рівні.

Щоб переконатися, чи можна використовувати цю модель, її необхідно протестувати на тестовому наборі даних, щоб оцінити її здатність до розпізнавання нових даних. Першою метрикою тестування є нормалізована матриця плутанини, зображена на рисунку 4.15, яка відображає, наскільки точно модель визначила класи об'єктів на зображенні.

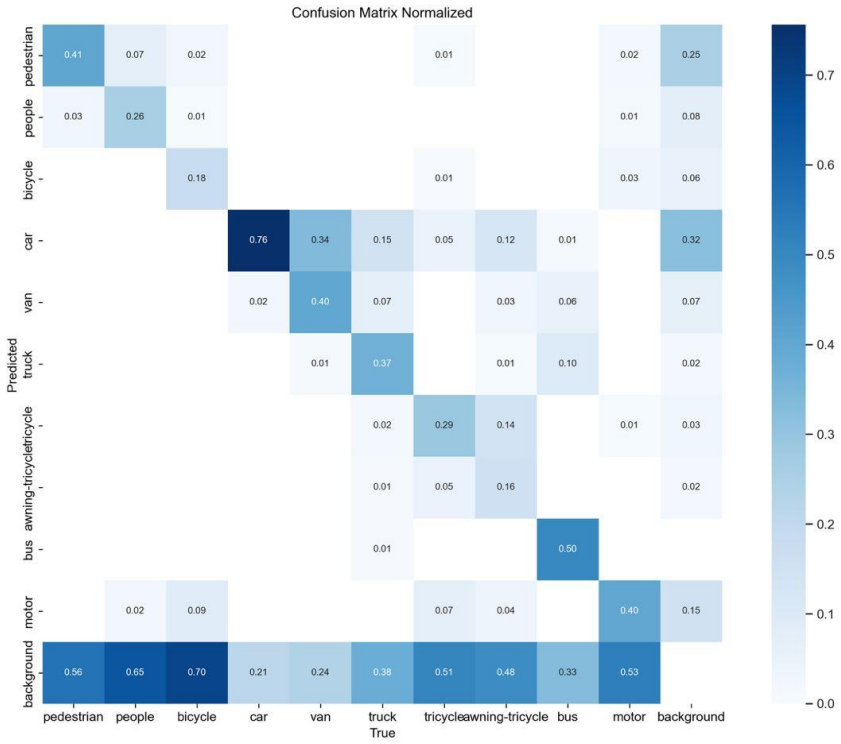

Рисунок 4.15 – Нормалізована матриця плутанини[власний рисунок]

За отриманими результатами видно, що модель успішно впоралася з ідентифікацією всіх типів транспорту. Показники для автобусів та інших

автомобілів трохи нижчі, але все ще дуже високі. Для поліпшення результатів потрібно оновити набір даних, оскільки модель з поточними параметрами показує відмінні результати.

Наступним параметром для оцінки є PR-крива (рис. 4.16). Згідно з її результатами, можемо зрозуміти, що модель ефективно впорався із виявленням та класифікацією транспортних засобів, але показники для автомобілів залишаються низькими. Цей показник використовується для розрахунку середньої точності (mAP) моделі. При IoU=0.5 ми спостерігаємо високі показники точності виявлення на рівні 0.95.

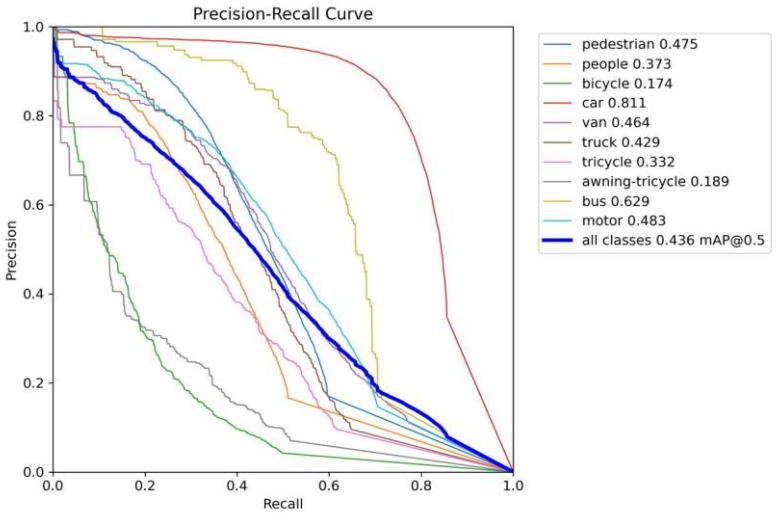

Рисунок 4.16 – PR-крива моделі[власний рисунок]

Ще одним методом для оцінки точності є F1-міра (див. Рис. 4.17). F1 є показником, який об'єднує точність і повноту в одне число. Вона дає комплексне уявлення про продуктивність моделі. Точність і повнота часто протистоять один одному, утворюючи компроміс. Модель з високою точністю дуже впевнена у своїх позитивних прогнозах, але може пропустити багато реальних позитивних прикладів. Навпаки, модель з високим рівнем повноти ідентифікує більшість позитивних прикладів, але може помилково вважати багато негативних прикладів за позитивні. Об'єднуючи ці дві характеристики, F1-міра намагається збалансувати цей компроміс . Можна сказати, що F1 є більш комплексною мірою точності, ніж просто точність. F1-міра розраховується наступним чино:

$$F1 = \frac{2*P*R}{P+R}, \tag{4.2}$$

де P – точність;

R – повнота.

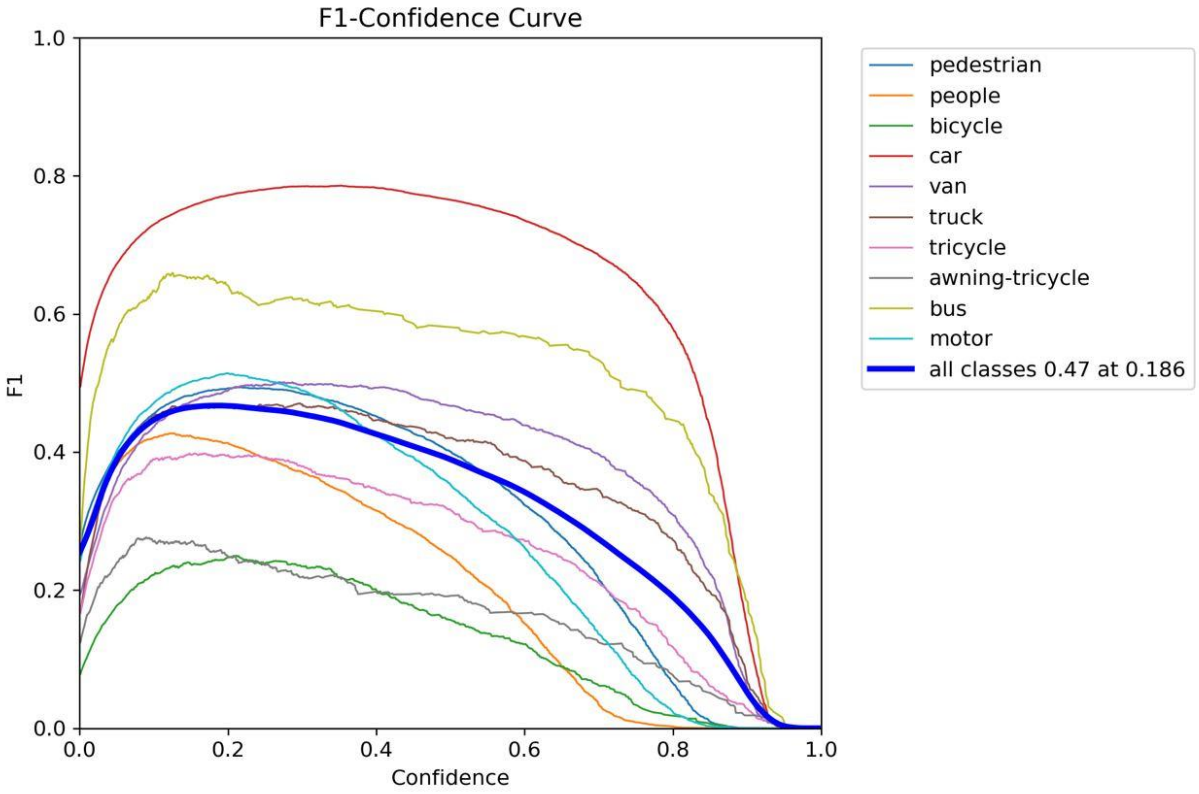

Рисунок 4.17 – Крива F1 моделі[власний рисунок]

Щоб сформулювати криву F1, потрібно обчислити значення F1 для кожного рівня впевненості моделі у розпізнаванні. З рисунка 4.17 видно, що при впевненості моделі на рівні 0.69 досягаємо точності F1 у 0.91. Це вказує на те, що ці показники відмінні для роботи в реальному часі. Таким чином, аналізуючи результати тестування моделі на тестовому наборі, можна стверджувати, що вона успішно виконує свою задачу, і можна продовжувати розробку системи та впровадження цього методу розпізнавання.

5 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ЧАСТИНИ

5.1 Мова програмування та середовище розробки

Також Python є однією з найпопулярніших мов програмування для розробки нейромереж і роботи з машинним навчанням. Існує кілька бібліотек та фреймворків, які роблять використання нейромереж в Python надзвичайно зручними. Ось декілька ключових компонентів для роботи з нейромережами в Python:

- PyTorch: PyTorch - інший популярний фреймворк для глибокого навчання, який отримав широку популярність завдяки своєму простому та динамічному обчислювальному графу. Він особливо популярний серед дослідників і лабораторій зі штучного інтелекту;
- бібліотеки для обробки даних: Python має ряд потужних бібліотек для обробки даних, такі як NumPy, pandas та matplotlib, які допомагають підготовити дані для навчання нейромереж та візуалізувати результати.

З використанням Python та цих інструментів ви можете розробляти, навчати та впроваджувати різноманітні типи нейромереж, включаючи згорткові нейромережі для зображень, рекурентні нейромережі для обробки послідовностей та багато інших.

Python також дозволяє використовувати передові алгоритми та інструменти для вирішення різних задач машинного навчання та глибокого навчання.

Перелік застосованих нами бібліотек Python:

- cvzone;
- ultralytics;
- hydra-core;
- matplotlib;
- numpy;
- opencv-python;
- pillows;
- pyyaml;
- requests;

- scipy;
- torch;
- torchvision;
- tqdm;
- filterpy;
- scikit-imaqe;
- lap.

5.2 Інструменти розробки

Для розробки програми було використано кілька бібліотек та програмних реалізацій. У цьому процесі для тренування та інтеграції моделі YOLO використовувалася бібліотека ultralytics. Вона надає інструменти для налаштування та тренування мереж, а також містить попередньо треновані моделі, які можна використовувати для навчання на нових даних [21]. Також використовувалася бібліотека PyTorch, яка надає інструменти для використання відеокарт з CUDA для прискорення обчислень моделі, що суттєво підвищило її продуктивність. IDE для розробки було вибрано Pycharm. Також є можливість при навчанні використовувати Colaboratory, або Colab, який в свою чергу надає безкоштовний доступ до хмарного сервісу Google і дає можливість запускати код Python у середовищі Jupyter Notebook. Colab надає доступ до великого обсягу обчислювальних ресурсів. Ми в свою чергу в експерименті та навчанні використовували операційну систему Windows 11 з Python 3.12.1 , PyTorch версії 2.2, Cuda 8.7. Навчання нейронної мережі виконувалось на апаратному забезпеченні Zotac Nvidia GTX 1080TI з 11 GB GDDR5x пам’яті, Intel Core i7 6800k розігнаному до 4.0 ГГц при 1.275V , DDR4 пам'ять 32 ГБ 2833 Мгц, SSD Transcend 1ТБ/560МБ/с. Гіперпараметри при навчанні та тестуванні та перевірці залишались незмінними та послідовними ,всі зображення були маштабовані до роздільної здатності 640x480 пікселів .Кількість епох була встановлена на рівні 100 тому що ми маємо власний датасет з невеликою кількістю зображень.

5.3 Результат реалізації програми

Різні варіанти розмірів YOLOv8 були розглянуті для вибору найкращого під конкретні умови завдання. Архітектура YOLOv8 була досліджена як система одноетапного виявлення, розглянуто її компоненти, такі як магістраль для отримання карт ознак зображення, шия для об'єднання отриманих карт ознак для збереження основних і підсилення головних ознак, а також голова для класифікації та виявлення об'єктів на зображенні.

Було проведено тренування моделі на створеному датасеті за вказаними параметрами та аналіз отриманих результатів за ключовими метриками. Перед початком розробки було вибрано програмне середовище та розглянуто необхідні модулі та бібліотеки. Основними бібліотеками були вибрані ті, які дозволяють захоплювати та редагувати зображення, працювати з моделями виявлення та розпізнавання тексту, а також підключати відеокарту для обчислень. Так як результати yolov8n був найкращих ,але все рівно недостатній для точного розпізнавання об’єктів основне навченная та реалізацію дотатку було зроблено на версії yolov8m ,яка по своїм характеристикам та навантаженню на систему найкраще підходить на мою думку.

Після навчання на наборі даних VisDrone , та власному наборі даних точність виявлення та взагалі можливість виявлення військової техніки та людських ресурсів ворога значно виросла. Точність виявлення наглядно продемонстрована на Рисунку 5.1 та 5.2.

Рисунок 5.1 – Виявлення військової техніки за допогою навченної моделі[власний рисунок]

Рисунок 5.2 – Виявлення військової техніки за допомогою навченої моделі[власний рисунок]

Була розроблена архітектура програми, яка складається з кількох модулів модулів. Спочатку здійснили взаємодію моделей Yolo із нашим веб додатком за

допомогою фреймворку Django. Django – представляє собою високорівневий Python фреймворк для розробки веб додатків ,який суттєво спрощує створення складних сайтів, він забезпечує ефективний і контрольований спосіб використання інструментів для створення веб додатку. Взаємодія здійснюється за допомогою методу «post» отримує зображення від користувача див.рис.5.3, зберігає його в таблиці бази даних ,зчитує зображення з бази даних та надсилає його моделі Yolo, зберігає результат детекції та надсилає клієнту. Після чого здійснюється виведення зображення із розпізнавальними об'єктами.

```
def post(self, request, *args, **kwargs):
    f1 = request.POST.get('model')
    if f1 == "Model 2":
        context = {"inference_img": 1
        }
    else:
        model_path = r'C:\Users\Admin\Desktop\webYOLO\myproject\weights\army.pt'
        model = YOLO(model_path)
        # Отримання завантаженого зображення з POST-запиту
        img = request.FILES.get('photo')
        img_instance = Image(image=img)
        img_instance.save()
```

Рисунок 5.3 – Метод обробки зображення за допомогою «post» [власний рисунок]

```
Performing system checks...

System check identified no issues (0 silenced).
June 17, 2024 - 12:56:50
Django version 5.0.6, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рисунок 5.4 – Запуск сервера.[власний рисунок]

Після запуску сервера рис.5.4, ми переходимо за адресою локального сервера http://127.0.0.1:8000/ для відкриття веб сторінки, на якій є вікно для завантаження фото і відео матеріалів, та детектування. Після чого ми бачимо на сторінці додатку виведене зображення з об’єктами ,а в терміналі бачимо швидкість виконання та знайдені об’єкти.

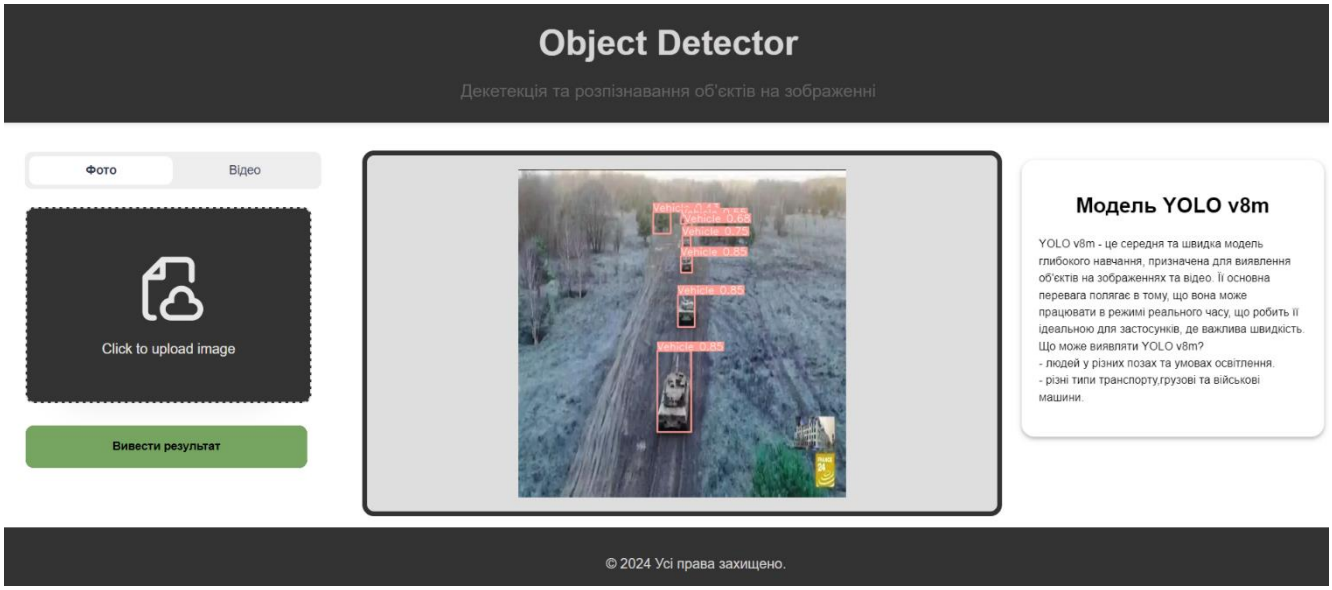

Рисунок 5.5 – Вивід результату розпізнавання об’єкта у веб додатку. [власний рисунок]

Як можна бачити на рисунку 5.5 навчена модель показує дуже гарну впевненість від 0.65 до 0.85, це означає що наша модель впевнена на 65-85% що на зображенні військові машини, в свою чергу це доказ того, наскільки важливою є якість вхідних даних таких як навчальний датасет, та цільовий вибір архітектури YOLO та моделі по вазі та швидкодії. Цей додаток може слугувати зразком для більш комерційної або державної програми.

ВИСНОВКИ

У результаті проведеного дослідження ефективності різних архітектур YOLO для виявлення об'єктів на зображеннях, отриманих від безпілотних літальних апаратів (БПЛА), можна зробити кілька важливих висновків.

Перше, архітектури YOLO демонструють високу точність у виявленні об'єктів при використанні в аерокосмічних технологіях. Вони виявляються досить швидкими та дозволяють забезпечити потрібну реакцію БПЛА на зміни у навколишньому середовищі.

Друге, результати показують, що різні архітектури YOLO можуть мати різний рівень ефективності в залежності від конкретних умов та завдань. Важливо ретельно вибирати архітектуру з урахуванням специфічних вимог, таких як ресурсоємність чи необхідність високої точності.

Третє, у дослідженні було виявлено, що врахування різноманітних метрик, таких як час виявлення та ресурсоємність, є критичним для повного розуміння ефективності моделей. Це дозволяє визначити баланс між точністю та швидкістю виявлення, що є ключовим у виборі оптимальної архітектури для конкретного застосування.

У цілому, результати дослідження вказують на великий потенціал використання архітектур YOLO в парі з БПЛА та надають підстави для подальших досліджень у цьому напрямку. Оптимальний вибір моделей YOLO для конкретних завдань визначається конкретними умовами та вимогами, і врахування цих факторів є ключовим у розробці та вдосконаленні систем виявлення об'єктів на зображеннях, отриманих від БПЛА.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. N. Bilous, O. Svidin, I. Ahekian, и V. Malko, «A skeleton-based method for exercise recognition based on 3D coordinates of human joints», IJ-AI, т. 13, вып. 2, с. 1805, июн. 2024, doi: 10.11591/ijai.v13.i2.pp1805-1816.

2. Н. В. Білоус, І. А. Агекян, О. В. Рассоха, and О. В. Грамм, “Дослідження методів для розробки програмної системи розпізнавання емоцій та визначення стану здоров’я людини,” Біоніка інтелекту. – Харків : ХНУРЕ, vol. №1 (94), pp. 65–70, 2020, doi: 10.30837/bi.2020.1(94).10.

3. N. V. Bilous, O. V. Rassokha, I. A. Ahekian, и O. V. Gramm, «Study of methods for the development of a software system for recognizing emotions and determining the state of human health», Bionics of Intelligence, т. 94, сс. 65–70, doi: 10.30837/bi.2020.1(94).10.

4. A. O. Rakova и N. V. Bilous, «REFERENCE POINTS METHOD FOR HUMAN HEAD MOVEMENTS TRACKING», RIC, т. 0, вып. 3, Art. вып. 3, ноя. 2020, doi: 10.15588/1607-3274-2020-3-11.

5. V. Krylov, G. Shcherbakova, R. Pisarenko, и N. Bilous, «Signal restoration by means of blind deconvolution based on optimization with wavelet transformation», в 2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), Kharkiv, Ukraine: IEEE, окт. 2016, сс. 21–23. doi: 10.1109/INFOCOMMST.2016.7905324.

6. N. V. Bilous, I. A. Ahekian, и V. V. Kaluhin, «DETERMINATION AND COMPARISON METHODS OF BODY POSITIONS ON STREAM VIDEO», RIC, вып. 2, Art. вып. 2, июн. 2023, doi: 10.15588/1607-3274-2023-2-6.

7. AI’s Role in Image Processing URL - https://medium.com/@aiconfidential/ais-role-in-image-processing-d0ac08c45bc6 (дата звернення: 27.11.2023).

8. Штучні нейронні мережі: що це таке? URL - https://futurum.today/shtuchni-neironni-merezhi-shcho-tse-take/ (дата звернення: 27.11.2023).

9. Kukil, & Kukil. Mean Average Precision (mAP) in Object Detection. LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow With Examples and Tutorials. [Електронний ресурс] – режим доступу: https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/

10. YOLO models for Object Detection Explained [YOLOv8 Updated]. Encord. [Електронний ресурс] – режим доступу: https://encord.com/blog/yolo-object-detection-guide/

11. Boesch, G. Object Detection in 2023: The Definitive Guide. viso.ai. [Електронний ресурс] – режим доступу: https://viso.ai/deep-learning/object-detection/

12. Tan, L., Huangfu, T., Wu, L., & Chen, W. Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification. BMC Medical Informatics and Decision Making, 21(1). [Електронний ресурс] – режим доступу:https://doi.org/10.1186/s12911-021-01691-8

13. K, B. Object Detection Algorithms and Libraries. neptune.ai. [Електронний ресурс] – режим доступу: https://neptune.ai/blog/object-detection-algorithms-and-libraries

14. Hui, J. Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3). Medium. [Електронний ресурс] – режим доступу:https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

15. YOLOv8 Model. [Електронний ресурс] – режим доступу: https://github.com/RangeKing

16. Solawetz, J. What is YOLOv8? The Ultimate Guide. Roboflow Blog. [Електронний ресурс] – режим доступу: https://blog.roboflow.com/whats-new-in-yolov8/

17. Papers with Code - SiLU Explained. [Електронний ресурс] – режим доступу: https://paperswithcode.com/method/silu

18. Imane, C. YOLO v5 model architecture [Explained]. OpenGenus IQ: Computing Expertise & Legacy. [Електронний ресурс] – режим доступу: https://iq.opengenus.org/yolov5/

19.Solawetz, J. What is YOLOv5? A Guide for Beginners. Roboflow Blog. [Електронний ресурс] – режим доступу: https://blog.roboflow.com/yolov5-improvements-and-evaluation/

20.Khandelwal, R. Different IoU Losses for Faster and Accurate Object Detection. Medium. [Електронний ресурс] – режим доступу: https://medium.com/analytics-vidhya/different-iou-losses-for-faster-and-accurate-object-detection-3345781e0bf

21.Distribution Focal Loss. [Електронний ресурс] – режим доступу: https://arxiv.org/pdf/2006.04388v1.pdf

22.Martinek, V. Cross-entropy for classification - Towards Data Science. Medium. [Електронний ресурс] – режим доступу: https://towardsdatascience.com/cross-entropy-for-classification-d98e7f974451

23.Roboflow: [Електронний ресурс] – режим доступу: https://roboflow.com/ 39.Le, B. Get Started with Training a YOLOv8 Object Detection Model.

24.[Електронний ресурс] – режим доступу: https://www.datature.io/blog/get-started- with-training-a-yolov8-object-detection-model