МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

Філімонов Данило Олегович

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
канд. техн. наук, доцент

_____Оксана ЗЕЛІНЬСКА
«__»_____20__ р.

ДОСЛІДЖЕННЯ МЕТОДІВ І ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЕТЕКЦІЇ ОБ'ЄКТІВ НА ТЕРИТОРІЯХ АКТИВНИХ БОЙОВИХ ДІЙ

Спеціальність 122 Комп'ютерні науки

Кваліфікаційна (магістерська) робота
(відповідно до стандарту спеціальності та ОП)

Науковий керівник:
Р.М. Бабаков, професор кафедри
інформаційних технологій,
докт. техн. наук, доцент
_____
(підпис)

Оцінка: _____ / _____ /
_____
(бали/за шкалою ЄКТС/за національною шкалою)
Голова ЕК:_____
(підпис)

Вінниця — 2024

АНОТАЦІЯ

Філімонов Д.О. Дослідження методів і інструментальних засобів детекції об’єктів на територіях активних бойових дій. Спеціальність 122 «Комп’ютерні науки», Освітня програма «Комп’ютерні технології обробки даних». Донецький Національний університет імені Василя Стуса.

У кваліфікаційній роботі вирішено актуальну науково-практичну проблему детекції об’єктів в зонах активних бойових дій із використанням для цієї мети нейронної мережі YOLOv5. Вирішення цієї проблеми передбачає вирішення таких завдань, як створення набору даних за допомогою інструменту Make Sense, навчання моделі за допомогою Google Colab, розгортання та аналізу навченої моделі.

Досліджено сучасні інструменти та методології виявлення об’єктів, їх сильні сторони та недоліки. Проаналізовано архітектуру, переваги та відмінні риси нейронної мережі YOLOv5.

Розроблено нейронну мережу з використанням хмарних платформ таких як Google Colab на базі YOLOv5, яка здатна проводити точне виявлення об’єктів військової техніки, що знаходиться у процесі ведення бойових дій. Проведено оптимізацію розробленої нейронної мережі за допомогою ітеративного навчання та вагових коригувань.

Ключові слова: нейронна мережа, детекція об’єктів, YOLOv5, Google Colab, Make Sense, вагові корегування.

Кваліфікаційна робота складається з 3 розділів, основна частина складає 61 сторінку, 1 таблицю, 29 рисунків, 53 використаних джерела інформації.

ABSTRACT

Filimonov D.O. Research of methods and instrumental means of detection of objects in the territories of active hostilities. Specialty 122 "Computer science", Educational program "Computer data processing technologies". Donetsk National University named after Vasyl Stus.

In the qualification work, the actual scientific and practical problem of object detection in active combat zones was solved using the YOLOv5 neural network for this purpose. Solving this problem involves solving tasks such as creating a dataset using the Make Sense tool, training a model using Google Colab, and deploying and analyzing the trained model.

Modern tools and methodologies for object detection, their strengths and weaknesses **are investigated**. The architecture, advantages and distinctive features of the YOLOv5 neural network are analyzed.

A neural network **has been developed** using cloud platforms such as Google Colab based on YOLOv5, which is capable of accurately detecting military equipment in the process of combat operations. The optimization of the developed neural network is carried out by means of iterative training and weight adjustments.

Keywords: neural network, object detection, YOLOv5, Google Colab, Make Sense, weight adjustments.

The qualification work consists of 3 sections, the main part consists of 61 pages, 1 table, 29 figures, 53 used sources of information.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЕНЬ І ТЕРМІНІВ

LIDAR — Light Detection and Ranging — технологія виявлення об’єктів за допомогою світла;

HUMINT — Human Intelligence — людська розвідка;

CNN — Convolutional Neural Networks — Згорткові нейронні мережі;

MSE — Mean Squared Error— Середнє квадратичне відхилення. Це функція втрат, яка використовується для оцінки точності моделі;

NMS — Non-Maximum Suppression — Немаксимальне придушення;

БТР — Бронетранспортер;

ППО — Протиповітряна оборона;

mAP — mean Average Precision— Стандартна метрика для оцінки продуктивності моделей виявлення об’єктів.;

ВСТУП

У непередбачуваному та постійно мінливому ландшафті збройного конфлікту швидке та точне прийняття рішень стає вирішальним. Динамічний і непередбачуваний характер активних бойових зон створює специфічні проблеми для виявлення різноманітних об’єктів як військового, так і цивільного призначення. Швидке і точне виявлення сприяє захисту військовослужбовців та цивільних осіб, прискорює збір цінних розвідувальних даних про ворожу діяльність та забезпечує координацію зусиль між різними військовими підрозділами. Ефективна детекція особливо важлива для виявлення низки загроз, від важкої артилерії та бронетехніки до піхотних підрозділів. Йдеться не лише про прямі погрози; виявлення активів противника сприяє розвідці, яка допомагає стратегічному плануванню. Ця інформація стає основою тактичних рішень, відкриваючи можливості для нанесення скоординованих ударів і тим самим істотно збільшуючи шанси на успіх операції.

Наукове дослідження, проведене в даній роботі, спрямоване на адаптацію існуючих методів та інструментальних засобів розпізнавання зображень до обставин, пов’язаних із веденням активних бойових дій. В основу роботи покладене дослідження можливостей нейронної мережі YOLOv5 із розпізнавання об’єктів військового призначення під час їхнього функціонування.

Об’єктом цього дослідження є методи і засоби розпізнавання об’єктів на графічних зображеннях.

Предмет дослідження – технологія розпізнавання військової техніки за допомогою нейронної мережі YOLOv5.

У кваліфікаційній роботі вирішені наступні **завдання**:

1. Комплексний огляд традиційних і сучасних систем виявлення об’єктів.

2. Аналіз ефективності, точності і надійності подібних систем.

3. Проведено адаптацію можливостей нейронної мережі YOLOv5 до особливостей предметної області.

4. Підготовка дата сету для роботи нейронної мережі із використанням інструментів Make Sense.

5. Тренування нейронної мережі за допомогою підготовленого дата сету в хмарному середовищі Google Colab.

6. Тестування навченої нейронної мережі та оцінка ефективності її роботи.

Для досягнення цілей використані наступні наукові підходи та методи:

- Огляд літератури;
- Аналітика даних;
- Обчислювальне моделювання;
- Емпіричне тестування.

Наукова новизна дослідження полягає в інтеграції передових обчислювальних алгоритмів для підвищення можливостей виявлення об’єктів в активних зонах бойових дій.

Практична цінність цього дослідження:

1. Підвищення безпеки: покращене виявлення об’єктів може сприяти кращій обізнаності про ситуацію, тим самим зменшуючи кількість втрат.

2. Тактична перевага: точне виявлення об’єктів у реальному часі може забезпечити тактичну перевагу у військових операціях.

РОЗДІЛ 1 ТЕОРЕТИЧНА ОСНОВА ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ.

1.1 Важливість виявлення об’єктів у зонах бойових дій

На полі бою динаміка, що постійно змінюється, посилює терміновість виявлення об’єктів. Ескалація типів загроз, від традиційної піхотної тактики до використання передових технологій, таких як безпілотники та кібер-війна, підкреслює необхідність постійної пильності та адаптації. Одна з найбільш безпосередніх і важливих ролей, яку відіграє виявлення об’єктів у бойових ситуаціях, збереження життя. Знання розташування та характеру різних об’єктів, особового складу, зброї чи транспортних засобів противника дозволяє приймати більш обґрунтовані рішення. Озброєний точними та своєчасними даними, військовий персонал може вжити відповідних заходів для нейтралізації загрози або вжити заходів ухилення. Це не тільки підвищує безпеку наших військ, але й рятує життя на боці противника, надаючи можливості для капітуляції або переговорів, а не до смертоносних зіткнень.

Не менш важливим є захист, який він забезпечує життям цивільного населення. Системи виявлення об’єктів можуть ідентифікувати цивільних осіб та цивільну інфраструктуру, допомагаючи мінімізувати побічний збиток під час військових операцій. Це не лише моральний імператив, але й вирішальне значення для отримання підтримки та співпраці місцевого населення в зонах конфлікту. Зменшення шкоди цивільному населенню може допомогти зміцнити довіру та прокласти шлях для зусиль зі стабілізації та відновлення після конфлікту.

Виявлення об’єктів виходить далеко за межі негайної нейтралізації загрози. Це важлива частина збору розвідданих. Знання позицій, сил і можливостей противника стає найважливішим для планування стратегічних і тактичних маневрів. У цьому сенсі виявлення об’єктів функціонує як мультиплікатор військової ефективності, оскільки воно покращує якість

розвідувальних даних і тим самим збільшує шанси на успіх місії. Це дозволяє вищому командуванню отримати чітку картину поля бою, забезпечуючи вкрай необхідний контекст для оптимального розгортання військ і ресурсів. Ефективне виявлення об'єктів це не просто односторонній процес; це часто вимагає міжвідомчої координації. Інформація з різних джерел, у тому числі з повітряної розвідки, наземних радарів та людської розвідки, повинна бути інтегрована в цілісну картину. Такі скоординовані зусилля значно посилюються передовими технологіями виявлення цілей, створюючи синергію, яка покращує загальну військову ефективність.

В епоху спільної та інтернет-війни здатність виявляти та ділитися інформацією, пов'язаною з активами, у режимі реального часу є безцінною. Інтеграція даних із багатьох джерел дозволяє приймати більш обґрунтовані рішення, дозволяючи командирам швидко реагувати на зміну умов на полі бою. Крім того, обмін інформацією про виявлення об'єктів між різними військовими підрозділами та силами союзників покращує спільні операції, підвищуючи загальну ефективність зусиль коаліції у складних зонах бойових дій.

У постійно змінюваному ландшафті бойових дій, де супротивники постійно намагаються перегравати один одного, підтримка ефективності систем виявлення об'єктів вимагає непохитної відданості дослідженню та адаптації. Ця безперервна гонка озброєнь між технологіями виявлення та уникнення підкреслює динамічний характер цієї галузі, де випередження вимагає постійної уваги та інновацій. Сучасні системи виявлення часто вимагають значних ресурсів, включаючи значні інвестиції в технології, навчання та поточне обслуговування [1-2]. Виникають питання щодо стійкості, оскільки ці системи мають бути інтегровані в ширші військові бюджети та матеріально-технічне забезпечення, що спонукає до всебічної перевірки розподілу ресурсів і довгострокової життєздатності. Крім того, оперативна сумісність різних систем виявлення об'єктів стає ключовою, особливо в багатонаціональних коаліціях. Сумісні системи сприяють безперебійній співпраці між країнами-

партнерами та сприяють підвищенню показників успіху місій. Можливість обмінюватися даними та координувати зусилля підвищує загальну ефективність спільних операцій.

Варто зазначити, що прогрес у технології виявлення об’єктів, зокрема в розробці програмного забезпечення, став дедалі ефективнішим, адаптивним та доступнішим [3-4]. Гнучкість програмних рішень дозволяє швидко налаштовувати та оновлювати без необхідності масштабних модифікацій апаратного забезпечення чи великих фінансових витрат. Ця адаптивність є свідченням універсальності сучасних систем виявлення об’єктів. Підсумовуючи, ландшафт виявлення об’єктів у війні позначений постійним пошуком переваги. Щоб зберегти перевагу на цій динамічній арені, дослідження та здатність до адаптації є найважливішими. У той час як традиційні системи виявлення можуть викликати проблеми з ресурсами, розробка нового програмного забезпечення для виявлення об’єктів пропонує більш доступний і адаптований шлях вперед. Саме завдяки цим інноваціям військові сили можуть йти в ногу з вимогами сучасного конфлікту, що постійно змінюються, одночасно оптимізуючи використання наявних ресурсів.

1.2 Огляд існуючих інструментів та методів

Виявлення об’єктів у зонах активних бойових дій є ключовим і динамічним напрямком сучасної війни та загального розвитку технологій. Здатність ідентифікувати та знаходити об’єкти в таких критичних середовищах має важливе значення для захисту як військовослужбовців, так і цивільних осіб, а також для досягнення успіху військової місії. Рішення на основі програмного забезпечення виступають як авангард, використовуючи потужність передових технологій для підвищення ситуаційної обізнаності та сприяння стратегічній перевазі. У міру розвитку військових дій змінюється і арсенал методів виявлення об’єктів. Традиційно апаратні рішення, такі як

LIDAR і радар, були непохитними, забезпечуючи надійні засоби виявлення об’єктів на різних територіях і в навколишніх умовах. Однак швидкий розвиток технологій, керованих програмним забезпеченням, змінив ландшафт. Алгоритми глибокого навчання, що базуються на нейронних мережах і штучному інтелекті, започаткували еру неперевершеної точності та адаптивності. Мова програмування Python, універсальна та широко поширена, стала важливим кроком у розробці систем виявлення об’єктів. Простота використання, великі бібліотеки та надійна підтримка спільноти зробили її кращим вибором для дослідників і розробників. Python служить каналом, через який навчаються та розгортаються складні нейронні мережі, що дозволяє створювати високоефективні моделі виявлення об’єктів.

У сфері детекції об’єктів у зоні активних бойових дій важливу роль відіграють непрограмні методи, такі як LIDAR, радіолокаційна технологія та людський інтелект (HUMINT). Ці методології пропонують явні переваги, але також мають власний набір обмежень порівняно з їхніми аналогами на основі програмного забезпечення. Системи LIDAR використовують лазерні імпульси для вимірювання відстаней і створення детальних тривимірних зображень навколишнього середовища[5]. Історично використовувався переважно для картографування та аналізу місцевості, LIDAR знайшов застосування у виявленні об’єктів, особливо в умовах погіршення видимості, наприклад у тумані чи диму. Здатність LIDAR забезпечувати точні вимірювання відстані є помітною перевагою, що робить його ефективним для виявлення нерухомих і повільно рухомих об’єктів. Однак його використання в динамічних високошвидкісних бойових сценаріях може бути обмеженим. Крім того, системи LIDAR, як правило, громіздкі та дорожчі, ніж деякі альтернативи програмного забезпечення, що може вплинути на їх розгортання в умовах обмежених ресурсів[6-9]. Радіолокаційна технологія, яка давно зарекомендувала себе в авіації та на морі, є потужним інструментом для виявлення об’єктів. Радарні системи використовують радіохвилі для виявлення та відстеження об’єктів, незалежно від погодних умов. Ця стійкість

до несприятливих погодних умов, включаючи дощ, туман і темряву, є значною перевагою. Утиліта радара поширюється на виявлення не лише нерухомих об’єктів, але й цілей, що швидко рухаються, наприклад літаків і ракет [10]. Тим не менш, радару може бути важко ідентифікувати менші та непомітні об’єкти, і його точність може бути втрачена в нестійкому середовищі. Крім того, радіолокаційні системи можуть бути чутливими до електронних засобів протидії, що підкреслює необхідність постійного технологічного розвитку.

Традиційні методи розвідки, які покладаються на людей-інформаторів і польових агентів, залишаються безцінними для виявлення об’єктів у контекстах, які включають загрози, орієнтовані на людину, та ідентифікацію критичних цілей[11]. Людський інтелект пропонує гнучкість збору інформації за допомогою розмов, інтерв’ю та спостережень, що є особливо актуальним у складних динамічних середовищах. Однак HUMINT має свої обмеження, насамперед щодо своєчасності та надійності. Людські ресурси можуть зіткнутися з небезпекою, а інформація, яку вони надають, може бути суб’єктивною, залежною від індивідуальних упереджень або неповного розуміння. Крім того, людський фактор створює потенціал для шпигунства та контррозвідувальної діяльності [12-14].

На відміну від цих непрограмних методів, програмні системи виявлення об’єктів, керовані машинним навчанням, алгоритмами або штучним інтелектом, пропонують кілька переваг. Вони можуть швидко обробляти величезні обсяги даних, адаптуватися до мінливих сценаріїв і досягти успіху в ідентифікації складних шаблонів і об’єктів різного розміру. Крім того, вони можуть працювати незалежно та інтегруватися з іншими технологіями[15]. Глибоке навчання, підмножина машинного навчання, започаткувало зміну парадигми виявлення об’єктів. Згорткові нейронні мережі (CNN), клас моделей глибокого навчання, виявилися особливо ефективними для завдань виявлення на основі зображень. Принцип їх роботи полягає в наступному[16-19]:

1. Попередня обробка даних. Для алгоритмів глибокого навчання потрібен набір даних зображень із мітками, де кожне зображення пов’язане з об’єктами в ньому[20-22];

2. Згорткові шари. Дія CNN починається з серії згорткових шарів. Ці шари застосовують фільтри (ядра) до вхідного зображення. Кожен фільтр фокусується на виявленні певних особливостей, таких як краї, текстури або кольори;

3. Об’єднання шарів. Шари об’єднання зменшують дискретизацію карт функцій, створених згортковими шарами. Максимальне об’єднання, наприклад, зберігає найважливішу інформацію, одночасно зменшуючи просторові розміри[23];

4. Повністю підключені шари. Кінцеві рівні CNN є повністю пов’язаними. Вони беруть витягнуті ознаки та виконують завдання класифікації та локалізації;

5. Навчання. Під час навчання прогнози мережі порівнюються з базовими мітками (класами об’єктів і обмежувальними рамками). Різниця між прогнозами та базовою правдою кількісно визначається за допомогою функції втрат, наприклад середньоквадратичної помилки або втрат крос-ентропії;

6. Висновок. Під час логічного висновку навчена модель обробляє нове, невидиме зображення. Він виконує пряме поширення, щоб прогнозувати об’єкти, присутні на зображенні [24];

7. Вихід. Класи об’єктів: модель виводить виявлені класи об’єктів (наприклад, «автомобіль», «людина», «дерево»).

По суті, CNN досягають успіху у виявленні об’єктів, навчаючись розпізнавати особливості та шаблони на багатьох рівнях абстракції, від простих країв до складних структур об’єктів (рис. 1.1). Це ієрархічне виділення ознак дозволяє їм ідентифікувати об’єкти на зображеннях, що робить їх потужним інструментом для таких програм, як класифікація зображень, розпізнавання обличчя і автономні транспортні засоби. Здатність автоматично вивчати релевантні функції з необроблених даних те, що

відрізняє глибоке навчання та CNN, зокрема, як революційний підхід до виявлення об’єктів.

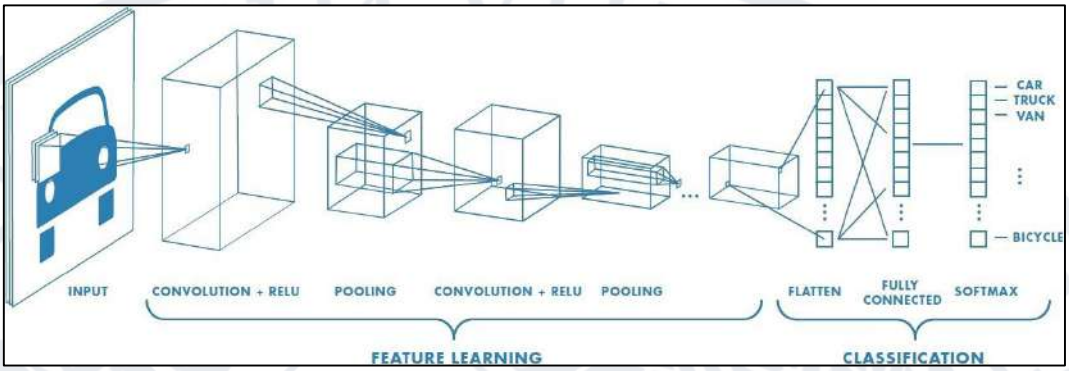


Рисунок 1.1 — Алгоритм роботи CNN

Наступним варто розглянути YOLO. Це відома система виявлення об’єктів у режимі реального часу, яка славиться своєю швидкістю та точністю. Інновація YOLO полягає в його здатності розділяти зображення на сітку та передбачати обмежувальні прямокутники та ймовірності класу для кожної комірки сітки за один прохід, звідси його назва «You Only Look Once»[25].

Алгоритм роботи YOLO наступний [26]:

1. YOLO ділить вхідне зображення на сітку комірок. Розмір цієї сітки залежить від обраної конфігурації, але зазвичай коливається від 7x7 до 19x19 комірок. Кожна комірка сітки відповідає за виявлення об’єктів у її межах [27];

2. Для кожної комірки в сітці YOLO передбачає кілька обмежувальних рамок. Ці обмежувальні рамки по суті є прямокутниками, які позначають положення, розмір і форму об’єктів. Кількість обмежувальних рамок, прогнозованих на комірку, зазвичай визначається конфігурацією моделі (наприклад, YOLOv3 передбачає 3 рамки на комірку)[27-28];

3. YOLO також передбачає ймовірність класу для кожної обмежувальної рамки. Ці ймовірності класів вказують на ймовірність того, що певний клас об’єктів (наприклад, «автомобіль», «людина», «собака») присутній у межах рамки;

4. Крім того, для кожного передбачення обмежувальної рамки YOLO обчислює оцінку достовірності. Ця оцінка відображає впевненість моделі в тому, що об’єкт існує всередині обмежувальної рамки та що передбачений клас є точним[29];

5. Одне з ключових нововведень YOLO полягає в тому, що всі ці прогнози (обмежувальні прямокутники, ймовірності класу та оцінки достовірності) робляться за один прохід через мережу. Це відрізняється від багатьох традиційних методів виявлення об’єктів, які включають кілька етапів і проходів;

6. Зробивши прогнози для всіх комірок і обмежувальних рамок, YOLO застосовує техніку, що називається немаксимальним придушенням (NMS). NMS усуває надлишкові та низько надійні виявлення. Він зберігає обмежувальну рамку з найвищим показником достовірності для кожного об’єкта, відкидаючи інші, які значно перекриваються;

7. Результатом процесу YOLO є набір обмежувальних рамок, кожна з яких пов’язана з міткою класу та оцінкою достовірності. Ці обмежувальні рамки точно визначають місцезнаходження та класифікують об’єкти у вхідному зображенні.

Підсумовуючи, принцип роботи YOLO обертається навколо поділу зображення на сітку (рис. 1.2).

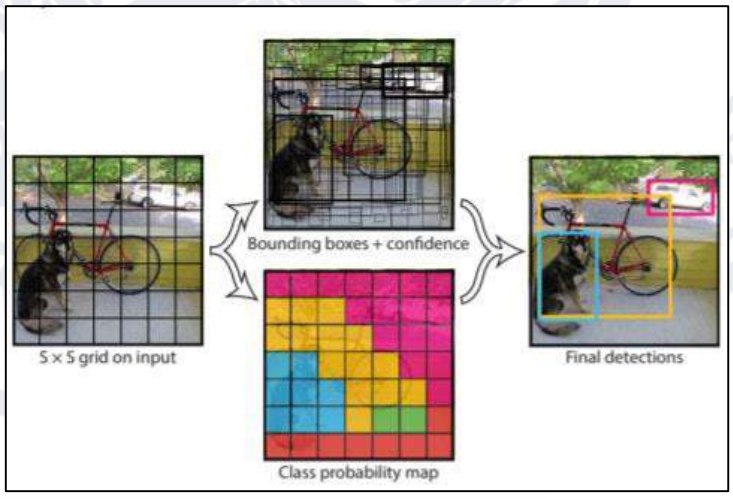


Рисунок 1.2 — Алгоритм роботи YOLO[30]

Прогнозування обмежувальних рамок, ймовірностей класу та балів достовірності для кожної комірки сітки, і все це за один прохід через мережу робить YOLO одним з найкращих методів детекції сьогодні. Його інноваційний однопрохідний підхід у поєднанні з немаксимальним придушенням забезпечує точне виявлення об’єктів у реальному часі, що робить його безцінним інструментом у різних програмах, які вимагають ефективного та швидкого розпізнавання об’єктів. За схожим принципом дії працює й Faster Region-CNN [31] це ще одна найсучасніша система детекції об’єктів. Це досягається шляхом створення набору пропозицій прямокутних областей, також відомих як обмежувальні рамки-кандидати. Ці пропозиції визначаються на основі опорних блоків, які є попередньо визначеними блоками різних масштабів і пропорцій.

OpenCV це комплексна бібліотека Python [32], яка надає попередньо підготовлені моделі виявлення об’єктів для таких завдань, як розпізнавання обличчя, виявлення пішоходів і загальне розпізнавання об’єктів. Віна широко використовується в різних областях, включаючи системи безпеки та робототехніку. Розберемо алгоритм роботи OpenCV для виявлення об’єктів детальніше[33]:

1. OpenCV починається з підготовки даних, яка включає отримання зображень або відеокадрів для виявлення об’єктів. Бібліотека підтримує різні формати зображень і відео та може захоплювати кадри з камер або відеофайли;

2. У контексті виявлення об’єктів OpenCV використовує попередньо навчені моделі глибокого навчання, які здатні витягувати значущі характеристики із зображень. Ці моделі навчені на великих наборах даних і можуть розпізнавати візерунки, текстури та форми на зображеннях;

2.1 OpenCV надає доступ до ряду попередньо навчених моделей, розроблених для конкретних завдань виявлення об’єктів. Наприклад: Каскади Хаара[34-35]: вони навчені виявляти об’єкти з чіткими рисами, як-от обличчя або тіла пішоходів; Моделі глибокого навчання: OpenCV можна інтегрувати з фреймворками глибокого навчання [36], такими як TensorFlow[37] і Caffe[38],

щоб використовувати попередньо підготовлені моделі для більш складних завдань розпізнавання об’єктів, включаючи загальне виявлення об’єктів;

3. Для виявлення об’єктів у різних масштабах OpenCV часто використовує піраміди зображень[39]. Піраміда зображення це масштабне представлення зображення з поступово зменшуваними версіями вихідного зображення (рис. 1.3). Це дозволяє алгоритмам виявлення об’єктів ефективно працювати з об’єктами різного розміру на одному зображенні;

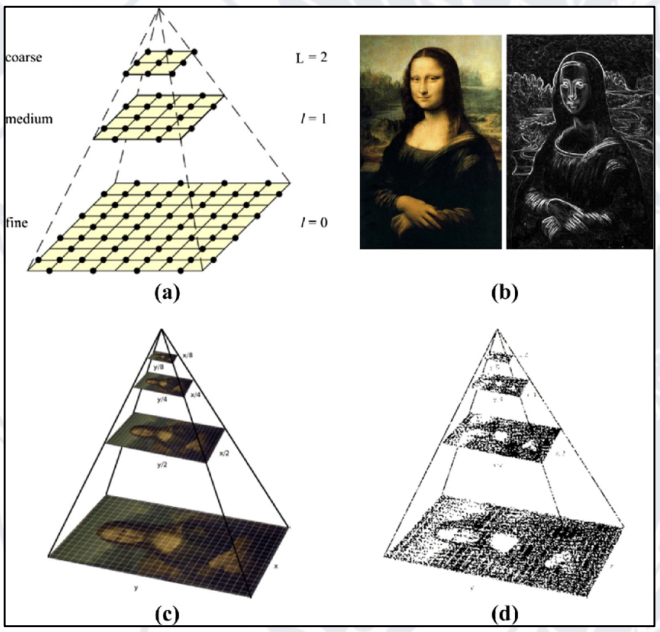


Рисунок 1.3 — Піраміда зображень

4. Після виявлення об’єкта OpenCV часто застосовує етапи постобробки, щоб уточнити та відфільтрувати результати. Поширені методи пост-обробки включають немаксимальне придушення (NMS) для усунення накладання неточності та порогове значення для збереження лише найбільш впевнених припущень;

1.3 Проблеми та обмеження в поточних методах

Основною метою кваліфікаційної роботи є критичний аналіз існуючих методів детекції об’єктів з метою розробки власного передового програмного

забезпечення. Щоб досягти цього, важливо оцінити обмеження та проблеми, пов’язані з такими популярними методами, як згорткові нейронні мережі (CNN), You Look Only Once (YOLO) і OpenCV. Розуміння цих обмежень не лише дає основу для розвитку, але й висвітлює сфери, які потребують інноваційних рішень та покращень.

У випадку з CNN, такі методи потребують багато обчислень через їх багаторівневу згортку, що робить їх непридатними для додатків у реальному часі або розгортання на малопотужних пристроях [40]. Наприклад, в наступному коді:

```
from tensorflow.keras import layers, models
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D((2, 2)),
]):
```

Є певні проблеми, а саме:

- Рівень Conv2D використовує 32 фільтри. Кожен фільтр сам є маленькою нейронною мережею, і це збільшує обчислення, необхідні на кожному рівні;
- Використовується розмір ядра (3,3), який знову потрібно розрахувати для кожного фільтра;
- Форма введення (150, 150, 3), що означає, що зображення має ширину та висоту 150 пікселів і 3 кольорові канали. Це пристойно висока роздільна здатність для завдань виявлення об’єктів і збільшує обчислювальні потреби;
- «ReLU» (Rectified Linear Unit) дешевший в плані обчислень, порівняно з деякими іншими функціями активації, але все одно збільшує обчислювальні можливості;
- MaxPooling зменшує просторові розміри (ширину та висоту) вхідного об’єму, але перед цим рівень Conv2D вже виконав основну частину обчислень;

- Можливі варіанти вирішення таких проблем:
- Використання меншої кількості фільтрів у шарах Conv2D. Це безпосередньо зменшить необхідну обчислювальну потужність, але може вплинути на точність;
- Кроки, які перевищують одиницю, зменшать просторові розміри результату, зменшуючи кількість обчислень. Однак вони можуть призвести до втрати деяких функцій;
- Якщо для завдання виявлення об’єктів не потрібна висока роздільна здатність, ви можете зменшити розмір вхідного зображення, щоб зменшити інтенсивність обчислення;
- Використання простішої функції активації, наприклад лінійної активації, може зменшити потреби в обчисленнях, але зазвичай ціною виразності моделі;
- Пропуск з’єднань може допомогти зменшити глибину мережі, зберігаючи при цьому продуктивність, таким чином заощаджуючи обчислювальні ресурси.

Підсумовуючи, аналіз коду CNN служить наочним прикладом обчислювальних проблем, пов’язаних із застосуванням таких складних методів машинного навчання для детекції об’єктів, особливо в середовищах з обмеженими ресурсами чи часом. Хоча CNN пропонують надійні можливості для розпізнавання складних об’єктів і були дуже успішними в різних програмах, вони не позбавлені своїх обмежень. Приклад з кодом виявив низку недоліків, пов’язаних з оптимізацією, починаючи від великої кількості використовуваних фільтрів, відсутності поступових згорток і закінчуючи загальною архітектурою мережі, усі вони сприяють неефективності обчислень. Мета цього аналізу полягала не в тому, щоб підірвати важливість чи можливості CNN, а радше підкреслити нагальну потребу в оптимізації. Це забезпечує інформаційну базу для майбутнього проекту.

Метод YOLO, також має в собі декілька доволі значних обмежень та проблем. Насамперед це його боротьба з захопленням дрібних деталей

об’єктів. Оскільки YOLO в першу чергу покладається на обмежувальні прямокутники для представлення виявлених об’єктів, він може не забезпечити точне окреслення об’єктів складної форми або складних меж. Це обмеження може бути критичним у програмах, як-от медична візуалізація, де точні межі об’єктів важливі для діагностики. Дизайн YOLO в першу чергу зосереджений на виявленні об’єкта одного зображення, а не на відстеженні об’єкта в кількох кадрах у відео. Це робить його менш придатним для відстеження об’єктів у часі, оскільки він за своєю суттю не підтримує ідентифікацію об’єктів. Хоча можна реалізувати відстеження об’єктів за допомогою YOLO, це часто вимагає додаткової пост-обробки та може бути менш точним порівняно з алгоритмами, призначеними для відстеження об’єктів. Моделі YOLO, особливо попередні версії, часто мають проблеми з узагальненням у широкому діапазоні масштабів об’єктів. Хоча YOLOv3 і наступні версії запровадили масштабне виявлення, вони все ще можуть працювати погано, коли об’єкти дуже різних розмірів співіснують на одній сцені. Це обмеження може перешкоджати продуктивності YOLO в сценаріях, де об’єкти демонструють значні варіації масштабу.

Також YOLO може мати проблеми, коли об’єкти в сцені частково закриті або перекриваються. Оскільки він передбачає об’єкти на основі їх видимих частин у клітинці сітки, закриті об’єкти можуть не отримати точного виявлення [41]. Це обмеження може бути проблематичним у сценаріях, де оклюзії є звичайним явищем, наприклад у багатолюдному міському середовищі або записах відеоспостереження з перекриттям об’єктів. Загалом, незважаючи на те, що YOLO є потужним і швидким алгоритмом виявлення об’єктів, він має кілька обмежень, зокрема чутливість до розміру об’єкта, високий рівень помилкових спрацьовувань, проблеми з дрібними деталями, відстеження об’єктів, узагальнення в масштабах, семантичну сегментацію, обробку оклюзій, дисбаланс даних навчання, обмежена адаптація до контексту сцени та відсутність підтримки виявлення 3D-об’єктів. Ці обмеження слід ретельно розглянути, вибираючи YOLO для конкретних програм, і, можливо,

знадобиться реалізувати потенційні рішення та обхідні шляхи для усунення цих недоліків в майбутньому проекті.

Хоча OpenCV можна вважати лідером серед вже описаних раніше методів, за рахунок її масштабності та підтримки. Існують деякі проблеми і в такого типу методах. Наприклад, класичні алгоритми OpenCV, такі як каскади HAAR, не є інваріантними щодо обертання, тобто вони можуть ефективно ідентифікувати лише об’єкти, які орієнтовані певним чином. Ця негнучкість може бути значним недоліком у реальних програмах, де об’єкти можуть з’являтися в різних орієнтаціях через рух або фактори навколишнього середовища. Хоча OpenCV пропонує більш просунуті методи виявлення об’єктів на основі машинного навчання, такі як реалізація Support Vector Machines (SVM) або Deep Learning, вони часто супроводжуються власним набором проблем. Ці методи можуть бути дорогими з обчислювальної точки зору та можуть потребувати спеціального апаратного прискорення для роботи в реальному часі. Крім того, процес навчання для цих моделей зазвичай, вимагає великого набору даних і значної обчислювальної потужності. Бібліотека має круту криву навчання, особливо при переході від традиційних методів комп’ютерної візуалізації до методів машинного навчання. Ця складність може перешкодити використовувати всі можливості OpenCV.

OpenCV, як правило, є бібліотекою низького рівня, для завдань комп’ютерної візуалізації та не пропонує інтегрованих рішень для аналітики вищого рівня чи процесів прийняття рішень. Наприклад, хоча він може виявити транспортний засіб, він не буде автоматично ідентифікувати тип або оцінювати рівень потенційної загрози. Це обмеження вимагає додаткових рівнів програмного забезпечення для ефективної інформації, що ускладнює архітектуру системи та потенційно сповільнює прийняття рішень у реальному часі. Розбираючи ці нюанси обмежень у можливостях виявлення об’єктів OpenCV, отримуємо більш чітке уявлення про труднощі, які чекають попереду. Незважаючи на численні переваги у швидкості та ефективності, OpenCV має кілька прогалин з точки зору надійності, гнучкості та складності.

Вирішення цих проблем має вирішальне значення, оскільки, необхідно створити програмне забезпечення для виявлення об’єктів, яке перевершує існуючі обмеження, надаючи більш надійні та універсальні рішення для різних умов реального світу. Проект прагне впроваджувати інновації в цьому просторі, використовуючи цей аналіз як базову структуру для розвитку більш оптимізованої, доступної та комплексної системи детекції об’єктів.

РОЗДІЛ 2 НЕЙРОМЕРЕЖА YOLOv5

2.1 Архітектура та порівняння YOLOv5

YOLOv5 (You Only Look Once, версія 5) сучасна нейронна мережа виявлення об’єктів, яка призначена для розпізнавання та класифікації об’єктів на зображеннях і відео в режимі реального часу. Принцип її роботи вже було описано в попередньому розділі роботи. В цьому розділі краще заглибитись в архітектуру цієї нейронної мережі щоб зрозуміти всі її елементи та яким чином їх можна використати для детекції об’єктів у зонах активних бойових дій.

Якщо розглядати цю модель без доповнень то її архітектура складається з трьох основних частин Backbone (магістраль), Neck (шия), та Head (голова) (рис. 2.1).

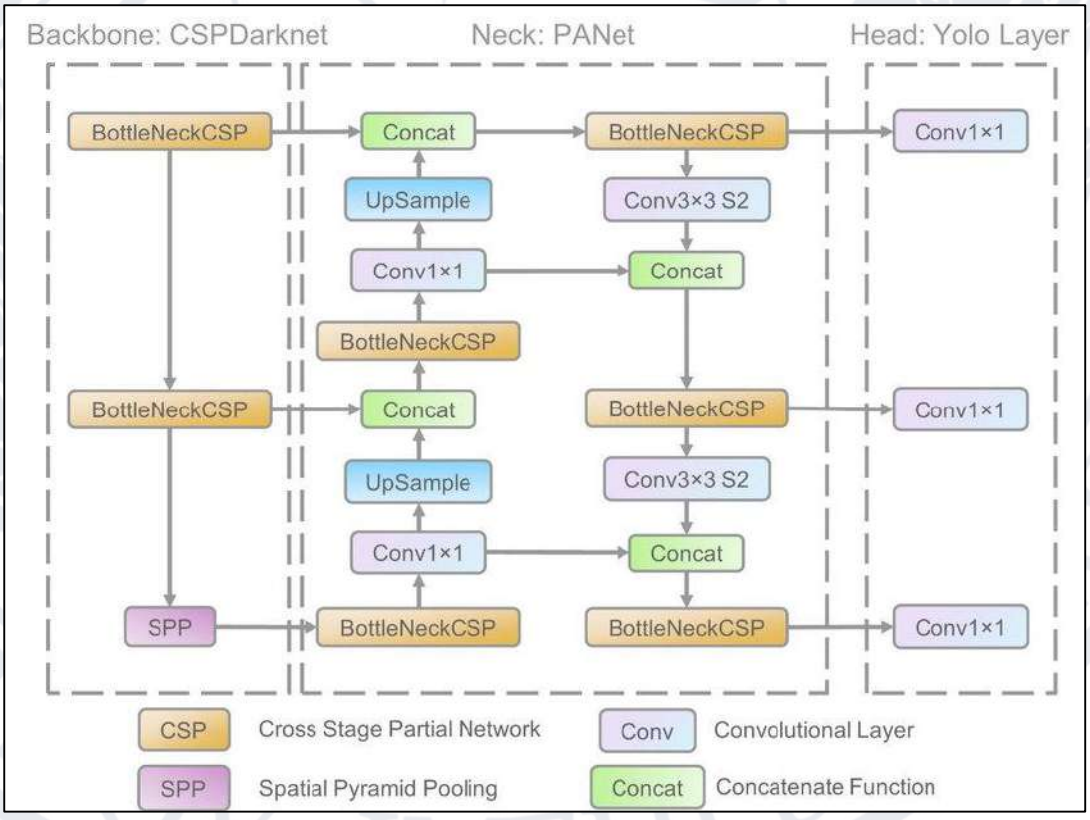


Рисунок 2.1 — Архітектура Yolov5 [42]

Conv1x1: Згортковий шар з розміром ядра 1x1. Цей шар використовується для зменшення кількості каналів на карті ознак.

Bottle NeckCSP: CSP-модуль з шаром вузького місця. Модуль CSP розділяє карту ознак на дві частини, одна з яких пропускається через шар вузьких місць, а інша через звичайний згортковий шар. Потім обидва результати об’єднуються разом. Шар вузьких місць зменшує кількість каналів на карті особливостей, що підвищує ефективність мережі.

Concat: Шар конкатенації. Цей шар об’єднує виходи двох попередніх шарів.

UpSample: Шар підвищеної вибірки. Цей шар збільшує просторову роздільну здатність карти об’єктів.

Conv3x3 S2: шар згортки з розміром ядра 3x3 і кроком 2. Цей шар витягує об’єкти з карти об’єктів підвищеної вибірки.

Блок BottleneckCSP використовується багато разів у магістральній мережі CSPDarknet53. Кількість каналів у карті особливостей збільшується в міру того, як мережа заглиблюється. Це дозволяє мережі виокремлювати більш складні ознаки з вхідного зображення.

За рахунок такої архітектури Yolov5 відрізняється доволі стійкою роботою на відміну від вже оглянутих конкурентів та інших версій Yolov5. Для демонстрації переваг саме цієї версії моделі порівняємо її з аналогами за наступними критеріями [43]:

- Магістраль: Основний компонент архітектури, що відповідає за вилучення ознак. Визначає здатність моделі виокремлювати значущі ознаки з вхідних зображень, що безпосередньо впливає на ефективність виявлення. Різні основи пропонують різні компроміси між точністю та обчислювальною ефективністю;
- Шия: Частина архітектури яка відповідає за злиття ознак і додаткову обробку після вилучення ознак. Хороша «шия» може значно покращити ефективність виявлення різномасштабних об’єктів, об’єднуючи ознаки з різних етапів архітектури;

- Шари виявлення: Кількість масштабів, на яких модель робить свої прогнози. Кілька рівнів розпізнавання дозволяють моделі ефективніше виявляти об’єкти різних розмірів;
- Якорі: Заздалегідь визначені межі, які модель використовує як орієнтир для прогнозування фактичних меж об’єктів. Вони мають вирішальне значення для визначення форми та розміру виявлених об’єктів;
- Вхідний розмір: Це стосується просторових розмірів вхідного зображення. Постійний розмір вхідного зображення може оптимізувати швидкість обробки, але гнучкість розміру вхідного зображення може бути корисною для різної роздільної здатності;
- Функція втрат: Функція втрат вимірює різницю між прогнозами моделі та фактичними значеннями (істиною). Вона керує оптимізацією нейронної мережі. Різні функції втрат або їхні комбінації можуть призвести до різної точності виявлення та збіжності моделей;
- Навчання з перенесенням: Цей критерій перевіряє, чи може модель використовувати попередні знання іншої навченої моделі для покращення процесу навчання. Навчання з перенесенням може значно прискорити навчання та покращити продуктивність, особливо коли навчальні дані обмежені;
- Точність: Це здатність моделі правильно виявляти і класифікувати об’єкти. Висока точність часто є першочерговою метою, але вона може досягатися за рахунок швидкості або обчислювальних ресурсів;
- Швидкість: Показує, наскільки швидко модель може обробити вхідне зображення і виявити об’єкти. Для додатків, що працюють в режимі реального часу, таких як спостереження або автономна робота, швидкість має першорядне значення;
- Продуктивність: Показує, наскільки ефективно модель може працювати з великими обсягами даних. Для додатків, які вимагають обробки великої кількості зображень, продуктивність є важливим фактором;

- Потреби в ресурсах: Показує, скільки обчислювальних ресурсів потрібно для навчання та використання моделі. Для додатків, які працюють на пристроях з обмеженими ресурсами, важливо вибрати модель з низькими потребами в ресурсах.

Такі критерії були обрані тому, що вони дають всебічний огляд дизайну, ефективності та результативності моделі.

Таблиця 2.1

Порівняння моделей

Модель/ критерії	**Yolov4**	**Yolov5**	**Yolov6**	**Yolov7**	**Yolov8**	**OpenCV**
Магістраль	Darknet-53	Darknet-53	CSPDarknet-53	CSPDarknet-53	CSPDarknet-53	MobileNetV3
Шия	Повна зв’язка	Повна зв’язка	Повна зв’язка	Повна зв’язка	Повна зв’язка	Повна зв’язка
Шари Виявлення (млн. параметрів)	3, 133	6, 272	6, 272	6, 272	6, 272	2, 22
Якорі (млн. параметрів)	9, 341	9, 341	9, 341	9, 341	9, 341	9, 341
Вхідний розмір	416x416	416x416	640x640	640x640	640x640	640x640
Функція втрат	MSE	MSE	MSE	MSE	MSE	L2
Навчання	Так	Так	Так	Так	Так	Так
Точність (COCO)	93,8%	95,8%	96,8%	97,2%	97,6%	85%
Швидкість (FPS)	30	40	25	20	15	10
Продуктивність	висока	висока	висока	висока	висока	низька
Потреби в ресурсах	високі	середні	високі	високі	високі	низькі

Аналізуючи порівняння моделей різних версій YOLO та OpenCV, YOLOv5 однозначно є оптимальним вибором для багатьох застосувань. Кожна версія YOLO від v4 до v8 послідовно спирається на магістральні архітектури, засновані на варіантах Darknet і CSPDarknet[44], і всі вони однаково використовують повне зв’язування шиї. Така узгодженість між версіями означає, що вдосконалення не пов’язані виключно зі змінами в архітектурі, а пояснюються методичною оптимізацією і доробкою.

Цікаво, що хоча YOLOv5 демонструє більшу кількість параметрів у критерії шарів виявлення ніж YOLOv4, вона є однаковою з наступними версіями аж до YOLOv8. Крім того, однорідність параметрів якоря в усіх цих моделях YOLO вказує на стандартизований механізм якоря, натякаючи на те, що підвищення продуктивності в нових ітераціях може не залежати від цього фактора. Переходячи до області вхідних розмірів і обчислень втрат, YOLOv5, як і його наступники, підтримує надійний вхідний розмір 640x640, на відміну від YOLOv4 416x416, пропонуючи більш тонку деталізацію розпізнавання об’єктів. Спільне використання функції втрат MSE, на відміну від L2 у OpenCV, ще більше уніфікує їхній методологічний підхід.

Однак, де YOLOv5 справді виділяється, так це в компромісі між точністю і швидкістю. З вражаючою точністю 95,8% на наборі даних COCO, він лише трохи відстає від новіших версій YOLO. Втім, приріст точності в наступних версіях є відносно незначним, що робить незначний недолік YOLOv5 майже несуттєвим. У порівнянні з 85% точністю OpenCV, перевага YOLOv5 не викликає сумнівів. Більше того, з похвальною швидкістю 40 FPS, YOLOv5 випереджає як свого попередника, так і всі пізніші ітерації YOLO. Таке поєднання швидкої обробки з високою точністю забезпечує YOLOv5 чудовий профіль продуктивності, спеціально розроблений для завдань у реальному часі.

YOLOv5, з огляду на свої помірні вимоги до ресурсів, стає особливо вигідним рішенням для різноманітних застосувань, де обмежені ресурси, такі як обчислювальна потужність і пам’ять, можуть бути фактором обмеження. Ця модель володіє балансом між високою швидкістю обробки відео та зображень, вражаючою точністю виявлення об’єктів і оптимізацією використання ресурсів. Завдяки цим особливостям, YOLOv5 може з успіхом застосовуватися в різних галузях, включаючи медицину, автономні транспортні засоби, безпеку та багато інших, роблячи його потужним і універсальним інструментом для завдань детекції.

2.2 Формування набору даних

Формування набору даних є фундаментальним кроком у конвеєрі глибокого навчання. Це особливо важливо в задачах виявлення об'єктів, подібних до тих, які вирішує YOLOv5. Добре сформований набір даних не лише забезпечує точність моделі, але й її стійкість у різних сценаріях. Продуктивність і точність YOLOv5 тісно пов'язані з калібром і різноманітністю навчальних даних. Зокрема, набір даних, який ретельно маркований, рівномірно збалансований за категоріями та охоплює широкий спектр об'єктів, має подвійну перевагу. По-перше, він надає моделі можливість ідентифікувати широкий спектр об'єктів за різних умов, підвищуючи її застосовність та універсальність. По-друге, такий набір даних, збагачений екземплярами об'єктів, знятих при різному освітленні, під різними кутами і в різних умовах, слугує захистом від надмірного налаштування, гарантуючи, що модель зберігає високий ступінь адаптивності і не зациклюється на певних підмножинах даних. Крім того, рівномірний розподіл даних між класами має першорядне значення. Такий баланс запобігає будь-якому притаманному моделі упередженню до певного класу, сприяючи більш цілісному та неупередженому виявленню об'єктів.

Ключові файли, необхідні для набору даних YOLOv5 [11]:

1. Файли зображень або відео: Це джерела візуальних даних, які містять об'єкти, що підлягають детекції. Для зображень найчастіше використовуються формати .jpg і .png. Однак YOLO також може обробляти інші формати зображень, такі як .bmp, .tiff і .webp. Що стосується відео, YOLO є універсальним і може обробляти такі формати, як .mp4, .avi, .mov і .mkv. Обробка відео передбачає аналіз кожного кадру як окремого зображення, що дозволяє виявляти об'єкти в реальному часі протягом усього відео. Перевага роботи з відео полягає в тому, що вона дозволяє безперервно відстежувати і виявляти об'єкти в часі, надаючи більш динамічну перспективу порівняно зі статичними зображеннями;

2. Файли міток: Важливі для процесу навчання, файли міток надають анотації, які детально описують, де розташовані об’єкти на відповідних зображеннях, і до яких категорій ці об’єкти належать. Кожен файл мітки відповідає одному зображенню і повинен бути збережений з тим самим ім’ям, але зазвичай з розширенням .txt. Ці файли зберігаються у спеціальному каталозі міток, окремому від зображень, але з тією самою організаційною структурою. Формат вмісту в файлі міток:

- Кожен об’єкт на зображенні представлений окремим рядком;
- Кожен рядок дотримується формату: class x_center y_center width height, де кожне значення відокремлюється пробілом;
- Значення x_center, y_center, width та height нормалізовані, тобто масштабуються між 0 та 1. Ця нормалізація відбувається відносно ширини та висоти зображення, що забезпечує послідовне масштабування незалежно від фактичного розміру зображення.

Клас позначається цілим числом. Це число є індексом, що посилається на список класів, визначених для набору даних. Наприклад, у наборі даних з класами автомобіль, пішохід і велосипедист ціле число 0 може відповідати автомобілю, 1 - пішоходу і 2 - велосипедисту. Ці анотації відіграють важливу роль на етапі навчання. Шляхом ітерацій модель намагається мінімізувати різницю між своїми прогнозами та цими базовими даними, покращуючи свою точність та здатність виявлення об’єктів з часом. Вручну анотувати великий набір даних може бути важко. Проте, існують такі інструменти, як Make Sense [46] та Label Studio [47], які полегшують анотування у форматі YOLO, спрощуючи процес, особливо для великих наборів даних.

3. Файл конфігурації даних (data.yaml): Файл data.yaml слугує компасом, який спрямовує модель YOLOv5 під час процесів навчання та валідації. Це не просто каталог це план, який визначає, як модель повинна отримувати доступ до даних і розуміти їх. Структура data.yaml файлу наступна:

- names: Це список, який містить усі назви класів у наборі даних. Наприклад, у наборі даних, що фокусується на об’єктах дорожнього руху, це

можуть бути такі мітки, як автомобіль, автобус, пішохід тощо. Цей список визначає текстове представлення класів і індексується відповідно до цілих чисел, що використовуються у файлах міток;

- nc: Скорочення від number of classes (кількість класів), це поле вказує, скільки унікальних класів модель має очікувати у наборі даних. Це важливо для ініціалізації кінцевого вихідного шару моделі, щоб врахувати всі передбачення класів. Наприклад, якщо є три класи - автомобіль, автобус і пішохід, nc буде встановлено на 3;
- train: Це критичне поле, яке спрямовує модель до місця розташування навчального набору даних. Зазвичай воно вказує на текстовий файл, який часто називають train.txt, що містить шлях до всіх зображень, використаних для навчання. Кожен рядок у цьому файлі відповідає шляху до одного зображення;
- val: Аналогічно до поля train, вказує моделі на валідаційний набір даних. Він вказує на інший текстовий файл, який часто називають val.txt, що перераховує шляхи до зображень, відкладених для перевірки. Перевірка на окремому наборі даних гарантує, що показники ефективності моделі, такі як точність і втрати, оцінюються на невидимих даних, пропонуючи уявлення про узагальнюючі можливості моделі.

Додаткові поля (необов'язкові): Залежно від специфіки проекту або набору даних, файл data.yaml може містити й інші поля. Наприклад:

- test: Шлях до тестового набору даних, подібний за функціями до train і val;
- augmentations: Опис будь-яких доповнень до даних, які будуть застосовані під час навчання для підвищення надійності моделі;
- significance: Файл data.yaml є важливим, оскільки він централізує метадані процесу навчання. Налаштувавши цей файл, можна легко перемикати набори даних, визначити класи або змінювати розподіл між навчанням і перевіркою, роблячи робочий процес навчання модульним і адаптивним.

4. Текстові файли навчання та перевірки: Ці файли містять шляхи до відповідного набору зображень для навчання та перевірки. Вони допомагають розділити дані, гарантуючи, що модель буде перевірена на невидимих даних.

2.3 Навчання нейронної мережі

Навчання нейронної мережі, особливо такої складної, як YOLOv5, схоже на передачу навичок. Процес навчання полягає в тому, щоб зробити модель вправною у виконанні свого завдання, в даному випадку, виявлення об’єктів. Алгоритм навчання YOLOv5 [48-51]:

1. Підготовка:

- Набір даних: Як ми вже обговорювали раніше, добре підготовлений набір даних є основоположним. Він охоплює анотовані зображення з відповідними файлами міток, організованими у структурований спосіб;
- Конфігураційні файли: Файл data.yaml має вирішальне значення, він деталізує назви класів, кількість класів і шляхи до навчальних та перевірочних наборів даних;
- Середовище: Відповідне середовище з необхідними бібліотеками є дуже важливим. Навчання глибоких нейронних мереж, таких як YOLOv5, вимагає потужних обчислювальних ресурсів, як правило, графічних процесорів, для ефективної обробки великих обсягів даних;

2. Ініціалізація

Weights (ваги): В основі будь-якої нейронної мережі, включаючи YOLOv5, лежать ваги. Це регульовані параметри всередині моделі, які визначають силу і напрямок впливу вхідних даних на вихідні. По суті, вони є серцем і душею здатності моделі до навчання. Коли вхідні дані проходять через мережу, вони множаться на ці ваги, об, а потім трансформуються за допомогою функції активації для отримання вихідних даних. Мета навчання нейронної мережі полягає в тому, щоб налаштувати ці ваги таким чином, щоб

мінімізувати похибку між прогнозованим виходом і фактичними мітками. Кожна ітерація під час навчання (прохід вперед і назад) уточнює ці ваги за допомогою алгоритмів оптимізації, таких як градієнтний спуск [52].

3. Пряме розповсюдження (перехід):

- Згорткові шари: Ці шари сканують вхідне зображення за допомогою фільтрів, щоб виділити особливості на низькому рівні, такі як краї, текстури та кольори. Коли тренування просувається вглиб мережі, ці особливості стають більш абстрактними, представляючи візерунки, частини об’єктів і, врешті-решт, цілі об’єкти;
- Функції активації: Після кожного шару згортки застосовується функція активації типу ReLU (Rectified Linear Unit випрямлена лінійна одиниця). Це вносить нелінійність у модель, дозволяючи їй вивчати більш складні взаємозв’язки в даних;
- Шари об’єднання: Ці шари зменшують просторові розміри вхідних даних, зберігаючи найбільш важливу інформацію. Це допомагає зменшити обчислювальне навантаження і зробити модель більш інваріантною до незначних змін або спотворень зображення;
- Рамки присвоєння: YOLOv5, як і інші версії YOLO, використовує рамки заздалегідь визначені форми обмежувальних рамок. Модель прогнозує коригування, які потрібно зробити в цих прив’язках, щоб вони відповідали реальним формам об’єктів на зображенні;
- Прогнозування класів: Разом з коригуванням обмежувальних рамок модель також прогнозує розподіл ймовірностей по всіх класах для кожної рамки. Це дає уявлення про те, до якого класу належить об’єкт, що перебуває в обмежувальній рамці;
- Останні шари об’єднують всю цю інформацію, створюючи прогнози моделі. Для YOLOv5 первинний результат складається з координат обмежувальної рамки (x, y, ширина, висота), оцінок об’єктності (ймовірність того, що рамка містить об’єкт) та ймовірностей класів (ймовірність того, що об’єкт у рамці належить до кожного класу).

Ця інформація потім обробляється для створення чітких обмежувальних рамок з мітками класів. Немаксимальне придушення (NMS) застосовується для видалення надлишкових областей, залишаючи лише найбільш достовірний прогноз для кожного об’єкта. Після отримання прогнозів обчислюються втрати шляхом порівняння цих прогнозів з істинними мітками. Ця втрата показує, наскільки прогнози моделі далекі від істини. В YOLO ця втрата є комбінацією локалізації (наскільки точними є обмежувальні рамки) і класифікації (наскільки точними є прогнози класів). Під час прямого проходу модель отримує необроблені дані зображення, обробляє їх за допомогою численних перетворень і виводить структуровані прогнози, які дають чітке уявлення про те, де знаходяться об’єкти і до яких класів вони належать. Цей результат слугує основою для зворотного розповсюдження, де модель вчиться уточнювати свої ваги та покращувати свої прогнози.

4. Обчислення втрат:

Передбачення моделі порівнюється з фактичними мітками (істиною в останній інстанції) за допомогою функції втрат. Для YOLOv5 це часто середньоквадратична похибка (MSE) або її різновид. Ця втрата кількісно показує, наскільки далеко передбачення моделі від фактичних міток. Під час навчання алгоритм оптимізації постійно намагається мінімізувати ці втрати. Зменшення втрат вказує на те, що прогнози моделі стають більш наближеними до істини, що означає покращення продуктивності. Однак дуже важливо стежити за тим, щоб не допустити надмірного пристосування, коли модель може працювати дуже добре на навчальних даних, але погано на нових, ще не бачених даних. Процес розрахунку втрат діє як механізм зворотного зв’язку під час навчання, вказуючи на сильні сторони моделі та висвітлюючи її слабкі місця. Саме цей зворотний зв’язок є рушієм ітеративного вдосконалення моделі, що приводить її до кращої продуктивності.

5. Розповсюдження та оптимізація:

На цьому етапі відбувається вдосконалення моделі. Використовуючи втрати як зворотний зв’язок, модель коригує свої ваги, щоб зменшити ці

втрати на наступних ітераціях. Щоб запобігти надмірному пристосуванню і забезпечити узагальнену модель, під час оптимізації можуть застосовуватися різні методи регулювання. Приклади включають L1 та L2, відсіювання та ранню зупинку. Ці методи накладають обмеження або штрафи на модель, не дозволяючи їй занадто точно відповідати навчальним даним. Замість того, щоб оновлювати ваги після кожної окремої точки даних (онлайн-навчання) або після всього набору даних (пакетне навчання), зазвичай використовують міні-пакетне навчання. У цьому підході ваги моделі оновлюються після обробки невеликої кількості точок даних. Це дозволяє досягти балансу між обчислювальною ефективністю та стабільністю оновлення ваг.

6. Ітеративне навчання:

Кроки з 3 по 5 повторюються для декількох ітерацій (епох). З кожною ітерацією модель покращується, зменшуючи втрати та покращуючи точність прогнозування. Терміни ітерації та епохи часто використовують як взаємозамінні, але вони мають різні значення. Епоха відноситься до одного циклу через весь навчальний набір даних, тоді як ітерація відноситься до одного оновлення ваг моделі, зазвичай після обробки пакету даних. У контексті глибокого навчання можливо мати кілька ітерацій протягом однієї епохи, особливо якщо ви використовуєте міні-пакетне навчання.

Нейронні мережі, за своєю природою, не є ідеальними після одного проходу через дані. Вони потребують повторного впливу на навчальний набір даних для коригування та уточнення вагових коефіцієнтів. Такий ітеративний підхід дозволяє моделі поступово мінімізувати помилки, відточувати точність і забезпечувати надійність прогнозів. Під час цих ітерацій дуже важливо періодично оцінювати роботу моделі на окремому валідаційному наборі даних. Це допомагає переконатися, що модель не просто запам’ятовує навчальні дані (перенавчання), а дійсно вивчає закономірності, які можуть узагальнюватися на нових, небачених даних. За допомогою повторюваних циклів прямих і зворотних проходів модель відточує свої ваги та упередження, щоб досягти балансу між пристосуванням до навчальних даних і хорошим

узагальненням нових даних. Саме це ітеративне вдосконалення перетворює наївну, визначену мережу на потужний інструмент детекції.

7. Кінцевий результат навчання:

Навчені ваги є кульмінацією ітеративного навчання, представляючи собою дистильовані знання, які модель отримала з набору даних. Вони є ключовими для формування висновків у реальних сценаріях і можуть бути використані в навчанні з перенесенням для прискорення процесу навчання на суміжних завданнях. Точність вимірює точність позитивних прогнозів, в той час як відкликання оцінює, скільки фактичних позитивних прогнозів модель фіксує. Показник F1 - це баланс між точністю та запам’ятовуванням. Карта в YOLO показує середню точність для різних класів об’єктів і рівнів пригадування, слугуючи цілісним показником ефективності виявлення. Візуальні інструменти незамінні для розшифровки процесу навчання моделі. Наприклад, графіки втрат дають уявлення про збіжність і можуть натякати на потенційну надмірну адаптацію. Візуалізація також може допомогти зрозуміти, в яких сферах модель досягає успіху, а в яких потребує подальшого доопрацювання.

По суті, кінець навчання це не просто припинення ітеративного навчання, а народження готової до розгортання моделі, підтвердженої метриками і висновками візуальної аналітики. Метрики ефективності: Такі метрики, як точність, тестування та оцінка F1, дають кількісну оцінку досконалості моделі. У контексті YOLO такі показники, як середня точність (mAP), є особливо важливими.

Навчання такої моделі, як YOLOv5, є важким процесом, що вимагає поєднання добре підготовлених наборів даних, обчислювальної майстерності та ітеративних стратегій навчання. Від ініціалізації з відповідними вагами до проходження циклів прогнозів і уточнень, модель розвивається, покращуючи свою продуктивність з кожною епохою. Результатом цього шляху є надійна нейронна мережа.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ НЕЙРОМЕРЕЖІ

3.1 Створення дата сету

У зонах активних бойових дій швидке і точне виявлення потенційних загроз, об’єктів цивільного та військового призначення є важливим як для стратегічного планування, так і для захисту життя людей.

Кожен обраний клас має особливе значення:

- Танк: Танки є основними наземними штурмовими машинами, здатними завдати значної шкоди. Раннє виявлення може мати вирішальне значення для оборони та контр-операцій;
- Бронетранспортер (БТР): БТР транспортують війська в захищеному середовищі, що робить їх важливими засобами для виявлення як для тактичних маневрів, так і для оцінки загроз;
- Бронеавтомобіль: Подібно до БТР, інші броньовані машини можуть свідчити про переміщення військ або наміри ескалації бойових дій;
- Гелікоптер: Часто використовується для швидкого розгортання військ, доставки вантажів або навіть для проведення атак класу повітря-земля. Їх раннє виявлення може забезпечити вирішальний час для реагування;
- Літак: Військові літаки можуть виконувати різні функції від розвідки до безпосереднього ведення бойових дій. Їх виявлення може сигналізувати про майбутні повітряні нальоти або спостереження;
- Дрони та безпілотні літальні апарати (безпілотники) : Дрони і безпілотники стають все більш поширеними в сучасній війні для спостереження, наведення на ціль і навіть доставки невеликих корисних вантажів. Їх виявлення є ключовим для контррозвідки та оборони;
- Ракети: Можливо, одна з найбільш нагальних загроз, раннє виявлення запуску ракети може означати різницю між перехопленням і ударом;

- Протиповітряна оборона (ППО): виявлення систем протиповітряної оборони може вказувати на можливості протиповітряної оборони району, що є важливою інформацією для планування повітряних операцій;
- Військовий об’єкт: Розпізнавання військовослужбовців може дати уявлення про концентрацію військ, їх переміщення та потенційні зони бойових дій;
- Цивільний об’єкт : не менш важливо виявляти цивільних, щоб мінімізувати супутні збитки і захистити життя невинних людей;
- Гранати, зброя та міномети: Виявлення цих предметів може надати пряму інформацію про безпосередні загрози на місцевості, вказуючи на зони високої інтенсивності бойових дій.

Після формування імен(класів) для детекції, потрібно зібрати дата сет, в сценарії роботи це будуть зображення попередньо описаних класів (рис. 3.1).

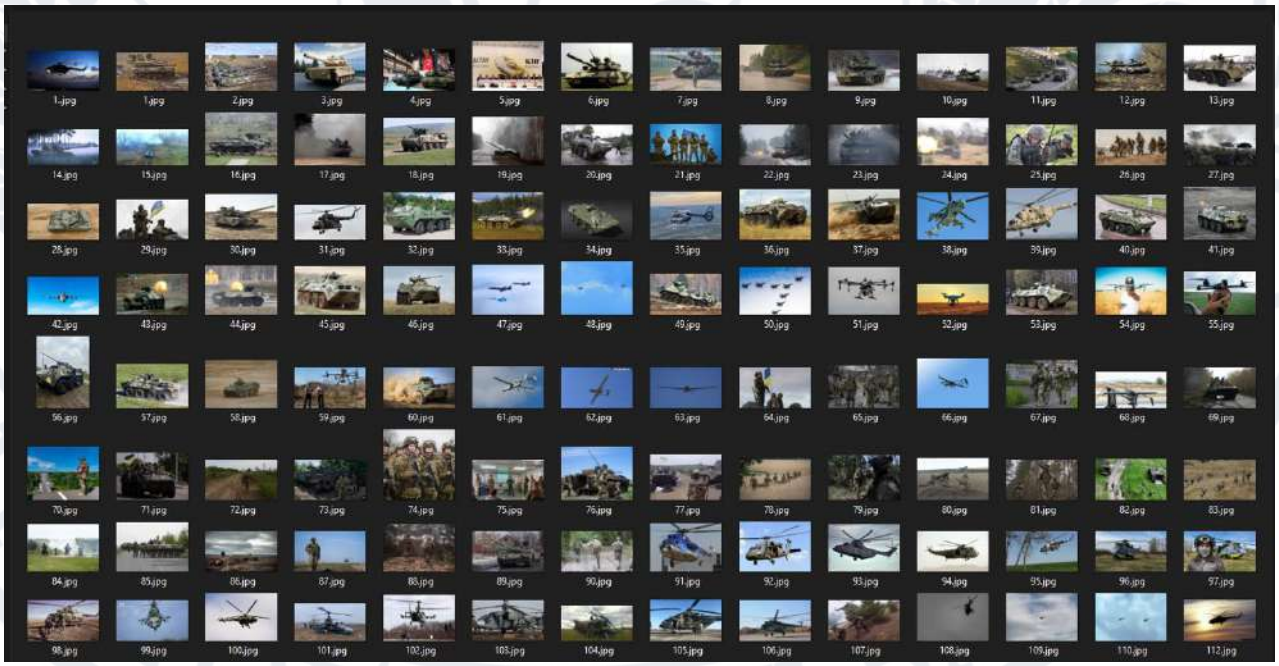


Рисунок 3.1 — Сформований дата сет

Для зручності всі зображення в наборі даних послідовно пронумеровані від 1 до 529, що відповідає загальній кількості навчальних зображень,

призначених для процесу навчання нейронної мережі. На додаток до них є 112 перевірочних зображень, що робить загальний розмір набору даних 641 зображення. Хоча це може здатися скромним розміром набору даних, важливо розуміти, що якість і різноманітність анотацій часто може переважати над їхнім обсягом. Завдяки ретельному маркуванню та репрезентативним вибіркам, що охоплюють широкий спектр обраних класів, навіть набір даних такого розміру може бути досить потужним. Таким чином, незважаючи на свій розмір, він є достатньо надійним для навчання нейронної мережі, і можна очікувати, що він дасть похвальні результати.

Наступним потрібно зробити маркери для зображень в .txt форматі, для цього використовується Make Sense онлайн-інструмент, розроблений для того, щоб зробити процес маркування зображень для розпізнавання об’єктів більш спрощеним та інтуїтивно зрозумілим. Ось загальний огляд того, як відбувається цей процес:

1. Першим кроком, як правило, є завантаження набору даних, який містить зображення, які ви маєте намір анотувати. Цей інструмент підтримує масові завантаження, що дозволяє завантажувати великі набори даних (рис.3.2).

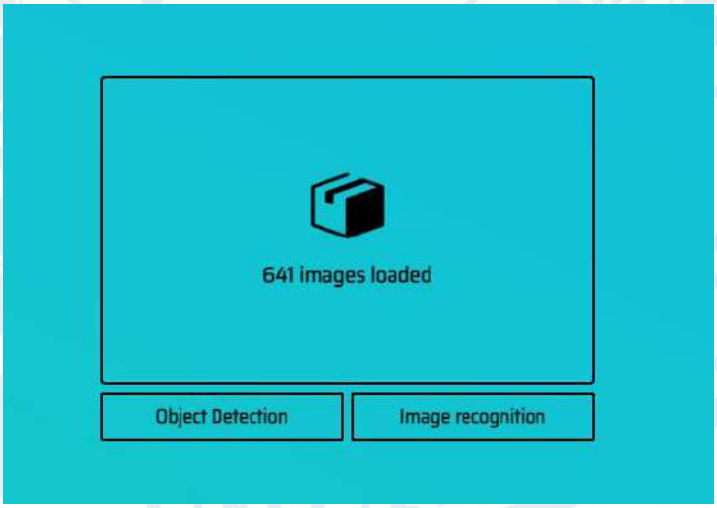


Рисунок 3.2 — Завантаження дата сету до Make Sense

2. Далі необхідно визначити класи об’єктів детекції. Для нашого набору даних, присвяченого військовій тематиці, це стосується таких категорій, як танк, ракета, безпілотник та інших. Кожен клас можна позначити кольором для легкої ідентифікації під час процесу маркування (рис. 3.3).

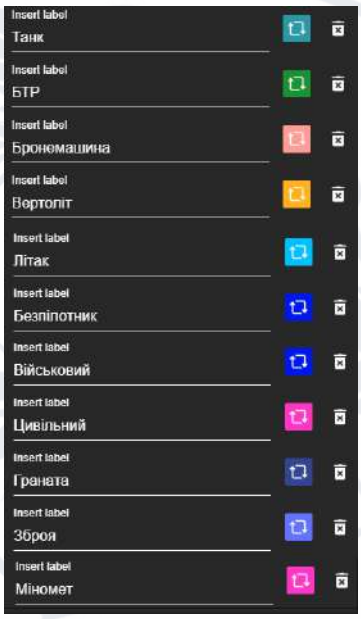


Рисунок 3.3 — Створення класів в Make Sense

3. Після визначення класів починається власне процес маркування. Для кожного зображення малюють обмежувальні рамки навколо об’єктів, що їх цікавлять (рис. 3.4.).

Рисунок 3.4 — Маркування класів в Make Sense

Коли ці рамки намальовані, платформа пропонує призначити заздалегідь визначений клас для кожної рамки, фактично маркуючи об’єкт(рис.3.5).

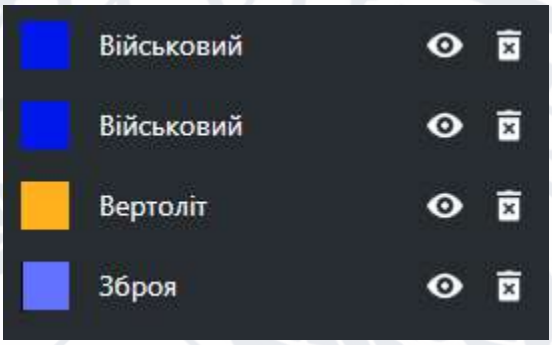


Рисунок 3.5 — Маркування класів в Make Sense

4. Після того, як всі зображення позначені, експортуються файли маркування. Для нейронної мережі YOLO ці анотації будуть у певному форматі, що містить координати обмежувальних рамок і пов’язані з ними мітки класів (рис. 3.6).

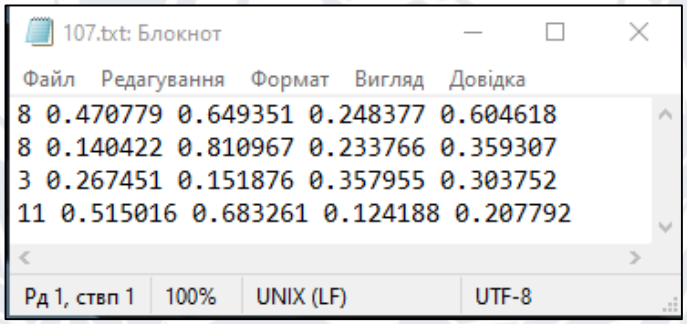


Рисунок 3.6 — Маркування класів в текстовому форматі

Як можна побачити на рисунку всі 4 класи (об’єкти) які були марковані в Make Sense перенеслись в текстовий файл з іменем зображення яке маркувалось. Як вже раніше описувалось перша цифра в рядку це порядковий номер класу, а далі координати рамки.

5. Після завантаження дуже важливо організувати ці файли міток таким чином, щоб вони відповідали оригінальному набору даних зображень. Належна організація гарантує, що на етапі навчання нейронна мережа

правильно асоціює кожне зображення з відповідними анотаціями. Структура збережених файлів зображена на рисунку 3.7.

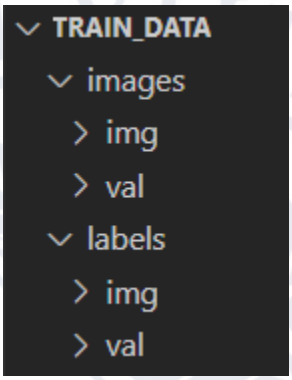


Рисунок 3.7 — Структура збережених файлів

Після цих кроків, можна переходити до навчання Yolov5.

3.2 Навчання через Google Colab

Google Colab, скорочено від Colaboratory це безкоштовна хмарна платформа від Google, яка дозволяє користувачам писати і виконувати код на Python в інтерактивному веб-середовищі. Однією з її особливостей є надання безкоштовного доступу до графічних процесорів (GPU) та тензорних процесорів (TPU), які допомагають ефективно навчати нейронної мережі.

Алгоритм навчання Yolov5 через Colaboratory наступний:

1. Ініціалізація:

Після переходу на сторінку Colaboratory з Yolov5 [53], перше що потрібно зробити це встановити всі необхідні залежності та клонувати репозиторій нейронної мережі, запустивши цей код:

```
!git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
%pip install -qr requirements.txt comet_ml # install
import torch
import utils
display = utils.notebook_init()
```

Тут використовується команда git clone для клонування репозиторію YOLOv5 зі сторінки GitHub. Це означає, що весь вихідний код, конфігурації та інші файли, пов'язані з YOLOv5, витягуються та зберігаються локально в середовищі Colaboratory. Команда %cd змінює поточний каталог блокнота на папку yolov5, яку щойно клонували. Це схоже на використання команди cd у терміналі для переходу до каталогу. Наступна команда встановлює пакети Python, перелічені у файлі requirements.txt зі сховища YOLOv5. Цей файл містить усі залежності, необхідні YOLOv5 для правильної роботи. Використовуються такі прапори:

-q: Тихий режим, який означає, що показуються лише важливі результати, що робить процес встановлення менш докладним;

-r: цей прапорець повідомляє pip встановити пакунки з наданого файлу вимог.

На додаток до у requirements.txt також встановлено пакет comet_ml. Comet ML платформа для відстеження, порівняння, пояснення та оптимізації експериментів і моделей. Далі імпортується бібліотека PyTorch, яка є основною структурою глибокого навчання, яку використовує YOLOv5. Таким чином, цей код готує середовище Colaboratory для запуску та експериментів із YOLOv5 шляхом клонування репозиторію, переходу до його каталогу, встановлення необхідних пакетів та ініціалізації деяких конфігурацій.

1.1 Після ініціалізації потрібно завантажити до Colaboratory наступні файли:

- Теку train_data: результат формування та маркування дата сету, його в форматі zip (архів) потрібно завантажити до платформи в довільне місце, після чого витягнути з нього файли, використовуючи наступний код:

```
!unzip train_data.zip
```

- Файл .yaml: файл необхідний нейронної мережі щоб зрозуміти яким чином працювати. Створений cutom_data.yaml (рис. 3.8) потрібно вставити у теку за шляхом: yolov5/data де зберігаються інші yaml файли.

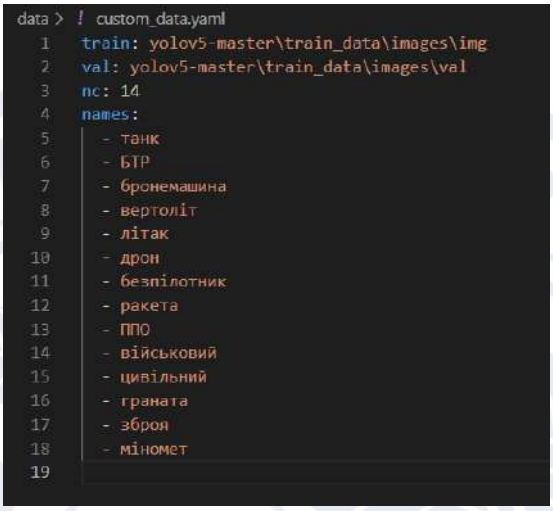


Рисунок 3.8 — Структура файлу cutom_data.yaml

В цій структурі:

- names: Список назв класів;
- nc: Загальна кількість класів;
- train: Шлях до навчального набору даних у теці train_data;
- val: Шлях до перевірочного набору даних.

2. Початок процесу навчання:

За замовчуванням команда для запуску процесу навчання на вибірковому наборі даних, виглядає так:

```
!python train.py --img 640 --batch 16 --epochs 3 --data coco128.yaml --weights yolov5s.pt –cache
```

Де:

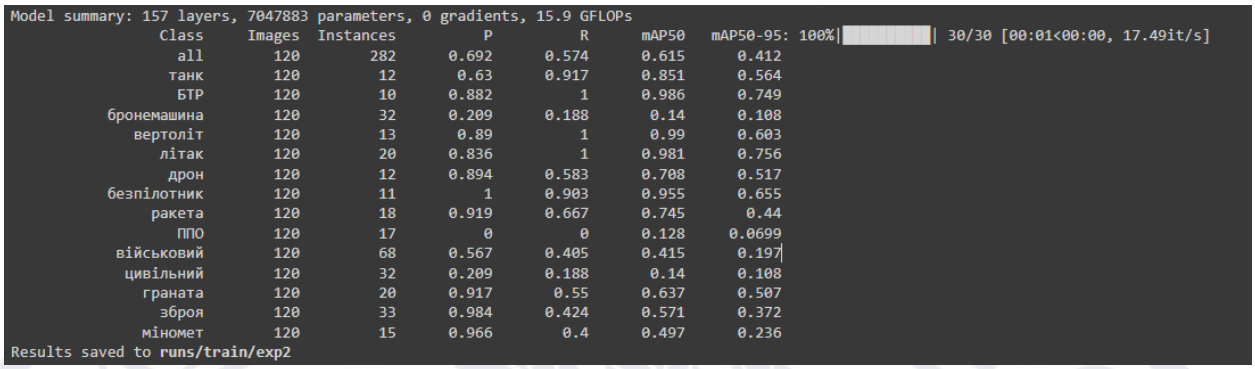
- !python train.py: це виконує сценарій train.py за допомогою Python. Скрипт відповідає за навчання моделі YOLOv5;
- --img 640: встановлює розмір зображень на 640x640 пікселів. YOLOv5 змінює розмір вхідних зображень до квадратної форми, і цей аргумент визначає розміри цього квадрата;

- --batch 16: розмір пакету встановлено на 16. У глибокому навчанні замість обробки всього набору даних за один раз ми розбиваємо його на менші фрагменти або пакети. Цей аргумент вказує, що 16 зображень обробляються та оновлюються під час кожної ітерації;
- --epochs 3: встановлює кількість циклів або епох навчання до 3. Одна епоха означає один повний прохід вперед і назад усіх прикладів навчання. Модель буде навчено протягом 3 повних циклів набору даних;
- --data coco128.yaml: це визначає файл конфігурації, який містить деталі про набір даних, як-от шлях до наборів даних для навчання та перевірки, кількість класів, імена класів тощо. У цьому випадку він вказує на набір даних за замовчуванням під назвою coco128 (менша версія набору даних COCO);
- --weights yolov5s.pt: це надає початкові ваги для моделі. Аргумент yolov5s.pt означає, що він починається з ваг з маленької моделі YOLOv5. Можливо використовувати це для перенесення навчання, коли модель може використовувати знання з попередньо навченої моделі для прискорення процесу навчання;
- --cache: кешує зображення для швидшого подальшого завантаження. Зображення набору даних зберігається в оперативній пам’яті, що забезпечує швидший доступ і покращує час навчання.

Необхідно змінити цю команду, щоб вона вказувала на власний набір даних і конфігурацію:

```
!python train.py --img 640 --batch 16 --epochs 80 --data custom_data.yaml --weights yolov5s.pt –cache
```

Після чого запустити процес навчання нейронної мережі. Враховуючи швидкість Colaboratory, відносно малий дата сет та кількість епох навчання буде тривати 40 хвилин до 120хвилин. Після пройденого часу навчання з’явиться результат навчання (рис. 3. 9).



```
Model summary: 157 layers, 7047883 parameters, 0 gradients, 15.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|████████| 30/30 [00:01<00:00, 17.49it/s]
                   all        120        282      0.692      0.574      0.615      0.412
                  танк        120         12       0.63      0.917      0.851      0.564
                   БТР        120         10      0.882          1      0.986      0.749
           бронемашина        120         32      0.209      0.188       0.14      0.108
              вертоліт        120         13       0.89          1       0.99      0.603
                 літак        120         20      0.836          1      0.981      0.756
                  дрон        120         12      0.894      0.583      0.708      0.517
           безпілотник        120         11          1      0.903      0.955      0.655
                ракета        120         18      0.919      0.667      0.745       0.44
                   ППО        120         17          0          0      0.128     0.0699
             військовий        120         68      0.567      0.405      0.415      0.197
              цивільний        120         32      0.209      0.188       0.14      0.108
                граната        120         20      0.917       0.55      0.637      0.507
                  зброя        120         33      0.984      0.424      0.571      0.372
               міномет         120         15      0.966        0.4      0.497      0.236
Results saved to runs/train/exp2
```

Рисунок 3.9 — Результат тренування нейронної мережі

Розберемо результат детальніше:

MAPS8-95: 100%| — Це означає, що модель досягла mAP 100% на навчальному наборі даних. mAP для класів означає середню точність та відкликання для конкретного класу об’єктів. Це показник того, наскільки добре модель виявляє та локалізує об’єкти цього класу.

mAP для класів:

- Танк — 0.851;
- БТР — 0.986;
- Бронемашина — 0.814;
- Гелікоптер — 0.899;
- Літак — 0.981;
- Дрон — 0.8708;
- Безпілотник — 0.955;
- Ракета — 0.745;
- ППО — 0.667;
- Військовий — 0.415;
- Цивільний — 0.14;
- Граната — 0.55;
- Зброя — 0.571;
- Міномет — 0.497.

Model summary: Цей розділ містить інформацію про архітектуру моделі, включаючи кількість шарів, параметри та кількість операцій з плаваючою комою за секунду (GFLOPS). Images кількість зображень у навчальному наборі даних, що містять екземпляри цього класу. Instances загальна кількість екземплярів цього класу в навчальному наборі даних. P точність, тобто частка правильних прогнозованих обмежувальних рамок. R Відтворення, тобто частка виявлених істинних граничних областей. MAPS8 середня точність для даного класу, яка є середньозваженим значенням точності та відгуку.

Модель досягла загального значення mAP 8,655, що означає, що вона змогла правильно виявити і локалізувати 86,55% об'єктів у навчальному наборі даних. Це хороший результат, особливо враховуючи невеликий розмір набору даних.

3.3 Використання моделі та аналіз результатів

Після тренування нейронної мережі найкраща її варіація буде збережена по шляху yolov5\runs\train, в нашому випадку найкраща версія моделі це файл best.pt. Файл являє собою збережений стан моделі нейронної мережі YOLOv5, яка досягла найкращої продуктивності на перевірочному наборі даних в процесі навчання.

Для використання моделі використовується наступна код:

```
!python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images
```

Ця команда є викликом скрипту detect.py з репозиторію YOLOv5 для виявлення об'єктів на заданих зображеннях.

Розглянемо кожен компонент:

- !python detect.py: Ініціює виконання скрипта detect.py мовою Python. Символ ! на початку це спосіб запускати команди командного інтерпретатора безпосередньо з блокнотів Jupyter або Google Colab;

- --weights yolov5s.pt: Вказує шлях до файлу ваг моделі, який буде використано для визначення. У цьому випадку використовується файл yolov5s.pt, який зазвичай позначає ваги для малої версії моделі YOLOv5. Це можуть бути попередньо навчені ваги або будь-які інші збережені ваги моделі;
- --img 640: Визначає розмір, до якого мають бути змінені вхідні зображення перед завантаженням їх у модель. Тут зображення буде змінено до розміру 640x640 пікселів. Зміна розміру необхідна для забезпечення узгодженості і часто для досягнення кращої якості розпізнавання;
- --conf 0.25: Встановлює поріг достовірності для прогнозів. Буде збережено лише виявлення з достовірністю, вищою за цей поріг (тобто 0,25 або 25% у цьому випадку). Це допомагає відфільтрувати слабші, потенційно невірні детекції.
- Джерело даних/зображень: Це шлях до джерела вхідних даних. Це може бути каталог із зображеннями, окремий файл зображення, відеофайл, джерело веб-камери або навіть URL-адреса для потокового передавання. У цьому прикладі вказано шлях до каталогу з назвою data/images.

Для перевірки працездатності моделі створимо теку, в якій будуть теки всіх класів а в них по декілька зображень для перевірки роботи мережі (рис. 3.10).

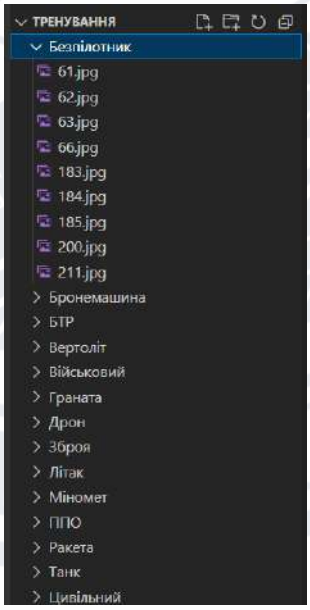


Рисунок 3.10 — Створений набір даних для тестування моделі

Після завантаження архіву для тренування та всіх необхідних даних, виконаємо наступний код:

```
!python detect.py --weights runs/train/best.pt --img 640 --conf 0.25 --source /content/yolov5/Тренування/БТР
```

Він запустить нашу найкращу версію моделі та проаналізує зображення в теці БТР. Після виконання отримаємо результат зображений на рисунку 3.11.

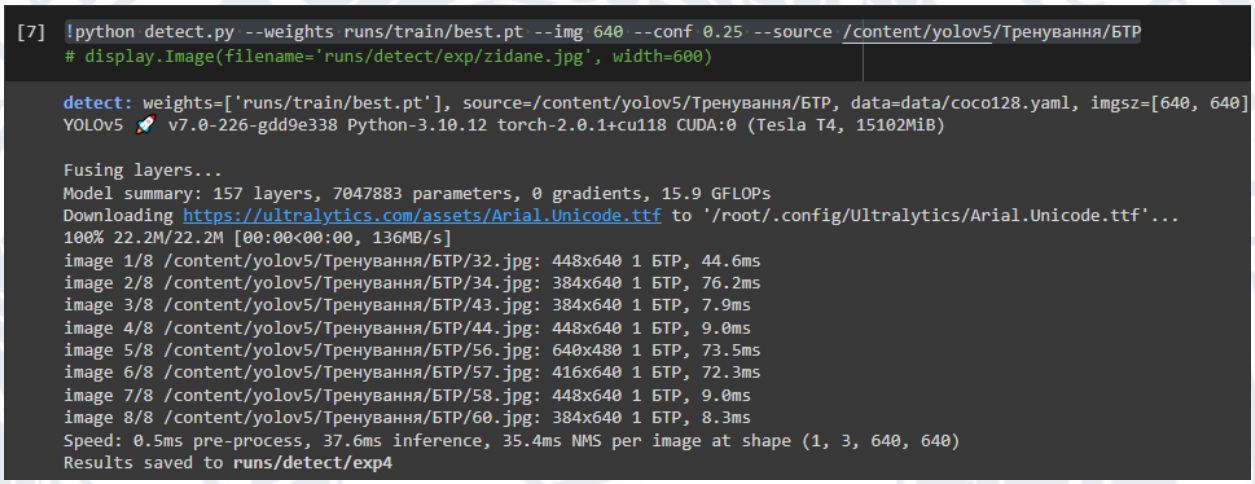


Рисунок 3.11 — Результат аналізу теки з зображеннями БТР

Де:

1. Аргументи командного рядка:

- weights=[‘runs/train/best.pt’]: Тут вказується шлях до тренованих ваг, у цьому випадку використовується файл best.pt, який зазвичай є вагами моделі, що досягли найменших втрат валідації під час навчання;
- source=/content/yolov5/Тренування/БТР: Шлях до вхідних даних, на яких потрібно зробити виявлення;
- data=data/custom_data.yaml: YAML-файл, що містить специфічну для набору даних інформацію;
- imgsz=[640, 640]: Розмір вхідного зображення для моделі. Це квадратне зображення розміром 640x640 пікселів;
- conf_thres=0.25: Довірчий поріг. Виявлення з достовірністю нижче цього значення будуть відкинуті;

- iou_thres=0.45: Поріг перетину з об’єднанням. Використовується для не-максимального придушення (NMS), щоб усунути перекриття обмежувальних рамок;
- project=runs/detect: Каталог, до якого буде збережено результати детектування.

2. Ініціалізація:

- Надає інформацію про версію YOLO, версію Python, версію PyTorch та пристрій CUDA, що використовується (у цьому випадку графічний процесор Tesla T4).

3. Злиття моделей та підбиття підсумків:

- Крок Fusing layers часто поєднує шари згортки та пакетної нормалізації для покращення швидкості розпізнавання;
- Короткий опис моделі містить інформацію про кількість шарів, параметри та обчислювальні витрати (GFLOP) моделі.

4. Завантаження шрифту:

- YOLOv5 часто завантажує певний шрифт (у цьому випадку Arial.Unicode.ttf) для позначення виявлених об’єктів на вихідних зображеннях.

5. Етап виявлення:

- Кожен рядок зображення X/Y представляє одне зображення, яке обробляється. Він містить роздільну здатність зображення, кількість виявлених об’єктів (наприклад, 1 БТР) і час, витрачений на виявлення;
- Також надається інформація, пов’язана зі швидкістю, наприклад, час попередньої обробки, час виведення та час NMS на одне зображення.

6. Результати:

- Результати збережено до runs/detect/exp: Вказує на каталог, до якого зберігаються результати розпізнавання.

Таким чином, цей вивід записує процес розпізнавання YOLOv5 на наборі зображень, вказуючи, звідки завантажено навчені ваги, джерело зображень, різні використані параметри та результати розпізнавання. Це дає

чітке розуміння того, як модель працювала на кожному зображенні і де можна знайти результати.

Використовуючи модель до кожної теки з класів об’єктів отримає теки де будуть відображені результати (рис. 3.12).

Рисунок 3.12 — Створені теки з результатами аналізу моделі

Розглянемо результати для кожного з класів:

Дрон (рис. 3.13).

Рисунок 3.13 — Результат аналізу зображень з дронами

Як видно з результатів модель добре розпізнає дрона, навіть на зображеннях де погано видно.

Зброя (рис. 3.14).

Рисунок 3.14 — Результат аналізу зображень з зброєю

Модель добре розрізняє зброю в хорошому освітленні, далі буде продемонстровано детекцію зброї в польових умовах.

Літак (рис. 3.15).

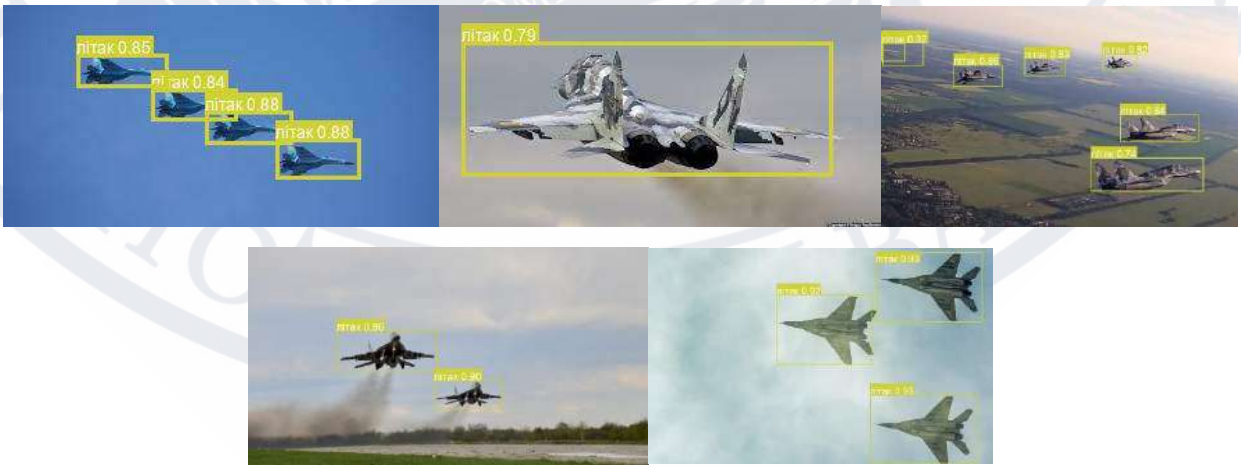


Рисунок 3.15 — Результат аналізу зображень з літаками

Літаки модель виявляє добре, з різних ракурсів що теж є вагомим плюсом.

Міномет (рис. 3.16).

Рисунок 3.16 — Результат аналізу зображень з мінометами

Міномети теж пройшли тестування, хоча з похибкою, так як модель виявляє і ППО за візуальну схожість з мінометом.

ППО (рис. 3.17).

Рисунок 3.17 — Результат аналізу зображень з ППО

Модель виявляє системи ППО та ракети які летять з них, проте потрібно збільшувати кількість дата сету з різними видами таких установок.

Ракета (рис. 3.18).

Рисунок 3.18 — Результат аналізу зображень з ракетами

Модель добре виявляє ракети навіть на розмитих зображеннях, проте потрібно поновлювати дата сет, за рахунок різноманітних видів ракет.

Танк (рис. 3.19).

Рисунок 3.19 — Результат аналізу зображень з танками

Якщо на зображенні є більше ніж один танк, модель може робити похибки, але не значні. Тому можна вважати що тестування на детекцію танків модель також пройшла.

Цивільний об’єкт (рис. 3.20).

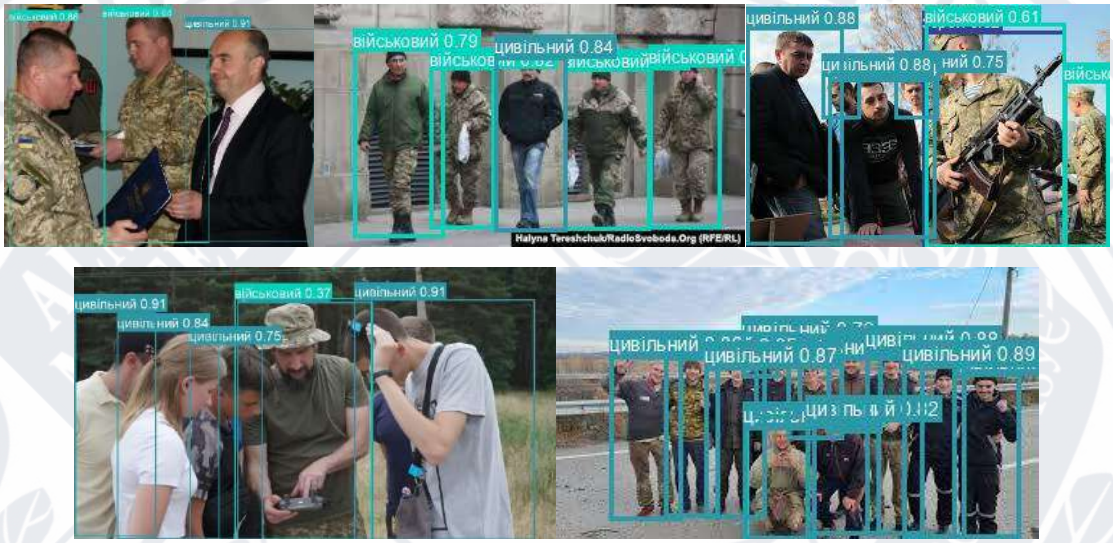


Рисунок 3.20 — Результат аналізу зображень з цивільними об’єктами

Якщо на зображенні багато об’єктів детекції, то їх маркери накладаються, що не дає змоги оцінити точність маркування. Проте, візуально модель справилась з виявленням цивільних об’єктів.

БТР (рис. 3.21).

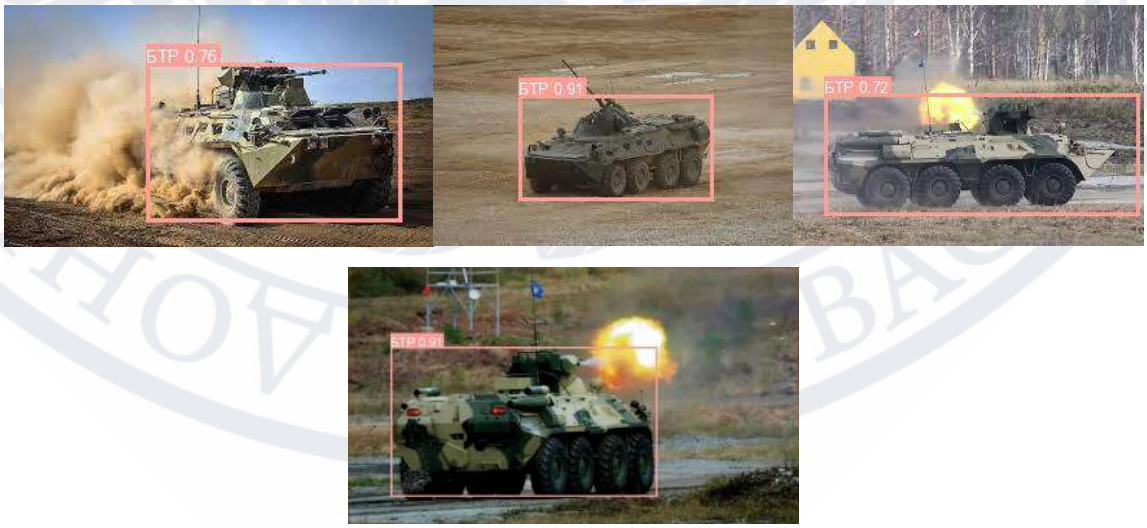


Рисунок 3.21 — Результат аналізу зображень з БТР

Модель розпізнає БТР з різних ракурсів, тому можна вважати що тестування пройдено.

Безпілотник (рис. 3.22).

Рисунок 3.22 — Результат аналізу зображень з безпілотниками

Модель розрізняє літак та безпілотник що вже є хорошим результатом, враховуючи детекцію з різних ракурсів можна зробити висновки що тестування пройдено успішно.

Бронемашина (рис. 3.23).

Рисунок 3.23 — Результат аналізу зображень з бронемашинами

Модель відрізняє бронемашини від танків, БТР що добре. Також виявляє військових та зброю, що є хорошим результатом та показником успішно пройденого тестування.

Гелікоптер (рис. 3.24).

Рисунок 3.24 — Результат аналізу зображень з гелікоптерами

Модель добре виявляє такі об’єкти навіть якщо вони близько один до одного. Тестування пройдено.

Граната (рис. 3.25).

Рисунок 3.24 — Результат аналізу зображень з гранатами

Хоча модель і гарно виявляє гранати на зображеннях, на полях бою гранати не будуть мати такої чіткої форми що може заплутати модель. Хоча і тестування можна вважати вдалим, модель все одно потрібно вдосконалювати та збільшувати дата сет.

Результати аналізу зображень свідчать про те, що модель демонструє похвальну продуктивність в цілому. Точність виявлення на вибірковому наборі даних свідчить про надійну архітектуру та ретельне навчання. Однак реальні зони бойових дій вносять певний рівень складності, який не завжди присутній у контрольованих наборах даних. На реальному полі бою об’єкти часто можуть бути частково приховані димом, уламками або просто динамічним характером бойового середовища. Така часткова видимість ускладнює точне виявлення об’єктів. Крім того, помітним обмеженням є накладання маркувань одне на інше. Ця проблема потенційно може перешкоджати читабельності виявлених об’єктів, особливо в захаращених сценах, де багато об’єктів знаходяться в безпосередній близькості один від одного. В оперативних сценаріях, де рішення, прийняті за частки секунди, можуть мати значні наслідки, забезпечення чіткості ідентифікації об’єктів має першорядне значення.

Для того, щоб підвищити корисність моделі в зонах активних бойових дій, необхідно і надалі покращувати її здатність розпізнавати частково видимі об’єкти і вдосконалювати етапи пост-обробки, щоб уникнути накладання маркувань. Постійний зворотній зв’язок та ітеративне вдосконалення з використанням реальних даних з бойових сценаріїв будуть ключовими у вирішенні цих завдань.

ВИСНОВКИ

У кваліфікаційній роботі описано ключову роль виявлення об’єктів у зонах активних бойових дій. В умовах, коли кожна секунда і кожне рішення можуть бути важливими, автоматизоване виявлення допомагає приймати швидкі і точні рішення, потенційно рятуючи життя і оптимізуючи стратегічні переміщення. Вивчаючи існуючий ландшафт інструментів і методів, стало очевидно, що, незважаючи на значний прогрес, існує низка проблем і обмежень. Деяким інструментам може не вистачати точності, необхідної для таких сценаріїв з високими ставками, тоді як інші можуть бути недостатньо пристосовані до динамічної природи зон бойових дій.

В результаті виконання роботи було представлено нейронну мережу YOLOv5 як перспективне рішення для подолання деяких обмежень, визначених у роботі. Досліджено її архітектурні нюанси, щоб зрозуміти, чому вона вирізняється швидкістю і точністю. Підкреслено важливість формування набору даних, детально описано ретельний процес відбору, маркування та підготовки даних для навчання. Підкреслено, що продуктивність YOLOv5 значною мірою залежить від якості навчального набору даних. Отримавши базові знання про мережу, далі було заглиблення в механіку навчання нейронної мережі, підкресливши її ітеративний характер і ключову роль ваг, обчислення втрат і оптимізації.

Наступним йде перехід від теорії до практики, описуючи реальний шлях впровадження. Описано практичний процес створення наборів даних, наголошуючи на таких інструментах, як Make Sense для точних анотацій. Маючи готові дані, досліджено практичні аспекти навчання нейронних мереж за допомогою Google Colab, використовуючи його обчислювальні можливості та зручність навчання в хмарі. Висновком стало використання навченої моделі, аналіз її продуктивності, висновки з отриманих результатів і роздуми над потенційними покращеннями або коригуваннями для майбутніх ітерацій.

Підсумовуючи, ця робота висвітлює потенціал виявлення об’єктів у зонах бойових дій, особливо коли його використовують за допомогою передових нейронних мереж, таких як YOLOv5. Однак вона також підкреслює складність і ретельність, необхідну на кожному кроці, від створення набору даних до навчання і розгортання моделі. Оскільки технології продовжують розвиватися, а бойові сценарії стають дедалі складнішими, синергія між військовими стратегіями і технологічними досягненнями, такими як YOLOv5, буде ставати дедалі більш необхідною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Computer Vision and Computing Power — What processing power do you need to do Computer Vision ?. Rajan Sharma URL: https://medium.com/analytics-vidhya/computer-vision-and-computing-power-what-processing-power-do-you-need-to-do-computer-vision-c5c45e51a837;

2. Which hardware components (CPU, RAM, GC, etc.) are needed for a machine learning/deep learning home PC/computer to run fast? URL: https://www.quora.com/Which-hardware-components-CPU-RAM-GC-etc-are-needed-for-a-machine-learning-deep-learning-home-PC-computer-to-run-fast;

3. Improved YOLOv5-Based Lightweight Object Detection Algorithm for People with Visual Impairment to Detect Buses URL: https://www.mdpi.com/2076-3417/13/9/5802;

4. Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks URL: https://hal.science/hal-01869779v2/document;

5. All You Know About LIDAR Systems and Applications. URL: https://www.elprocus.com/lidar-light-detection-and-ranging-working-application/;

6. Object detection, shape fitting, and tracking in lidar point cloud data. URL: mathworks.com/help/lidar/detectiontracking.html;

7. How does 3D data become actionable information? – Object detection and tracking. URL: https://www.blickfeld.com/blog/object-detection-and-tracking/;

8. A Survey on Deep-Learning-Based LiDAR 3D Object Detection for Autonomous Driving. URL: https://www.mdpi.com/1424-8220/22/24/9577;

9. Real time object detection using LiDAR and camera fusion for autonomous driving. URL: https://www.nature.com/articles/s41598-023-35170-z;

10. Принцип роботи радіолокатора. RAdio (Aim) Detecting And Ranging URL: https://www.radartutorial.eu/01.basics/rb06.uk.html;

11. Artificial Intelligence vs. Human Intelligence URL: https://www.simplilearn.com/artificial-intelligence-vs-human-intelligence-article;

12. Intelligence URL: https://www.fbi.gov/about/leadership-and-structure/intelligence;

13. A Guide to Human Intelligence (HUMINT) URL: https://greydynamics.com/a-guide-to-human-intelligence-humint/;

14. Psychology of Intelligence Analysis. Richards J. Heuer, Jr. CENTER for the STUDY of INTELLIGENCE Central Intelligence Agency.

15. INCRYPTED. Що таке машинне навчання? Усе, що вам потрібно знати URL: https://incrypted.com/ua/mashynne-navchannja/;

16. Data Science Blogathon. Basic Introduction to Convolutional Neural Network in Deep Learning. URL: analyticsvidhya.com/blog/2022/03/basic-introduction-to-convolutional-neural-network-in-deep-learning/;

17. Згорткова нейронна мережа – просте пояснення CNN та її застосування. URL: https://evergreens.com.ua/ua/articles/cnn.html;

18. ЗАСТОСУВАННЯ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ. Ковальчук А.М. ,Марчук Г.В., Марчук Д.К. Державний університет «Житомирська політехніка». URL: tech.vernadskyjournals.in.ua/journals/2019/4_2019/part_1/15.pdf;

19. ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ. URL: https://conf.ztu.edu.ua/wp-content/uploads/2019/02/45-1.pdf;

20. Застосування нейронної мережі для стилізованої обробки зображень. URL: https://conf.ztu.edu.ua/wp-content/uploads/2019/02/45-1.pdf;

21. W. Chen, Z. Fu, D. Yang, and J. Deng, “Single-Image Depth Perception in the Wild,” Adv. Neural Inf. Process. Syst., pp. 730–738, 2016.;

22. A. G. Howard, Some Improvements on Deep Convolutional Neural Network Based Image Classification. 2013. URL: https://arxiv.org/abs/1312.5402

23. P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays, “Transient attributes for high-level understanding and editing of outdoor scenes,” ACM Trans. Graph., vol. 33, no. 4, pp. 1–11, 2014, URL: dl.acm.org/doi/10.1145/2601097.2601101;

24. C. Li and M. Wand, “Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis,” 2016,, URL: https://arxiv.org/abs/1601.04589;

25. YOLO: Real-Time Object Detection. URL: pjreddie.com/darknet/yolo/;

26. YOLO — You only look once, real time object detection explained. URL: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006;

27. Comparative Analysis of YOLO and SSD. URL: https://medium.com/@nikitamalviya/comparative-analysis-of-yolo-and-ssd-7433e249d429;

28. The Faster R-CNN algorithm Only Look Once (YOLO). URL: https://www.neuralception.com/objectdetection-yolo/;

29. Object Detection using the YOLO Model. URL: https://medium.com/@siddhijadhav98/object-detection-using-the-yolo-model-20596b725ad6;

30. YOLO object detection with OpenCV. URL: https://pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/;

31. Faster R-CNN Explained for Object Detection Tasks. URL: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/;

32. Detect an object with OpenCV-Python. URL: geeksforgeeks.org/detect-an-object-with-opencv-python/;

33. Object Detection with Python, Deep Learning, and OpenCV. URL: https://dontrepeatyourself.org/post/object-detection-with-python-deep-learning-and-opencv/;

34. OpenCV Haar Cascades. Adrian Rosebrock URL: https://pyimagesearch.com/2021/04/12/opencv-haar-cascades/;

35. Object Detection using Haar feature-based cascade classifiers. URL: docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html;

36. opencv-python 4.8.1.78. URL: https://pypi.org/project/opencv-python/;

37. Create production-grade machine learning models with TensorFlow. URL: https://www.tensorflow.org/;

38. Caffe. URL: https://caffe.berkeleyvision.org//;

39. Image Pyramids with Python and OpenCV. URL: https://pyimagesearch.com/2015/03/16/image-pyramids-with-python-and-opencv/;

40. Designing efficient accelerator of depthwise separable convolutional neural network on FPGA. Ding Wei, Huang Zeyu, Huang Zunkai , Tian Li, Wang Hui. URL:sciencedirect.com/science/article/abs/pii/S1383762118304612;

41. Object detection using YOLO: challenges, architectural successors, datasets and applications. URL: https://link.springer.com/article/10.1007/s11042-022-13644-y;

42. Renjie Xu, Haifeng Lin, Kangjie Lu, Lin Cao, A Forest Fire Detection System Based on Ensemble Learning, лютий 2021 року, 7 ст.;

43. Aditya Sharma. Introduction to the YOLO Family URL: https://pyimagesearch.com/2022/04/04/introduction-to-the-yolo-family/;

44. CSP-DarkNet. URL: https://huggingface.co/docs/timm/models/csp-darknet;

45. Glenn Jocher. This guide explains how to train your own custom dataset with YOLOv5. URL: https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data/;

46. Make Sense. URL: https://www.makesense.ai/;

47. Quickly Create Datasets for Training YOLO Object Detection with Label Studio. URL: https://labelstud.io/blog/quickly-create-datasets-for-training-yolo-object-detection-with-label-studio//;

48. The practical guide for Object Detection with YOLOv5 algorithm. URL: towardsdatascience.com/the-practical-guide-for-object-detection-with-yolov5-algorithm-74c04aac4843;

49. How Does the YOLO Algorithm Work?. URL: exxactcorp.com/blog/Deep-Learning/YOLOv5-PyTorch-Tutorial;

50. ATC-YOLOv5: Fruit Appearance Quality Classification Algorithm Based on the Improved YOLOv5 Model for Passion Fruits. URL: https://www.mdpi.com/2227-7390/11/16/3615;

51. Ultralytics. YOLOv5. URL: https://github.com/ultralytics/yolov5;

52. Understanding Backpropagation: The Engine Behind Neural Network Learning. URL: medium.com/@evertongomede/understanding-backpropagation-the-engine-behind-neural-network-learning-a7c2e1acdbf;

53. YOLOv5 Tutorial. This YOLOv5 notebook by Ultralytics presents simple train, validate and predict examples to help start your AI adventure. URL: colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=wbvMlHd_QwMG;

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____

_____

Повністю вказується ПІБ та статус (посада для працівників, освітня (освітньо-наукова) програма – для здобувачів вищої освіти)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)