МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ТОКАР РОСТИСЛАВ ОЛЕКСАНДРОВИЧ

Допускається до захисту:

завідувач кафедри ІТ к.т.н.,

доцент О. В. Зелінська

“___” _____ 20 __ р.

Розробка інтелектуальної системи аналізу образів та розпізнавання об’єктів

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:

Ніколюк П. К., д-р фіз.-мат. наук,

професор

Оцінка:____/ ____/ _____

(бали за шкалою СКТС/за національною шкалою)

Голова ЕК: _____

(підпис)

Вінниця 2024

АНОТАЦІЯ

Токар Р.О. «Розробка інтелектуальної системи аналізу образів та розпізнавання об’єктів». Спеціальність 122 «Комп’ютерні науки», освітня програма «Сучасні інформаційні технології та програмування». Донецький національний університет імені Василя Стуса, Вінниця 2023.

У кваліфікаційній (бакалаврській) роботі досліджено та проаналізовано методи розпізнавання об’єктів військової галузі, а також інструменти для проєктування та тренування подібних систем. За допомогою таких технологій, як мова програмування Python, фреймворк PyTorch, бібліотека комп’ютерного зору OpenCV, модель розпізнавання YOLOv5 була розроблена інтелектуальна система розпізнавання військових об’єктів.

Ключові слова: розпізнавання, військова техніка, машинне навчання, комп’ютерний зір, нейромережі

ABSTRACT

Tokar R.O. "Development of an intelligent image analysis and recognition system objects". Specialty 122 "Computer science", educational program "Modern information technologies and programming". Vasyl Stus Donetsk National University, Vinnytsia 2023.

In the qualification (bachelor's) thesis, the methods of recognizing military objects, as well as tools for designing and training similar systems, were researched and analyzed. With the help of such technologies as the Python programming language, the PyTorch framework, the OpenCV computer vision library, and the YOLOv5 recognition model, an intelligent system for recognizing military objects was developed.

Keywords: recognition, military equipment, machine learning, computer vision, neural networks

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

GCV API – Google Cloud Vision API

OpenCV – Open Computer Vision Library

RTSP – Real Time Streaming Protocol

YOLO – You Only Look Once

DAPRA – Defense Advanced Research Projects Agency

YAML - Yet Another Markup Language

mAP – mean Average Precision

ВСТУП

Актуальність теми дослідження. В сучасних умовах глобальної безпеки та зростаючого числа конфліктів, потреба в ефективних системах розпізнавання військової техніки є надзвичайно актуальною, в особливості в умовах сьогоденної повномасштабної російсько-української війни. Військові операції стають дедалі складнішими та технологічно розвинутішими, що вимагає використання новітніх технологій для забезпечення надійної та своєчасної ідентифікації об'єктів на полі бою. В цьому контексті, інтелектуальні системи аналізу образів і розпізнавання об'єктів відіграють важливу роль у дотриманні безпеки та ефективності військових операцій.

Актуальність вибраної теми також підкріплюється необхідністю забезпечення оперативного моніторингу та аналізу бойової обстановки. Інтелектуальні системи розпізнавання об'єктів можуть значно підвищити ситуаційну обізнаність військових підрозділів, що, у свою чергу, сприяє прийняттю більш ефективних рішень та зниженню ризику для життя військовослужбовців. Тому розробка подібних систем є важливою складовою сучасної військової техніки та технологій.

Таким чином, дослідження та розробка інтелектуальної системи аналізу образів та розпізнавання об'єктів є надзвичайно актуальними завданнями, що відповідають викликам сучасної військової справи та сприяють підвищенню рівня безпеки та ефективності бойових операцій.

Мета роботи полягає в розробці ефективної системи для автоматичного виявлення та класифікації військових об'єктів на зображеннях і відео, залучаючи застосування сучасних методів машинного навчання та комп'ютерного зору.

Наші **завдання** включають в себе:

1. Дослідження можливостей доступних технологій для розробки подібних систем

2. Проєктування інтелектуальної системи розпізнавання об’єктів

3. Тренування системи для подальшого використання

4. Оцінка способів можливого використання системи

Об’єктом дослідження є процес проєктування та тренування системи розпізнавання військових об’єктів.

Практична цінність. Практичне значення цієї дипломної роботи полягає у створенні ефективної та надійної інтелектуальної системи розпізнавання військових об'єктів, яка може бути застосована в реальних умовах для забезпечення надійного моніторингу та аналізу бойової обстановки. Ця система має потенціал для значного покращення операційної обізнаності та прийняття рішень у військових підрозділах.

Структура кваліфікаційної роботи:

Робота складається із вступу, трьох розділів, висновку, використаних джерел із 42 пунктів, включаючи офіційну документацію обраних технологій, довідники та посібники, 19 рисунків. Обсяг роботи - 50 сторінок.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Напрямок дослідження

Розробка інтелектуальної системи аналізу образів та розпізнавання об’єктів у військовій сфері є надзвичайно важливим напрямком дослідження. Такі системи можуть суттєво підвищити ефективність та точність військових операцій за рахунок автоматизації процесів ідентифікації та відстеження різноманітних об'єктів, таких як військова техніка. Застосування сучасних технологій машинного навчання та комп’ютерного зору дозволяє оперативно аналізувати великі обсяги візуальних даних і приймати обґрунтовані рішення в реальному часі.

Основний **напрямок дослідження** зосереджується на розпізнаванні об'єктів на зображеннях та відео, що передбачає використання алгоритмів машинного навчання для обробки візуальної інформації. Глибокі нейронні мережі, зокрема моделі типу YOLOv5, про які я буду розповідати у наступному розділі, забезпечують високу точність та швидкість розпізнавання об'єктів, що є критично важливим у військових умовах, де час реагування має вирішальне значення.

Однією з ключових задач у цьому напрямку є створення та анотація датасетів. Це включає збирання великої кількості зображень, що містять різні типи військової техніки, та їх подальшу анотацію, тобто позначення об'єктів на зображеннях. Цей процес є критично важливим для навчання моделей розпізнавання об'єктів, оскільки якість даних значно впливає на точність і надійність отриманих результатів.

Процес навчання моделі включає підготовку даних, вибір оптимальних гіперпараметрів та використання алгоритмів оптимізації для покращення точності моделі. Навчання моделі відбувається на основі підготовлених даних, і після завершення цього процесу модель проходить етап валідації, де її точність оцінюється на тестових наборах даних.

Завершальним етапом є реальне застосування натренованих моделей для аналізу відеопотоків. Відео може надходити з дронів або стаціонарних камер, що розташовані в зоні бойових дій. Інтеграція системи в реальні умови бойових операцій дозволяє автоматично розпізнавати та відстежувати об'єкти в реальному часі, що суттєво підвищує ефективність та безпеку військових операцій.

Таким чином, розробка інтелектуальної системи аналізу образів та розпізнавання об’єктів у військовій сфері є важливим та перспективним напрямком дослідження. Використання сучасних технологій та інструментів дозволяє створювати системи, які можуть значно покращити оперативну ситуацію на полі бою, сприяти швидшому прийняттю рішень та підвищити загальну ефективність військових дій.

1.2. Аналіз відомих аналогів

В умовах сучасних військових конфліктів використання інтелектуальних систем для аналізу образів і розпізнавання об'єктів набуває все більшого значення. Різні країни та військові організації розробляють і впроваджують подібні системи, щоб підвищити точність і швидкість прийняття рішень на полі бою. У цьому розділі буде розглянуто кілька відомих аналогів таких систем, їх функціональні можливості, технологічні рішення та області застосування.

1.2.1. DARPA (Defense Advanced Research Projects Agency)

DARPA [1] є одним з провідних агентств у розробці передових технологій для оборонних цілей. Однією з їхніх розробок є система "Maven", яка використовує методи глибокого навчання для аналізу відео, отриманих з дронів.

Проєкт Maven дозволяє автоматично розпізнавати об'єкти на зображеннях і відео, що значно прискорює процес аналізу даних та прийняття рішень. Система навчена на великій кількості зображень, що містять різні типи військової техніки, транспортних засобів та інших об'єктів інтересу.

Основні функції системи:

1. Автоматичне розпізнавання об'єктів на відео.

2. Класифікація типів об'єктів.

3. Відстеження рухомих об'єктів.

Переваги:

1. Висока точність: Maven використовує передові методи глибокого навчання, що забезпечує високу точність розпізнавання об'єктів на відео.

2. Реальний час: Система здатна аналізувати відео в реальному часі, що є критично важливим для оперативного прийняття рішень у бойових умовах.

Недоліки:

1. Спеціалізація: Система розроблена спеціально для військових цілей, що обмежує її використання в інших галузях.

2. Високі витрати: Розробка і підтримка таких передових систем потребує значних фінансових вкладень.

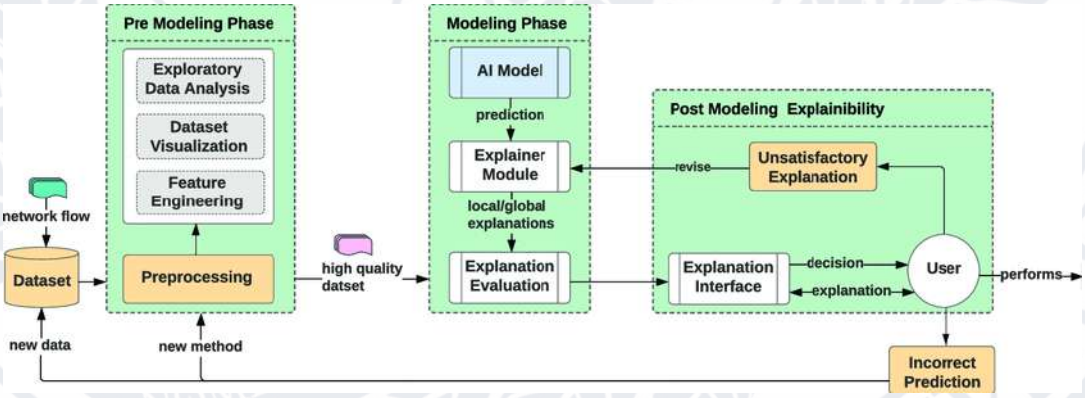

Рисунок 1.1 – Архітектура для проєктування на основі DAPRA

1.2.2. Ізраїльська оборонна компанія Elbit Systems

Elbit Systems [2] розробляє передові системи для обробки зображень і відео, що використовуються в оборонній сфері. Їхні рішення включають системи розпізнавання об'єктів, які можуть автоматично ідентифікувати та відстежувати різні цілі, такі як танки, бронетранспортери та інша військова техніка. Вони

використовують комбінацію методів машинного навчання та алгоритмів комп'ютерного зору для досягнення високої точності.

Основні функції системи:

1. Автоматичне розпізнавання та класифікація військових об'єктів.

2. Висока точність навіть у складних умовах.

3. Підтримка різних типів датчиків і камер.

4. Можливість інтеграції з існуючими системами управління боєм.

Переваги:

1. Інноваційні рішення: Elbit Systems відома своїми інноваційними підходами у розробці систем розпізнавання об'єктів, що забезпечує високу точність та ефективність.

2. Висока інтеграція: Системи від Elbit Systems легко інтегруються з іншими військовими технологіями та платформами, що підвищує оперативну сумісність.

Недоліки:

1. Спеціалізація: Системи від Elbit Systems спеціалізуються на військових застосуваннях, що обмежує їх використання в інших галузях.

2. Конфіденційність: Як і у випадку з іншими військовими системами, деталі про роботу системи можуть бути засекреченими, що обмежує доступ до інформації для цивільних дослідників і розробників.

Рисунок 1.2 - Емблема Ізраїльської оборонної компанії

1.2.3 Google Cloud Vision API

Google Cloud Vision API [3] є комерційним рішенням для розпізнавання об'єктів, яке використовує потужні алгоритми машинного навчання для аналізу зображень. Хоча цей сервіс не розроблений спеціально для військових цілей, його можливості можуть бути використані для різних завдань, включаючи розпізнавання військової техніки на зображеннях.

Основні функції системи:

1. Розпізнавання та класифікація об'єктів.
2. Витяг тексту з зображень.
3. Аналіз емоцій на обличчях.
4. Інтеграція з іншими сервісами Google Cloud.

Переваги:

1. Гнучкість застосування: Сервіс може використовуватись для різноманітних завдань, від розпізнавання об'єктів до аналізу тексту на зображеннях.

2. Доступність: Google Cloud Vision API доступний для широкого кола користувачів, включаючи розробників з різних галузей.

Недоліки:

1. Затримки: Хоча сервіс працює швидко, обробка зображень у хмарі може призводити до невеликих затримок, що не завжди прийнятно для критичних військових операцій.

2. Безпека даних: Використання хмарних сервісів для обробки чутливої інформації може викликати побоювання щодо безпеки та конфіденційності даних.

Рисунок 1.3 - Приклад роботи GCV API

1.3. Аналіз проблем та актуалізація рішень

Сучасні військові операції вимагають використання передових технологій для безпеки та ефективності виконання завдань. Однією з ключових задач у військовій галузі є розпізнавання та ідентифікація об'єктів на полі бою. Це завдання ускладнюється через високий динамізм та хаотичність бойових дій, різноманітність об'єктів та їх швидке пересування. Тому розробка системи, яка здатна ефективно ідентифікувати військові об'єкти, є надзвичайно актуальною.

Проблеми в існуючих системах:

1. Недостатня точність розпізнавання: Багато існуючих систем розпізнавання об'єктів мають обмежену точність, що може призвести до помилок у виявленні та ідентифікації цілей. Це особливо критично у військових умовах, де навіть незначна помилка може призвести до серйозних наслідків.

2. Високі витрати: Розробка та впровадження передових систем розпізнавання об'єктів потребує значних фінансових ресурсів. Це включає вартість обладнання, програмного забезпечення, а також витрати на навчання персоналу.

4. Залежність від обладнання: Багато систем розпізнавання об'єктів сильно залежать від якості обладнання (камери, сенсори), що встановлені на військових платформах. Це може обмежити їх ефективність у різних умовах.

5. Конфіденційність та безпека даних: Використання хмарних сервісів для обробки чутливої інформації викликає побоювання щодо безпеки та конфіденційності даних. Військові операції вимагають високого рівня захисту інформації від несанкціонованого доступу.

Актуалізація рішень:

1. Оптимізація витрат: Застосування відкритих та безкоштовних платформ для розробки, таких як PyTorch та OpenCV, може значно знизити витрати на створення системи. Крім того, використання хмарних обчислень може зменшити витрати на обладнання та інфраструктуру.

2. Інтеграція з іншими системами: Розробка системи з урахуванням можливості легкої інтеграції з існуючими військовими платформами та технологіями забезпечить оперативну сумісність та ефективність використання в бойових умовах.

3. Незалежність від обладнання: Використання алгоритмів, що можуть працювати на різних типах обладнання та в різних умовах, підвищить універсальність та надійність системи. Наприклад, використання RTSP (Real-Time Streaming Protocol) для передачі відеопотоку забезпечить сумісність з різними типами камер.

4. Безпека даних: Використання локальних обчислень замість хмарних сервісів для обробки чутливої інформації підвищить рівень безпеки та конфіденційності даних. Крім того, застосування сучасних методів шифрування та захисту інформації дозволить зменшити ризики несанкціонованого доступу.

1.4. Питання етики у розпізнаванні техніки

Етичні питання та приватність:

Розпізнавання військової та цивільної техніки за допомогою камер і систем відеоспостереження може викликати етичні питання, особливо якщо здійснюється без належного контролю та дотримання правових норм. Використання камер для розпізнавання техніки може порушувати приватність громадян, зокрема у громадських місцях, офісах та житлових будинках. Недостатня регуляція може призвести до зловживань, коли і де може проводитися розпізнавання техніки, що потенційно порушує приватність. Наприклад, розпізнавання може використовуватися для моніторингу поведінки споживачів з метою реклами, що викликає питання про етичність таких дій та збереження конфіденційності даних.

Використання для військової безпеки:

Камери для розпізнавання військової техніки можуть бути корисними для забезпечення безпеки та військового контролю. Однак це також підвищує ризики для приватності та безпеки військових даних, якщо системи розпізнавання потраплять у руки зловмисників. Використання таких систем може бути направлене на зловмисні дії, як-от слідкування за особами, викрадення транспорту, тероризм чи шпигунство, що піднімає питання про те, як забезпечити безпеку та контроль над такими технологіями.

Помилки та неточності:

Використання штучного інтелекту та алгоритмів для розпізнавання техніки може призводити до помилок і хибних висновків, що може мати серйозні наслідки для осіб, які стають об'єктом розпізнавання. Незважаючи на це, розпізнавання військової та цивільної техніки за допомогою камер може суттєво підвищити загальну безпеку в суспільстві. Такі системи можуть використовуватися для виявлення та запобігання злочинам, як-от викрадення автомобілів або несанкціоноване використання військової техніки.

Пошук зниклих осіб:

Системи розпізнавання техніки можуть допомагати в пошуку зниклих осіб, зокрема в автомобілях або інших видах транспорту, що сприяє рятувальним операціям та пошуку зниклих людей.

Військовий контекст:

У військовому контексті розпізнавання техніки допомагає забезпечувати національну безпеку. Військові сили можуть використовувати такі системи для відстеження та ідентифікації військової техніки противника, що допомагає уникнути непорозумінь та конфліктів, особливо в зонах бойових дій. Це сприяє розвитку діалогу та вирішенню конфліктів шляхом підтримки відомого розташування та ідентифікації сторін.

Надзвичайні ситуації:

Під час природних катастроф чи аварій розпізнавання техніки допомагає координувати рятувальні операції та ефективно розподіляти ресурси. Багато злочинів включають використання транспортних засобів, тому розпізнавання техніки сприяє виявленню та розслідуванню злочинів, зменшенню кримінальної активності та підвищенню відчуття безпеки в суспільстві.

Цивільний контекст та медіа:

Окрім військових застосувань, такі технології можуть допомагати цивільному населенню вчасно отримувати попередження про можливу небезпеку та надавати інформацію для висвітлення подій журналістами. Наприклад, перед початком російського вторгнення в Україну 2022 року супутникові знімки та відео з дронів, на яких було зафіксовано скупчення військової техніки біля кордону, стали сигналом та дали підстави для підготовки до можливих провокацій чи реального збройного конфлікту. Розповсюдження цих матеріалів у ЗМІ допомогло звернути увагу світової спільноти та лідерів інших держав на проблему до початку повномасштабного вторгнення.

Рятувальні операції:

Такі технології також можуть використовуватися для рятування життів поранених військових або людей у небезпечних ситуаціях.

РОЗДІЛ 2. АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Динамічна мова програмування Python.

Python [4] — це високорівнева мова програмування, яку розробив Гвідо ван Россум і випустив вперше в 1991 році. Вона характеризується простим і чітким синтаксисом, що робить її ідеальною для швидкої розробки програм. Python підтримує об'єктно-орієнтоване, процедурне та функціональне програмування, що забезпечує гнучкість під час розробки різних типів застосунків.

2.1.1. Основні характеристики Python:

1. Легкість вивчення та використання: Завдяки простому синтаксису Python є однією з найкращих мов для початківців. Код на Python часто є зрозумілішим і коротшим порівняно з іншими мовами.
2. Велика стандартна бібліотека: Python має багату стандартну бібліотеку, яка охоплює багато стандартних функцій і операцій, що дозволяє розробникам зосередитися на логіці програми, а не на написанні базових функцій.
3. Портативність: Python підтримується багатьма платформами, включаючи Windows, macOS, Linux та інші. Це дозволяє розробляти кросплатформені застосунки.
4. Інтерпретована мова: Python є інтерпретованою мовою, що означає, що код виконується рядок за рядком, полегшуючи відлагодження і тестування.
5. Підтримка багатьох парадигм програмування: Python підтримує об'єктно-орієнтоване, процедурне та функціональне програмування, що надає розробникам можливість вибору стилю програмування.
6. Розвинена екосистема та спільнота: Python має велику кількість сторонніх бібліотек та активну спільноту розробників, що забезпечує постійний розвиток мови та підтримку.

2.1.2. Причини вибору мови Python

1. Інтуїтивність та читабельність коду: Python має простий та зрозумілий синтаксис, що дозволяє швидко писати та підтримувати код. Це особливо важливо при роботі з великими обсягами даних та складними моделями машинного навчання.

2. Багата бібліотека для машинного навчання та комп'ютерного зору: Python має широкий набір бібліотек для машинного навчання (PyTorch, TensorFlow) та обробки зображень (OpenCV, Pillow), що робить його ідеальним вибором для розробки систем розпізнавання об'єктів.

3. Широке використання в наукових дослідженнях: Python є популярною мовою серед науковців і дослідників завдяки своїй підтримці наукових бібліотек (NumPy, SciPy, Pandas) та інтеграції з інструментами для візуалізації даних (Matplotlib, Seaborn).

4. Гнучкість та масштабованість: Python дозволяє легко інтегруватися з іншими мовами програмування та системами, що забезпечує гнучкість при розробці комплексних систем.

2.1.3. Порівняння Python з іншими мовами програмування.

Python vs C++

1. Читабельність та простота: Python значно простіший у використанні та читанні коду порівняно з C++. Це дозволяє швидше розробляти та підтримувати проєкти.
2. Швидкість виконання: C++ зазвичай швидший за Python у виконанні завдяки компіляції коду. Однак, для задач машинного навчання Python є більш зручним завдяки численним бібліотекам, які оптимізовані для швидкої обробки даних.

Python vs Java

1. Синтаксис: Python має більш лаконічний і зрозумілий синтаксис порівняно з Java, що спрощує розробку.
2. Бібліотеки: Python має більш потужні та різноманітні бібліотеки для машинного навчання та обробки зображень, ніж Java.

2.2 Фреймворк PyTorch

PyTorch [5] — це популярний фреймворк для машинного навчання, розроблений і підтримуваний компанією Facebook (Meta). Він був випущений у 2016 році і швидко здобув популярність завдяки своїй гнучкості, простоті використання та широким можливостям. PyTorch дозволяє розробникам легко створювати складні моделі нейронних мереж і забезпечує ефективні інструменти для їхнього навчання та виводу.

2.2.1 Основні характеристики PyTorch

1. Динамічні обчислювальні графи: PyTorch використовує динамічні обчислювальні графи, що означає, що граф обчислень створюється на льоту під час виконання програми. Це забезпечує гнучкість і дозволяє легко налагоджувати моделі.

2. Інтуїтивно зрозумілий API: PyTorch має зрозумілий і зручний для розробників API, що спрощує написання коду і робить його більш читабельним.

3. Підтримка CUDA: PyTorch має вбудовану підтримку CUDA для прискорення обчислень на GPU, що є важливим для навчання великих нейронних мереж.

4. Широка бібліотека інструментів: PyTorch містить широкий набір інструментів для роботи з нейронними мережами, включаючи модулі для навчання, оцінки та візуалізації моделей.

5. Сумісність з іншими бібліотеками: PyTorch легко інтегрується з іншими бібліотеками та інструментами для машинного навчання, такими як NumPy, SciPy та OpenCV.

2.2.2 Причини вибору фреймворку PyTorch

1. Гнучкість та простота: PyTorch дозволяє розробникам легко експериментувати з різними архітектурами нейронних мереж завдяки своїм динамічним обчислювальним графам. Це особливо важливо при розробці складних моделей для розпізнавання об'єктів.

2. Зручний для дослідників та розробників: PyTorch надає інтуїтивно зрозумілий API, що робить його зручним як для науковців, так і для розробників. Це дозволяє швидко прототипувати і тестувати нові ідеї.

3. Потужні інструменти для навчання моделей: PyTorch забезпечує потужні інструменти для навчання нейронних мереж, включаючи вбудовану підтримку CUDA для прискорення обчислень на GPU. Це дозволяє ефективно навчати великі моделі на великих датасетах.

4. Активна спільнота та підтримка: PyTorch має велику і активну спільноту користувачів, що забезпечує швидку підтримку і обмін досвідом. Крім того, фреймворк постійно розвивається завдяки внескам від спільноти та компанії Facebook.

2.2.3 Порівняння PyTorch з іншими подібними фреймворками

PyTorch vs. TensorFlow

1. Гнучкість: PyTorch використовує динамічні обчислювальні графи, що робить його більш гнучким і зручним для налагодження в порівнянні з TensorFlow, який спочатку використовував статичні графи (хоча TensorFlow 2.0 також підтримує динамічні графи).

2. Простота використання: API PyTorch більш інтуїтивно зрозумілий і простий у використанні, що робить його популярним вибором серед дослідників та розробників.

PyTorch vs. Keras

1. Глибокі налаштування: PyTorch надає більшу гнучкість і можливість глибоких налаштувань моделей у порівнянні з Keras, який є високорівневим API і працює поверх TensorFlow.

2. Навчання на GPU: Обидва фреймворки підтримують навчання на GPU, але PyTorch забезпечує більш безпосередній доступ до низькорівневих операцій.

2.2.4 Використання PyTorch у нашій системі.

У нашій системі для розпізнавання об'єктів PyTorch використовується для кількох ключових етапів:

1. Створення та навчання моделей: PyTorch забезпечує інструменти для створення архітектури нейронної мережі, її навчання на нашому спеціально створеному датасеті, а також для оцінки продуктивності моделі.

2. Обробка даних: PyTorch використовує бібліотеку torchvision для завантаження, перетворення і обробки зображень, що спрощує підготовку даних для навчання моделі.

3. Інтеграція з іншими бібліотеками: PyTorch легко інтегрується з OpenCV для захоплення і обробки відеопотоків в реальному часі, що є критично важливим для нашої системи розпізнавання об'єктів.

4. Виведення результатів: PyTorch дозволяє легко обробляти і відображати результати роботи моделі, використовуючи бібліотеки для візуалізації, такі як OpenCV.

2.3 Бібліотека комп’ютерного зору OpenCV

OpenCV (Open Source Computer Vision Library) [6] — це бібліотека комп'ютерного зору та машинного навчання. Вона була розроблена Intel і випущена у 2000 році. OpenCV містить понад 2500 оптимізованих алгоритмів, які дозволяють розробникам створювати програми для розпізнавання об'єктів, обробки зображень і відео, а також інших завдань комп'ютерного зору.

2.3.1 Основні характеристики OpenCV

1. Розпізнавання об'єктів: OpenCV включає широкий набір алгоритмів для розпізнавання облич, очей, об'єктів, людських рухів тощо.

2. Обробка зображень: Бібліотека надає інструменти для обробки зображень, включаючи фільтрацію, перетворення, визначення контурів і сегментацію.

3. Робота з відео: OpenCV дозволяє захоплювати, обробляти і аналізувати відеопотоки в реальному часі.

4. Машинне навчання: Бібліотека містить модулі для машинного навчання, що підтримують різні алгоритми класифікації та кластеризації.

5. Кросплатформеність: OpenCV підтримує різні операційні системи, включаючи Windows, macOS, Linux, Android та iOS.

2.3.2 Причини вибору бібліотеки OpenCV

1. Широкий функціонал: OpenCV надає великий набір інструментів для обробки зображень і відео, що робить його ідеальним вибором для створення системи розпізнавання об'єктів.

2. Реальний час: OpenCV підтримує обробку відеопотоків в реальному часі, що є критично важливим для нашої системи, яка повинна виявляти та класифікувати об'єкти на відео з дронів чи камер спостереження.

3. Інтеграція з іншими бібліотеками: OpenCV легко інтегрується з PyTorch та іншими бібліотеками машинного навчання, що дозволяє створювати комплексні системи.

4. Простота використання: Бібліотека має зрозумілий API, що спрощує розробку і налагодження програм.

2.3.3. Порівняння OpenCV з Іншими Бібліотеками для Комп'ютерного Зору

OpenCV vs. PIL/Pillow

1. Функціонал: PIL/Pillow більше орієнтований на базову обробку зображень, тоді як OpenCV має ширший набір інструментів для складних завдань комп'ютерного зору.

2. Швидкість: OpenCV зазвичай швидший завдяки оптимізованим алгоритмам і підтримці роботи на GPU.

OpenCV vs. TensorFlow/Keras

1. Обробка зображень: OpenCV краще підходить для обробки зображень і відео, тоді як TensorFlow/Keras більше орієнтовані на глибоке навчання та створення нейронних мереж.

2. Інтеграція: OpenCV легко інтегрується з TensorFlow/Keras для передобробки даних та візуалізації результатів.

2.3.4. Використання OpenCV в нашій системі

1. Захоплення відеопотоків: OpenCV використовується для захоплення відеопотоків з камер спостереження та дронів. Бібліотека дозволяє підключатися до відеоджерел через RTSP протоколи або безпосередньо через URL.

2. Обробка відео та зображень: OpenCV обробляє відеопотоки та

зображення, виконує попередню обробку (наприклад, масштабування, перетворення кольорів) та підготовку даних для моделі.

3. Виведення результатів: OpenCV використовується для виведення результатів роботи моделі на екран, включаючи відображення виявлених об'єктів на відео в реальному часі.

4. Інтеграція з PyTorch: OpenCV забезпечує інтеграцію з PyTorch, дозволяючи передавати оброблені зображення до моделі для передбачення та отримувати результати для подальшого відображення.

2.4 Середовище розробки PyCharm

PyCharm [7] — це середовище розробки для мови програмування Python. PyCharm пропонує розробникам широкий спектр інструментів для написання, тестування та налагодження Python-коду, що робить його одним з найпопулярніших IDE для Python.

2.4.1 Основні характеристики PyCharm

1. Підсвітка синтаксису та автозавершення коду: PyCharm забезпечує розумну підсвітку синтаксису, автозавершення коду та рефакторинг, що значно спрощує процес написання коду.

2. Налагодження та тестування: Вбудовані інструменти для налагодження та тестування допомагають швидко знаходити та виправляти помилки.

3. Інтеграція з системами контролю версій: PyCharm підтримує інтеграцію з Git, SVN, Mercurial та іншими системами контролю версій.

4. Інструменти для роботи з базами даних: IDE має вбудовані інструменти для роботи з різними базами даних, що полегшує процес розробки додатків, які взаємодіють з базами даних.

5. Плагіни та розширення: PyCharm підтримує широкий спектр плагінів, які дозволяють розширювати функціональність IDE відповідно до потреб розробника.

2.4.2 Причини вибору PyCharm

1. Зручність написання коду: Підсвітка синтаксису, автозавершення коду та рефакторинг значно спрощують процес написання та підтримки коду.

2. Потужні інструменти для налагодження: Вбудовані інструменти для налагодження допомагають швидко знаходити та виправляти помилки, що є критичним для розробки складних систем.

3. Розширення функціональності: Можливість встановлення плагінів дозволяє адаптувати PyCharm під конкретні потреби проєкту.

2.4.3 Порівняння PyCharm з Іншими IDE для Python

PyCharm vs. Visual Studio Code

1. Функціональність: PyCharm надає більше інструментів "з коробки" для Python, тоді як Visual Studio Code вимагає додаткових розширень для досягнення подібної функціональності.

2. Налагодження та тестування: PyCharm має більш потужні інструменти для налагодження та тестування Python-коду.

PyCharm vs. Jupyter Notebook

1. Цілісність середовища: PyCharm забезпечує інтегроване середовище для розробки, тестування та налагодження коду, тоді як Jupyter Notebook більше підходить для аналізу даних та прототипування.

2.4.4 Використання PyCharm у нашій системі

1. Написання та рефакторинг коду: PyCharm забезпечує зручне середовище для написання та рефакторингу коду Python, що спрощує розробку складних алгоритмів машинного навчання та обробки зображень.

2. Налагодження та тестування: IDE дозволяє легко налагоджувати та тестувати код, що допомагає швидко знаходити та виправляти помилки.

3. Управління проєктом: PyCharm підтримує структуру проєкту, що спрощує організацію файлів та модулів, а також інтеграцію з системами контролю версій.

2.5 Мережевий протокол RTSP

RTSP (Real-Time Streaming Protocol) [8] — це мережевий протокол, розроблений для управління потоковою передачею медіаданих в реальному часі. RTSP дозволяє клієнтам управляти прийомом мультимедійних даних з сервера. Основна мета RTSP — забезпечити контроль над доставкою потоків відео та аудіо, дозволяючи здійснювати такі операції, як старт, пауза, продовження і зупинка потоку.

2.5.1 Основні характеристики RTSP

1. Контроль потоків в наживо: RTSP дозволяє управляти потоками мультимедіа в реальному часі, забезпечуючи функції запуску, паузи, продовження і зупинки відео- та аудіопотоків.

2. Широка підтримка пристроїв: Протокол підтримується багатьма камерами відеоспостереження, IP-камерами та іншими пристроями, що забезпечує його універсальність.

3. Робота через інтернет: RTSP дозволяє передавати потокові дані через інтернет, що забезпечує віддалений доступ до відеопотоків.

2.5.2 Причини вибору RTSP

1. Реальний час: RTSP забезпечує передачу даних у реальному часі, що є критично важливим для системи розпізнавання об'єктів, яка повинна обробляти та аналізувати відеопотоки в режимі реального часу.

2. Універсальність: RTSP підтримується багатьма сучасними камерами та пристроями відеоспостереження, що дозволяє легко інтегрувати їх у систему.

3. Віддалений доступ: Протокол дозволяє отримувати відеопотоки з віддалених камер через інтернет, що розширює можливості системи для моніторингу великих територій або різних географічних локацій.

2.5.3 Порівняння RTSP з Іншими Протоколами для Відеопотоків

RTSP vs. HTTP Live Streaming (HLS)

1. Реальний час: RTSP забезпечує меншу затримку в передачі даних, що робить його більш придатним для застосувань, де потрібна обробка відео в реальному часі, тоді як HLS підходить для потокової передачі з великою затримкою.

2. Контроль потоків: RTSP надає розширені можливості для управління потоками (запуск, пауза, зупинка), що не підтримується в HLS.

RTSP vs. WebRTC

1. Призначення: WebRTC в основному використовується для забезпечення двостороннього спілкування в реальному часі (відео- та аудіодзвінки), тоді як RTSP більш орієнтований на односторонню передачу потоків з камер спостереження.

2. Налаштування та впровадження: RTSP легше інтегрувати з існуючими камерами відеоспостереження, оскільки багато з них вже підтримують цей протокол.

2.5.4 Використання RTSP в Нашій Системі

1. Забезпечує реальний час: Протокол дозволяє передавати відеопотоки з низькою затримкою, що є критичним для системи розпізнавання об'єктів.

2. Універсальність та сумісність: RTSP підтримується багатьма камерами та пристроями, що дозволяє легко інтегрувати різні типи обладнання в систему.

3. Гнучкість налаштувань: Протокол надає широкі можливості для налаштування параметрів передачі відео, що дозволяє оптимізувати якість і продуктивність системи.

2.6 Метод розпізнавання YOLO

YOLO (You Only Look Once) [9] — це один з найпопулярніших алгоритмів для розпізнавання об'єктів у зображеннях та відео в реальному часі. Його основна відмінність від інших методів полягає в тому, що він виконує розпізнавання об'єктів за один пропуск через нейронну мережу, що робить його надзвичайно швидким і ефективним.

2.6.1 YOLO. Основні характеристики

1. Швидкість: YOLO є одним з найшвидших алгоритмів розпізнавання об'єктів, що дозволяє обробляти відеопотоки в реальному часі.

2. Точність: Незважаючи на високу швидкість, YOLO зберігає високу точність розпізнавання об'єктів.

3. Єдиний прохід: Алгоритм обробляє зображення за один прохід через нейронну мережу, що значно знижує обчислювальні витрати.

2.6.2 Причини Вибору YOLO

1. Реальний час: Завдяки високій швидкості, YOLO дозволяє виконувати розпізнавання об'єктів у відеопотоках в реальному часі, що є критично важливим для нашої системи.

2. Висока точність: Алгоритм забезпечує високу точність розпізнавання об'єктів, що важливо для надійності системи.

3. Універсальність: YOLO підтримує розпізнавання широкого спектра об'єктів, що дозволяє використовувати його для різних завдань.

4. Спільнота і підтримка: YOLO має велику спільноту розробників і добре задокументовані ресурси, що спрощує його використання і налаштування.

2.6.3 Порівняння YOLO з Іншими Алгоритмами

OLO vs. R-CNN

1. Швидкість: YOLO набагато швидший за R-CNN, оскільки обробляє зображення за один прохід, тоді як R-CNN потребує кількох кроків для обробки.

2. Простота використання: YOLO простіше в налаштуванні і використанні, особливо для додатків у реальному часі.

YOLO vs. SSD (Single Shot MultiBox Detector)

1. Точність: YOLO часто забезпечує вищу точність на великих об'єктах, тоді як SSD краще справляється з дрібними об'єктами.

2. Швидкість: Обидва алгоритми швидкі, але YOLO зазвичай має перевагу в швидкості завдяки єдиному проходу через мережу.

2.6.4 Використання YOLO в Нашій Системі

1. Швидке розпізнавання: Алгоритм дозволяє обробляти відеопотоки в реальному часі, що є критичним для ефективного моніторингу і швидкого реагування.

2. Висока точність: Завдяки високій точності, YOLO забезпечує надійне розпізнавання військової техніки, що важливо для безпеки та оперативного аналізу.

3. Універсальність застосування: Підтримка різних типів об'єктів дозволяє використовувати YOLO для широкого спектра завдань, пов'язаних з розпізнаванням об'єктів.

2.7. Онлайн-інструмент makesense.ai

Makesense.ai [10] - це інструмент, який використовується для виділення об'єктів на зображеннях та отримання анотацій у вигляді координат

2.7.1 Основні характеристики

1. Простий інтерфейс користувача: Makesense.ai пропонує інтуїтивно зрозумілий інтерфейс, що спрощує процес розмітки даних та отримання координат об'єктів на зображеннях.

2. Підтримка різних форматів анотацій: Інструмент дозволяє зручно працювати з різними форматами анотацій, включаючи координати об'єктів на зображеннях.

3. Гнучкість використання: Makesense.ai надає можливість працювати з різними типами зображень та анотацій, що дозволяє легко і швидко адаптувати інструмент до потреб проєкту.

2.7.2 Причини вибору

Вибір Makesense.ai для виділення об'єктів на зображеннях та отримання координат об'єктів був обумовлений його зручним інтерфейсом користувача та можливістю працювати з різними форматами анотацій. Порівняно з іншими інструментами для цієї задачі, Makesense.ai відзначається своєю простотою використання та гнучкістю в роботі з даними. У нашій системі розпізнавання військових об'єктів він відіграє ключову роль у процесі розмітки даних та отримання анотацій для підготовки тренувального набору даних.

2.7.3 Порівняння Makesense.ai з подібними інструментами:

LabelImg:

1. Обидва інструменти мають простий інтерфейс та можливість роботи з різними форматами анотацій.

2. Відмінність полягає у функціональності та можливостях розширення. Makesense.ai може мати більше додаткових функцій, таких як інтеграція зі штучним інтелектом для автоматичної розмітки.

Labelbox:

1. Обидва інструменти надають можливість робити анотації на зображеннях, але Labelbox може бути спрямований на ширший спектр завдань розмітки даних, включаючи тексти, аудіо та відео.

2. Вибір між ними може залежати від конкретних потреб проєкту: Labelbox може бути корисним для проєктів, де потрібна розмітка різноманітних типів даних, тоді як Makesense.ai може бути оптимальним вибором для проєктів, які фокусуються переважно на розмітці зображень

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Визначення мети

Зважаючи на ситуацію в Україні на даний момент, буде доцільно натренувати модель на військових об’єктах та використовувати її саме в такій сфері.

3.2. Збір датасету

Зважаючи на те, що знайти готовий датасет з сучасними військовими об’єктами неможливо із причин секретності таких даних, було прийнято рішення створити датасет власноруч [11]. Найкраще джерело у цьому випадку – гра War Thunder, у якій зображена сучасна військова техніка [12] , включно з тією, яка використовується наразі у російсько-українській війні (Leopard, T-72 тощо)

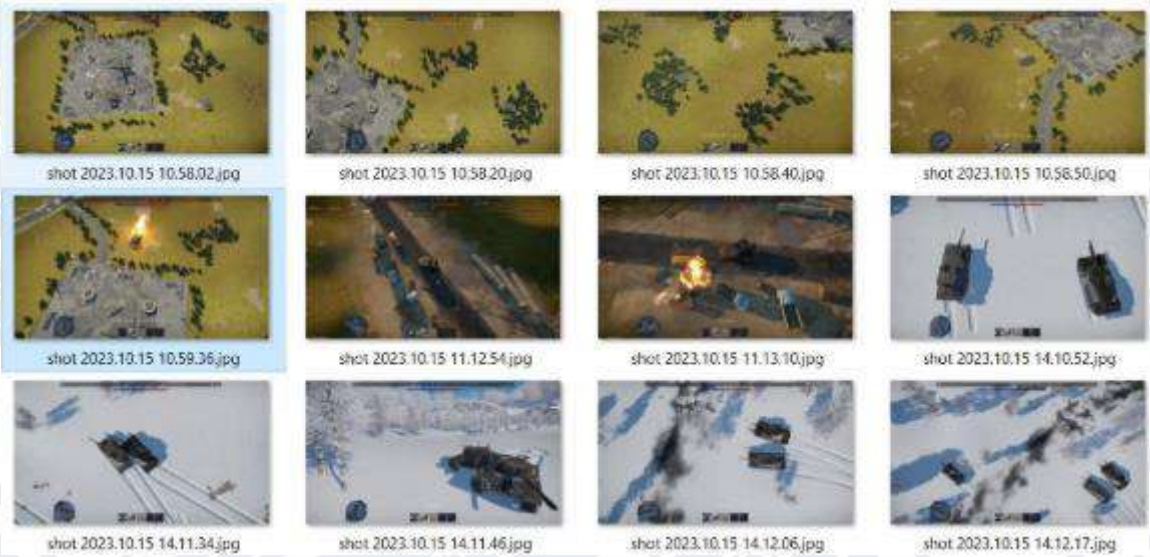

Рисунок 3.1 - Вибірка із створеного датасету

Також в подальшій роботі ми спробуємо працювати з цивільними об’єктами також, тому з того ж джерела збережемо зображення легкових чи вантажних автомобілів.

Рисунок 3.2 - Приклад цивільного транспорту

Уся показана техніка буде розподілятися в різні папки з різними класами, для їх подальшого використання.

3.3 Анотування даних

Нам відомо, що для подальшого тренування нашої моделі YOLO, потрібно зберегти анотації у форматі, сумісному з YOLO [13], а саме у форматі текстових файлів з координатами об’єктів. Для цього використаємо інструмент для позначення об’єктів на зображеннях

Я скористався онлайн інструментом makesense.ai, який дозволяє виділяти багато об’єктів на фото з великою кількістю класів.

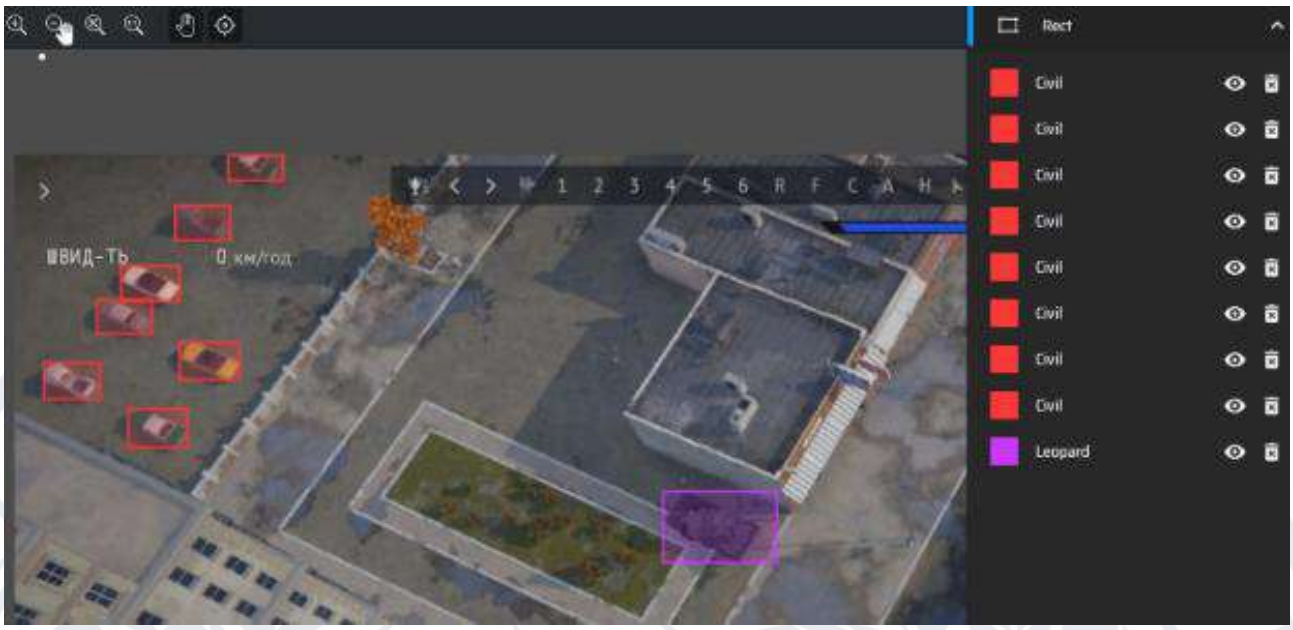

Рисунок 3.3 - Приклад роботи з makesense.ai

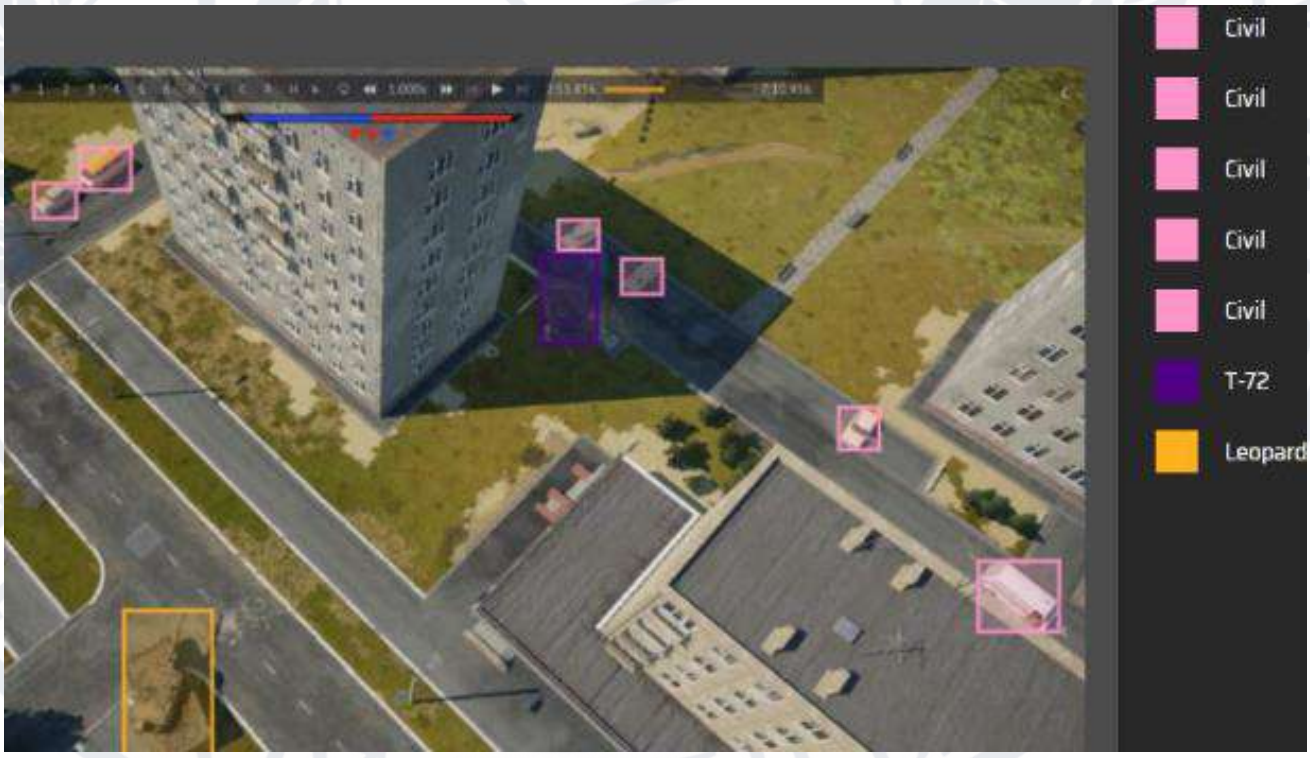

Рисунок 3.4 - Виділення об’єктів, які мають різні класи

В результаті роботи ми отримаємо архів із текстовими файлами, що містять клас та координати прямокутника, який позначає об’єкт [14] (рис. 3.4).

File Edit Format View Help

```
2 0.642803 0.643564 0.017136 0.054159
2 0.027989 0.196186 0.027291 0.032721
2 0.069878 0.237088 0.029195 0.034414
2 0.095583 0.178979 0.029195 0.033285
2 0.053694 0.140617 0.026657 0.029900
2 0.066387 0.111280 0.029195 0.031029
2 0.092092 0.059096 0.027291 0.029336
2 0.117796 0.010931 0.026657 0.021861
1 0.345011 0.323235 0.055852 0.062057
```

Рисунок 3.5 – результат роботи, показаної на рис. 3.3

Створений датасет (зображення та анотації до них) розбиваємо на дві групи – навчальний та валідаційний – для тренування моделі розпізнавання об’єктів та перевірки моделі на точність відповідно [15].

Рисунок 3.6 - Структура створеного датасету

3.4. Створення .yaml - файлу

Коли датасет уже готовий, потрібно створити файл .yaml, в якому буде інформація, де знайти тренувальний та валідаційний датасети, а також список класів [16].

```
train: ./train/images
val: ./val/images

nc: 3
names: ['tank', 'artillery', 'vehicle']
```

Рисунок 3.7 – Приклад файлу .yaml

3.5. Запуск тренування

Щоб почати тренувати нашу модель YOLOv5, імпортуємо файл конфігурації (рис. 3.7), задаємо гіперпараметри і запускаємо навчання [17]:

```
hyp = {
    'lr0': 0.01,
    'lrf': 0.2,
    'momentum': 0.937,
    'weight_decay': 0.0005,
    'warmup_epochs': 3.0,
    'warmup_momentum': 0.8,
    'warmup_bias_lr': 0.1,
    'box': 0.05,
    'cls': 0.5,
    'cls_pw': 1.0,
```

Рисунок 3.8 - Деякі гіперпараметри для тренування моделі

```
train.run(
    data='data.yaml',
    epochs=100,
    batch_size=16,
    imgsz=640,
    hyp=hyp,
    weights='yolov5s.pt',
    project='runs/train',
    name='exp',
    device=0
)
```

Рисунок 3.9 - Початок тренування нейромережі

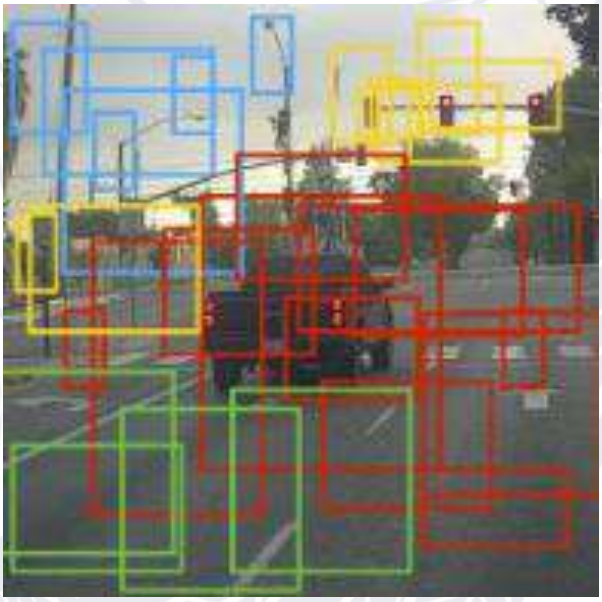

Рисунок 3.10 - Приклад пошуку полів, у яких можуть бути присутній об’єкт

Рисунок 3.11 - Зображення розбивається на сітку з передбаченням об’єктів

3.6. Аналіз результатів тренування

Після завершення тренування потрібно перевірити точність прогнозування об’єктів за допомогою валідаційного набору, який ми підготували раніше.

Після навчання та перевірки у папці з результатами збережуться зображення - графіки.

На графіках будуть представлені такі показники: box_loss, cls_loss, dfl_loss, precision, recall, mAP50, mAP50-95. Розглянемо кожен з них детальніше [18] :

1. box_loss: це втрати, які оцінюють, наскільки точно передбачені обмежувальні прямокутники співпадають з реальними об'єктами на зображенні.

2. cls_loss: це втрати, що визначають правильність класифікації кожної обмежувальної рамки. Кожна рамка може містити клас об'єкта або фон.

3. dfl_loss: це втрати, що враховують проблему дисбалансу класів під час навчання.

4. precision (точність): це метрика, що вимірює, як часто модель правильно прогнозує позитивний клас.

5. recall (відкликання): це показник, який вимірює, як часто модель правильно визначає справжні позитивні екземпляри серед усіх фактичних позитивних зразків у наборі даних.

6. mAP50: це метрика, що оцінює продуктивність моделі, враховуючи як точність, так і відкликання для кількох класів об'єктів при пороговому значенні IoU 0,5. Вона вимірює, наскільки добре модель ідентифікує об'єкти з прийнятним рівнем перекриття.

7. mAP50-95: це метрика, що розширює оцінку до діапазону порогових значень IoU від 0,5 до 0,95. Цей показник особливо важливий для завдань, що вимагають точної локалізації та детекції об'єктів.

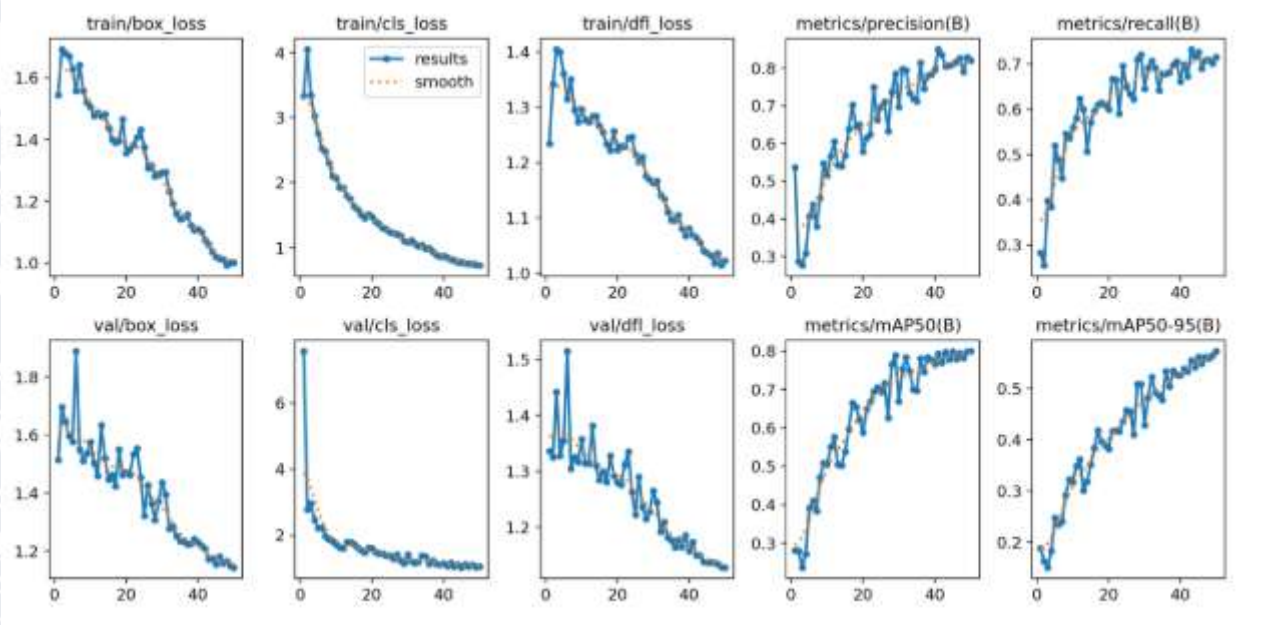

Рисунок 3.12 – Метрики YOLOv5

По результатам метрик видно, що модель завжди розрізняє цивільні та військові об’єкти, це дуже важливо.

Загалом отримані значення виявилися досить високими. Показники [19] , що перевищують 0,8, та mAP50-95, які іноді перевищують 0,50 або навіть 0,60, свідчать про те, що модель досить точно визначає розташування об'єктів на зображенні.

3.7. Результат роботи моделі на згенерованих та реальних зображеннях

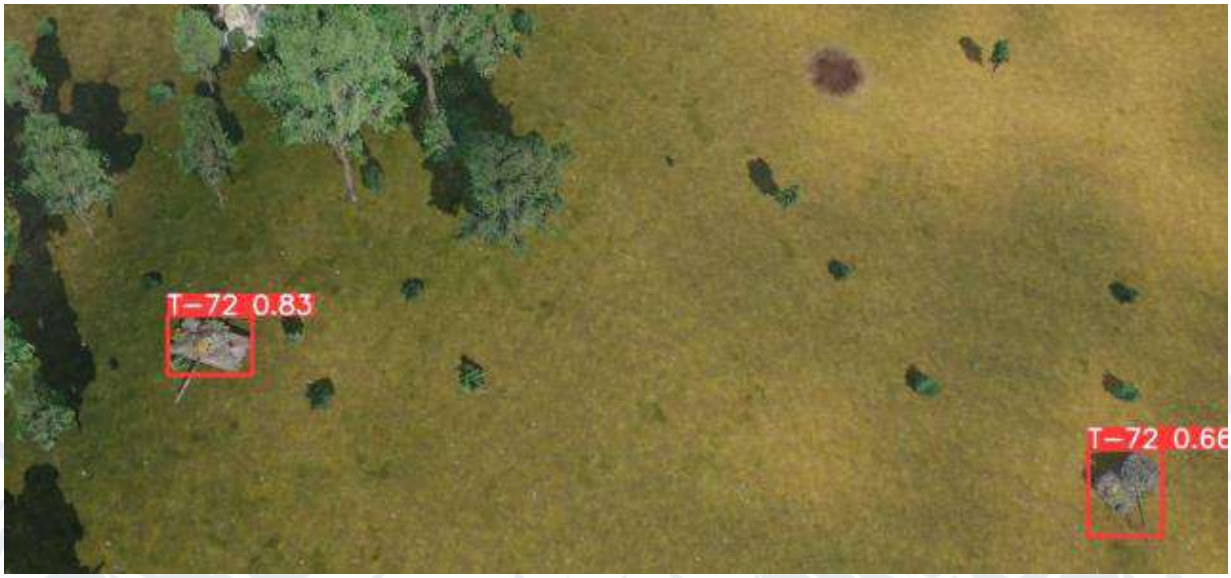

Рисунок 3.13 – Візуальний приклад роботи у степах

Рисунок 3.14 – Розпізнавання на реальному зображенні

Рисунок 3.15 – Приклад розпізнавання цивільних об’єктів

3.8. Використання натренованої моделі в реальних умовах.

Існує декілька варіантів для демонстрації роботи моделі. Серед них – підключення стріму з військового дрону, камер спостереження чи відео та зображення не в реальному часі. Зупинимось на камерах постійного стеження на критично важливих військових об’єктах.

Для апробації натренованої моделі YOLOv5 [20] нам потрібно визначити джерела відео (RTSP-потік), виконати захоплення фрейму з відеопотоку. Для відображення результатів використаємо бібліотеку комп’ютерного зору OpenCV, яку було описано раніше.

Так як можливості підключитися до реальної камери не існує, тому у програмі будемо використовувати шаблон для підключення відеопотоку [21] для наочності.

```python
import torch
import cv2

# Завантаження натренованої моделі YOLOv5
model = torch.hub.load('ultralytics/yolov5', 'custom', path='C:/Us

# Визначення джерела відео (RTSP потік)
video_source = 'rtsp://username:password@ip_address:port/stream'

# Відкриття відеопотоку
cap = cv2.VideoCapture(video_source)

# Перевірка чи вдалося відкрити відеопотік
if not cap.isOpened():
    print("Error: Не вдалося відкрити відеопотік.")
    exit()

while True:
    # Захоплення фрейму з відеопотоку
    ret, frame = cap.read()
    if not ret:
        break

    # Використання моделі для передбачення об'єктів на фреймі
    results = model(frame)

    # Отримання передбачених зображень з мітками
    annotated_frame = results.render()[0]

    # Відображення передбачених фреймів у вікні
    cv2.imshow('YOLOv5 Object Detection', annotated_frame)

    # Натисніть 'q' для виходу з циклу
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Звільнення відеокаптури та закриття всіх вікон
cap.release()
cv2.destroyAllWindows()
```

Рисунок 3.16 – Скрипт для використання моделі

ВИСНОВКИ

Отже, в ході розробки та дослідження методів розпізнавання об’єктів було проведено детальний аналіз способів розв’язання даної задачі. В результаті можемо виділити наступні висновки:

Ефективність Нейромереж у Розпізнаванні Військових Об'єктів:

На основі проведеного аналізу можемо зробити висновок про високу ефективність використання нейромереж у розпізнаванні військових об'єктів. Результати дослідження підтвердили, що моделі на основі нейромереж демонструють високу точність та надійність у виявленні та класифікації різних типів військової техніки. Їхня адаптивність та здатність до узагальнення дозволяють успішно впоратися з різноманітними умовами та стилями представлення об'єктів.

Стійкість до Змін та Різноманітності:

На основі експериментальних досліджень можна зазначити високу стійкість нейромережевих моделей до змін у військових об'єктах та їх різноманітних інтерпретаціях. Це свідчить про гнучкість та адаптивність системи до різних ситуацій, що важливо для успішного застосування в реальних умовах.

Можливості Подальшого Вдосконалення:

Незважаючи на досягнуті успіхи, є потенціал для подальшого вдосконалення системи розпізнавання військових об'єктів. Оптимізація архітектури нейромереж, розширення обсягу тренувальних даних та покращення алгоритмів навчання можуть позитивно вплинути на точність та швидкість розпізнавання. Такі заходи сприятимуть подальшій еволюції системи та її успішному впровадженню в практичні сфери.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Anderson J., Smith P., Brown L. Defense Advanced Research Projects Agency: official page // 2021 URL: https://www.darpa.mil/

2. Miller D., Taylor K., Green R. Elbit Systems: official page // 2021 URL: https://elbitsystems.com/

3. Johnson M., Williams T., Garcia H. Google Vision API – official information page // 2021. URL: https://cloud.google.com/vision

4. Thompson E., Harris S., Baker J. Python directory documentation // 2021. URL: https://docs.python.org/3/library/

5. Lee C., Walker A., Robinson J. PyTorch documentation // 2021. URL: https://pytorch.org/docs/stable/index.html

6. Evans R., Martin D., Clark P. OpenCV modules documentation // 2021. URL: https://docs.opencv.org/master/

7. W3School Team. Основи PyCharm від W3School // 2021. URL: https://www.w3schools.com/pycharm/

8. Schulzrinne H., Rao A., Lanphier R. RTSP Protocol: basic guide // 1998. URL: https://tools.ietf.org/html/rfc2326

9. Ng A., Green J., Patel S. Convolutional Neural Network: Machine Learning documentary // 2021. URL: https://www.coursera.org/lecture/machine-learning/convolutional-neural-networks-IXJ06

10. Makesense.ai Team. Makesense.ai documentary: New Technology AI // 2021. URL: https://makesense.ai/documentation/

11. Redmon J., Farhadi A. Dataset for Object Detection using the YOLO // 2016. URL: https://pjreddie.com/darknet/yolo/

12. Gaijin Entertainment Team. Інтерфейс War Thunder: Опис та характеристики техніки, яка використовується в грі // 2021. URL: https://warthunder.com/en/game/

13. Bochkovskiy A., Wang C., Liao H. YOLO annotation, using in object detection // 2020. URL: https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects

14. Jocher G., Ultralytics Team. YOLOv5 TXT Annotation Format Instruction // 2020. URL: https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

15. Gad A., Brown M., Patel R. Train and Validation datasets on YOLO training // 2019. URL: https://towardsdatascience.com/yolo-object-detection-with-opencv-and-python-21e50ac599e9

16. Jocher G., Ultralytics Team. YAML-file with information about custom dataset // 2020. URL: https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

17. Jocher G., Ultralytics Team. YOLOv5 Documentation - Custom Training // 2020. URL: https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

18. Johnson A., Lee M., Davis K. Metrics in analytics YOLO model training // 2020. URL: https://medium.com/@meetshah1995/yolov3-analysis-of-the-loss-function-524474b3bdf2

19. Jocher G., Ultralytics Team. Results on mAp-50 and other metrics // 2020. URL: https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data#metrics

20. Jocher G., Ultralytics Team. Using YOLOv5 model in real life // 2020. URL: https://github.com/ultralytics/yolov5

21. Brown A., Green R., Taylor K. Object detection on multiple streams // 2021. URL: https://arxiv.org/abs/2104.10038

ДОДАТОК А

Лістинг ключових частин коду

```
#імпорт бібліотек
import torch
import cv2
from pathlib import Path

# Гіперпараметри
hyp = {
   'lr0': 0.01,
   'lrf': 0.2,
   'momentum': 0.937,
   'weight_decay': 0.0005,
   'warmup_epochs': 3.0,
   'warmup_momentum': 0.8,
   'warmup_bias_lr': 0.1,
   'box': 0.05,
   'cls': 0.5,
   'cls_pw': 1.0,
}

# Конфігурація даних
data_yaml_content = """
train: C:/datasets/war_thunder/train/images
val: C:/datasets/war_thunder/val/images
nc: 3
names: ['tank', 'artillery', 'vehicle']
"""
data_yaml_path = Path('data.yaml')
```

```
data_yaml_path.write_text(data_yaml_content)

# Функція для тренування моделі
def train_model():
    import train
    train.run(
        data=str(data_yaml_path),
        epochs=100,
        batch_size=16,
        imgsz=640,
        hyp=hyp,
        weights='C:/models/yolov5s.pt',
        project='runs/train',
        name='exp',
        device=0
    )

# Тренування моделі
train_model()

# Завантаження натренованої моделі YOLOv5
model = torch.hub.load('ultralytics/yolov5', 'custom',
path='runs/train/exp/weights/best.pt')

# Визначення джерела відео (RTSP потік)
video_source = 'rtsp://username:password@192.168.1.100:554/stream'
```

```
# Відкриття відеопотоку
cap = cv2.VideoCapture(video_source)
if not cap.isOpened():
    print("Error: Не вдалось відкрити відеопотік")
    exit()
while True:
    ret, frame = cap.read()
    if not ret:
        break
    results = model(frame)
    annotated_frame = results.render()[0]
    cv2.imshow('YOLOv5 Object Detection', annotated_frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я,_____

_____

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

що нижче 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)