Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Одноетапне розпізнавання військових об'єктів на аерофотознімках

(тема)

Виконав:
студент 2 курсу, групи СШІм-22-1
Любименко Р.С.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту

(повна назва спеціалізації)
Керівник проф. Кулішова Н.Є.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Любименку Роману Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Одноетапне розпізнавання військових об'єктів на аерофотознімках

затверджена наказом університету від 1 квітня 20 24 р. № 260Ст

2. Термін подання студентом роботи до екзаменаційної комісії 7 червня 20 24 р.

3. Вихідні дані до роботи науково-методична література, дані інтернет-джерел, документація мови програмування Python, документація Ultralytics та Flask, набір зображень військової техніки, середовище розробки PyCharm.

4. Перелік питань, що потрібно опрацювати в роботі

1) оглянути та проаналізувати інформаційні джерела в області комп'ютерного зору та військової справи;

2) знайти оптимальні підходи до вирішення задачі розпізнавання військових об'єктів за допомогою методів комп'ютерного зору;

3) визначити набір даних, який міститиме аерофотознімки військових об'єктів для постановки експерименту;

4) розробити систему розпізнавання військових об'єктів на аерофотознімках;

5) провести аналіз результатів експерименту.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.04.2024	виконано
2	Аналіз поставленого завдання	05.04.2024	виконано
3	Огляд інформаційних джерел із заданої тематики	14.04.2024	виконано
4	Огляд існуючих рішень та формулювання вимог	19.04.2024	виконано
5	Пошук та аналіз набору даних	23.04.2024	виконано
6	Огляд засобів розробки	26.04.2024	виконано
7	Навчання моделей	28.04.2024	виконано
8	Розробка та тестування програмного продукту	01.05.2024	виконано
9	Написання пояснювальної записки	14.05.2024	виконано
10	Перевірка на академічний плагіат	15.05.2024	виконано
11	Нормоконтроль	16.05.2024	виконано
12	Підготовка презентації та доповіді	31.05.2024	виконано
13	Попередній захист	03.06.2024	виконано
14	Рецензування	04.06.2024	виконано
15	Захист перед екзаменаційною комісією	07.06.2024	виконано

Дата видачі завдання 1 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Кулішова Н.Є.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 74 с., 47 рис., 3 табл., 1 дод., 34 джерела.

ВИЯВЛЕННЯ ОБ'ЄКТІВ, ГЛИБОКЕ НАВЧАННЯ, КОМП'ЮТЕРНИЙ ЗІР, ОДНОЕТАПНИЙ АЛГОРИТМ, YOLO.

Об'єкт дослідження – процес одноетапної ідентифікації об'єктів на зображеннях.

Предмет дослідження – система розпізнавання військових об'єктів на аерофотознімках засобами комп'ютерного зору.

Мета роботи – підвищення точності розпізнавання військових об'єктів на аерофотознімках на базі одноетапного алгоритму ідентифікації.

Методи дослідження – методи глибокого навчання, які базуються на аналізі інформаційних джерел та програмних засобах: Python, PyTorch та Flask.

У результаті роботи проведено теоретичний аналіз підходів до ідентифікації об'єктів на зображеннях. Була обрана та детально проаналізована одноетапна модель розпізнавання YOLO. Було навчено три моделі YOLOv8n за різних оптимізаторів на наборі даних військової техніки та обраний найкращий оптимізатор на основі порівняння метрик якості навчання. Отримана модель була інтегрована в веб-застосунок, який дозволяє ідентифікувати військові об'єкти на аерофотознімках.

ABSTRACT

Master’s thesis contains: 74 pp., 47 fig., 3 tabl., 1 ann., 34 references.

OBJECT DETECTION, DEEP LEARNING, COMPUTER VISION, ONE-STAGE ALGORITHM, YOLO.

The object of research is the process of one-stage identification of objects in images.

The subject of research is a system for recognizing military objects in aerial imagery using computer vision.

The purpose of research is to improve the accuracy of military object recognition in aerial imagery based on a one-stage identification algorithm.

The research methods are deep learning methods based on the analysis of information sources and software tools: Python, PyTorch and Flask.

As a result of the work, a theoretical analysis of approaches to object identification in images was carried out. The one-stage YOLO recognition model was selected and analyzed in detail. Three YOLOv8n models were trained with different optimizers on the military objects dataset and the best optimizer was selected based on a comparison of training quality metrics. The resulting model was integrated into a web application that allows identifying military objects on aerial imagery.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БПЛА – безпілотний літальний апарат;

ШІ – штучний інтелект;

Adam – Adaptive Moment Estimation – метод адаптивної оцінки моментів;

AP – Average Precision – середня точність;

CLI – Command-line interface – інтерфейс командного рядка;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

DFL – Distribution Focal Loss – розподіл фокусних втрат;

FPN – Feature Pyramid Network – мережа піраміди ознак;

GPU – Graphics Processing Unit – графічний процесор;

IoU – Intersection Over Union – перетин над об’єднанням;

mAP – Mean Average Precision – середня точність;

NMS – Non-maximum Suppression – немаксимальне придушення;

R-CNN – Region-based CNN – регіональна згорткова нейронна мережа;

RetinaNet – уніфікована нейронна мережа, що складається з магістральної мережі та двох підмереж;

RMSProp – Root Mean Squared Propagation – алгоритм розповсюдження кореня середнього квадрату градієнта;

SGD – Stochastic Gradient Descent – стохастичний градієнтний спуск;

SPP – Spatial Pyramid Pooling – просторове об’єднання пірамід;

SSD – Single Shot Detector – одноетапний алгоритм розпізнавання об’єктів;

Ultralytics – бібліотека, яка містить YOLOv5 та YOLOv8;

YOLO – You Only Look Once – ви дивитесь лише раз.

ВСТУП

На сьогодні штучний інтелект все частіше використовується для розв’язання прикладних задач в різних галузях людської діяльності. Зокрема методи глибокого навчання та комп’ютерного зору широко застосовуються у сфері оборонно-промислового комплексу для розпізнавання та класифікації об’єктів військового та цивільного призначення.

Використання технології розпізнавання військових об’єктів є ключем до покращення формування обстановки на полі бою, розвідки, спостереження та прийняття рішень командуванням. Сучасні військові операції вже неможливо представити без розвідки повітрям за допомогою безпілотних літальних апаратів або космічних знімків. Вони відіграють чи не найвирішальнішу роль та є важливим фактором для перемоги у війні.

Точне виявлення військових об’єктів допомагає відстежувати ворожі підрозділи та розуміти динаміку дій противника. Своєчасне отримання таких даних дозволяє приймати миттєві рішення, які можуть кардинально змінити обстановку на полі бою, надаючи неабияку перевагу одній із сторін та, за відповідних засобів, вберегти або попередити жертви серед військових та мирного населення.

Технології комп’ютерного зору набули широкого застосування в різних галузях, включаючи відеоспостереження, пілотування безпілотників і аналіз даних військової розвідки, завдяки швидкому зростанню глибокого навчання.

На сучасному щаблі розвитку алгоритмів розпізнавання об’єктів на зображеннях сформувалося дві великі групи двоетапних та одноетапних алгоритмів за характеристиками їх роботи. Різниця між цими групами алгоритмів полягає в тому, що одноетапні прогнозують розташування та об’єкт класу за один прохід, тоді як двоетапні спочатку виділяють області-

кандидати, які є місцями ймовірного розташування об’єктів на зображенні, а потім класифікують ці об’єкти в межах кожної області.

Враховуючи специфіку предметної галузі та в умовах швидкоплинності бойових дій і наявності обчислювальних та апаратних обмежень, метою даної роботи є підвищення точності розпізнавання військових об’єктів на аерофотознімках на базі одноетапного алгоритму ідентифікації.

1 ТЕОРЕТИЧНІ АСПЕКТИ КОМП'ЮТЕРНОГО ЗОРУ

1.1 Теоретичні відомості

Комп'ютерний зір – це підгалузь штучного інтелекту, яка займається обчислювальними методами, що допомагають комп'ютерам розуміти та інтерпретувати вміст візуальних даних, що надходять з камер або датчиків. Це передбачає отримання зображення, його обробку та отримання значущої інформації для вжиття відповідних дій.

З точки зору нейропсихології розпізнавання об'єктів людиною розпочинається з передачі сигналу від ока до мозку, де подана інформація аналізується, виділяються краї об'єктів, які потім з'єднуються разом у більш складне уявлення, яке утворює форму, глибину, величину, колір та інші характеристики об'єкта.

Подібним чином працюють й системи комп'ютерного зору. Розпізнавання починається з визначення країв зображення, які потім об'єднуються та формують об'єкт. Значна відмінність полягає в тому, що комп'ютери розглядають зображення як набір числових значень, тому система комп'ютерного зору потребує конкретного методу інтерпретації кожного пікселя, що складає зображення. Для визначення країв зображення кожному пікселю ставиться у відповідність певне значення і, досліджуючи різницю цих значень між однією областю пікселів та іншою, система може розпізнати краї.

Загальний вигляд системи комп'ютерного зору у порівнянні з зоровою системою людини можна переглянути на рисунку 1.1.

Алгоритми комп'ютерного зору здатні ефективно та точно обробляти великі масиви візуальних даних, перевершуючи людські можливості в таких завданнях, як розпізнавання об'єктів та класифікація зображень. Це дозволило їм знайти широкий спектр застосувань, який включає

робототехніку, автономні транспортні засоби, системи спостереження та ідентифікації, медичну візуалізацію та доповнену реальність.

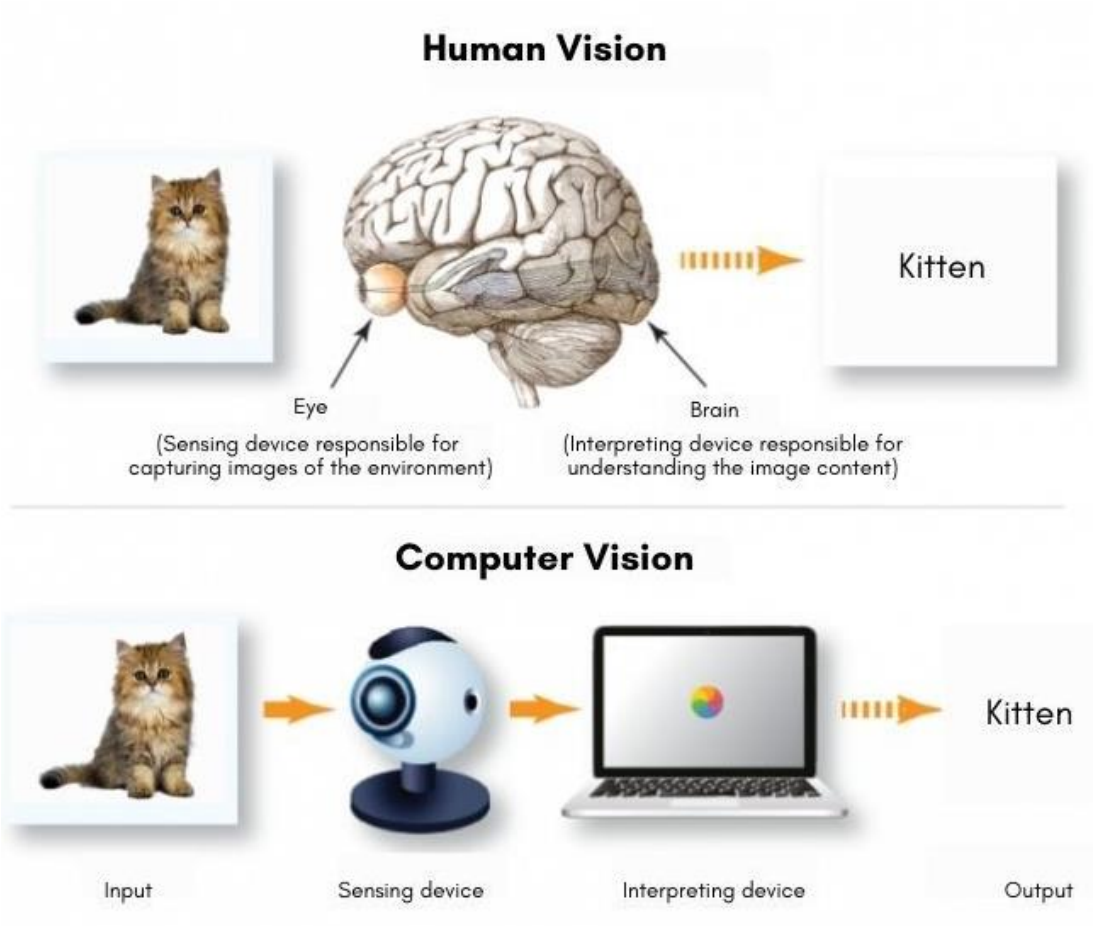


Рисунок 1.1 – Зорова система людини та система комп’ютерного зору

1.2 Історія розвитку

Історія комп’ютерного зору нерозривно пов’язана з віхами розвитку штучного інтелекту. Дана сфера зазнала значної еволюції протягом десятиліть, ставши наразі однією з фундаментальних технологій з численним спектром застосувань [1].

Зачатки комп’ютерного зору сягають 1950-х років, коли дослідники вперше почали досліджувати можливість навчання обчислювальних систем інтерпретації візуальних даних. Першим кроком на цьому шляху стала

створена Ф. Розенблаттом у 1957 році модель перцептрона, яка була примітивною формою штучної нейронної мережі. Вона мала здатність розпізнавати прості закономірності на зображеннях і заклала основи для подальших досягнень у машинному навчанні та комп'ютерному зорі.

Протягом 1960-х і 1970-х років в галузі комп'ютерного зору було досягнуто значних успіхів. З найпомітніших створення загального розв'язувача задач (GPS), який міг розв'язувати геометричні задачі у візуальних сценах. У 1980-х роках з'явилися більш вдосконалені методи та алгоритми обробки та аналізу зображень. Важливою подією стало введення концепції масштабно-просторової теорії Т. Ліндебергом, що надала основу для аналізу зображень на різних масштабах і з різною роздільною здатністю. Це відкрило нові можливості для виявлення ознак та розпізнавання об'єктів.

З розвитком обчислювальних потужностей, у 1990-х роках сфера комп'ютерного зору пережила стрімкий розвиток та зростання. Це десятиліття ознаменувалося розробкою алгоритму для розпізнавання обличь у реальному часі П. Віолою та М. Джонсом у 2001 році.

Галузь комп'ютерного зору пережила новий сплеск у 2000-х, коли з'явилися такі методи машинного навчання, як машини опорних векторів і згорткові нейронні мережі (CNN). У 2012 році було започатковано проект ImageNet Large Scale Visual Recognition Challenge, що значно прискорив розвиток у цій галузі, надаючи стандартний набір даних і структуру оцінки для алгоритмів класифікації зображень.

Це призвело до створення архітектури AlexNet через два роки, що стало одним з найбільших проривів у галузі комп'ютерного зору [2]. Використання глибокого навчання продемонструвало величезну перевагу над традиційними алгоритмами комп'ютерного зору, які розглядають об'єкти як сукупність характеристик форми і кольору.

На сьогодні CNN стала де-факто стандартною обчислювальною одиницею в комп'ютерному зорі. Оптимізовані та легкі моделі штучного інтелекту дозволили розгортати додатки комп'ютерного зору на недорогому

обладнанні та мобільних пристроях, а спеціальні апаратні рішення для ШІ, такі як апаратні прискорювачі глибокого навчання, уможливлюють високоефективну обробку вхідної інформації.

Завдяки прогресу в галузі штучного інтелекту та інноваціям у глибокому навчанні та нейронних мережах ця сфера змогла зробити значний стрибок за останні роки.

В результаті нещодавніх розробок у таких сферах, як штучний інтелект і обчислювальні можливості, сфера комп'ютерного зору стала все більше проникати в повсякденне життя. Очікується, що до 2030 року ринок комп'ютерного зору наблизиться до 41,11 мільярда доларів, а середній річний темп зростання становитиме 16,0% між 2020 і 2030 роками [3].

1.3 Машинне навчання у комп'ютерному зорі

До появи глибокого навчання завдання, які міг виконувати комп'ютерний зір, були дуже обмеженими і вимагали багато ручного кодування та зусиль з боку розробників і людей-операторів. Наприклад, для вирішення задачі розпізнавання обличчя, потрібно було б виконати наступні кроки:

– створення бази даних. Зробити окремі зображення всіх об'єктів для відстеження у певному форматі;

– анотування зображення. Для кожного окремого зображення потрібно визначити кілька ключових параметрів, таких як відстань між очима, ширина перенісся, відстань між верхньою губою і носом та десятки інших вимірів, які визначають унікальні характеристики кожної людини;

– отримання нових зображень. Потрібно отримати нові зображення, чи то з фотографій, чи то з відео контенту і знову виконати процес вимірювання, позначаючи ключові точки на зображеннях. При створенні фотографій також треба враховувати кут, під яким робиться знімок.

Після цієї ручної роботи програма зможе порівняти вимірювання на новому зображенні з тими, що зберігаються в її базі даних, і повідомити, чи відповідає знімок якомусь із профілів, які вона вже відстежувала. Насправді, автоматизація є дуже незначною, більша частина роботи виконувалася вручну і ймовірність помилки все ще була високою.

Машинне навчання забезпечило інший підхід до вирішення проблем комп'ютерного зору. Завдяки машинному навчанню розробникам більше не потрібно вручну кодувати кожне окреме правило. Замість цього визначаються спеціальні функції, які можуть виявляти певні ознаки на зображеннях. Потім використовується такі алгоритми як лінійна регресія, логістична регресія, дерева рішень або машини опорних векторів для виявлення закономірностей, класифікації зображень і виявлення об'єктів на них.

Машинне навчання допомогло вирішити багато проблем, які історично були складними для класичних інструментів та підходів до розробки програмного забезпечення. В порівнянні з класичним підходом, створення гарного алгоритму глибокого навчання у більшості випадків зводиться до збору великої кількості навчальних даних і налаштування таких параметрів, як тип, кількість шарів нейронних мереж, епохи навчання та інших.

Загалом методи машинного навчання поділяються на чотири типи: кероване навчання, некероване навчання, напівкероване навчання та навчання з підкріпленням. Термін глибоке навчання передбачає використання декількох шарів у мережі.

1.3.1 Кероване навчання

У цьому підході до навчання алгоритм отримує знання з попередніх і поточних даних, які позначені мітками. Модель заздалегідь знає правильні відповіді на деяких прикладах, і її завдання полягає в тому, щоб навчитися

відображати зв’язок між вхідними даними та виходами, щоб потім передбачати виходи на основі нових даних.

Кероване навчання передбачає ітеративний процес, під час якого модель вдосконалюється шляхом аналізу прикладів і коригування своїх ваг для зменшення різниці між передбаченими та фактичними відповідями. Після достатнього навчання система буде здатна виявляти помилки і змінювати дії, які вона виконує для прогнозування вихідних даних [4].

1.3.2 Некероване навчання

У даному типі навчання система самостійно намагається вивчати приховані закономірності, присутні у вхідних даних, та на основі цього корегувати свої ваги. Це може бути корисно для вирішення задач класифікації, кластеризації, пониження розмірності даних та інших. Також його використання є доцільним, коли навчальні дані не мають визначених відповідей або міток, щоб використовувати навчання з учителем [4].

1.3.3 Напівкероване навчання

Це тип навчання є поєднанням методів керованого та некерованого навчання. У напівкерованому навчанні частково надаються мітки для даних. Це може бути корисно, коли тільки частина даних є позначеною, і бракує кваліфікованої експертизи для позначення іншої частини набору даних або сам процес є складним та дорогим.

Напівкероване навчання використовується у широкому спектрі задач, включаючи класифікацію, регресію, сегментацію даних та розпізнавання образів [5].

1.3.4 Навчання з підкріпленням

Навчання з підкріпленням – це напрямок машинного навчання у якому присутній агент, що навчається приймати рішення в реальному чи симульованому середовищі з метою отримання зворотного зв’язку від середовища у формі винагороди або штрафу. Завдяки такій взаємодії агент виробляє стратегії, які максимізують його винагороду протягом часу.

Даний тип навчання використовується для навчання роботів, оптимізації виробничих процесів, автоматизації торгівлі на фінансових ринках та під час розробки ігор [6].

1.3.5 Глибоке навчання

Глибокі нейронні мережі – це мережі з кількома шарами нейронів. За сучасними стандартами розмір глибокої нейронної мережі коливається від п’яти до кількох тисяч шарів. Цей тип мереж надає потужні засоби для вивчення закономірностей з великих обсягів даних і показує неймовірні результати в галузі класифікації зображень, розпізнавання мови, машинного перекладу, виявлення об’єктів та в багатьох інших застосуваннях.

1.4 Підходи до розпізнавання об’єктів

Розвиток алгоритмів розпізнавання об’єктів можна розділити на два етапи: традиційні алгоритми виявлення об’єктів та алгоритми виявлення об’єктів на основі глибокого навчання. У свою чергу алгоритми виявлення об’єктів на основі глибокого навчання також поділяються на два основні підходи: одноетапні та двоетапні алгоритми.

Традиційні методи зазвичай базуються на методах ковзного вікна та штучного вилучення ознак, що складаються з трьох основних етапів: пропозиція регіону, вилучення ознак та класифікаційна регресія. Однак ці

методи мають свої недоліки, такі як висока обчислювальна складність, обмежена репрезентативність ознак і складність оптимізації. До них відносяться, наприклад, детектор Віоли-Джонса та детектор пішоходів HOG.

На рисунку 1.2 представлено розвиток технологій розпізнавання об’єктів з 2001 по 2023 рік.

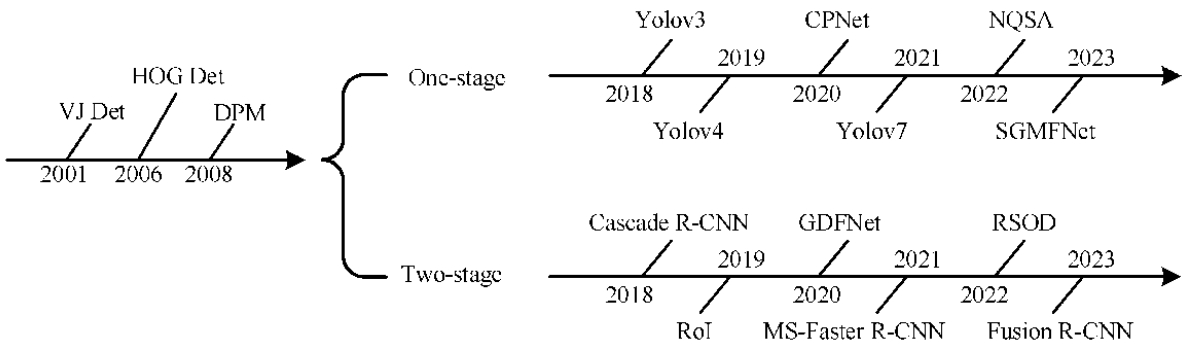


Рисунок 1.2 – Розвиток алгоритмів виявлення об’єктів

Із поширенням згорткових нейронних мереж виявлення об’єктів досить швидко еволюціонувало. Глибокі нейронні мережі в основному почали використовуватися в алгоритмах розпізнавання об’єктів для автоматичного вилучення високорівневих ознак із вхідних зображень і класифікації об’єктів.

Побудовані на цій основі двоетапні алгоритми розпізнавання об’єктів мають високу швидкість, точність і надійність. Як показано на рисунку 1.3, двоетапні алгоритми вирішують проблеми виявлення граничних об’єктів, неточності локалізації та різномасштабність об’єктів. Ці алгоритми генерують області пропозицій на першому етапі, а на другому етапі виконують класифікацію та регресію вмісту в цікавих для нас областях. Хоча такі алгоритми зазвичай демонструють високу точність, швидкість їх роботи може бути низькою. Найпопулярнішими з цих алгоритмів є R-CNN (Region-based CNN), Fast R-CNN, Faster R-CNN та інші [7].

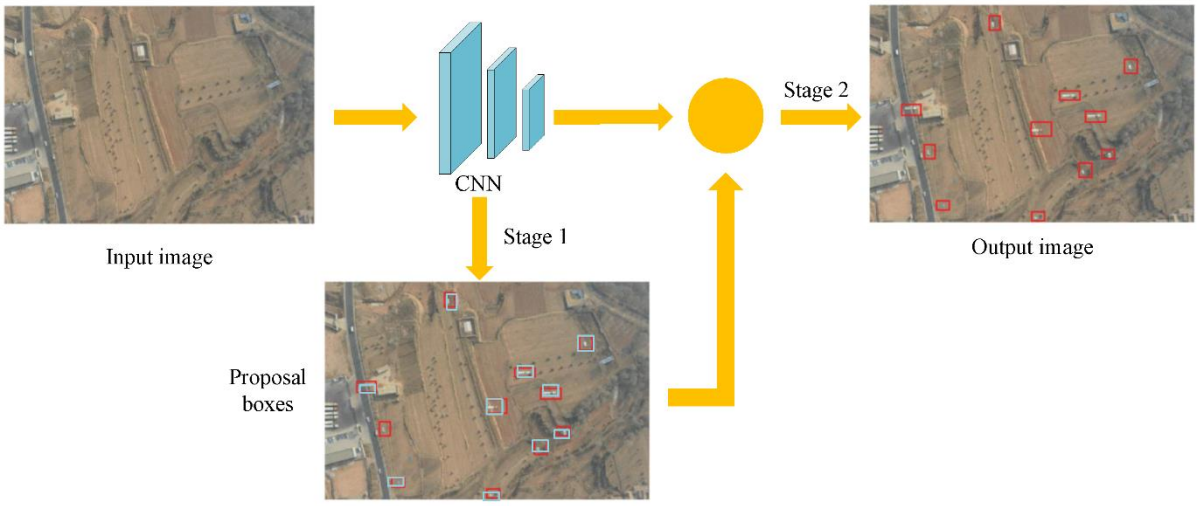


Рисунок 1.3 – Двоетапні алгоритми виявлення об’єктів

Оскільки двоетапні детектори вимагають двох етапів для виконання виявлення, перший етап створює велику кількість областей пропозицій, що призводить до значних обчислень і повільної обробки, яка не є оптимальною для сценаріїв реального часу.

На відміну від двоетапних, алгоритми одноетапного розпізнавання об’єктів безпосередньо генерують місцезнаходження і клас об’єкта на зображеннях за один етап, усуваючи процес створення областей пропозицій, тим самим забезпечуючи вищу швидкість обробки. Однак, одноетапні алгоритми виявлення об’єктів схильні до неправильного виявлення при локалізації та розпізнаванні невеликих об’єктів через велику кількість щільно згенерованих областей пропозицій. Крім того, класифікаційна та регресійна гілки одноетапних методів зазвичай є простими і важко вловлюють детальні характеристики об’єкту, що призводить до нестабільної продуктивності виявлення. Репрезентативні алгоритми включають SSD, серію моделей YOLO, RetinaNet тощо [8], [9]. На рисунку 1.4 показана загальна структура одноетапних алгоритмів.

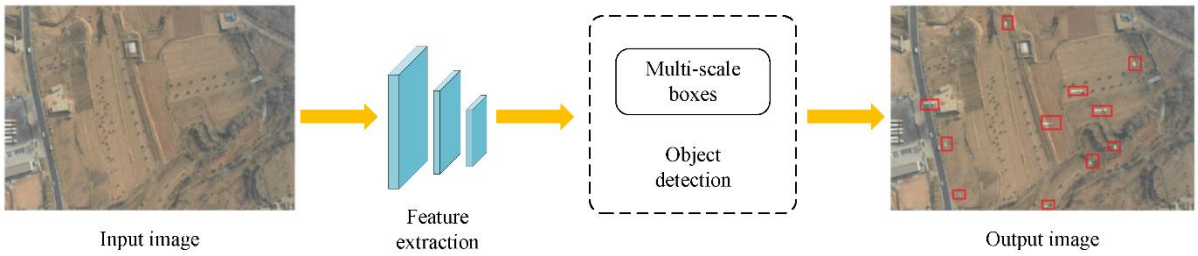


Рисунок 1.4 – Одноетапні алгоритми виявлення об’єктів

Виявлення об’єктів у військовій сфері найчастіше залежить від швидкодії, точності та можливості розгортання додатків в середовищах з обмеженими обчислювальними ресурсами. Оскільки якість результатів розпізнавання відіграє важливу роль у прийнятті будь-яких рішень та незначна недбалість може призвести до масових руйнувань, втрат людських життів та територій, подальший розгляд буде обмежений одноетапними методами. Архітектури деяких з найбільш популярних представлені в наступних підрозділах.

1.4.1 YOLO

Під час роботи алгоритму YOLO (You Only Look Once) [8] на вихідне зображення накладається квадратна сітка та розраховуються ймовірності належності до певного класу. Комірки, що мають імовірність класу вище порогового значення, вибираються і використовуються для визначення місця розташування об’єкта на зображенні. В кінці обробки комірки, що мають найвищі ймовірності, групуються в обмежувальні рамки. Загальний вид того, як працює дана модель, запропонований на рисунку 1.5.

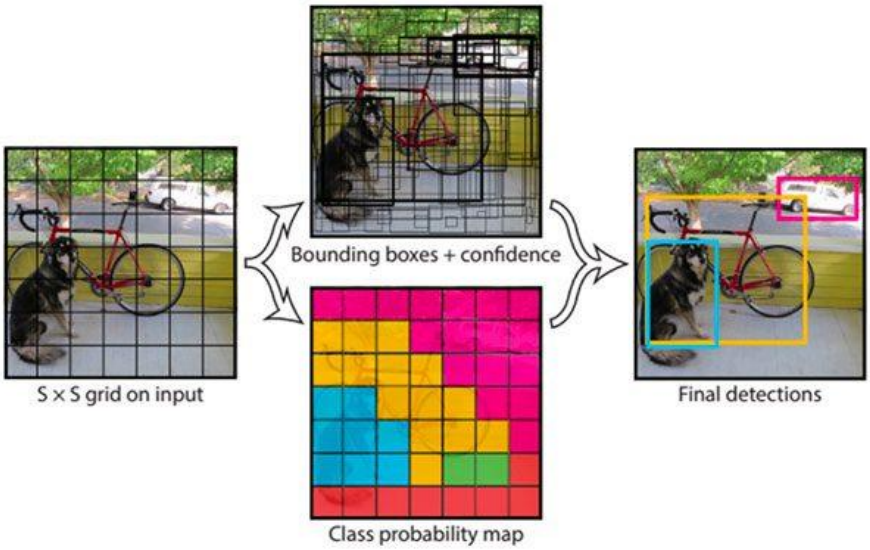


Рисунок 1.5 – Алгоритм YOLO

1.4.2 SSD

Архітектура SSD (Single Shot Detector) [8] (рисунок 1.6) складається з базової мережі, наприклад, VGG або ResNet, яка попередньо навчається на великому наборі даних класифікації зображень і в якої відсутній повнозв’язний шар класифікації. До цієї базової мережі додається кілька додаткових шарів згорток, які надбудовуються над базовою мережею. Ці додаткові шари відповідають за виявлення об’єктів різного масштабу, а їх виходи інтерпретуються, як обмежувальні рамки та класи об’єктів.

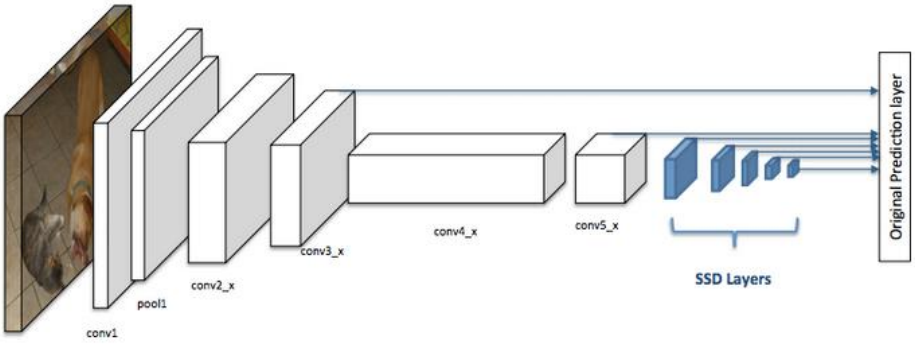


Рисунок 1.6 – Алгоритм SSD

1.4.3 RetinaNet

Дана мережа використовує функцію фокальних втрат (Focal Loss) для усунення дисбалансу класів під час навчання. Завдяки застосуванню цієї функції модель більше фокусується на прикладах, в яких помиляється, а не на тих, які вона може впевнено передбачити, гарантуючи, що прогнози на складних прикладах з часом будуть покращуватися. Це єдина уніфікована мережа, що складається з магістральної мережі та двох підмереж, призначених для конкретних завдань. Магістральна мережа відповідає за обчислення згорткової карти ознак на всьому вхідному зображенні. Перша підмережа виконує класифікацію об'єктів, а друга підмережа визначає обмежувальні рамки для об'єктів [8]. Архітектура даної мережі зображена на рисунку 1.7.

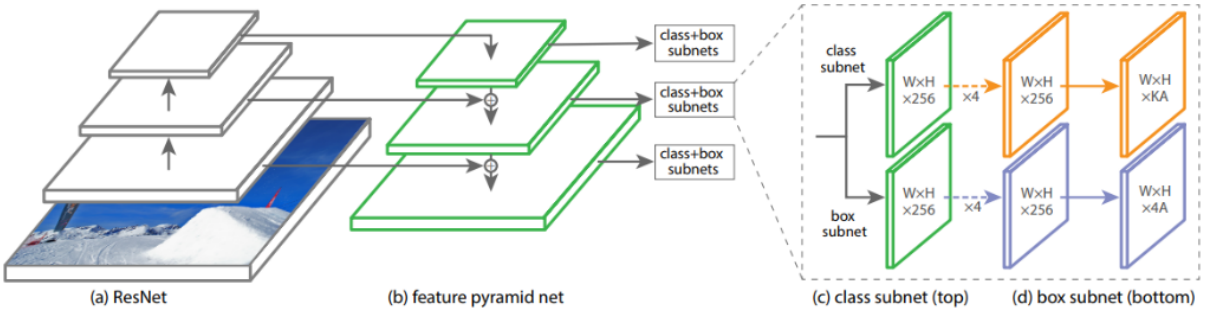


Рисунок 1.7 – Алгоритм RetinaNet

1.5 Використання комп'ютерного зору у військовій сфері

1.5.1 Особливості застосування

Однією з найважливіших організацій в рамках будь-якої держави є збройні сили, які використовуються для захисту суверенітету країни у разі нападу, забезпечення загальної безпеки громадян, підтримки громадянських

інституцій всередині країни та за кордоном, а також виконують багато інших важливих функцій.

Забезпечуючи безпеку громадян країни, збройні сили мають виконувати широкий спектр різноманітних завдань. Командири несуть відповідальність за точне виконання місій, тому для забезпечення успішності операцій вони мають глибоко розуміти район проведення операції та поставлені завдання, уявляти рішення проблеми та чітко описувати кроки її реалізації своїм підлеглим. Командири часто стикаються зі складними рішеннями. Постійна невизначеність війни, конфліктів та мирного часу може ускладнити ситуацію ще більшою мірою. Додатковим чинником, який іноді ускладнює ситуацію, є етичні питання, з якими стикаються командири на всіх рівнях управління, а також солдати, що виконують накази в умовах бойових дій.

Військовий штаб складається з різних функціональних підрозділів та персоналу, які беруть участь у процесі планування операцій. Процес прийняття військових рішень – це циклічний процес планування, в якому беруть участь командир підрозділу, співробітники штабу, підпорядковані штаби та підпорядковані підрозділи. Він використовується для розуміння місії та її контексту, оцінки ключових компонентів місії, створенні, порівнянні та оцінці потенційних варіантів дій, визначенні найкращого варіанту дій для даної місії, формуванні оперативного наказу та контролю за ходом виконання місії.

Для прийняття найкращого рішення командир повинен мати доступ до актуальної, точної та об’єктивної інформації про місцевість, метеорологічну обстановку, сили опозиції, цивільні структури тощо.

Військове керівництво зазвичай отримує таку інформацію через розвідувальні служби. Ті, в свою чергу, історично користуються традиційними джерелами збору інформації: людський інтелект [10], розвідка сигналів [11] та розвідка зображень [12].

Хоча традиційні джерела збору розвідувальних даних були надзвичайно важливими протягом всієї історії війни, важливо підкреслити, що сучасні джерела збору розвідувальних даних, такі як розвідка з відкритих джерел та розвідка соціальних мереж, демонструють стрімкий розвиток в контексті швидкого розширення сфери кібер-війни. Крім того, сучасні джерела збору розвідувальних даних значно дешевші у використанні порівняно з традиційними джерелами.

Розвідка зображень має ключове значення в сучасному зборі даних та включає отримання інформації з електрооптичних, інфрачервоних, радіолокаційних і лідарних датчиків на різних платформах, таких як наземні, морські, повітряні і космічні платформи. Також варто зазначити, що останнім часом безпілотні літальні апарати (БПЛА) набули величезного значення у виконанні різноманітних місій в рамках сучасних систем ведення війни та національної безпеки [13]. Після збору даних з наведених пристроїв інформація обробляються і представляється кінцевим користувачам. Для цього найкращим чином підходять різноманітні рішення побудовані на базі комп’ютерного зору.

1.5.2 Проблеми виявлення

У більшості випадків дослідники комп’ютерного зору є доволі обмеженими у виборі наборів даних для навчання своїх моделей. Особливо це стосується глибоких мереж, які потребують значного обсягу даних. Це пов’язано з конфіденційністю та секретністю такої інформації. Тому загальнодоступні набори даних, що використовуються для алгоритмів виявлення об’єктів у військовій сфері, здебільшого містять матеріали з ручних камер або з фіксованих позицій, тим самим ми отримуємо зображення тільки з певного ракурсу.

Тим не менш на поточному щаблі розвитку воєнних технологій та активних конфліктів у відкритому доступі з’являється все більше відео та

фотоматеріалів, зроблених за допомогою камер БПЛА. Повітряні знімки мають відмінні характеристики від звичайних зображень, оскільки вони зроблені зверху вниз. Це означає, що алгоритми виявлення об’єктів у звичайному вигляді не можуть бути безпосередньо застосовані до таких знімків.

На якість отриманих матеріалів з систем БПЛА впливає багато факторів, таких як нестабільність обладнання, що спричиняє тремтіння, розмиття, низька роздільна здатність, зміна освітлення, спотворення зображення тощо. Ці проблеми необхідно попередньо обробляти перед подачею до моделі, щоб покращити ефект виявлення об’єктів.

Також найчастіше щільність об’єктів на аерофотознімках не є постійною, а розмір дуже малий [14]. Наприклад, пішоходи та автомобілі можуть займати багато пікселів у звичайному вигляді, але лише кілька пікселів на аерофотознімках та розподілені нерівномірно, що спричиняє деформацію об’єкта, збільшує складність виявлення декількох об’єктів і вимагає спеціальних мережевих модулів для вилучення ознак.

Оклюзія на аерофотознімках також відрізняється від звичайного вигляду. У звичайному вигляді об’єкт може бути закритий іншими об’єктами, наприклад, людиною перед автомобілем. Однак на аерофотознімку об’єкт може бути закритий навколишнім середовищем, наприклад, будівлями та деревами. Тому неможливо безпосередньо застосовувати алгоритми виявлення декількох об’єктів, навчені на наборах даних зі звичайним оглядом, до зображень, отриманих таким чином.

Не менш важливими є також проблеми обертання об’єктів, складний фон, камуфляж, збільшення кількості дрібних об’єктів, низька ефективність виявлення, спричинена зміною масштабу, а також розріджений і нерівномірний розподіл класів об’єктів.

Окрім вищезгаданих труднощів з обмеженим доступом до вхідних даних, також мають місце більш специфічні проблеми, які стосуються апаратних обмежень, на яких буде запущена побудована модель, наявності

безпечних каналів передачі даних та інші. Існування вищезгаданих проблем відкриває величезний простір для дослідників в цій області.

1.5.3 Огляд існуючих досліджень

Оскільки інформованість про обстановку на полі бою – важлива прикладна тема для національної оборони, вчені багатьох країн розпочали серію досліджень у цій галузі. A. K. Jain [15] та інші запропонували метод сегментації для виявлення цілей на складному фоні та застосували його до військової розвідки. B. N. Nelson [16] запропонував новий метод виявлення та класифікації цілей за допомогою системи нечіткого виведення, який використовувався для ідентифікації танків і військових транспортних засобів. Jun та ін. [17] запропонували алгоритм автоматичного виявлення цілей (ATD) для зображень з ПЗС-матрицями, який був використаний для виявлення таких цілей, як бойові танки і бронетранспортери, в ситуаціях «земля-земля». V.E. Neagoe та ін. [18] запропонували новий метод автоматичного розпізнавання цілей на аерофотознімках радару із синтезованою апертурою (SAR) за допомогою нейронної мережі із загальним показником успішності 97,36%.

AlexNet є важливим поворотним моментом у розвитку глибокого навчання [19]. З моменту створення AlexNet ця мережева модель використовувалася в багатьох сферах, включаючи виявлення військових цілей. В останні роки вчені послідовно застосовують методи, засновані на глибокому навчанні, для розпізнавання обстановки на полі бою. R-CNN – це перший метод застосування згорткової нейронної мережі для виявлення цілей. Порівняно з традиційними алгоритмами, такими як Adaboost і DPM, його продуктивність була значно покращена. З того часу було запропоновано багато ефективних алгоритмів виявлення цілей, таких як SPP-Net, Fast R-CNN, Faster R-CNN та R-FCN. Ці методи виправляють недоліки R-CNN та підвищують точність і швидкість виявлення цілей.

Однак, вищезгадані методи все ще мають проблему низької швидкості виявлення в практичному застосуванні, та великою мірою не задовольняють вимоги розгортання моделі в умовах обмежених апаратних ресурсів, тому для подальшого дослідження була обрана модель YOLO, яка є репрезентативним одноетапним детектором. Її найхарактернішою особливістю є висока швидкість розпізнавання, що може бути використана в ситуаціях реального часу.

Дана модель широко використовується у багатьох додатках комп'ютерного зору, оскільки вона реалізує компроміс між швидкістю та точністю виявлення. Однак, враховуючи специфіку предметної області, результати виявлення військових цілей у складних умовах не є оптимальними, тому подальша робота буде націлена на покращення результатів роботи даної моделі в умовах військового застосування.

1.6 Постановка задачі дослідження

З аналізу різних підходів до виявлення військових об'єктів за допомогою методів комп'ютерного зору визначені два різних підходи до ідентифікації об'єктів: одноетапний і двоетапний. Оскільки у військовій сфері такі характеристики розпізнавання, як швидкодія та точність, є вирішальними, найбільш широкого застосування зазнали одноетапні алгоритми, такі як YOLO, SSD та RetinaNet.

Традиційні способи ведення розвідки відкривають нові можливі області застосування комп'ютерного зору у військовій сфері, але через брак необхідної вхідної інформації для навчання, апаратні та безпекові обмеження, пошук найбільш прийнятного алгоритму розпізнавання об'єктів продовжується. Після огляду наявних досліджень у даній тематиці для подальшого дослідження була обрана модель одноетапного детектора YOLO.

Таким чином, метою даної роботи є підвищення точності розпізнавання військових об’єктів на аерофотознімках на базі одноетапного алгоритму ідентифікації.

Об’єкт дослідження – процес одноетапної ідентифікації об’єктів на зображеннях.

Предмет дослідження – система розпізнавання військових об’єктів на аерофотознімках засобами комп’ютерного зору.

Методи дослідження – методи глибокого навчання, які базуються на аналізі інформаційних джерел та програмних засобах: Python, PyTorch та Flask.

Для досягнення мети дослідження потрібно розв’язати наступні завдання:

– оглянути та проаналізувати інформаційні джерела в області комп’ютерного зору та військової справи;

– знайти оптимальні підходи до вирішення задачі розпізнавання військових об’єктів за допомогою методів комп’ютерного зору;

– визначити набір даних, який міститиме аерофотознімки військових об’єктів для постановки експерименту;

– розробити систему розпізнавання військових об’єктів на аерофотознімках;

– провести аналіз результатів експерименту.

2 ОГЛЯД ОДНОЕТАПНОЇ МОДЕЛІ YOLO

2.1 Принцип роботи

YOLO є одним із найсучасніших алгоритмів виявлення об’єктів у реальному часі. Дана модель розглядає задачу розпізнавання об’єктів як задачу регресії, а не класифікації, що є значним зрушенням від традиційного підходу до ідентифікації об’єктів. Оригінальна версія цього алгоритму була представлена у 2015 році Д. Редмоном, С. Діввалом, Р. Гіршиком та А. Фархаді [20].

Розпізнавання об’єктів розпочинається з розбиття вихідного зображення на квадратну сітку (рисунок 2.1), де кожна комірка сітки відповідає за передбачення обмежувальних рамок та локалізацію об’єкта, визначаючи значення ймовірності належності його до певного класу [21].

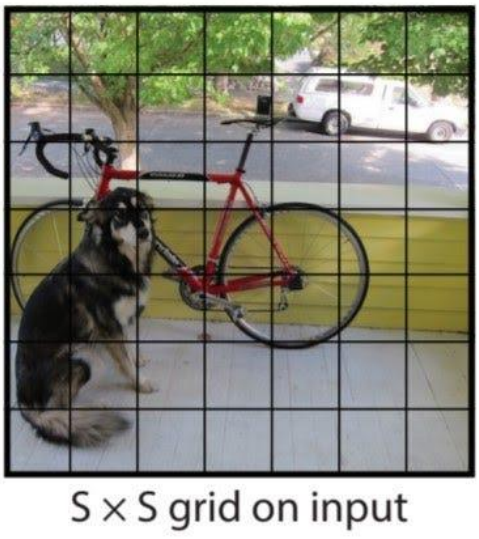


Рисунок 2.1 – Розбиття зображення на клітинки

На наступному кроці визначаються обмежувальні рамки для кожної комірки та довірчі ймовірності. Обмежувальні рамки являють собою прямокутники різного розміру та довжини, які виділяють всі можливі

об’єкти на зображенні. Оцінки правильності класифікації мають розподіл від 0 до 1, де 0 – найнижчий рівень, який означає, що в комірці не зафіксовано жодного об’єкта, а 1 – це найвищий рівень, який позначає впевненість моделі у тому, що в комірці присутній об’єкт.

Для кожної обмежувальної рамки даний алгоритм визначає певні атрибути у наступному векторному форматі: оцінка ймовірності сітки, що містить об’єкт, позиція x, позиція y, висота, ширина та довірчі ймовірності. Координати x та y позначають центр прогнозованої рамки.

Для подальшого навчання моделі потрібні два набори векторів обмежувальних рамок: набір векторів з прогнозами розташування, який отримано на попередньому кроці, та вектор з вже наперед розміченими областями розташування об’єктів (істинний).

Для визначення довірчої ймовірності використовується метрика перетину над об’єднанням IoU (Intersection Over Union). IoU – це широко прийнята метрика для вимірювання точності локалізації та її кількісної оцінки в моделях виявлення об’єктів. Розрахунок цієї метрики розпочинається з визначення площі перетину між прогнозованою областю і істинною областю для одного і того самого об’єкта. Ця спільна область називається «перетином». Паралельно обчислюється загальна площа покриття цими двома областями, яку зазвичай називають «об’єднанням». Розділивши «перетин» на «об’єднання», ми отримаємо співвідношення, яке показує ступінь перекриття по відношенню до загальної площі. Це співвідношення дає гарну оцінку того, наскільки точно прогнозована рамка збігається з вихідною рамкою. Приклад розрахунку цієї метрики на зображенні показаний на рисунку 2.2.

Після визначення IoU метрик для усіх обмежувальних рамок зазвичай постає проблема, коли алгоритм прогнозує кілька обмежувальних рамок для одного класу. Задля вирішення її можна було б вибрати лише одну рамку для кожного класу з найбільшою ймовірністю, але в такому випадку

алгоритм не зможе зафіксувати інші об’єкти цього ж класу на зображенні. Тому для цього використовується алгоритм не максимального придушення.

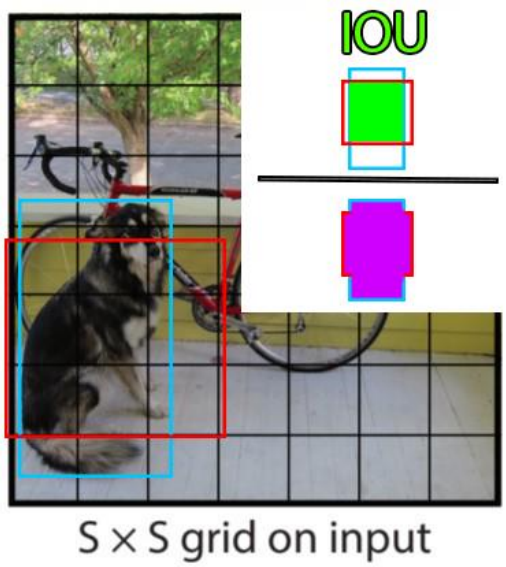


Рисунок 2.2 – Метрика IoU

Немаксимальне придушення (NMS) – це метод пост-обробки, який використовується в алгоритмах виявлення об’єктів для зменшення кількісті областей, що перекриваються, і поліпшення якості виявлення. NMS відфільтровує надлишкові та нерелевантні області-кандидати, залишаючи лише найточніші.

Алгоритм починає з вибору рамки з найвищою ймовірністю. Потім порівнює цю область з усіма іншими областями того самого класу, використовуючи метрику перетину над об’єднанням. Якщо значення IoU перевищує певний поріг (наприклад, 0,5), то область з меншою ймовірністю виключається. Таким чином, якщо дві рамки мають високе значення IoU, це може вказувати на те, що вони обидві позначають один і той самий об’єкт на зображенні, і можна виключити рамку з меншою ймовірністю. Цей процес повторюється, поки всі області не будуть прийняті, як прогнози об’єктів або виключені зі списку кандидатів.

В кінцевому результаті створюється набір передбачених обмежувальних рамок і міток класів для кожного об’єкта на зображенні, як показано на рисунку 2.3.

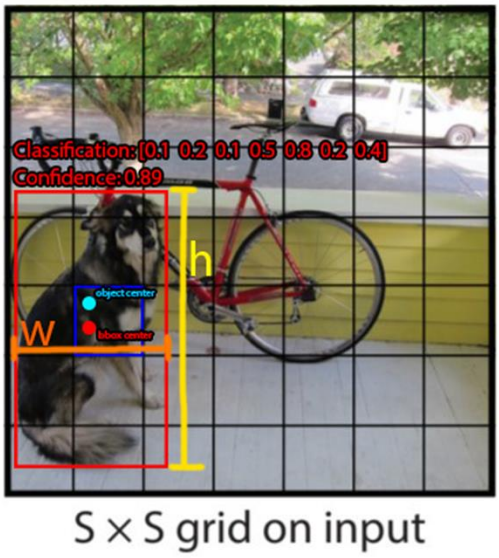


Рисунок 2.3 – Результат роботи моделі

В даному прикладі справжній центр собаки позначений блакитною крапкою. Таким чином комірка сітки, яка відповідає за виявлення та визначення межі, визначається коміркою, яка містить блакитну крапку та позначена темно-синім кольором. Прогнозована обмежувальна рамка складається з чотирьох елементів: червона крапка, яка позначає центр рамки (x, y), ширина та висота позначені помаранчевим і жовтим кольорами відповідно.

Важливо зауважити, що модель прогнозує центр обмежувальної рамки за допомогою ширини та висоти, а не положення лівого верхнього та правого нижнього кутів. З рисунку видно, що об’єкт найймовірніше належить з вірогідністю 0.8 до 5 класу.

2.2 Огляд версій

Як вже зазначалося, модель YOLO була вперше представлена публіці ще в 2015 році. За цей час відбувся величезний розвиток комп'ютерного зору і з'явилося багато різних версій даного алгоритму, які були спрямовані на покращення результатів виявлення, усуваючи недоліки попередників.

Хоча новаторська ідея регресії замість класифікації зробила цю модель швидшою за існуючі детектори об'єктів того часу, однак помилка локалізації була більшою, порівняно з методами двоетапного виявлення, такими як Fast R-CNN. Це відбувалося через те, що модель могла виявити щонайбільше два об'єкти одного класу в комірці сітки, що обмежувало її здатність передбачати сусідні об'єкти. Вона намагалася передбачити об'єкти із співвідношенням сторін, яких не було в навчальних даних, та навчалася на грубих ознаках об'єкта.

На рисунку 2.4 показана хронологія розвитку різних версій сімейства моделей YOLO [22].

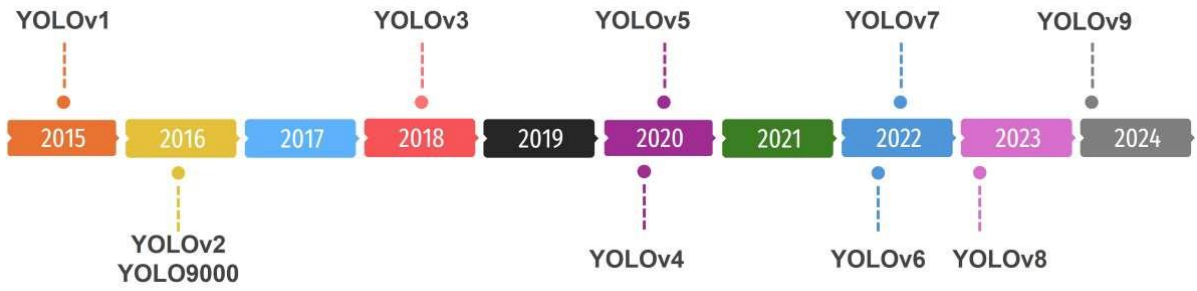


Рисунок 2.4 – Хронологія YOLO

2.2.1 YOLOv2

YOLOv2 (також відомий YOLO 9000) був представлений Д. Редмоном у 2016 році [23], [24], [25]. Ця модель є покращеною версією та націлена на зменшення недоліків, які спостерігалися в оригінальній версії, зберігаючи

при цьому швидкість. Вона використовує іншу магістраль CNN під назвою Darknet-19. Також у внутрішню архітектуру була впроваджена пакетна нормалізація для покращення збіжності моделі, що збільшило швидкість навчання та покращило продуктивність на 2 % mAP.

Перша версія YOLO працювала з вхідним зображенням розміром 224×224 пікселів на етапі навчання, а на етапі виявлення була можливість використовувати зображення з вищою роздільністю 448×448 пікселів. Це призводило до зниження точності, тому автори YOLOv2 натренували модель на зображеннях вищої роздільності 448×448 пікселів протягом 10 епох на наборі даних ImageNet. Це призвело до збільшенню точності mAP ще на 4 %.

Одним з найбільших покращень в даній версії моделі стало використання якірних рамок, які являють собою попередньо визначений набір обмежувальних рамок з різними співвідношеннями сторін і масштабами. Під час прогнозування YOLOv2 використовує комбінацію цих якірних рамок та прогнозованих зсувів для визначення остаточної обмежувальної рамки. Це дозволяє алгоритму ефективно обробляти об’єкти з різними розмірами та пропорціями.

В YOLOv2 також використовується стратегія багато масштабного навчання, що включає навчання моделі на зображеннях у різних масштабах та подальше усереднення їх прогнозів. Це допомагає покращити ефективність виявлення невеликих об’єктів, оскільки модель може краще адаптуватися до різних розмірів об’єктів під час навчання.

З точки зору продуктивності, YOLOv2 забезпечила 76,8 mAP при 67 FPS і 78,6 mAP при 40 FPS. За цими результатами модель продемонструвала перевагу архітектури над тогочасними SSD та Faster R-CNN. На рисунку 2.5 показане порівняння точності YOLOv2 з іншими моделями.

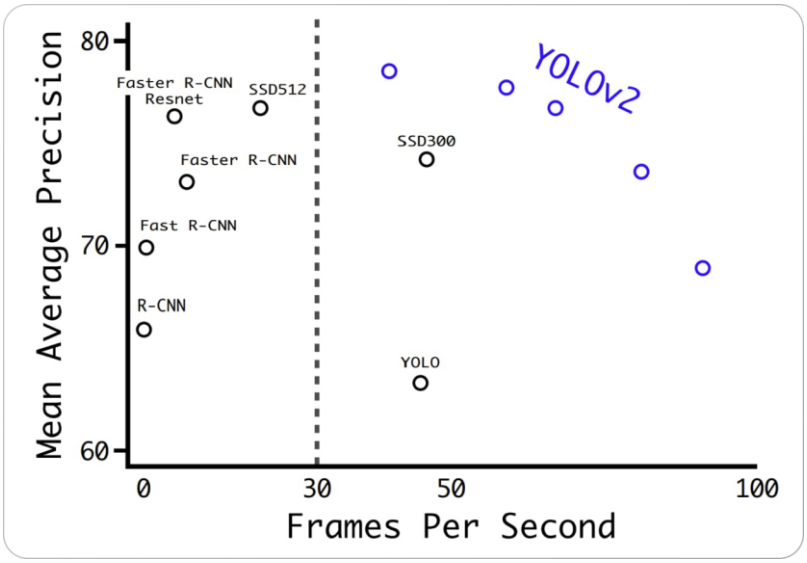


Рисунок 2.5 – Графік точності YOLOv2

2.2.2 YOLOv3

Третя ітерація даної моделі була представлена в 2018 році та була спрямована на підвищення точності та швидкості [23], [24], [25]. В цій версії моделі була використана нова мережева архітектура під назвою Darknet-53. Вона є різновидом архітектури ResNet і розроблена спеціально під задачу визначення об’єктів на зображенні. Дана мережа складається зі 106 нейронів та є більшою і точнішою за Darknet-19, яка була використана в YOLOv2.

Однією з важливих змін стало впровадження якірних рамок з різним масштабом та співвідношенням сторін на відміну від YOLOv2, де якірні рамки були фіксованого розміру. Це дозволило YOLOv3 краще охоплювати об’єкти різного розміру та форми на зображенні.

Також було введено поняття FPN (feature pyramid networks) – мережі пірамідальних ознак, яка допомагає виявляти об’єкти різного масштабу через побудову піраміди карт ознак, де кожен рівень піраміди використовується для виявлення об’єктів різного масштабу. Це допомагає покращити ефективність розпізнавання невеликих об’єктів.

YOLOv3 також використовує комбінацію втрат класифікації та втрат локалізації, що дозволяє моделі вивчати як ймовірності класів, так і координати обмежувальних рамок.

На рисунку 2.6 показане порівняння точності YOLOv3 з іншими моделями на наборі даних COCO.

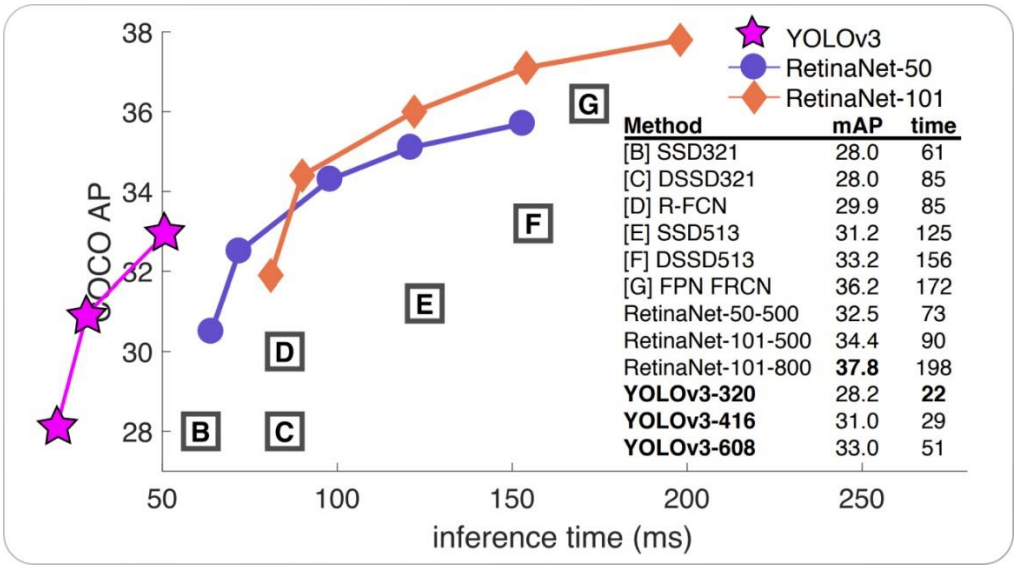


Рисунок 2.6 – Графік точності YOLOv3

2.2.3 YOLOv4

Четверта версія була першою «не офіційною» версією YOLO, так як творець даного алгоритму Д. Редмонд покинув спільноту ШІ та більше не брав участі у розробках подальших версій даної моделі. Тільки у квітні 2020 року О. Бочковський, Ч.-Я. Ванг та Х.-Ю.М. Ляо опублікували статтю, в якій описали їх погляд на розвиток моделі YOLO [23], [24], [25]. Версія цих авторів зберігала ту ж саму філософію YOLO та пропонувала доволі значущі покращення, тому спільнота ШІ сприйняла цю версію моделі як офіційне продовження.

В даній версії була використана більш досконала нейронна мережа на базі CSPDarknet-53. Був введений новий блок SPP (Spatial Pyramid Pooling)

для визначення ознак на зображенні в різних масштабах і роздільній здатності. Також FPN був замінений на PANet, як агрегатора ознак, що є деякою мірою вдосконаленням мережі пірамідальних ознак.

В YOLOv4 впроваджено новий метод генерації опорних квадратів, який є кластеризацією за методом k-середніх. Цей метод передбачає використання алгоритму кластеризації для групування базових обмежувальних рамок у кластери, а потім використання центроїдів кластерів як опорних рамок. Такий підхід дозволяє опорним рамкам більш точно відповідати розміру та формі виявлених об’єктів.

Також було представлено перехресну міні-пакетну нормалізацію, яка полегшує роботу на GPU. Тим самим модель оптимізована для паралельних обчислень, тому більш підходить для виробничих систем.

На рисунку 2.7 нижче показано, що YOLOv4 перевершує YOLOv3 за точністю приблизно на 10%.

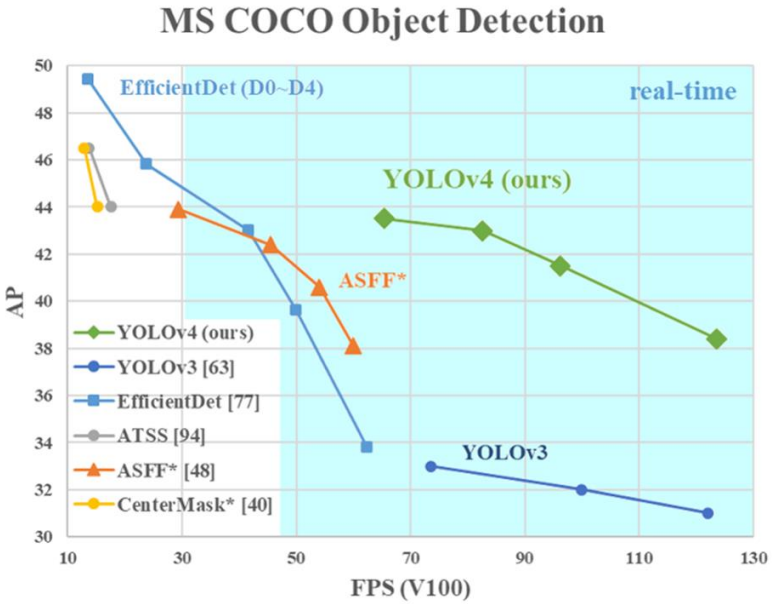


Рисунок 2.7 – Графік точності YOLOv4

2.2.4 YOLOv5

YOLOv5 був випущений у 2020 році Г. Джочером в рамках проекту Ultralytics [23], [24], [25]. На відміну від інших версій, дана не описана у науковій статті та, варто зазначити, що це була перша версія, написана за допомогою фреймворку PyTorch, а не Darknet.

PyTorch дав можливість використовувати усталену екосистему з широкою базою користувачів і надав допоміжну інфраструктуру для розгортання на мобільних пристроях. Випуск включає такий ряд моделей від найменшої до найбільшої: YOLOv5s, YOLOv5m, YOLOv5l і YOLOv5x.

П’ята версія була побудована на основі архітектури EfficientDet, що є розширенням архітектури класифікації зображень EfficientNet. Використання більш складної архітектури дозволило досягти більшої точності для ширшого спектру класів об’єктів. Рисунок 2.8 демонструє перевагу YOLOv5 над EfficientDet.

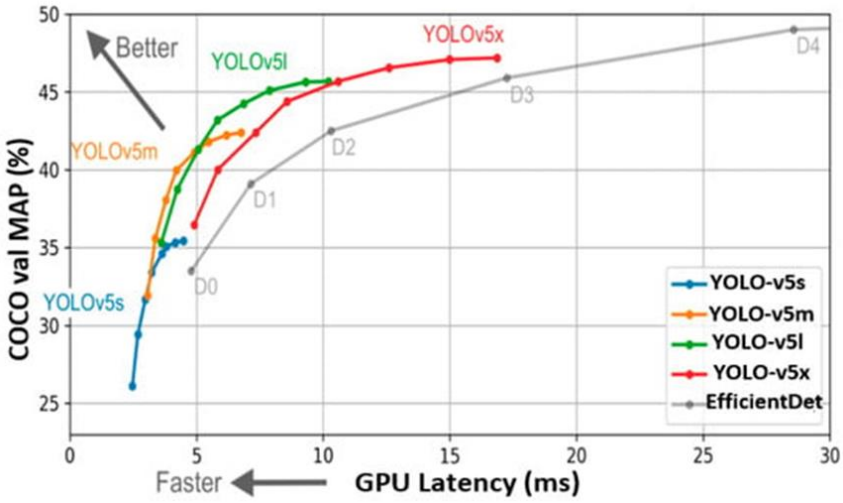


Рисунок 2.8 – Графік точності YOLOv5

У YOLOv5 використовується новий метод генерації якірних рамок, відомий як динамічні якірні рамки. Цей метод передбачає використання алгоритму кластеризації для групування базових обмежувальних рамок у

кластери, а потім використання центроїдів кластерів, як опорних рамок. Це дозволяє опорним рамкам краще відповідати розміру та формі виявлених об’єктів, забезпечуючи більш точні прогнози.

YOLOv5 також розширює та покращує шар SPP, дозволяючи ефективніше виявляти невеликі об’єкти на зображенні. Вводиться поняття CIoU втрат, що є покращенням функції втрат IoU, за допомогою якої збільшується продуктивність моделі на незбалансованих наборах даних.

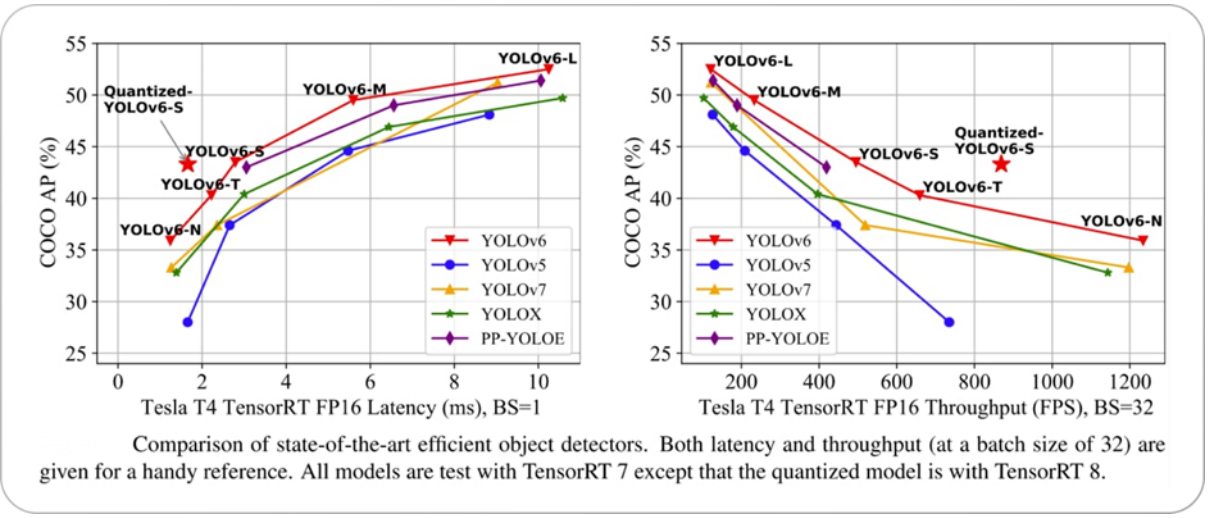
2.2.5 YOLOv6

Модель YOLOv6 була випущена відділом Vision AI китайської компанії Meituan [23], [24], [25]. Дана версія як і її попередниця була побудована на базі фреймворку PyTorch, та є націленою на промислове використання. Її архітектура була дещо модернізована та отримала назву EfficientNet-L2. Це легша та більш ефективна архітектура нейронної мережі, що дозволяє даній версії працювати набагато швидше і з меншими обчислювальними ресурсами. Для користувачів було запропоновано ряд варіантів для використання, починаючи з YOLO-v6-nano, яка поставляється з найменшою кількістю параметрів та закінчуючи YOLO-v6-large з високою точністю та швидкістю обробки.

Під час навчання YOLOv6 використовує методи доповнення даних, що збільшує надійність та точність виявлення об’єктів. Для цього над вхідним зображенням застосовують випадкові перетворення: обертання, перевертання та масштабування.

YOLOv6 також представляє новий метод генерації якірних рамок, який називається щільні якірні рамки.

Результат порівняння YOLOv6 з іншими моделями, показаний на рисунку 2.9.



Comparison of state-of-the-art efficient object detectors. Both latency and throughput (at a batch size of 32) are given for a handy reference. All models are test with TensorRT 7 except that the quantized model is with TensorRT 8.

Рисунок 2.9 – Графік точності YOLOv6

YOLOv6-N досяг 35,9% AP на наборі даних COCO при пропускній здатності 1234 FPS на графічному процесорі NVIDIA Tesla T4. YOLOv6-S досягнув 43,3% AP при 869 FPS, тоді як YOLOv6-M та YOLOv6-L також показали результати на рівні 49,5% та 52,3% при однаковій швидкості виведення.

2.2.6 YOLOv7

У 2022 році, окрім шостої версії, було випущено сьому [23], [24], [25], в якій, головним чином, було зроблені дві значні архітектурні зміни:

– впроваджена інтегрована ефективна мережа агрегації рівнів (E-ELAN) в магістраль моделі, яка дозволяє моделі вивчати більше різноманітних ознак для кращого навчання;

– масштабування для генерації моделей різного розміру. Таким чином модель задовольняє ширший спектр вимог для різних типів використання, надаючи більше пріоритету точності виявлення або швидкості обробки зображення.

Окрім цього, YOLOv7 використовує дев’ять опорних рамок, розширюючи можливості виявлення об’єктів різних форм і розмірів, що, в свою чергу, зменшує кількість неправильних спрацьовувань.

На відміну від інших версій, була замінена функція втрат з перехресної ентропії на функцію фокальних втрат. Це дозволило враховувати складність виявлення малих об’єктів, регулюючи вагу втрат на добре класифікованих прикладах і приділяючи більше уваги складним для розпізнавання об’єктам.

YOLOv7 може обробляти зображення з роздільною здатністю 608×608 пікселів та досягати при цьому швидкості до 155 кадрів за секунду, що показано на рисунку 2.10.

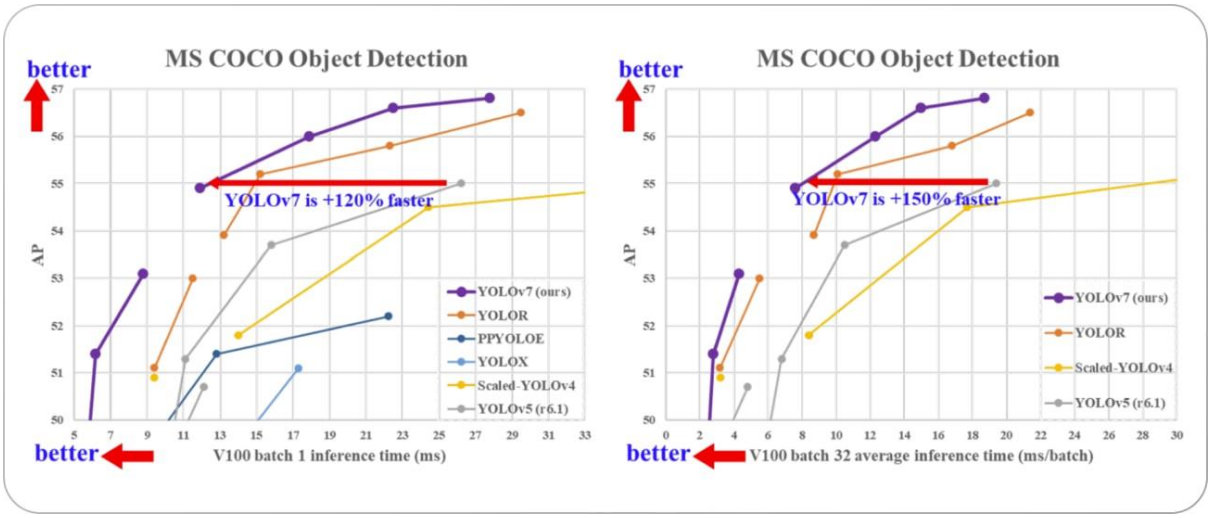


Рисунок 2.10 – Графік точності YOLOv7

Щодо точності, YOLOv7 досягає значних результатів порівняно з іншими алгоритмами виявлення об’єктів. На популярному наборі даних COCO він досягає середньої точності на рівні 37,2% при порозі IoU 0,5.

Порівняння кількісних показників показано нижче на рисунку 2.11.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

Рисунок 2.11 – Порівняння точності YOLOv7 з іншими моделями

2.2.7 YOLOv8

У 2023 році компанією Ultralytics була випущена восьма версія популярного одноетапного детектора YOLO [23], [24], [25].

YOLOv8 має п’ять версій для різних сценаріїв використання від меншої до більшої: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l і YOLOv8x.

В даній версії використовується нова покращена магістральна мережа CSPDarknet з деякими змінами на рівні CSPLayer, який тепер називається C2f-модулем. Модуль C2f поєднує високо рівневі ознаки з контекстною інформацією для підвищення точності виявлення.

YOLOv8 використовує безякірну модель з відокремленою головкою, яка дозволяє кожній гілці зосередитися на своєму завданні і підвищує загальну точність моделі.

У вихідному шарі YOLOv8 використовується сигмоїдна функція активації для оцінки ймовірності того, що обмежувальна рамка містить

об’єкт. Для оцінки ймовірностей класів використовується функція softmax, що відображає ймовірність належності об’єктів до різних класів.

У YOLOv8 застосовуються функції втрат CIoU та DFL для визначення втрат у граничній області та бінарна крос-ентропія для втрат під час класифікації. Ці методи допомагають покращити ефективність виявлення об’єктів, особливо при роботі з невеликими об’єктами.

YOLOv8 можна запустити з інтерфейсу командного рядка (CLI) або встановити як PIP-пакет. Крім того, він постачається з численними інтеграціями для маркування, навчання та розгортання, що значно полегшує використання.

Алгоритм YOLOv8 демонструє збільшення середньої точності на 1,2% порівняно з YOLOv7. Цього вдалося досягти при зменшенні розміру файлу моделі на 80,6 мегабайт, що робить модель більш ефективною і легшою для розгортання в середовищах з обмеженими ресурсами.

Порівняння точності YOLOv8 з її попередницями показана на рисунку 2.12.

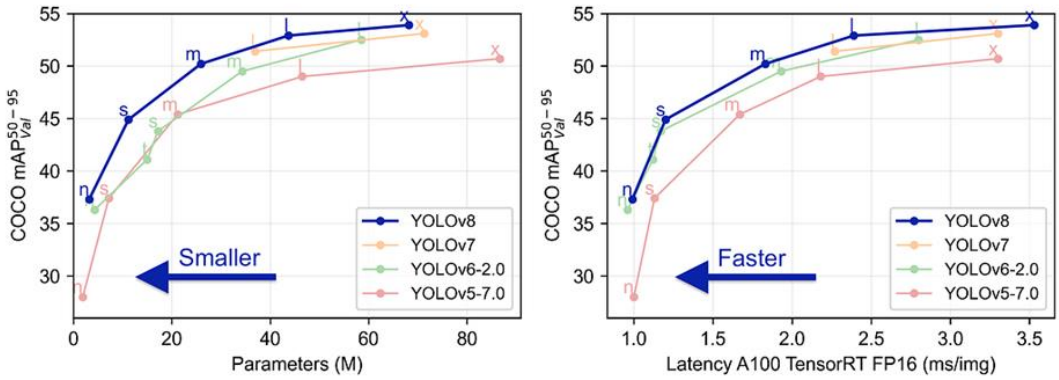


Рисунок 2.12 – Графік точності YOLOv8

2.2.8 YOLOv9

У кінці лютого 2024 року групою вчених з Тайбейського національного технологічного університету в Тайвані була опублікована стаття, в якій вони запропонували нову версію YOLO з інноваційною легкою архітектурою узагальненої ефективної мережі агрегації рівнів (GELAN) та механізму програмованої градієнтної інформації (PGI) [26].

Впровадження PGI дозволяє вирішувати проблеми втрати інформації, притаманні глибоким нейронним мережам. Цей механізм підвищує здатність до навчання і забезпечує збереження важливої інформації протягом усього процесу виявлення, що призводить до більш надійного оновлення градієнта.

GELAN дозволяє користувачам обирати відповідні обчислювальні блоки для різних пристроїв виведення, тим самим підвищуючи гнучкість та ефективність моделі.

Розробники даної версії YOLO домоглися зменшення розміру моделі на 49%, обсягів обчислень на 43% та кількості параметрів на 49%, порівняно з YOLOv8. Незважаючи на менший розмір, дана модель підвищила середню точність на наборі даних MS COCO на 0,6%.

На рисунку 2.13 показане порівняння продуктивності моделі YOLOv9 з іншими популярними моделями виявлення об’єктів.

За майже 10 років невпинної еволюції моделі YOLO стали по праву однією з найбільш ефективних, точних та швидких рішень для ідентифікації об’єктів на зображеннях і відео. Завдяки своїй революційній архітектурі, яка розглядає розпізнавання об’єктів як регресійну задачу, YOLO став відмінним рішенням для додатків, які працюють у режимі реального часу.

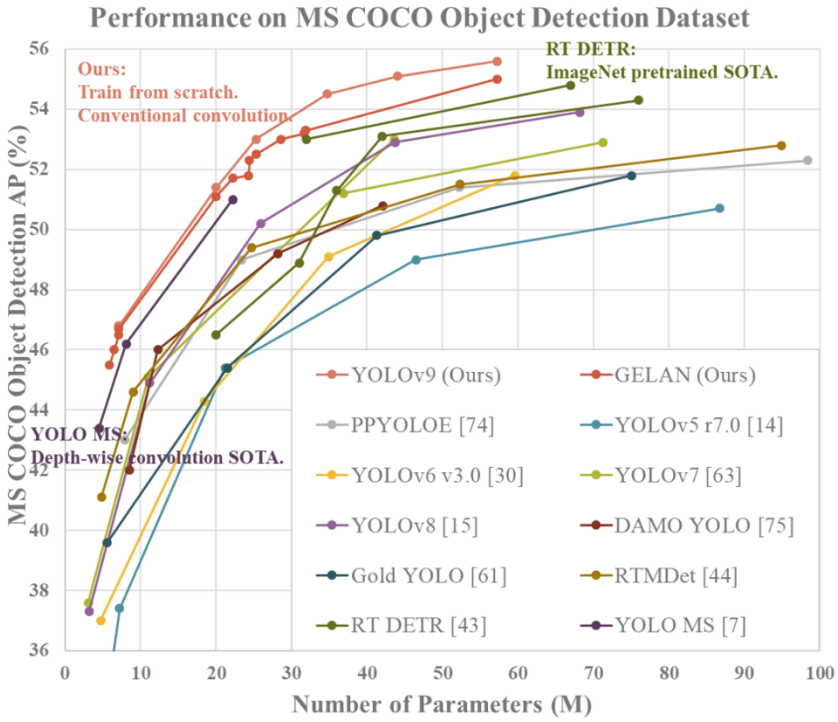


Рисунок 2.13 – Графік точності YOLOv9

Підсумовуючи даний розділ, для подальшого експерименту виявлення військових об’єктів на аерофознімках була обрана модель YOLOv8. Дана версія має високий рівень точності розпізнавання об’єктів, хоча одним з її недоліків є схильність виявляти неіснуючі об’єкти, що призводить до збільшення кількості хибно позитивних спрацьовувань. На відміну від цього, YOLOv9 використовує більш строгий підхід до розпізнавання, який зменшує кількість хибно позитивних розпізнавань, але тим самим деякі об’єкти можуть бути пропущені, що в кінцевому результаті призводить до нижчої якості виявлення.

Як можна побачити, відмінності у виявленні є незначними, і кінцеве рішення використовувати YOLOv8 було зроблено через наявність величезної підтримки спільноти та швидкості і зручності використання даної моделі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ ВІЙСЬКОВИХ ОБ'ЄКТІВ

3.1 Огляд засобів розробки

На початку розробки будь-якого додатку постає питання у виборі відповідних технологій, які дозволять найефективніше вирішити поставлене завдання.

Вибір мови програмування – це один із найважливіших аспектів, який постає відразу перед розробниками. Мова програмування Python широко використовується в додатках аналізу даних, машинного навчання та веб розробки, що робить її ідеальним кандидатом для використання в роботі.

З моменту створення, Python є динамічно типізованою та інтерпретованою мовою програмування, якою можна писати в об'єктно-орієнтованому та процедурному стилі. Синтаксис Python простий і нагадує повсякденну англійську мову, що скорочує час для вивчення, розуміння та написання програм. Python є платформо незалежною мовою, що означає, що код легко конвертується без будь-яких модифікацій для різних платформ, зокрема Linux, Windows, MacOS і UNIX. Крім того, дана мова підтримує широкий вибір бібліотек, фреймворків та модулів, які спрощують процес розробки, тестування і впровадження додатків штучного інтелекту. Одні з найвідоміших – це NumPy, Pandas, SciPy, TensorFlow, PyTorch, scikit-learn і Keras. Ці бібліотеки значно скорочують час і спрощують процес написання коду. Python має одну з найбільших спільнот користувачів серед мов програмування, що сприяє обміну досвідом та знаннями. Всі вищезгадані переваги даної мови підкріплюються індексом TIOBE у 2024 році, за яким Python є найпопулярнішою мовою програмування у світі [27].

Для ефективного використання та налагодження роботи різних пакетів та бібліотек був обраний дистрибутив Anaconda, який спеціально призначений для наукового обчислення та аналізу даних. Цей дистрибутив

поставляється з власним менеджером пакетів Conda, який дозволяє легко управляти залежностями на проекті. За допомогою Conda можна без особливих зусиль створити своє власне віртуальне середовище ізольоване від решти обчислювальної платформи і в рамках нього використовувати різні набори бібліотек та пакетів. Менеджер пакетів допомагає визначити сумісність тих чи інших бібліотек і повідомити користувачеві про це.

Навчання, тестування та розгортання моделі у веб-додаток було здійснено на персональній машині з наступними параметрами, які зображені в таблиці 3.1.

Таблиця 3.1 – Параметри персональної машини

Компоненти	Характеристики
Процесор	Intel i7-13700HX (8 фізичних та 8 віртуальних ядр, 24 потоки)
Відеокарта	NVIDIA GeForce RTX 4050 (6 гігабайт)
Оперативна пам’ять	32 гігабайти
SSD	1 терабайт

Для того, щоб навчання відбувалося на графічному процесорі, необхідно було подбати про встановлення CUDA – програмно-апаратної архітектури для паралельних обчислень, яка дозволяє істотно збільшити обчислювальну продуктивність завдяки використанню графічних процесорів фірми Nvidia та бібліотеку cuDNN для навчання глибоких нейронних мереж.

Зручність розробки є ключовим фактором, тому у якості інтерактивного середовища для розробки та виконання програмного коду було обрано Jupyter Notebook та середовище PyCharm.

Для навчання та використання моделей було використано бібліотеку ultralytics, яка надає доступ до моделі YOLOv8 [28]. Ultralytics також має

документацію, приклади використання та інші ресурси для розробників, щоб спростити процес роботи з їх продуктами.

Після навчання моделей був побудований веб-додаток за допомогою фреймворку Flask, що гарно підходить для швидкої розробки веб-рішень з мінімальними зусиллями, використовуючи простий та елегантний синтаксис.

3.2 Аналіз набору даних

В даній роботі для навчання моделей було вирішено використовувати набір даних Military Decision-Making Dataset [29]. Цей набір даних був отриманий з відкритих ресурсів та містить інформацію про різноманітну військову техніку та системи озброєння, які використовується сучасними арміями світу. Зображення були зроблені з різних ракурсів і в різних умовах. Більшість матеріалів становить кадри, зняті під час війни між Україною та Російською Федерацією. Також були використані відео та фотоматеріали з виставок та військових навчань.

Даний набір даних представлений 11 класами:

– танк (TANK);

– бойова машина піхоти (IFV);

– бронетранспортер (APC);

– інженерна машина (EV);

– бойовий вертоліт (AH);

– транспортний вертоліт (TH);

– штурмовий літак (AAP);

– транспортний літак (TAP);

– зенітна установка (AAH);

– буксирувана артилерія (TA);

– самохідна артилерія (SPA).

Анотація відбувалася за допомогою інструменту анотування DarkLabel [30]. В результаті для кожного зображення був створений відповідний файл з розширенням .txt, як показано на рисунку 3.1.

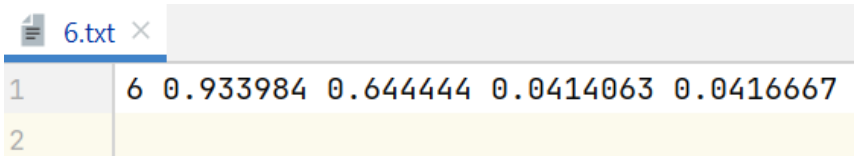


Рисунок 3.1 – Анотація даних для моделі YOLO

Кожен рядок у файлі представляє собою одне спостереження та містить інформацію про індекс класу об’єкта, координату центра x, координату центра y, ширину та висоту обмежувальної рамки. Усі координати є нормалізованими.

На рисунку 3.2 показаний розподіл класів в наборі даних.

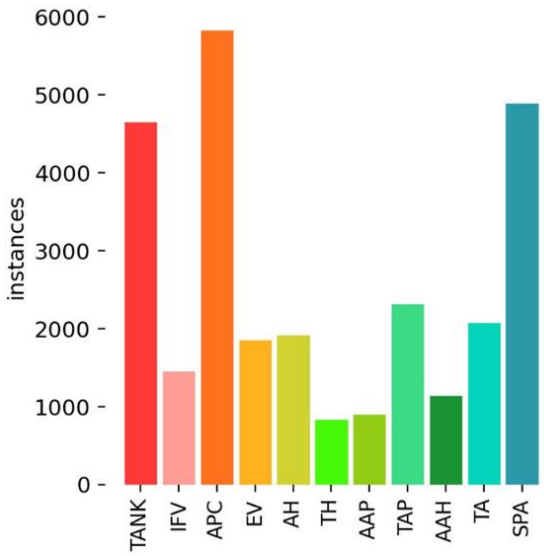


Рисунок 3.2 – Розподіл класів

Набір даних складається з 16 527 зображень військової техніки. Можна переконатися, що класи розподілені нерівномірно, що є проблемою

для виявлення певних типів об’єктів. Дані були розділені на три частини: навчальну, тестову та перевірочну вибірки. В навчальній міститься 11 613 зображень, у тестовій 3 234, а в перевірочній 1 680 зображень.

Деякі зображення із набору даних показані на рисунках 3.3 – 3.5.

Рисунок 3.3 – Приклад зображення з набору даних

Рисунок 3.4 – Приклад зображення з набору даних

Рисунок 3.5 – Приклад зображення з набору даних

Навчений на цьому наборі даних алгоритм може поліпшити ситуативну обізнаність військових командирів та штабів щодо структури та можливостей опозиційних сил, особливо в інтенсивних бойових операціях, де швидкий потік інформації має важливе значення.

З точки зору машинного навчання, значення цього набору даних полягає в підвищенні надійності та узагальненні алгоритмів. Різноманітність контрастів сприяє ефективному навчанню, роблячи алгоритм здатним розрізняти різні рівні яскравості і, отже, збільшуючи його стійкість до змін у світлових умовах. Додатково, різноманіття кольорів та текстур допомагає алгоритму розуміти різні типи об’єктів та фонів.

Включення зображень із різних перспектив у набір даних допомагає уникнути спеціалізації моделі на конкретних положеннях або кутах огляду. Різні розміри об’єктів та відстаней дозволяють моделі розвивати навички точного виявлення об’єктів у різних контекстах. І що не менш важливіше, це те, що дані містять зображення, зроблені в різних часових періодах та сезонах, що підвищує продуктивність моделі на реальних даних.

Тому використання цього набору даних відмінно підходить для вирішення поставленої задачі, та надає можливість задовільнити потреби військових, надаючи цінну інформацію про виявлення техніки.

3.3 Опис тренування моделей

Алгоритм YOLOv8 потребує спеціальної структури директорій для того, щоб мати змогу знайти інформацію про мітки та зображення під час навчання. На рисунку 3.6 показана структура організації файлів в директоріях.

```
├─ yolov8
└─ military-vehicles
   ├─ images
   |  ├─ train
   |  |  ├─ file1.jpg
   |  |  └─ file2.jpg
   |  └─ val
   |  |  ├─ file3.jpg
   |  |  └─ file4.jpg
   |  └─ test
   |     ├─ file5.jpg
   |     └─ file6.jpg
   └─ labels
      ├─ train
      |  ├─ file1.txt
      |  └─ file2.txt
      └─ val
      |  ├─ file3.txt
      |  └─ file4.txt
      └─ test
         ├─ file5.txt
         └─ file6.txt
```

Рисунок 3.6 – Розташування зображень та міток

Після рознесення файлів по директоріях був підготовлений конфігураційний файл з розширенням .yaml (рисунок 3.7), який містить інформацію про розташування набору даних та інформацію про класи.

```yaml
# Train\val\test sets as 1) dir: path\to\imgs, 2) file: path\to\imgs.txt
train: F:\education\Університи\ХНУРЕ\6_курс\Диплом\yolo-military-detection\datasets\military-vehicles\images\train
val: F:\education\Університи\ХНУРЕ\6_курс\Диплом\yolo-military-detection\datasets\military-vehicles\images\val
test: F:\education\Університи\ХНУРЕ\6_курс\Диплом\yolo-military-detection\datasets\military-vehicles\images\test

# Classes with military objects
names:
  0: TANK
  1: IFV
  2: APC
  3: EV
  4: AH
  5: TH
  6: AAP
  7: TAP
  8: AAH
  9: TA
  10: SPA
```

Рисунок 3.7 – Конфігураційний файл

Враховуючи специфіку предметної галузі, з різних варіантів була обрана найменша модель YOLOv8n. Її детальна архітектура запропонована на рисунку 3.8.

	from	n	params	module	arguments
0	-1	1	464	ultralytics.nn.modules.conv.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.conv.Conv	[16, 32, 3, 2]
2	-1	1	7360	ultralytics.nn.modules.block.C2f	[32, 32, 1, True]
3	-1	1	18560	ultralytics.nn.modules.conv.Conv	[32, 64, 3, 2]
4	-1	2	49664	ultralytics.nn.modules.block.C2f	[64, 64, 2, True]
5	-1	1	73984	ultralytics.nn.modules.conv.Conv	[64, 128, 3, 2]
6	-1	2	197632	ultralytics.nn.modules.block.C2f	[128, 128, 2, True]
7	-1	1	295424	ultralytics.nn.modules.conv.Conv	[128, 256, 3, 2]
8	-1	1	460288	ultralytics.nn.modules.block.C2f	[256, 256, 1, True]
9	-1	1	164608	ultralytics.nn.modules.block.SPPF	[256, 256, 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	148224	ultralytics.nn.modules.block.C2f	[384, 128, 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	37248	ultralytics.nn.modules.block.C2f	[192, 64, 1]
16	-1	1	36992	ultralytics.nn.modules.conv.Conv	[64, 64, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.conv.Concat	[1]
18	-1	1	123648	ultralytics.nn.modules.block.C2f	[192, 128, 1]
19	-1	1	147712	ultralytics.nn.modules.conv.Conv	[128, 128, 3, 2]
20	[-1, 9]	1	0	ultralytics.nn.modules.conv.Concat	[1]
21	-1	1	493056	ultralytics.nn.modules.block.C2f	[384, 256, 1]
22	[15, 18, 21]	1	753457	ultralytics.nn.modules.head.Detect	[11, [64, 128, 256]]

Model summary: 225 layers, 3012993 parameters, 3012977 gradients, 8.2 GFLOPs

Рисунок 3.8 – Архітектура YOLOv8n

Впродовж цієї роботи було навчено три моделі YOLO версії нано за різних оптимізаторів. У якості оптимізаторів були обрані три алгоритми градієнтного спуску: стохастичний градієнтний спуск (SGD), метод RMSProp та метод адаптивної оцінки моментів (Adam). Характеристики цих алгоритмів:

– стохастичний градієнтний спуск є популярним та простим алгоритмом навчання моделей нейронних мереж. Цей алгоритм оновлює кожну вагу, використовуючи поточний градієнт помножений на швидкість навчання;

– алгоритм RMSprop є методом оптимізації на основі градієнта, що використовується під час навчання нейронних мереж. Градієнти дуже складних функцій, таких як нейронні мережі, мають тенденцію або зникати або вибухати в міру поширення даних через шари. RMSprop вирішує вищезгадану проблему, використовуючи ковзне середнє квадратів

градієнтів для нормалізації поточного градієнту. Ця нормалізація врівноважує розмір кроку (імпульсу), зменшуючи крок для великих градієнтів, щоб уникнути вибуху, і збільшуючи крок для малих градієнтів, щоб уникнути зникнення;

– Adam є одним із найпопулярніших алгоритмів оптимізації. У цьому оптимізаційному алгоритмі використовуються ковзні середні як самих градієнтів, так і їх других моментів. Даний оптимізатор використовується за замовчуванням у багатьох бібліотеках машинного навчання, таких як TensorFlow та PyTorch.

Після підготовки набору даних наступним кроком проводиться навчання. Бібліотека ultralytics надає змогу навчати модель YOLOv8, використовуючи зручний Python API або через CLI. На рисунках 3.9 та 3.10 показаний процес створення моделі та запуску навчання через Python.

```python
import os
import torch
from ultralytics import YOLO

torch.cuda.set_device(0)
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

# known issue
os.environ['KMP_DUPLICATE_LIB_OK'] = 'True'

# build a new model from scratch
model_adam = YOLO("yolov8n.pt")
model_adam.to(device=device)
```

Рисунок 3.9 – Створення моделі YOLOv8n

```python
# define hyperparameters
epochs = 100
img_size = 640
dataset_yaml = "military-dataset-structure.yaml"
optimizer = "Adam"

# train the model
model_adam.train(
    data=dataset_yaml,
    epochs=epochs,
    imgsz=img_size,
    optimizer=optimizer,
    device=0,
    verbose=True
)
```

Рисунок 3.10 – Запуск навчання

Всі моделі були навчені протягом 100 епох із розміром зображень 640×640 пікселів. Важливо також зазначити, що у методі train був встановлений параметр device у значення 0. Тим самим це дозволило запустити навчання на дискретній відеокарті, що значно пришвидшує процес проходження тренування моделі.

В результаті навчання в директорії під назвою weights можна знайти декілька файлів: best.pt та last.pt, які можна розгортати в подальшому в веб-застосунку для розпізнавання об’єктів.

3.4 Розробка веб-застосунку

Після успішного навчання моделі для зручності її використання був розроблений веб-застосунок за допомогою фреймворку Flask. Даний додаток надає можливості ідентифікації об’єктів на відео та фотоматеріалах.

Структура даного застосунку складається з папки static, у якій містяться файли стилів, JavaScript файли, шрифти та результати розпізнавання об’єктів на зображенні. Для створення стильного та адаптивного інтерфейсу був використаний фреймворк Bootstrap. В директорії templates розташовані файли з розширенням .html. Також в проекті міститься два файли, які відповідають за логіку роботи додатку.

Один з них містить допоміжний клас YoloHelper, який є ядром застосунку та відповідає за виявлення об’єктів та нанесення обмежувальних рамок на результуюче зображення. Для ініціалізації даного класу потрібні наступні параметри: шлях до папки для збереження результатів та шлях до ваг моделі YOLO. Також розраховується масив colors, який допомагає помітити кожен клас на зображенні унікальним кольором.

На рисунку 3.11 показаний процес ініціалізації класу YoloHelper вищезгаданими параметрами.

```
class YoloHelper:
    def __init__(self, out_dir, weight_name):
        self.out_dir = out_dir
        self.model = YOLO(weight_name)
        self.colors = np.random.uniform( low: 0,  high: 255, size=(len(self.model.names), 3))
```

Рисунок 3.11 – Клас YoloHelper

На рисунку 3.12 показаний метод get_predictions, який робить прогнози та формує обмежувальні рамки.

```
def get_predictions(self, image, img_path):
    if img_path is None:
        predictions = self.model(image)
    else:
        predictions = self.model(img_path)

    prediction = predictions[0]
    boxes = prediction.boxes
    for box in boxes:
        label_number = box.cls.numpy()[0]
        label = prediction.names.get(label_number)
        confidence = box.conf.numpy()[0]
        x1, x2, x3, x4 = map(int, box.xyxy.numpy()[0])

        cv2.rectangle(image, (x1, x2), (x3, x4), self.colors[int(label_number)], 1)
        cv2.putText(
            image,
            text: f'{label} {confidence:.2f}',
            org: (x1, x2),
            cv2.FONT_HERSHEY_SIMPLEX,
            fontScale: 0.4,
            self.colors[int(label_number)],
            thickness: 1,
            cv2.LINE_AA
        )
```

Рисунок 3.12 – Метод get_predictions

В іншому файлі міститься інформація, необхідна для запуску веб-додатку, та маршрути, які визначаються за допомогою декораторів та пов’язують функції з певними URL-адресами. Приклад маршруту, відповідального за розпізнавання об’єктів, показаний на рисунку 3.13.

```python
@app.route( rule: '/object_detection', methods=['GET', 'POST'])
def detect_objects():
    cleaning_dir(app.config['RESULT_FOLDER'])
    if request.method == 'POST':
        if 'file' not in request.files:
            return render_template( template_name_or_list: 'index.html', error_msg='No file found')
        file = request.files['file']
        if file.filename == '':
            return render_template( template_name_or_list: 'index.html', error_msg='No file found')
        if file:
            path = save_upload(file)
            result_path, result_name, filetype = predict_file(path)

    return render_template(
        template_name_or_list: 'index.html',
        title='Home',
        out_path=result_path,
        out_name=result_name,
        filetype=filetype
    )
```

Рисунок 3.13 – Маршрут, відповідальний за визначення об’єктів

В результаті створено застосунок зі зручним користувацьким інтерфейсом, який показаний на рисунку 3.14. Верхня секція дозволяє обирати зображення або відеофайл, а в нижній ми отримаємо результат розпізнавання об’єктів.

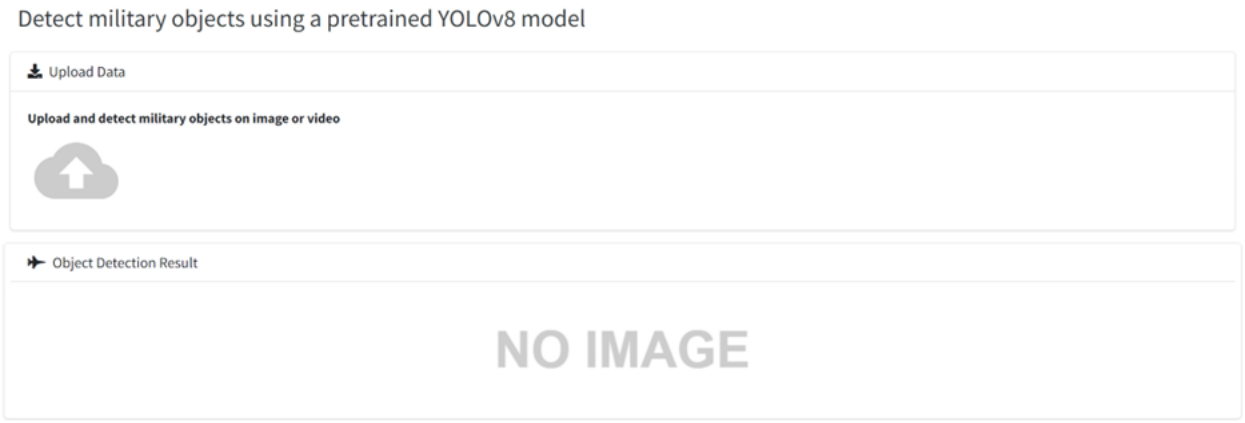


Рисунок 3.14 – Інтерфейс застосунку

У цьому розділі проведено розгляд програмних засобів, які були використані для реалізації системи розпізнавання військових об’єктів.

Детально описаний набір даних, який використовувався для навчання моделей YOLOv8n, проведено попередню підготовку перед його поданням. Для тестування моделі YOLOv8n було обрано три оптимізатори: SGD, RMSprop та Adam. Навчання відбувалося протягом 100 епох з використанням графічного ядра. В результаті навчання були отримані ваги, які було використано для розгортання моделі у веб-застосунку.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ МОДЕЛІ YOLOv8n ДЛЯ РОЗПІЗНАВАННЯ ВІЙСЬКОВОЇ ТЕХНІКИ НА ЗОБРАЖЕННЯХ

4.1 Оцінка моделей

Після підготовки набору даних та написанні програмного коду настає етап навчання моделей. Як зазначалося у минулому розділі, у якості основної архітектури була обрана YOLOv8n. Тренування відбувалося протягом 100 епох на графічному ядрі за різних оптимізаторів. Процес проходження навчання на останніх епохах зображений на рисунку 4.1.

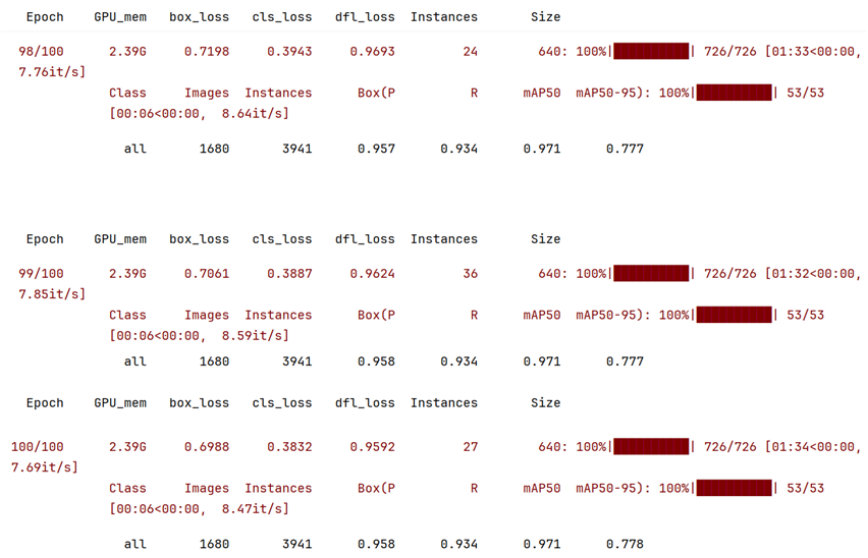


Рисунок 4.1 – Процес навчання моделі

Найважливіший етап після навчання – перевірка точності та ефективності роботи моделі. Для цього під час навчання генерується певний набір метрик, які визначають наскільки ефективно модель може виявляти та локалізувати об’єкти на зображенні. Уміння інтерпретувати ці метрики грає вирішальне значення при розгляданні можливостей впровадження цих

моделей в додатки. Тому в таблиці 4.1 показані запропоновані метрики та їх характеристики, які присутні на рисунках 4.2 – 4.4 [31], [32], [33].

Таблиця 4.1 – Метрики розпізнавання об’єктів

Назва	Характеристика
box_loss	Показує наскільки добре модель прогнозує розташування та розмір обмежувальних рамок навколо об’єктів. Менше значення вказує на більшу точність локалізації.
cls_loss	Вимірює точність передбачення правильного класу об’єктів в межах обмежувальних рамок. Нижче значення вказує на більш точну класифікацію.
dfl_loss	Функція втрат, яка усуває дисбаланс класів шляхом коригування фокусних втрат на основі розподілу класів у наборі даних. Це допомагає моделі при навчанні краще справлятися з виявленням об’єктів усіх класів, незалежно від їхньої частоти в наборі даних.
precision	Показує частку правильно ідентифікованих позитивних результатів від усіх позитивних результатів, передбачених моделлю. Чим вище значення, тим краще модель розпізнає об’єкти.
recall	Визначає як часто модель правильно ідентифікує позитивні екземпляри з усіх реальних позитивних зразків у наборі даних. Високий показник означає, що модель добре виявляє об’єкти і не пропускає багато з них.
mAP50	Показує середню точність для кожного класу при пороговому значенні IoU 0,5, а потім обчислює середнє значення для всіх класів. Це означає, що об’єкт вважається правильно виявленим, якщо він збігається з еталонним об’єктом щонайменше на 50%. Чим більше значення тим краще продуктивність.
mAP50-95	Середня точність при різних порогах IoU в діапазоні від 0,50 до 0,95. Це дає комплексне уявлення про роботу моделі на різних рівнях складності розпізнавання.
confusion matrix	Це таблиця, яка підсумовує результати роботи моделі, порівнюючи передбачені нею мітки з істинними мітками. Вона відображає кількість істинно позитивних, істинно негативних, хибно позитивних і хибно негативних прогнозів моделі.

На рисунках 4.2 – 4.4 представлені графіки показників, які були розраховані в процесі навчання моделей за різних оптимізаторів.

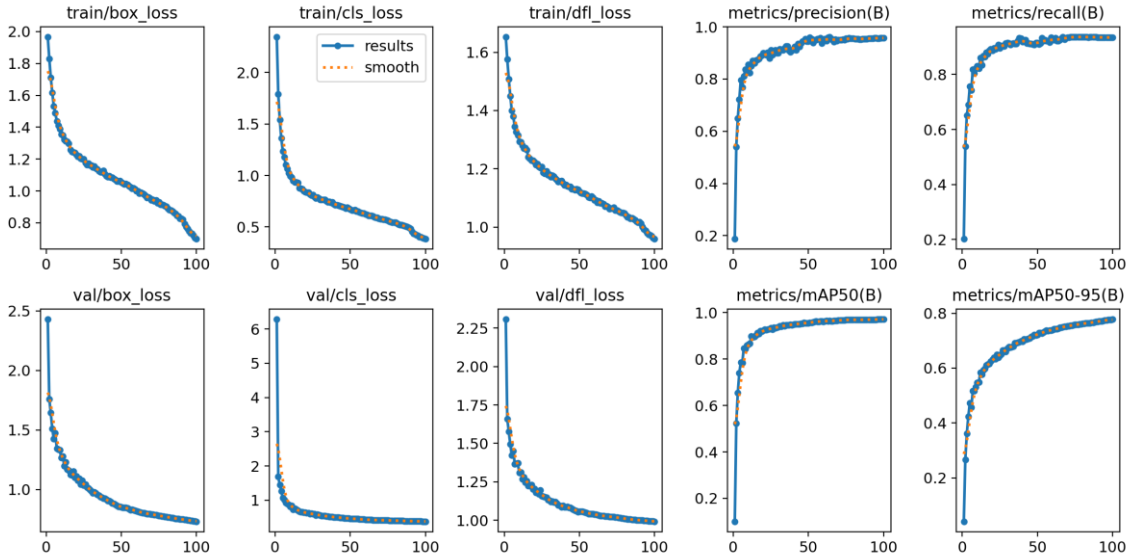


Рисунок 4.2 – Метрики за оптимізатора Adam

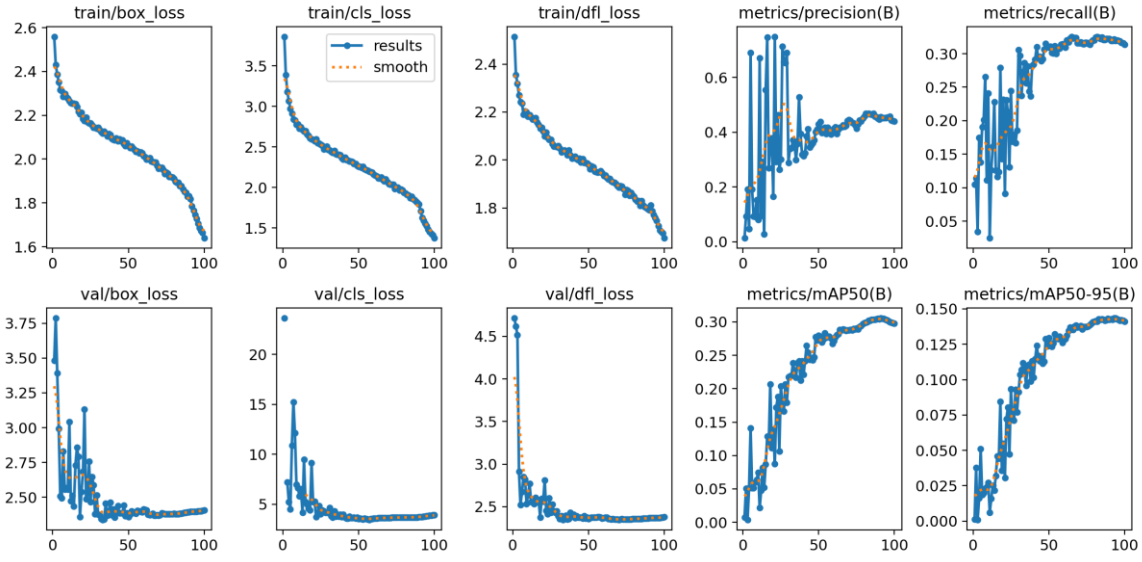


Рисунок 4.3 – Метрики за оптимізатора RMSProp

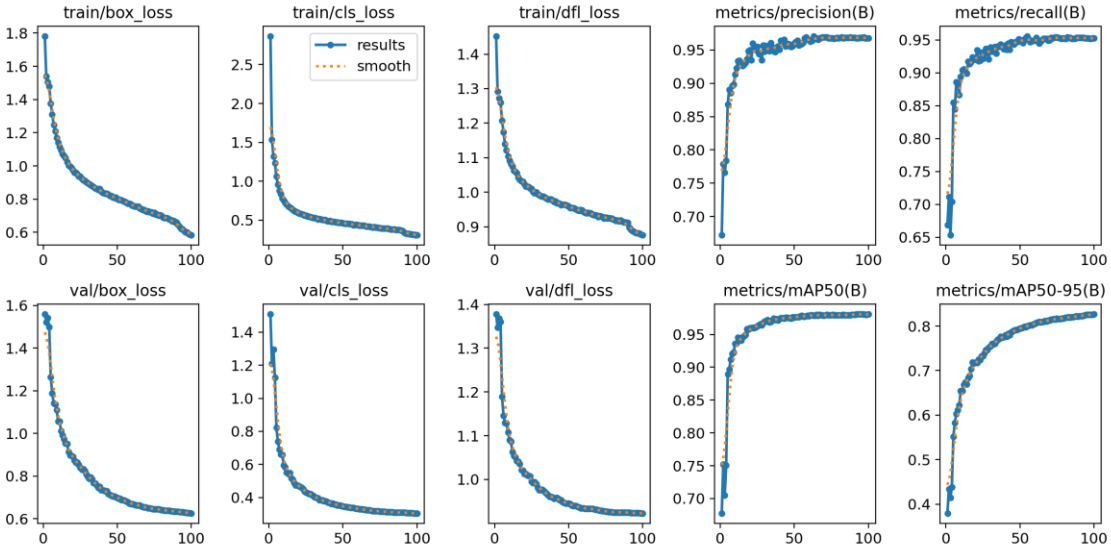


Рисунок 4.4 – Метрики за оптимізатора SGD

На рисунках 4.5 – 4.7 запропоновані нормалізовані матриці невідповідностей для заданих класифікаторів.

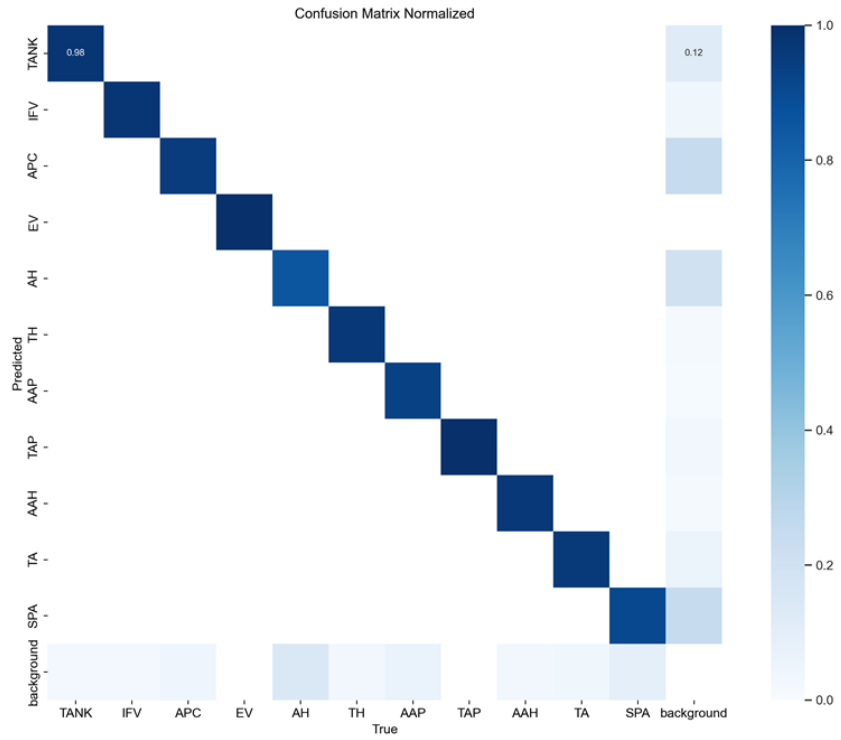


Рисунок 4.5 – Матриця невідповідностей за оптимізатора Adam

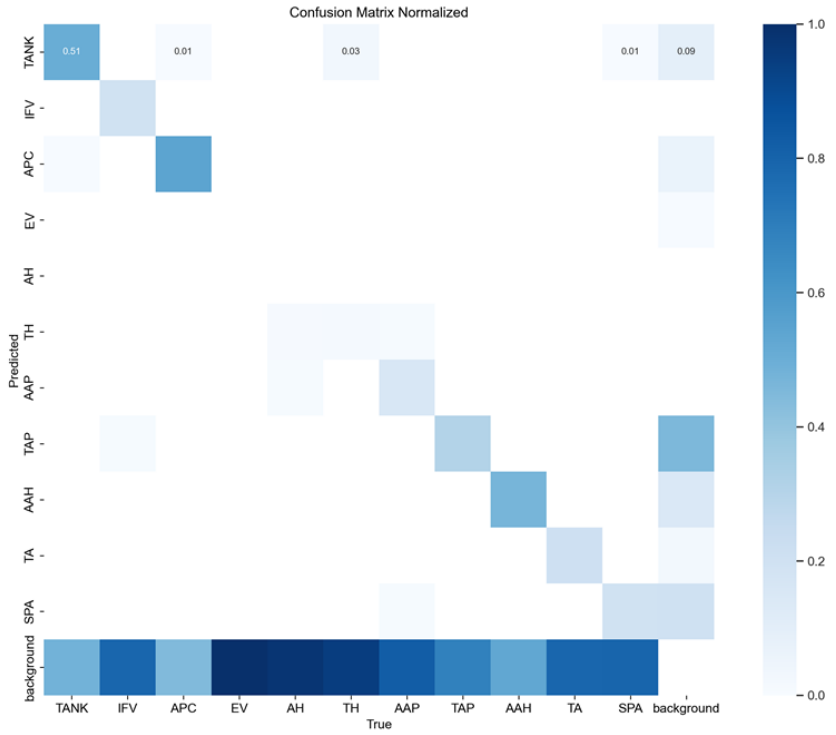


Рисунок 4.6 – Матриця невідповідностей за оптимізатора RMSProp

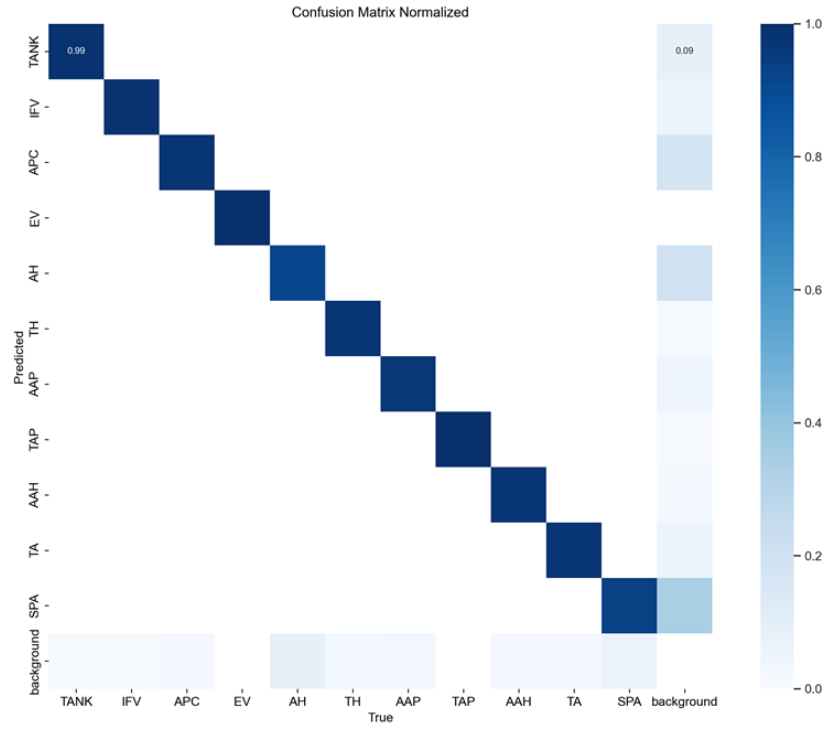


Рисунок 4.7 – Матриця невідповідностей за оптимізатора SGD

Підсумовуючи, оформимо отримані метрики у вигляді таблиці 4.2 для кожного класифікатора. В процесі навчання зберігаються ваги з найкращими показниками mAP50. Тому при побудові цієї таблиці був врахований цей факт.

Таблиця 4.2 – Порівняння метрик для різних оптимізаторів

Оптимізатор/ метрика	precision	recall	mAP50	mAP50-95
Adam	0.95805	0.93437	0.97147	0.77686
RMSProp	0.455	0.31964	0.3057	0.14294
SGD	0.96854	0.95283	0.98086	0.82519

За результатами навчання було визначено, що оптимізатор SGD досяг найкращих показників за усіма метриками. Модель, яка використовувала алгоритм Adam, досягла доволі непоганих результатів, які трохи програють SGD, але все одно є дуже високими та точними. Найгірші показники виявилися у алгоритму RMSProp. Значення mAP50 в середньому на 219,33% гірше за своїх конкурентів, що робить модель, навчену за цим оптимізатором, абсолютно не придатною до використання в реальних застосунках.

Досліджуючи матрицю невідповідностей оптимізатора SGD, можна дійти висновку, що модель показує винятковий рівень точності у визначенні об’єктів на зображенні. Також варто звернути увагу, що елементи діагоналі матриці позначені найтемнішим кольором зі спектру, на відміну від всіх інших клітинок, що свідчить про те, що модель не зробила помилкових класифікацій під час оцінювання і всі об’єкти були визначені правильно.

Ці висновки підкреслюють високу точність моделі у виявленні об’єктів з обраного набору даних. Опираючись на даний аналіз, для подальшого використання була обрана модель YOLOv8n, натренована з використанням оптимізатора SGD.

4.2 Результат роботи веб-застосунку

Після визначення найкращих ваг моделі YOLO слід подбати про їх розгортання в застосунку. Для цього достатньо перемістити файл з моделлю у корінь застосунку і запустити додаток. Перевірка спроможності додатку відбувалася на тестовій частині набору даних. На рисунках 4.8 – 4.10 представлені результати розпізнавання військових об’єктів на зображеннях з точки зору користувача додатку, а на рисунках 4.11 – 4.13 показані зображення роботи додатку, збережені в папку з результатами.

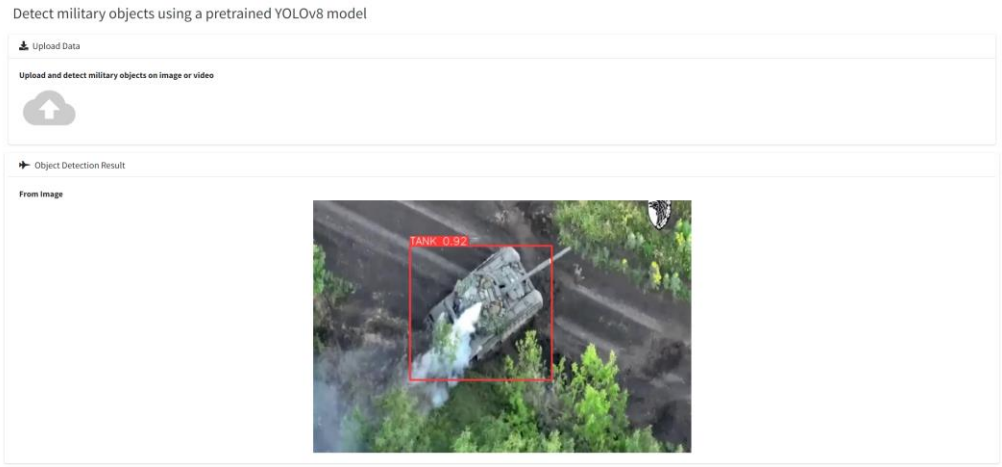


Рисунок 4.8 – Результат роботи застосунку

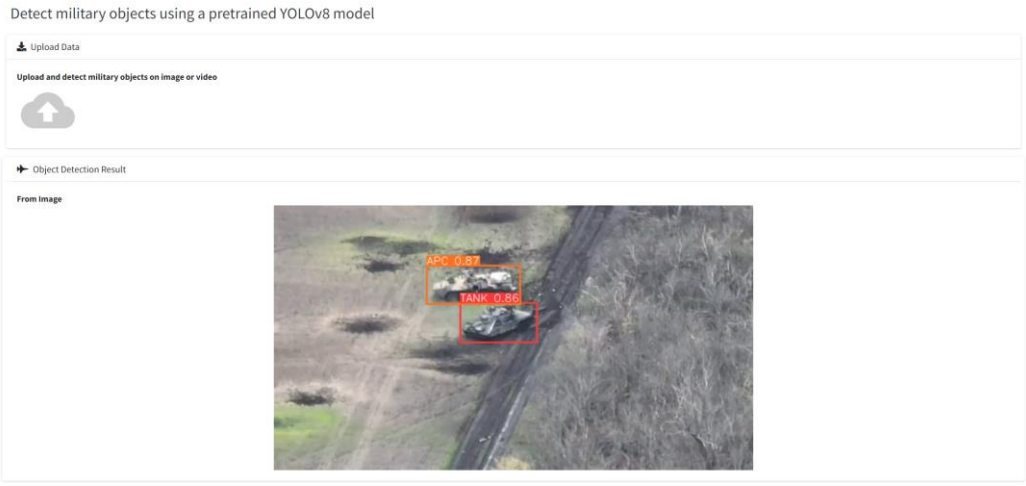


Рисунок 4.9 – Результат роботи застосунку

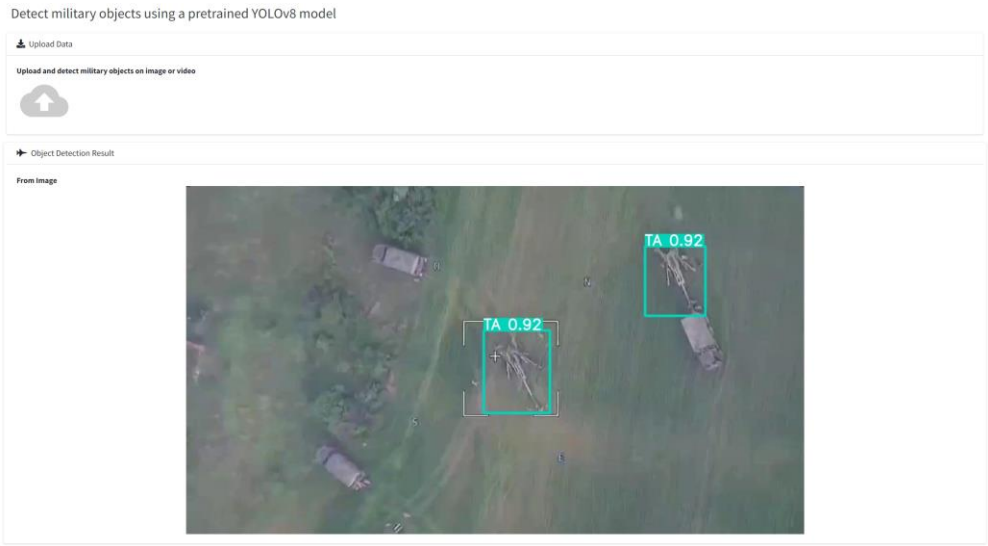


Рисунок 4.10 – Результат роботи застосунку

Рисунок 4.11 – Результат роботи застосунку

Рисунок 4.12 – Результат роботи застосунку

Рисунок 4.13 – Результат роботи застосунку

Таким чином, можна зазначити, що найкращим оптимізаційним алгоритмом для наявного набору даних є SGD, що підкреслюється високими значеннями отриманих метрик. Найгірший результат був досягнутий алгоритмом RMSProp, що свідчить про необхідність додаткового конфігурування даного оптимізатора та загалом параметрів моделі. Після відбору ваг їх було розгорнуто у веб-застосунку та було проведено тестування роботи моделі на різних зображеннях з тестового набору даних. Тестування показало високу точність моделі, яка розпізнає техніку на знімках з різним рівнем освітлення, текстурним оточенням та за наявності завад.

ВИСНОВКИ

У рамках кваліфікаційної роботи були розглянуті та проаналізовані інформаційні джерела в області комп’ютерного зору та військової справи. Визначені два підходи до ідентифікації об’єктів на зображеннях: одноетапний і двоетапний та, враховуючи, що у військовій справі такі характеристики як швидкодія та точність є вирішальними, подальший розгляд був обмежений одноетапними алгоритмами розпізнавання.

Було вивчено більш детально можливості використання технології комп’ютерного зору у військовій сфері та визначено, що дана предметна галузь через брак необхідної вхідної інформації для навчання, апаратних та безпекових обмежень та інших проблем є актуальною та перспективною для вивчення.

Був детально вивчений алгоритм розпізнавання YOLO та засади, які зробили цей алгоритм інноваційним. В результаті було встановлено, що модель YOLOv8 відмінно підходить для подальшої постановки експерименту та має величезну наявність підтримки у спільноти ШІ, зручність використання та впровадження.

Описані засоби розробки та набір даних, який використовувався в даній роботі. Була навчена найменша модель із сімейства YOLOv8n за трьох оптимізаторів: SGD, RMSprop та Adam. В результаті навчання визначено, що найкращим оптимізаційним алгоритмом для наявного набору даних є SGD, що підкреслюється високими значеннями отриманих метрик. Алгоритм RMSProp показав найгірші результати зі значенням mAP50 в середньому на 219,33% меншим за своїх конкурентів.

В кінцевому результаті модель навчена за допомогою оптимізаційного алгоритму SGD була розгорнута у веб-застосунок та протестована на різних аерофотознімках з тестової частини набору даних.

Результати дослідження апробовано у вигляді тез доповідей [34].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Що таке комп’ютерний зір (Computer Vision, CV)? | TheTransmitted. *TheTransmitted*. URL: https://thetransmitted.com/adlucem/shho-take-kompyuternyj-zir-computer-vision-cv (дата звернення: 16.05.2024).

2. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*. 2017. Т. 60, № 6. С. 84–90. URL: https://doi.org/10.1145/3065386 (дата звернення: 16.05.2024).

3. Computer Vision Market Size, Industry Forecast - 2032. *Allied Market Research*. URL: https://www.alliedmarketresearch.com/computer-vision-market-A12701 (дата звернення: 16.05.2024).

4. Saravanan R., Sujatha P. A state of art techniques on machine learning algorithms: a perspective of supervised learning approaches in data classification. *2018 second international conference on intelligent computing and control systems (ICICCS)*, м. Madurai, India, 14–15 черв. 2018 р. 2018. URL: https://doi.org/10.1109/iccons.2018.8663155 (дата звернення: 16.05.2024).

5. Alzubi J., Nayyar A., Kumar A. Machine learning from theory to algorithms: an overview. *Journal of physics: conference series*. 2018. Т. 1142. С. 012012. URL: https://doi.org/10.1088/1742-6596/1142/1/012012 (дата звернення: 16.05.2024).

6. Ling Q. Machine learning algorithms review. *Applied and computational engineering*. 2023. Т. 4, № 1. С. 91–98. URL: https://doi.org/10.54254/2755-2721/4/20230355 (дата звернення: 16.05.2024).

7. Spatial pyramid pooling in deep convolutional networks for visual recognition / K. He та ін. *IEEE transactions on pattern analysis and machine intelligence*. 2015. Т. 37, № 9. С. 1904–1916.

URL: https://doi.org/10.1109/tpami.2015.2389824 (дата звернення: 16.05.2024).

8. Zhang H., Cloutier R. S. Review on one-stage object detection based on deep learning. *EAI endorsed transactions on e-learning*. 2022. Т. 7, № 23. С. 174181. URL: https://doi.org/10.4108/eai.9-6-2022.174181 (дата звернення: 16.05.2024).

9. Kulishova N., Bodyanskiy Y., Pliss I. The extended generalized neo-fuzzy network and its online learning in image recognition problem. *2019 10th IEEE international conference on intelligent data acquisition and advanced computing systems: technology and applications (IDAACS)*, м. Metz, France, 18–21 верес. 2019 р. 2019. URL: https://doi.org/10.1109/idaacs.2019.8924367 (дата звернення: 16.05.2024).

10. Lau S. The good, the bad, and the tradecraft: HUMINT and the ethics of psychological manipulation. *Intelligence and national security*. 2022. С. 1–19. URL: https://doi.org/10.1080/02684527.2022.2129159 (дата звернення: 16.05.2024).

11. Aldrich R. J. From sigint to cyber: a hundred years of Britain’s biggest intelligence agency. *Intelligence and national security*. 2021. Т. 36, № 6. С. 910–917. URL: https://doi.org/10.1080/02684527.2021.1899636 (дата звернення: 16.05.2024).

12. IMINT, FININT, and MASINT. *The U.S. domestic intelligence enterprise*. 2015. С. 289–312. URL: https://doi.org/10.1201/b18794-8 (дата звернення: 16.05.2024).

13. Ciolponea C.-A., Bârsan G. NATO corps HQ – land component – integration of unmanned aerial vehicles (uavs). *Land forces academy review*. 2022. Т. 27, № 4. С. 323–332. URL: https://doi.org/10.2478/raft-2022-0041 (дата звернення: 16.05.2024).

14. Real-Time small drones detection based on pruned yolov4 / H. Liu та ін. *Sensors*. 2021. Т. 21, № 10. С. 3374. URL: https://doi.org/10.3390/s21103374 (дата звернення: 16.05.2024).

15. Jain A. K., Ratha N. K., Lakshmanan S. Object detection using gabor filters. *Pattern recognition*. 1997. Т. 30, № 2. С. 295–309. URL: https://doi.org/10.1016/s0031-3203(96)00068-4 (дата звернення: 16.05.2024).

16. Nelson B. N. Automatic vehicle detection in infrared imagery using a fuzzy inference-based classification system. *IEEE transactions on fuzzy systems*. 2001. Т. 9, № 1. С. 53–61. URL: https://doi.org/10.1109/91.917114 (дата звернення: 16.05.2024).

17. Sun S.-G. Automatic target detection using binary template matching. *Optical engineering*. 2005. Т. 44, № 3. С. 036401. URL: https://doi.org/10.1117/1.1869997 (дата звернення: 16.05.2024).

18. Neagoe V.-E., Carata S.-V., Ciotec A.-D. An advanced neural network-based approach for military ground vehicle recognition in sar aerial imagery. *Scientific research and education in the air force*. 2016. Т. 18, № 1. С. 41–48. URL: https://doi.org/10.19062/2247-3173.2016.18.1.5 (дата звернення: 16.05.2024).

19. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*. 2017. Т. 60, № 6. С. 84–90. URL: https://doi.org/10.1145/3065386 (дата звернення: 16.05.2024).

20. You only look once: unified, real-time object detection / J. Redmon та ін. *2016 IEEE conference on computer vision and pattern recognition (CVPR)*, м. Las Vegas, NV, USA, 27–30 черв. 2016 р. 2016. URL: https://doi.org/10.1109/cvpr.2016.91 (дата звернення: 16.05.2024).

21. Aggarwal A. YOLO explained. *Medium*. URL: https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31 (дата звернення: 16.05.2024).

22. What is YOLO? An in-depth introduction to object detection in computer vision. *Open Source Tools for Scaling Computer Vision Projects*. URL: https://www.ikomia.ai/blog/what-is-yolo-introduction-object-detection-computer-vision (дата звернення: 16.05.2024).

23. Terven J., Córdova-Esparza D.-M., Romero-González J.-A. A comprehensive review of YOLO architectures in computer vision: from yolov1 to yolov8 and YOLO-NAS. *Machine learning and knowledge extraction*. 2023. Т. 5, № 4. С. 1680–1716. URL: https://doi.org/10.3390/make5040083 (дата звернення: 16.05.2024).

24. Hussain M. YOLO-v1 to yolo-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection. *Machines*. 2023. Т. 11, № 7. С. 677. URL: https://doi.org/10.3390/machines11070677 (дата звернення: 16.05.2024).

25. YOLO algorithm for object detection explained [+examples]. *V7 Gen AI & Darwin | Accelerate AI Development & Automation*. URL: https://www.v7labs.com/blog/yolo-object-detection (дата звернення: 16.05.2024).

26. Xiao C. YOLOv7, YOLOv8, YOLOv9 and YOLO-World. *Mastering computer vision with pytorch and machine learning*. 2024. С. 8–1–8–22. URL: https://doi.org/10.1088/978-0-7503-6244-3ch8 (дата звернення: 16.05.2024).

27. TIOBE Index - TIOBE. *TIOBE*. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 16.05.2024).

28. Ultralytics | revolutionizing the world of vision AI. *Ultralytics | Revolutionizing the World of Vision AI*. URL: https://www.ultralytics.com/ (дата звернення: 16.05.2024).

29. Military Decision-Making Dataset. *Kaggle: Your Machine Learning and Data Science Community*. URL: https://www.kaggle.com/datasets/nzigulic/military-equipment/data (дата звернення: 16.05.2024).

30. GitHub - darkpgmr/darklabel: video/image labeling and annotation tool. *GitHub*. URL: https://github.com/darkpgmr/DarkLabel (дата звернення: 16.05.2024).

31. Ultralytics. YOLO performance metrics. *Home - Ultralytics YOLOv8 Docs*. URL: https://docs.ultralytics.com/guides/yolo-performance-metrics (дата звернення: 16.05.2024).

32. How to interpret a confusion matrix for a machine learning model. *Evidently AI - Open-Source ML Monitoring and Observability*. URL: https://www.evidentlyai.com/classification-metrics/confusion-matrix (дата звернення: 16.05.2024).

33. Accuracy vs. precision vs. recall in machine learning: what's the difference?. *Evidently AI - Open-Source ML Monitoring and Observability*. URL: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall (дата звернення: 16.05.2024).

34. Любименко Р.С. Виявлення військових цілей за допомогою методів комп'ютерного зору. *28-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті»:* матеріали форуму. Т. 7. Харків: ХНУРЕ, 2024. С. 76 – 77.