Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів визначення і відстеження технічних об'єктів за допомогою БПЛА
(тема)

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

Ткаченко Д.Д.
(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
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. каф. ПІ к.т.н., проф.
_____ Дудар З. В.
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ткаченку Дмитру Дмитровичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження методів визначення і відстеження технічних об’єктів за допомогою БПЛА»

Затверджена наказом по університету від «29» березня 2024 р. № 250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18.06.2024

3. Вихідні дані до роботи бібліотеки Python OpenCV, PyTorch, моделі YOLOv7, YOLOv7-tiny, YOLOv8nano, Anaconda, середовища розробки Visual Studio Code

4. Перелік питань, що потрібно опрацювати в роботі

мета роботи, постановка задачі, опис програмного забезпечення, аналіз результатів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	30.03.2024	*виконано*
2	Огляд існуючих методів	01.04.2024	*виконано*
3	Підготовка проведення експерименту	08.04.2024	*виконано*
4	Проведення експерименту	15.04.2024	*виконано*
5	Аналіз результатів отриманих під час експерименту	22.04.2024	*виконано*
9	Підготовка пояснювальної записки	08.06.2024	*виконано*
10	Підготовка презентації та доповіді	10.06.2024	*виконано*
11	Нормоконтроль	11.06.2024	*виконано*
12	Рецензування	16.06.2024	*виконано*
13	Занесення диплома в електронний архів	17.06.2024	*виконано*
14	Попередній захист	19.06.2024	*виконано*
15	Допуск до захисту у зав. кафедри	20.06.2024	*виконано*
16	Захист магістерського дослідження	21.06.2024	*виконано*

Дата видачі завдання 28 грудня 2024р.

Студент _____ Ткаченко Д.Д.
(підпис)

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

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 54 с., 13 рис., 1 табл., 10 джерел.

АНАЛІЗ, БПЛА, ЕФЕКТИВНІСТЬ, КОМП'ЮТЕРНИЙ ЗІР, МАШИННЕ НАВЧАННЯ, ОБ’ЄКТ, ОБРОБКА ЗОБРАЖЕНЬ, YOLO.

Об’єкт дослідження – моделі машинного навчання з різними алгоритмами визначення та відстеження.

Мета роботи – Дослідження моделей YOLO для виявлення та відстеження технічних об’єктів, проведення практичного дослідження моделей машинного навчання з різними алгоритмами визначення та відстеження технічних об’єктів за допомогою БПЛА.

Результат роботи – виконане порівняння трьох моделей YOLO з метою виявлення найбільш оптимальної для визначення та відстеження технічних об’єктів за допомогою БПЛА.

UAV, COMPUTER VISION, IMAGE PROCESSING, PERFORMANCE, OBJECT, ANALYSIS, YOLO, MACHINE LEARNING.

The object of the research is machine learning models with various detection and tracking algorithms.

The purpose of the work is to study YOLO models for detecting and tracking technical objects, conducting a practical study of machine learning models with various algorithms for identifying and tracking technical objects using UAVs.

The result of the work is a comparison of three YOLO models with a goal detection of the most optimal for the identification and tracking of technical objects using UAVs.

Я, Ткаченко Дмитро Дмитрович, студент гр.ІПЗм-22-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів визначення і відстеження технічних об’єктів за допомогою БПЛА», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

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

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

CNN – Convolutional Neural Network

YOLO – You Only Look Once

mAP – mean Average Precision

GPU – Graphics Processing Unit

FPS – Frames Per Second

IoU – Intersection over Union

AP – Average Precision

TP – True Positive

FP – False Positive

ВСТУП

В останні десятиліття технології безпілотних літальних апаратів (БПЛА) набули значного розвитку та стали незамінними інструментами в різних галузях, включаючи цивільне, комерційне та військове застосування. Однією з найважливіших завдань для військових БПЛА є виявлення і відстеження військової техніки, такої як танки, бронетранспортери, та інші. У зв'язку з цим сучасні методи комп'ютерного зору, засновані на глибоких нейронних мережах, відіграють ключову роль у забезпеченні високої точності та швидкості виявлення об'єктів.

Актуальність даного дослідження обумовлено необхідністю забезпечення високої точності та швидкості виявлення військової техніки в умовах реального часу. Використання БПЛА для цих цілей вимагає від моделей комп'ютерного зору не тільки високої продуктивності, але й ефективності у використанні обчислювальних ресурсів. Моделі YOLO, будучи одними з найпередовіших і найпопулярніших методів виявлення об'єктів, продовжують еволюціонувати, пропонуючи вдосконалені версії з меншими розмірами та підвищеною швидкістю.

При виборі інструментарію для реалізації цього дослідження особлива увага приділяється деяким бібліотекам Python, що являють собою потужні інструменти, що широко використовуються в галузі машинного навчання та комп'ютерного зору. OpenCV, відома своєю ефективністю у обробці зображень, також було використано PyTorch, який є одним з найбільш ефективних фреймворків для глибокого навчання, що дозволяє інтегрувати YOLO у роботу з комп'ютерним зором.

У даному магістерському дослідженні експериментально порівняно три нейронні мережі, а саме YOLOV7, YOLOV7-tiny та YOLOV8nano для визначення найбільш ефективного рішення для завдань виявлення військової техніки з використанням БПЛА, що має важливе значення для розвитку технологій у даній галузі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі дослідження

З інтеграцією комп'ютерних технологій у другій половині XX століття, зокрема, розвитком комп'ютерного зору, розширилися можливості управління безпілотними літальними апаратами (БПЛА). Початок використання комп'ютерного зору для обробки зображень та відео належить до 1980-х - 1990-х років, коли дослідники почали створювати алгоритми для розпізнавання об'єктів.

На сьогоднішній день військові та державні служби у всьому світі активно розробляють системи автоматизованого виявлення та розпізнавання військової техніки з метою забезпечення національної безпеки. Деякі ключові аспекти цього напряму включають:

- виявлення та ідентифікація об'єктів: Однією з основних функцій систем виявлення військової техніки є раннє виявлення та ідентифікація потенційно небезпечних об'єктів на полі бою або на військових об'єктах;
- важливою складовою військових операцій є швидкість та точність виявлення, що відіграють ключову роль у забезпеченні безпеки. Системи повинні мати можливість оперативно реагувати на потенційні загрози та забезпечувати високу точність у розпізнаванні, щоб уникнути помилкових ситуацій та мінімізувати ризик помилок;
- адаптація до різних умов навколишнього середовища є ключовим аспектом виявлення військової техніки. Це може включати погану видимість через погодні умови або особливості рельєфу місцевості. Системи комп'ютерного зору повинні працювати ефективно в різних умовах освітлення, забезпечуючи надійне виявлення об'єктів;
- з метою забезпечення більш широкого охоплення та підвищення ефективності систем виявлення військової техніки можлива їх інтеграція з іншими військовими технологіями, такими як радари, тепловізори та системи GPS. Це дозволяє створювати комплексні системи моніторингу та управління, що забезпечують повне покриття та контроль території.

Виявлення військових об'єктів є складним та водночас чутливим завданням, оскільки воно враховує безліч факторів і тонкощів, таких як локалізація, розпізнавання і класифікація об'єктів. При розробці систем виявлення військових об'єктів необхідно враховувати всі ці чинники, оскільки якість результатів розпізнавання має вирішальне значення для прийняття оперативних рішень. Використання новітніх технологій, таких як методи штучного інтелекту, машинне навчання та комп'ютерний зір, в останні десятиліття значно полегшило цей процес. Технології штучного інтелекту, зокрема комп'ютерний зір, дозволяють комп'ютерам точно виявляти, розпізнавати та аналізувати об'єкти. Один із популярних підходів до виявлення об'єктів у реальному часі - це YOLO, який використовується дослідниками в останні кілька років.

Визначення військових об'єктів на відеопотоці з використанням безпілотних літальних апаратів (БПЛА) у реальному часі потребує уваги до різних проблем і викликів, що можуть виникнути під час розробки та впровадження відповідних систем.

Одна з головних складнощів - це обмеження обчислювальної потужності, оскільки обробка відеопотоку в реальному часі потребує великих ресурсів, що може стати проблемою для БПЛА з обмеженими обчислювальними можливостями.

Друга складність – це оптимізація алгоритмів обробки, яка передбачає розробку та застосування оптимізованих методів для мінімізації часу відповіді та підвищення продуктивності.

Щодо виявлення та класифікації об'єктів, тут важливо враховувати різноманітність зовнішніх умов, таких як умови освітлення та погодні умови, що можуть ускладнити цей процес. Також існує проблема можливості маскування військових об'єктів, що потребує додаткового уваги при їх виявленні на відеопотоці.

Останній виклик – це проблеми з точністю та надійністю виявлення, а також обробці великих обсягів даних. Це включає в себе мінімізацію помилкових спрацьовувань та забезпечення надійної роботи алгоритмів у різних умовах

експлуатації, а також управління передачею та зберіганням великих обсягів даних.

Для вирішення цих проблем потрібно використовувати передові технології комп'ютерного зору, а також ретельно аналізувати дані та розробляти ефективні алгоритми визначення об'єктів на відеопотоці з використанням БПЛА.

PyTorch – це бібліотека машинного навчання з відкритим вихідним кодом для Python, розроблена та підтримувана компанією Facebook. Вона надає гнучкі інструменти для створення та навчання глибинних нейронних мереж, також є такі функції, як зручний інтерфейс для визначення, навчання та застосування нейронних мереж, підтримує динамічні обчислення, що полегшує розробку моделей зі змінною довжиною та формою даних. Бібліотека також активно використовується для досліджень у галузі глибокого навчання та має широку спільноту розробників та дослідників.

Існує кілька причин, чому використання PyTorch в даний час краще, ніж TensorFlow. По-перше, PyTorch має більш простий та інтуїтивно зрозумілий інтерфейс програмування, що робить процес розробки та експериментування з моделями глибокого навчання більш приємним та ефективним для дослідників та розробників. По-друге, PyTorch має більш гнучку та динамічну систему обчислень графів, що робить реалізацію складних моделей простіше та зручніше. Крім того, у PyTorch велика спільнота користувачів та активний внесок у розвиток відкритих вихідних кодів, що сприяє швидкому вирішенню проблем та розширенню функціональності бібліотеки. Також варто відзначити, що PyTorch активно використовується у сфері досліджень та академічному середовищі, де швидкість прототипування та можливість зосередитись на ідеях та концепціях є особливо важливими. Ці фактори сприяють зростанню популярності та застосування PyTorch у різних галузях, включаючи дослідження, розробку продуктів та вирішення завдань штучного інтелекту.

PyTorch широко використовується для реалізації алгоритмів комп'ютерного зору, таких як YOLO, адже він забезпечує гнучкість та ефективність у реалізації

та навчанні моделей, також дозволяючи легко інтегрувати їх у додатки та продукти.

Алгоритм YOLO (You Only Look Once) представляє собою новаторський підхід до виявлення об'єктів, що дозволяє проводити цю операцію та класифікацію в реальному часі з високою швидкістю. Він аналізує зображення лише один раз, що робить його швидким та ефективним для використання у ситуаціях, де важлива швидкість обробки даних. У контексті виявлення військової техніки YOLO може бути ефективним інструментом для оперативного та точного виявлення об'єктів на зображеннях, що містять військову техніку, що дозволяє оперативно реагувати на загрози та вживати відповідні заходи для забезпечення безпеки та захисту.

У порівнянні з традиційними алгоритмами розпізнавання об'єктів, які мають кілька кроків для пошуку об'єктів, YOLO має лише один крок для обробки всього зображення, що робить роботу швидшою та ефективнішою. Крім того, алгоритм YOLO використовує поля прив'язки для підвищення точності виявлення та обробки об'єктів різних розмірів та пропорцій.

Починаючи з появи, YOLO пройшов кілька еволюційних стадій, кожна з яких вдосконалювала попередню за швидкістю, точністю та функціональністю.

YOLOv8, який представляє останню версію алгоритму, демонструє найсучасніші результати на різних наборах тестових даних. Цей алгоритм широко використовується у різних застосунках, таких як автономне керування, спостереження та робототехніка. Проте, при виборі версії YOLO важливо враховувати кілька нюансів, зокрема швидкість обробки, особливо на БПЛА, через обмеженість обчислювальних потужностей GPU, який буде використовуватися на дронах.

Бібліотека OpenCV (Open Source Computer Vision Library) – це відкритий інструмент з обробки зображень та відео, який має широкий спектр функцій. Вона є ключовим інструментом у сфері комп'ютерного зору і використовується для різних завдань, включаючи виявлення об'єктів, розпізнавання облич, сегментацію та інше.

OpenCV часто використовується разом з алгоритмами глибокого навчання, такими як YOLO (You Only Look Once) та PyTorch, для виявлення та класифікації об'єктів на зображеннях та відео. Інтеграція з YOLO та PyTorch надає потужні інструменти для обробки зображень та відеопотоків.

Однією з переваг OpenCV є його підтримка різних форматів зображень та відео, а також можливість роботи з живими відеопотоками. OpenCV має зручне API для завантаження, попередньої обробки та аналізу зображень і відео, що робить його ідеальним для створення систем комп'ютерного зору в реальному часі.

Крім цього, OpenCV має функції для роботи з різними алгоритмами обробки зображень, такими як фільтрація, сегментація та інші, що дозволяє ефективно застосовувати методи глибокого навчання разом з класичними алгоритмами для розв'язання складних завдань у сфері комп'ютерного зору [1].

Таке поєднання OpenCV, YOLO та PyTorch створює потужний інструментарій для розробки та впровадження систем комп'ютерного зору з високою ефективністю та точністю. Ця комбінація дозволяє вирішувати широкий спектр завдань у сфері обробки зображень та відео і знаходить своє застосування у різних галузях, включаючи медицину, безпеку, автомобільну промисловість і багато інших.

1.2. Використання безпілотних літальних апаратів (БПЛА) у військовій сфері

Використання безпілотних літальних апаратів (БПЛА) в різних галузях приносить численні переваги, такі як швидкість реагування, ефективність та підвищення рівня безпеки. Однак важливо враховувати аспекти конфіденційності, безпеки передачі даних та викликів, пов'язаних із нормативно-правовими питаннями використання БПЛА у різних ситуаціях.

У військовій сфері БПЛА використовуються для різноманітних цілей, таких як аерофотозйомка, створення докладних карт та 3D-моделей місцевості, аналіз терену та планування військових операцій. Вони також забезпечують постійне

стеження за діяльністю противника, виявлення руху військових одиниць та об'єктів, надаючи важливу розвідувальну інформацію. БПЛА можуть бути оснащені для проведення повітряних атак на ворожі цілі, забезпечуючи точне та ефективне використання огневої сили. Крім того, вони використовуються для надання технічної підтримки військам, передачі живої інформації, допомоги в розташуванні ворожих сил та навігації під час бойових операцій. БПЛА також можуть використовуватися для оперативного попередження, спостерігаючи за місцевістю та виявляючи можливі загрози, а також для маскування своїх дій та розвідувальних операцій [2].

У військовій сфері методи комп'ютерного зору відіграють важливу роль у виявленні та розпізнаванні технічних об'єктів. Вони дозволяють автоматизований аналіз зображень та відеопотоків, що є критично важливим у сучасних військових конфліктах та операціях. Застосування методів комп'ютерного зору включає виявлення різних видів техніки, таких як військові машини, танки, авіація та морська техніка, що дозволяє в реальному часі відстежувати переміщення ворожих сил та вживати оперативних заходів для захисту власних позицій. Методи комп'ютерного зору використовують різні алгоритми обробки зображень і машинного навчання, зокрема глибоке навчання, що показує високу точність і надійність у розпізнаванні об'єктів на зображеннях.

Важливо зазначити, що використання БПЛА у військовій сфері породжує етичні питання, пов'язані з збереженням цивільних прав, контролем за автоматизованими системами та уникненням непередбачених наслідків. Необхідно враховувати нормативно-правові аспекти, такі як міжнародне гуманітарне право, для регулювання використання таких технологій у воєнних конфліктах.

1.3 Постановка задачі

Самою суттю нашого дослідження – є виявлення найкращої версії YOLO з трьох для пошуку та відстеження технічних об’єктів з БПЛА.

Розкладемо дану задачу на пункти, які необхідно буде виконати під час магістерського включають:

– зібрати теоретичний матеріал щодо теми та галузі дослідження – машинного навчання;
– знайти та підготувати датасет, на якому будуть навчатись моделі;
– привести основні відомості щодо алгоритмів YOLO;
– навчити відстеженню технічних об’єктів три різні версії YOLO, а саме YOLOV7, YOLOV7-tiny, YOLO8n на попередньо обраному датасеті;
– порівняти три отриманні нейронні мережі на однакових даних з метою пошуку найкращої для нашого типу задач. Результати повинні бути представлені у графічному вигляді;
– обґрунтувати висновки, що були зроблені з результату виконаного магістерського дослідження.

Після того, як нами було сформульовано технічне завдання для магістерського дослідження, наступним кроком є вибір інструментів та методів для його проведення, а також підготовка до самого дослідження.

1.4 Засоби, що буде використано для проведення дослідження

Проведення даного магістерського дослідження буде виконуватись завдяки мові програмування Python.

Сам код буде написано у середовищі розробки Visual Studio Code.

Також для підготовки датасету буде використано LabelImg - інструмент для розмітки об'єктів на зображеннях. Він дозволяє користувачам створювати розмітку для навчання нейронних мереж для виявлення об'єктів. Цей інструмент особливо корисний у комп'ютерному зору, коли потрібно підготувати дані для навчання моделей на основі глибокого навчання, таких як YOLO.

У ході виконання дослідження буде використано декілька бібліотек, що будуть описані нижче.

PyTorch – це фреймворк глибокого навчання з відкритим вихідним кодом, розроблений та підтримуваний компанією Facebook. Він надає гнучкі інструменти

для створення та навчання глибоких нейронних мереж.У магістерському дослідженні буде використатися для створення, компіляції та навчання моделі.

Matplotlib – це потужна бібліотека для створення графіків, діаграм, візуалізацій та ілюстрацій у Python. У контексті магістерського дослідження, цей інструмент буде використаний для зроблення візуальних представлень результатів розроблених систем та аналізу отриманих даних. Matplotlib дозволяє створювати різноманітні типи графіків і діаграм, що допомагає краще зрозуміти отримані результати та відобразити їх у зручній формі для подальшого аналізу і використання.

Pandas – це високорівнева бібліотека для аналізу даних у Python. У магістерському дослідженні вона буде використана для проведення аналізу та обробки даних, що допоможе в зробленні висновків та відображенні результатів дослідження.

NumPy – це open-source модуль, який забезпечує загальні математичні та числові операції у вигляді пре-компільованих, швидких функцій. У магістерському дослідженні він буде використаний для спрощення процесу розробки та виконання різноманітних обчислень, що допоможе в ефективному аналізі отриманих даних.

OpenCV (Open Source Computer Vision Library) – це бібліотека комп'ютерного зору з відкритим вихідним кодом, розроблена для обробки зображень та виконання завдань комп'ютерного зору в реальному часі.

2 МЕТОДИ ГЛИБИННОГО НАВЧАННЯ ДЛЯ ВИЗНАЧЕННЯ ПОЛОЖЕННЯ ТЕХНІЧНИХ ОБ’ЄКТІВ

2.1 Загальний опис методів глибинного навчання

Мета детектора об'єктів полягає в визначенні присутності технічного об'єкта або об'єктів на зображенні, повертаючи відповідний клас і місцезнаходження цих об'єктів. Існують дві основні підкатегорії виявлення об'єктів: двоетапні та одноетапні методи, що можна побачити на рисунку 2.1. У двоетапних методах на першому етапі обираються численні пропозиції, а на другому етапі виконується прогнозування регіонів, запропонованих на першому етапі. Приклади таких методів включають Fast R-CNN і Fast R-CNN, які, маючи високу точність, відрізняються низькою обчислювальною ефективністю. Одноетапні методи перетворюють завдання на задачу регресії, уникнувши потреби в початковому етапі відбору регіонів-кандидатів. Це дозволяє досягти вибору кандидата та прогнозування за один прохід. Хоча такі методи менш вимогливі до обчислень і забезпечують вищу частоту кадрів і швидкість виявлення, загальна їхня точність зазвичай нижча, ніж у двоетапних методів.

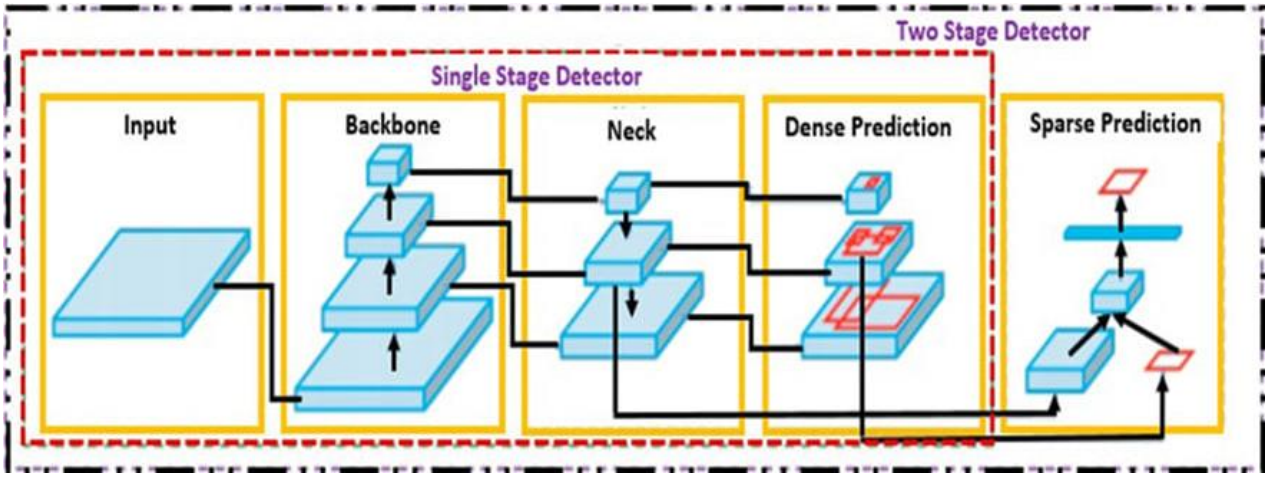

Рисунок 2.1 – Опис детекторів об’єктів (за даними [3])

Основна ідея, запропонована YOLO-v1, полягала у розділенні зображення на сітку розміром s×s і використанні цих осередків для виявлення об'єктів: якщо центр об'єкта потрапляв у комірку сітки, то ця комірка відповідала за виявлення цього об'єкта, спрощуючи процес для інших комірок[4].

З моменту створення у 2016 році до сьогоднішнього дня YOLO швидко розвивається. Хоча оригінальний автор припинив активну роботу над YOLO після версії YOLO-v3, ефективність та потенціал цієї уніфікованої концепції продовжили розширюватися завдяки внеску декількох інших авторів, останнім з яких став YOLO-v8. На рисунку 2.2 показана хронологія еволюції YOLO.

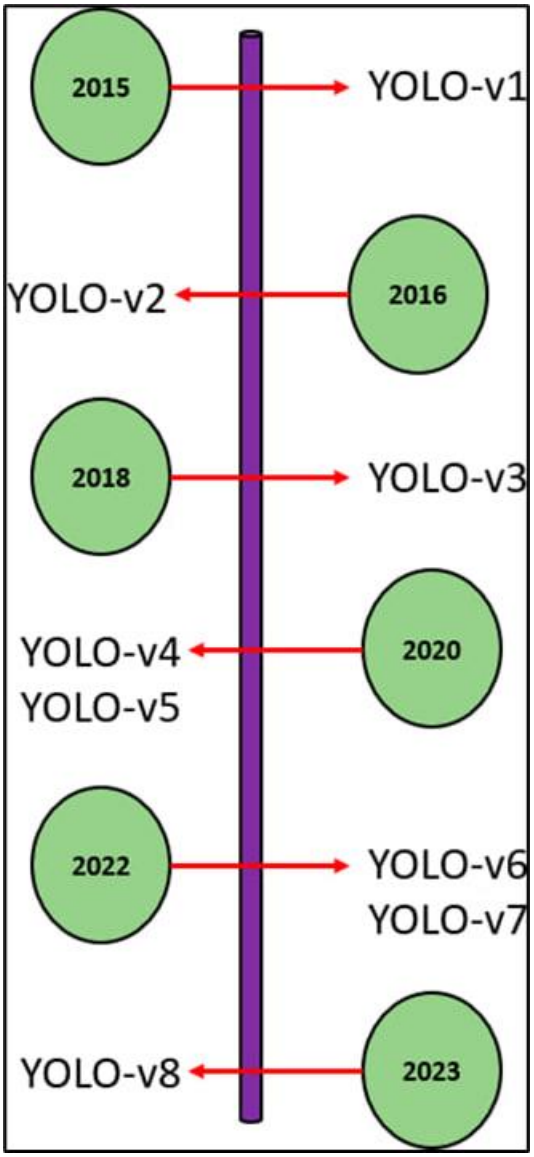

Рисунок 2.2 – Хронологія еволюції YOLO (за даними [5])

Зростання популярності YOLO можна пояснити двома ключовими факторами. По-перше, архітектурна конструкція варіантів YOLO, яка поєднує в собі функції виявлення та класифікації на одному етапі, робить їх обчислювально легкими порівняно з іншими детекторами. Однак саме ця архітектурна композиція не є єдиним чинником, що сприяв популярності YOLO, оскільки інші одноетапні

детектори, наприклад, MobileNets, також пропонують подібні можливості. Другий фактор - це доступність, яка стала значною по мірі розвитку версій YOLO, і особливо видна в YOLO-v5, що стало поворотним моментом в цьому процесі [6].

Існує також метод L2-нормалізації, що відноситься до методів глибокого навчання, який може бути використаний для підвищення точності та загальної продуктивності при розпізнаванні поз відео. У статті розглядаються методи визначення та порівняння поз тіла на потоковому відео. Досліджено бібліотеки OpenPose, PoseNet та BlazePose для їх придатності до розпізнавання та відстеження частин тіла та рухів у реальному часі. Найбільшу точність та продуктивність показало поєднання BlazePose та методу зважених відстаней, що забезпечує високу точність та надійність у різних складних сценаріях [7].

Зокрема, можливо використовувати згорткові нейронні мережі (CNN) для задач визначення вектора напрями голови людини. Ці моделі показали високу точність та незалежність від умов освітлення та часткового закриття обличчя [8]. У дослідженні було порівняно два способи визначення відчуття болі: визначення відчуття болю за наявністю сильних негативних емоцій та використання для тренування нейронної мережі датасету, з зображеннями людей, поділеними на класи «боляче» - «не боляче». Експериментально визначено, що спеціалізований набір даних краще справляється з поставленою задачею, не дивлячись на те, що має достатньо малу кількість зображень. Кількість епох навчання позитивно впливає на точність нейронної мережі, а збільшення швидкості навчання – негативно. Доведено, що відчуття болю у момент часу може досить точно бути визначено за допомогою відеоспостереження [9].

Також існує дослідження, яке представляє метод розпізнавання вправ на основі 3D-координат суглобів, використовуючи ARKit та BlazePose. Метод показав високу точність та стійкість до помилок даних, що робить його ефективним для застосування у фітнесі та реабілітації. За даними дослідження можна зробити висновок, що використання сучасних технологій відстеження рухів дозволяє забезпечити точний аналіз та контроль за виконанням вправ без необхідності у спеціалізованому обладнанні [10].

2.2 Підготовка датасету для тренування YOLO

Для успішного навчання нейронної мережі задля виконання завдання виявлення об'єктів, необхідно мати якісний і різноманітний набір даних. В рамках цього дослідження було проведено аналіз доступних ресурсів в інтернеті з метою збирання фотографій для створення датасету, в якому буде предоставлено 10 класів техніки, а саме це: BMP-BMD, BTR, KAMAZ, MTLB, RLS, RSZO, SAU, TANK, URAL, ZRK.

В результаті аналізу було знайдено набір із 9279 фотографій, що містять об'єкти, що підлягають подальшому виявленню. Дані фотографії були зібрані з різних джерел та являли собою зображення різних розмірів та якостей.

Для забезпечення ефективного навчання нейронної мережі було прийнято рішення про необхідність розмітки кожної фотографії, визначаючи на них об'єкти інтересу та їх місцезнаходження. Для цього був використаний інструмент розмітки об'єктів LabelImg.

За допомогою LabelImg було проведено розміщення 9279 фотографій, на яких було знайдено об'єкти інтересу. Кожному об'єкту була присвоєна відповідна позначка класу, а також зазначені координати прямокутної рамки, що обрамляє об'єкт на зображенні, на рисунку 2.3 можна побачити вигляд цього процесу.

Рисунок 2.3 – Екран роботи LabelImg (рисунок створено самостійно)

Отриманий датасет був розділений на навчальну і тестову вибірки у співвідношенні 20% до 80%, що склало 1855 фотографій у навчальній вибірці і 7424 фотографії в тестовій вибірці. Такий поділ дозволяє забезпечити баланс між обсягом даних для навчання та перевірки, а також оцінити узагальнюючу здатність навченої моделі на нових даних.

Підготовлений датасет є ключовим компонентом для успішного навчання нейронної мережі YOLO для задачі виявлення об'єктів на зображеннях. Він забезпечує необхідну інформацію для навчання моделі та оцінки її продуктивності, що дозволить подальшому дослідженню провести аналіз та отримати результати, що відповідають поставленим цілям та завданням дослідження.

2.3 Вибір моделей YOLO

У цьому розділі ми розглянемо вибрані моделі для навчання на нашому датасеті.

YOLO-моделі вимагають значних обчислювальних ресурсів для обробки зображень реального часу. На БПЛА існує обмеженість потужністю заліза, тому вибір легковагої версії YOLO може бути кращим.

Необхідно оптимізувати параметри моделі та розмір зображення для балансу між точністю та обчислювальною складністю.

Менший розмір зображення дозволяє зменшити обчислювальне навантаження. Однак це може вплинути на точність виявлення об'єктів, тому необхідно знайти баланс.

БПЛА працюють у реальному часі, тому важливо забезпечити високу швидкість обробки. YOLOv7-tiny і YOLOv8nano мають хорошу продуктивність при невеликому обчислювальному навантаженні.

YOLOv7 – одна з останніх версій архітектури You Only Look Once (YOLO). Вона поєднує у собі високу точність виявлення об'єктів та відносно невелику обчислювальну складність. Це важливо для нашого завдання, оскільки потужність заліза обмежена.

YOLOv7 має хорошу продуктивність і здатність виявляти об'єкти різних розмірів.

YOLOv7-tiny – це спрощена версія YOLOv7. Вона має меншу кількість шарів та параметрів, що робить її більш легковагою.

YOLOv7-tiny підходить для завдань, де потрібна висока швидкість обробки, але при цьому точність виявлення залишається на прийнятному рівні.

YOLOv8nano – це ще більш легка версія архітектури YOLO. Вона оптимізована до роботи з пристроями з обмеженими обчислювальними ресурсами.

YOLOv8nano може бути гарним вибором для нашого завдання, враховуючи обмежену потужність заліза.

Можна зробити висновок, що вибрані моделі YOLOV7, YOLOV7-tiny та YOLOV8nano мають гарний баланс між точністю та обчислювальною складністю, що робить їх придатними для виявлення технічних об'єктів з безпілотних літальних апаратів (БПЛА).

2.3.1 YOLOv7

В рамках цього дослідження було проведено навчання моделі YOLOv7 на підготовленому датасеті, що містить 9279 фотографій із розміченими об'єктами інтересу. Дана модель, заснована на архітектурі YOLO (You Only Look Once), була обрана як основна для вирішення завдання виявлення об'єктів у зображеннях, завдяки своїй продуктивності та точності.

Як приклад роботи навченої моделі YOLOV7, представлено зображення із застосуванням виявлення технічних об'єктів. Нижче наведено приклад фотографії з тестового набору даних, на якій успішно виявлені об'єкти інтересу, такі як танки. Отримані результати свідчать про високу точність та надійність навченої нейронної мережі YOLOv7 у виконанні завдання виявлення об'єктів у реальному світі, результат можна побачити на рисунку 2.4.

Рисунок 2.4 – Результат роботи YOLOv7 (рисунок створено самостійно)

Цей приклад є лише одним з багатьох успішних застосувань навченої моделі YOLOV7 в області комп'ютерного зору. Отримані результати підтверджують ефективність обраного підходу до навчання нейронних мереж на вирішення завдання виявлення технічних об'єктів на зображеннях.

2.3.2 YOLOv7-tiny

Аналогічно було проведено навчання моделі YOLOV7-tiny на підготовленому датасеті. YOLOv7-tiny є легковажною версією архітектури YOLO, призначену для більш швидкої обробки зображень при збереженні достатнього рівня точності.

Для наочної демонстрації роботи навченої моделі YOLOV7-tiny представлено зображення із застосуванням виявлення об'єктів. Нижче наведено приклад фотографії з тестового набору даних, на якій успішно виявлено два танки.

Отримані результати підтверджують ефективність навченої нейронної мережі YOLOV7-tiny у виконанні завданні виявлення технічних об'єктів у реальному світі, результат роботи можна побачити на рисунку 2.5.

Рисунок 2.5 – Результат роботи YOLOv7-tiny (рисунок створено самостійно)

Цей приклад ілюструє успішне застосування навченої моделі YOLOV7-tiny у сфері пошуку технічних технічних об’єктів з БПЛА.

2.3.3 YOLOv8nano

У ході дослідження було проведено навчання моделі YOLOV8-nano на основі такого ж підготовленого датасету, який був застосований при навчанні двох попередніх моделей. YOLOv8nano є легковажною версією архітектури YOLOv8, спеціально розробленою для завдань виявлення об'єктів з мінімальними вимогами до обчислювальних ресурсів, що робить її ідеальним рішенням для вбудованих систем і мобільних пристроїв.

Для наочного представлення роботи навченої моделі YOLOV8-nano наведено приклад фотографії з тестового набору даних, приклад роботи можна побачити на рисунку 2.6.

Представлений приклад роботи моделі YOLOv8-nano ілюструє успішність цієї архітектури в галузі комп'ютерного зору. Отримані результати свідчать про перевагу обраного підходу до навчання нейронних мереж для вирішення завдання виявлення об'єктів на зображеннях.

Рисунок 2.6 – Результат роботи YOLOv8nano (рисунок створено самостійно)

Впевнено можна стверджувати, що використання моделі YOLOv8nano дозволяє досягати високої точності в роботі з великим обсягом даних у реальному часі.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Налаштування середовища Anaconda для YOLOv8nano та тренування моделі

Необхідно створити три окремі середовища розробки Anaconda для роботи з YOLO. Нижче наведено покрокову інструкцію для налаштування середовища для YOLOv8nano, створення та активацію роботи з середовищем можна побачити на рисунку 3.1, інші середовища будуть створені за аналогією.

```
(base) C:\Users\dtkachenko34>conda create -n yolov8n_custom python=3.9
(base) C:\Users\dtkachenko34>conda activate yolov8n_custom
```

Рисунок 3.1 – Створення та активація середовища розробки (рисунок створено самостійно)

Для налаштування середовища розробки YOLOV8nano необхідно встановити декілька ключових бібліотек. Були завантажені бібліотеки LabelImg, NumPy, Pandas, Matplotlib та OpenCV за допомогою Anaconda. Також встановлено бібліотеки PyTorch за допомогою pip.

Для підготовки даних було створено дві основні папки: train та val. У папці train знаходяться зображення для навчання моделі, а також відповідні інструкції (мітки) до них. У папці val містяться зображення для валідації моделі та відповідні їм інструкції. Структура папок виглядає так:

а) “train”:
   1) “images”;
   2) “labels”;

б) “val”:
   1) “images”;
   2) “labels”.

Для поділу даних на навчальну та валідаційну вибірки було використано правило 80/20, згідно з яким 80% даних використовувалися для навчання, а 20% – для валідації. Це забезпечує достатню кількість даних для обох процесів,

покращуючи загальну продуктивність моделі.

Для визначення класів зображень, які ідентифікуватимуться, був створений файл data_custom.yaml. Приклад вмісту файлу data_custom.yaml можна побачити на рисунку 3.2. У цьому файлі вказані шляхи до папок train і val, а також перераховані класи об'єктів, які модель буде навчатиметься розпізнавати.

```
train: C:\Users\dtkachenko34\.conda\envs\yolov8s_custom\train
val: C:\Users\dtkachenko34\.conda\envs\yolov8s_custom\val

nc: 10
names: ['BMP-BMD', 'BTR', 'KAMAZ', 'MTLB', 'RLS', 'RSZO', 'SAU', 'TANK', 'URAL', 'ZRK']

```

Рисунок 3.2 – Вміст файлу data_custom.yaml (рисунок створено самостійно)

Отже, згідно файлу data_custom.yaml, будуть використовуватись такі класи, як:

– бойова машина піхоти;
– бронетранспортер;
– «Камаз»;
– багатоцільовий тягач легкоброньований;
– радіолокаційна станція;
– реактивна система залпового вогню;
– самохідна артилерійська установка;
– танк;
– «Урал»;
– зенітний ракетний комплекс.

Далі необхідно розпочати процес навчання, на рисунку 3.3 можна побачити параметри, що були встановлені для навчання

```
(yolov8n_custom) C:\Users\dtkachenko34>yolo task=detect mode=train epochs=100
 data=data_custom.yaml model=yolov8n.pt imgsz=640 batch=16
```

Рисунок 3.3 – Тренування моделі (рисунок створено самостійно)

Для навчання вибраних моделей YOLOV7, YOLOV7-tiny та YOLOV8nano були встановлені наступні параметри:

– розмір зображення: Для навчання було використано зображення розміром 640×640 пікселів. Цей розмір забезпечує хороший баланс між точністю та обчислювальною складністю;
– кількість епох: модель навчалася на 100 епохах, щоб досягти стабільної точності;
– для навчання нейронної мережі було вирішено вибрати розмір батча, що дорівнює 16. Вибір цього розміру обумовлений балансом між ефективністю навчання та використанням ресурсів обчислювальної системи. Розмір батча визначає кількість зразків, що обробляються нейронною мережею за один прохід навчання. В даному випадку, розмір батчу 16 дозволяє враховувати деякі особливості навчання YOLO, такі як необхідність передбачення різних класів об'єктів та їх координат. Крім того, використання батчу дозволяє прискорити процес навчання за рахунок ефективного використання ресурсів, таких як GPU, що особливо важливо для моделей, що потребують великого обсягу даних та обчислювальних потужностей.

Далі був написаний код, який можна побачити на рисунку 3.4, що використовується для тестування моделі yolov8nano, в якому відбувається замір метрик для майбутнього порівняння, для двох інших моделей зроблено аналогічно.

```
def predict_with_model_v8(model, images):
    predictions = []
    for image in images:
        results = model(image)
        boxes = results[0].boxes.data.cpu().numpy()
        predictions.append(boxes)
    return predictions
```

Рисунок 3.4 – Функція, в якій виконується передбачення (рисунок створено самостійно)

Спочатку виконується прогноз для зображення, далі отримуються передбачені рамки, що обмежують, і їх перетворюємо у numpy масив та додаємо прогнози до списку.

Після прогнозу зображень виконуються дві функції, що перетворюють прогнози на формат, зручний для подальшої обробки та перетворює мітки зображень у формат, аналогічний прогнозам. На рисунку 3.5 можна побачити основну функцію, що розраховує три метрики, а саме точність, повнота та середня точність.

```python
def calculate_metrics(predictions, labels, iou_threshold=0.5):
    precision_list = []
    recall_list = []
    ap_list = []

    unique_classes = np.unique(labels[:, 5]) if len(labels) > 0 else []

    for cls in unique_classes:
        cls_preds = predictions[predictions[:, 5] == cls]

        if len(cls_labels) == 0 or len(cls_preds) == 0:
            precision_list.append(0)
            recall_list.append(0)
            ap_list.append(0)
            continue

        cls_preds_sorted = sorted(cls_preds, key=lambda x: x[4], reverse=True)
        tp = np.zeros(len(cls_preds_sorted))
        fp = np.zeros(len(cls_preds_sorted))
        cls_labels_used = np.zeros(len(cls_labels))

        for i, pred in enumerate(cls_preds_sorted):
            ious = np.array([calculate_iou(pred[1:5], label[1:5]) for label in cls_labels])
            max_iou_idx = np.argmax(ious)
            if ious[max_iou_idx] >= iou_threshold and not cls_labels_used[max_iou_idx]:
                tp[i] = 1
                cls_labels_used[max_iou_idx] = 1
            else:
                fp[i] = 1

        cum_tp = np.cumsum(tp)
        cum_fp = np.cumsum(fp)
        precision = cum_tp / (cum_tp + cum_fp + np.finfo(float).eps)
        recall = cum_tp / (len(cls_labels) + np.finfo(float).eps)
        ap = np.sum((recall[1:] - recall[:-1]) * precision[1:])

        precision_list.append(precision[-1] if len(precision) > 0 else 0)
        recall_list.append(recall[-1] if len(recall) > 0 else 0)
        ap_list.append(ap)

    mean_precision = np.mean(precision_list)
    mean_recall = np.mean(recall_list)
    mean_ap = np.mean(ap_list)
```

Рисунок 3.5 – Розрахунок метрик (рисунок створено самостійно)

Ця функція розраховує метрики точності (precision), повноти (recall) та середньої точності (average precision, AP) для передбачених рамок, що обмежують, в порівнянні з істинними мітками. Для кожного унікального класу

вона збирає передбачення та мітки, сортує передбачення за впевненістю, потім обчислює істинно-позитивні (TP) та помилково-позитивні (FP) спрацьовування на основі значення IoU (Intersection over Union). На основі кумулятивних значень TP та FP вона розраховує точність та повноту, а потім обчислює AP для кожного класу. Наприкінці функція повертає середні значення метрики всім класів.

Далі відбувається візуалізація результатів, код цієї функції можна побачити на рисунку 3.6.

```
def visualize_predictions(images, predictions, labels, output_dir):
    os.makedirs(output_dir, exist_ok=True)
    for i, (image, preds, lbls) in enumerate(zip(images, predictions, labels)):
        img = image.copy()
        for box in preds:
            x1, y1, x2, y2, conf, cls = box
            cv2.rectangle(img, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2)
        for l in lbls:
            if len(l) != 5:
                with open("debug_log.txt", "a") as f:
                    f.write(f"Skipping invalid label {l} in image {i}\n")
                continue
            cls, x_center, y_center, width, height = map(float, l)
            x1 = int((x_center - width / 2) * img.shape[1])
            y1 = int((y_center - height / 2) * img.shape[0])
            x2 = int((x_center + width / 2) * img.shape[1])
            y2 = int((y_center + height / 2) * img.shape[0])
            cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 0), 2)
        output_path = os.path.join(output_dir, f"image_{i}.jpg")
        cv2.imwrite(output_path, img)
```

Рисунок 3.6 – Тренування моделі (рисунок створено самостійно)

Дана функція створює директорію для збереження зображень із рамками, якщо вона ще не існує, далі відбувається ітерація за зображеннями, передбаченнями та мітками, створюється копія зображення для малювання рамок, після цього проходить ітерація передбачуваних рамок для поточного зображення, витягуються координати (лівий верхній та правий нижній кути), впевненість та клас передбачуваної рамки, малюється зелена рамка для передбачення на зображенні, далі відбувається ітерація по справжнім міткам для поточного зображення, робиться перевірка на коректність мітки (має бути 5 елементів: клас та координати), пропускаються некоректні мітки, та перетворюються мітки в

координати та клас, розраховуються координати, малюються рамка синього коліру на зображені, далі воно зберігається у вказану директорію. Візуальну роботу функції можна побачити на рисунку 3.7.

Рисунок 3.7 – Візуальне зображення роботи функції (рисунок створено самостійно)

Ця функція дозволяє візуально оцінити якість передбачень моделі, порівнюючи їх із дійсними мітками на зображеннях.

4 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТУ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Проведення аналізу отриманих результатів

У цій роботі були розглянуті три різні моделі сімейства YOLO (You Only Look Once): YOLOV7, YOLOV8nano та YOLOV7tiny. Всі три моделі були навчені на тому самому наборі даних, що складається з 7424 зображень для навчання і 1855 зображень для валідації, що відповідає правилу 80 на 20. Навчання проводилося з використанням 100 епох і розміру батча 16. Метою даного розділу є порівняльний аналіз цих моделей за різними ключовими метриками, такими як середня точність (mAP), швидкість обробки зображень (FPS), розмір моделі, час навчання, час інференсу та споживання пам'яті (GPU Memory Usage).

Для порівняння моделей було обрано такі метрики:

– середня точність (mAP): середня точність у всіх класах;
– швидкість обробки зображень (FPS): кількість кадрів, що обробляються моделлю за секунду;
– розмір моделі: кількість параметрів моделі;
– час навчання: загальний час, витрачений навчання моделі;
– час інференсу: час, витрачений для обробки одного зображення;
– споживання пам'яті (GPU Memory Usage): споживання відеопам'яті під час інференсу.

Нижче будуть приведені формули, за якими розраховувались деякі з метрик, що були використані.

Середня точність(mAP) розраховується як середнє значення точності за всіма класами за різних порогів IoU (Intersection over Union), її було розраховано за допомогою формули 4.1:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{4.1}$$

де N – кількість класів,

AP_i – Average Precision для і-го класса.

Precision (точність), було розраховано за допомогою формули 4.2:

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

де TP - кількість справжніх позитивних,

FP - кількість хибних позитивних.

IoU використовується для визначення збігу передбаченої та істинної рамки, що обмежує, було розраховано за допомогою формули 4.3:

$$IoU = \frac{\text{Area of Overlap}}{Area\ of\ Union} \tag{4.3}$$

де Area of Overlap - площа перетину передбачуваної та істинної рамки, що обмежує,

Area of of Union - площа об'єднання цих рамок.

Нижче можна побачити таблицю 4.1 з порівняннями трьох моделей за попередньо обраними метриками, тести та навчання були проведені за однакових умов для кожної моделі.

Таблиця 4.1 – Порівняння трьох моделей YOLO (таблиця виконана самостійно)

Параметр	YOLOv7	YOLOv8nano	YOLOv7tiny
mAP	0.71	0.6	0.64
FPS	25	50	45
Розмір моделі	60M	7M	6M
Час інференсу	17 ms	8 ms	12 ms
Споживання пам'яті	10GB	4GB	5GB

Середня точність (mAP): YOLOv7 показала найвищу точність серед усіх трьох моделей завдяки більш складній архітектурі та більшій кількості параметрів. Це робить її кращою для завдань, де важливою є висока точність детекції. У той же час YOLOV8nano продемонструвала дещо меншу точність, що пояснюється спрощеною архітектурою, оптимізованою для роботи на мобільних та вбудованих пристроях. YOLOV7tiny, будучи полегшеним варіантом YOLOV7, показала точність, що знаходиться між YOLOV7 і YOLOV8nano, що робить її хорошим компромісом між точністю і швидкістю.

Швидкість обробки зображень (FPS): YOLOV8nano демонструє найвищу швидкість обробки зображень, що робить її найбільш придатною для завдань, де критично важлива швидкість інференсу. YOLOV7tiny також володіє високою швидкістю інференсу, перевищуючи YOLOV7, але поступаючись YOLOV8nano. YOLOV7, через складнішу архітектуру, має помірну швидкість інференсу, що може бути недоліком у завданнях реального часу.

Час інференсу: YOLOV8nano має найменший час інференсу, що робить її найшвидшою моделлю для обробки зображень. YOLOV7tiny має менший час інференсу в порівнянні з YOLOV7, але більший у порівнянні з YOLOV8nano. YOLOv7 вимагає найбільшого часу на інференс через складність архітектури.

Споживання пам'яті (GPU Memory Usage): YOLOv7 вимагає найбільшого обсягу пам'яті через велику кількість параметрів та складну архітектуру. YOLOV8nano має найнижче споживання пам'яті завдяки своїй компактній архітектурі. YOLOV7tiny споживає помірну кількість пам'яті, перебуваючи між YOLOV7 і YOLOV8nano.

Отже, кожна з розглянутих моделей має свої переваги та недоліки, і вибір конкретної моделі залежить від специфіки завдання. Якщо пріоритетом є висока точність, перевагу слід віддати YOLOV7. У випадку, коли важлива висока швидкість та компактність моделі, найкращим вибором буде YOLOV8nano. YOLOV7tiny є збалансованим варіантом, що підходить для завдань, що потребують компромісу між точністю і швидкістю.

ВИСНОВКИ

В результаті виконання даного магістерського дослідження було досягнуто усі поставлені цілі та виконано дослідницькі завдання, спрямовані на порівняння моделей YOLOv7, YOLOv8nano та YOLOv7tiny для задач детекції об’єктів.

Порівнюючи всі три моделі, можна зробити кілька ключових висновків. По-перше, модель YOLOv7 продемонструвала найвищу середню точність (mAP) серед усіх трьох моделей. Це обумовлено більш складною архітектурою та більшим обсягом параметрів, що робить її найбільш підходящою для задач, де важлива висока точність детекції, таких як системи безпеки та медичної діагностики.

По-друге, модель YOLOv8nano показала найвищу швидкість обробки зображень (FPS), що робить її оптимальною для використання в реальному часі, особливо у мобільних та вбудованих системах. Завдяки своїй компактній архітектурі та меншій кількості параметрів, вона забезпечує високу продуктивність при значно меншому споживанні ресурсів.

Модель YOLOv7tiny є збалансованим варіантом між YOLOv7 та YOLOv8nano. Вона демонструє середні показники як за точністю, так і за швидкістю, що робить її відповідною для задач, де необхідний компроміс між цими двома характеристиками. Зокрема, YOLOv7tiny може бути ефективно використана в БПЛА, де обчислювальними ресурсами є обмеженими.

Щодо споживання пам’яті, YOLOv7 потребує найбільшого обсягу GPU пам'яті, тоді як YOLOv8nano, завдяки своїй компактній архітектурі, споживає найменше. YOLOv7tiny споживає помірну кількість пам’яті, що робить її більш доступною для використання на пристроях з обмеженими ресурсами.

Отже, вибір конкретної моделі залежить від специфіки завдання та доступних ресурсів. У нашому випадку, YOLOv7tiny стане найкращим варіантом для завдань, що потребують компромісу між точністю та швидкістю.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Learning OpenCV3. - Келєр А., Бредски Г. - ISBN 978-5-97060-471-7. - 978-5-97060-471-7.pdf [Електронний ресурс]. - Режим доступу: https://dmkpress.com/files/PDF/978-5-97060-471-7.pdf. (Дата звернення 25.03.2024)

2. Настанова з тактичної розвідки/ Командування СВ ЗС України. – Київ: МОУ, 2017. – 127 с.

3. Redmon, J., Farhadi, A. YOLOv3: An Incremental Improvement // arXiv:1804.02767, 2018.

4. YOLO – You only look once, real time object detection explained // URL: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006 (дата звернення: 25.03.2024)

5. Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. У матеріалах конференції IEEE/CVF з комп'ютерного зору та розпізнавання образів, с. 16123–16133, 2022.

6. MS-YOLOv7:YOLOv7 Based on Multi-Scale for Object Detection on UAV Aerial Photography // URL: https://www.mdpi.com/2504-446X/7/3/188 (дата звернення: 25.03.2024).

7. N. V. Bilous, I. A. Ahekian, and V. V. Kaluhin, “DETERMINATION AND COMPARISON METHODS OF BODY POSITIONS ON STREAM VIDEO,” RIC, no. 2, p. 52, Jun. 2023, doi: 10.15588/1607-3274-2023-2-6.

8. N. V. Bilous and A. O. Rakova, “REFERENCE POINTS METHOD FOR HUMAN HEAD MOVEMENTS TRACKING,” Радиоэлектроника, информатика, управление, vol. 2020, no. 3, pp. 121–129, 2020, doi: 10.15588/1607-3274-2020-3-11.

9. 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.

10. N. Bilous, O. Svidin, I. Ahekian, and V. Malko, “A skeleton-based method for exercise recognition based on 3D coordinates of human joints,” IJ-AI, vol. 13, no. 2, p. 1805, Jun. 2024, doi: 10.11591/ijai.v13.i2.pp1805-1816.