МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Оксана
_____ ШОВКОПЛЯС
(підпис)
грудня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Веборієнтована інформаційна технологія аналізу даних аеророзвідки»
здобувача групи ІН.м – 34 Харченко Руслан В'ячеславович

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Руслан Харченко
(підпис)

Керівник доцент, кандидат технічних
наук, доцент В`ячеслав Москаленко _____
(підпис)

Суми – 2024

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
Оксана
_____ ШОВКОПЛЯС
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-34 Харченко Руслан В'ячеславович

1. Тема роботи: «Веборієнтована інформаційна технологія аналізу даних аеророзвідки» затверджую наказом по СумДУ від *« »* _____
2. Термін здачі здобувачем кваліфікаційної роботи *до 17 грудня 2024 року* _____
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Інформаційна технологія розшифровки даних аеророзвідки. 3) Програмна реалізація веб-орієнтованої системи аналізу даних аеровідеоспостереження. 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «19» жовтня 2024 р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз моделей штучного бачення.*	02.09.24 – 12.09.24	
2	*Ознайомлення, навчання та тестування моделі YOLOv8.*	13.09.24 – 19.09.24	
3	*Ознайомлення з Test-Time Augmentation.*	20.09.24 – 29.09.24	
4	*Ознайомлення з ансамблюванням та NMS.*	*29.09.24 – 03.10.24*	
5	*Налаштування середовища веб-додатку.*	*29.09.24 –*	

		03.10.24	
6	*Написання методів знаходження об’єктів на відео із застосуванням ТТА та ансамблюванням.*	*16.10.24 – 05.11.24*	

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 47 стр., 27 рис., 3 додаток, 6 використаних джерел.

Обґрунтування актуальності теми роботи – необхідність автоматизації аналізу аеророзвідувальних відеоданих, що містять критичну інформацію для прийняття рішень у військовій сфері. Використання моделі YOLOv8 підвищує швидкість і точність виявлення військової техніки на відео, що є важливим для національної безпеки та оборони.

Об’єкт дослідження — технології автоматизованого аналізу аеророзвідувальних відеоданих, зокрема методи та моделі комп’ютерного зору для виявлення військової техніки на відео.

Мета роботи — Розроблено веб-технологію для автоматизованого аналізу аеророзвідувальних відео, що забезпечує ефективне виявлення військової техніки з високою точністю, підвищуючи оперативність обробки та підтримуючи прийняття рішень у розвідці.

Методи дослідження — для досягнення мети застосовано методи машинного та глибокого навчання, зокрема YOLOv8 для детекції об'єктів на відео, обробку зображень, ансамблювання моделей, консенсусне голосування та NMS для підвищення точності й надійності.

Результати — Створено веб-додаток для автоматичної детекції військової техніки на аеророзвідувальних відео. Обробка здійснюється за допомогою ансамблю моделей YOLOv8, що підвищує точність і надійність. Користувач може завантажити відео, запустити аналіз і переглянути результати з позначеними об'єктами.

ІНФОРМАЦІЙНА СИСТЕМА, ОБРОБКА ДАНИХ, РОЗПІЗНАВАННЯ ВІЙСЬКОВОЇ ТЕХНІКИ, PYTHON, OPENCV, YOLOV8, DJANGO.

ЗМІСТ

ВСТУП 6

1. АНАЛІЗ ПРОБЛЕМ ТА ПОСТАНОВКА ЗАДАЧІ 7

1.1. СУЧАСНИЙ СТАН ТА ТЕНДЕНЦІЇ РОЗВИТКУ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ АНАЛІЗУ ДАНИХ АЕРОРОЗВІДКИ 7

1.2 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ І МЕТОДІВ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА МІСЦЕВОСТІ 7

1.3 ФОРМАЛІЗОВАНА ПОСТАНОВКА ЗАДАЧІ 8

2. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗШИФРОВКИ ДАНИХ АЕРОРОЗВІДКИ 9

2.1 МОДЕЛЬ ДЕТЕКТОРА ОБ'ЄКТІВ ІНТЕРЕСУ НА ЗОБРАЖЕННЯХ АЕРОРОЗВІДКИ 9

2.2 МЕТОД МАШИННОГО НАВЧАННЯ ДЕТЕКТОРА 10

2.2.1 НАВЧАННЯ МОДЕЛІ YOLOV8 ДЛЯ ДЕТЕКШЕНУ ОБ'ЄКТІВ 10

2.3 КРИТЕРІЇ ВАЛІДАЦІЇ ЕФЕКТИВНОСТІ МОДЕЛІ ДЕТЕКТОРА ... 21

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ АНАЛІЗУ ДАНИХ АЕРОВІДЕОСПОСТЕРЕЖЕННЯ 27

3.1 ФОРМУВАННЯ НАВЧАЛЬНИХ ТА ТЕСТОВИХ ДАНИХ 27

3.2 АРХІТЕКТУРА І КОРОТКИЙ ОПИС ВЕБ-ОРІЄНТОВАНОГО ДОДАТКУ/СЕРВІСУ 29

ВИСНОВОК 35

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 36

ДОДАТОК А: view.py 37

ДОДАТОК В: services.py 38

ДОДАТОК С: index.html 45

ВСТУП

В епоху стрімкого розвитку інформаційних технологій обробка великих обсягів даних стає ключовим завданням для різних галузей. Аеророзвідка, завдяки використанню безпілотників та інших систем спостереження, дозволяє отримувати важливі дані про об’єкти на поверхні землі. Ці дані потребують швидкої та точної обробки для ефективного прийняття рішень, особливо у військовій сфері.

Сучасні технології машинного навчання дозволяють автоматизувати процес аналізу та інтерпретації даних, зокрема, шляхом використання нейронних мереж для розпізнавання об’єктів на зображеннях та відео. У цій роботі ми розробляємо веб-орієнтовану інформаційну технологію для аналізу даних аеророзвідки з використанням моделей YOLOv8. Вона забезпечує можливість виявлення об'єктів військової техніки на аеровідео та відображення результатів через інтерактивний інтерфейс. Реалізація цієї системи сприятиме зниженню часу на обробку даних та підвищенню точності розпізнавання об'єктів інтересу, що є вкрай важливим в умовах швидкоплинних військових дій.

1. АНАЛІЗ ПРОБЛЕМ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. СУЧАСНИЙ СТАН ТА ТЕНДЕНЦІЇ РОЗВИТКУ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ АНАЛІЗУ ДАНИХ АЕРОРОЗВІДКИ

Інформаційні технології в обробці аеророзвідувальних даних розвиваються в напрямі автоматизації та інтеграції з алгоритмами машинного навчання. Завдяки вдосконаленню алгоритмів нейронних мереж стає можливим виявлення об’єктів на зображеннях, які мають складні риси або значно відрізняються від загального фону.

Основна тенденція у цій сфері – інтеграція алгоритмів розпізнавання у реальному часі, що дозволяє швидко аналізувати інформацію та одразу відображати результати. Дослідження також свідчать про підвищений інтерес до впровадження глибоких згорткових нейронних мереж, таких як YOLO (You Only Look Once), які дозволяють ефективно обробляти великі обсяги візуальних даних, зокрема аеровідео. Завдяки новим моделям YOLOv8, які були розроблені для більш точної та швидкої обробки зображень, з’являються можливості для створення систем, що можуть працювати в умовах обмежених ресурсів або в реальному часі.

1.2 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ І МЕТОДІВ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ НА МІСЦЕВОСТІ

Сучасні методи розпізнавання об'єктів на зображеннях можна розділити на дві основні групи: традиційні алгоритми комп’ютерного зору (таких як HOG, SIFT) і методи глибокого навчання (зокрема, згорткові нейронні мережі). Традиційні методи використовують фільтри для виділення ознак об'єкта, але мають обмежену здатність до адаптації і часто не здатні точно розпізнати об'єкти на складних зображеннях або відео. Глибоке навчання, навпаки, дозволяє автоматично навчати нейронну мережу на великій кількості зображень, завдяки чому модель краще адаптується до специфіки даних та здатна розпізнавати об'єкти у більш складних умовах.

Найпопулярнішими моделями у сфері розпізнавання об’єктів є Faster R-CNN, SSD та YOLO. Faster R-CNN забезпечує високу точність, але має низьку швидкість обробки. SSD є швидшим, але інколи поступається за точністю. YOLO, особливо її нові версії, поєднує високу точність з високою швидкістю, що робить її ідеальним варіантом для обробки аерознімків у реальному часі.

1.3 ФОРМАЛІЗОВАНА ПОСТАНОВКА ЗАДАЧІ

Задача цієї роботи полягає у розробці інформаційної технології для автоматизованого аналізу відеоданих аеророзвідки з метою ідентифікації військової техніки. На основі аналізу існуючих технологій, було прийнято рішення використовувати модель YOLOv8, яка має хорошу здатність до розпізнавання об'єктів у реальному часі і підходить для застосування у веб-орієнтованих системах.

Основні задачі роботи включають:

- Аналіз існуючих технологій та вибір оптимальної архітектури детектора об’єктів для розпізнавання військової техніки.
- Програмна реалізація системи розшифровки даних аеророзвідки з інтеграцією моделі YOLOv8.
- Проведення тестування розробленої системи та аналіз результатів експериментів, щоб оцінити ефективність моделі за критеріями точності та швидкості обробки даних.

2. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗШИФРОВКИ ДАНИХ АЕРОРОЗВІДКИ

2.1 МОДЕЛЬ ДЕТЕКТОРА ОБ'ЄКТІВ ІНТЕРЕСУ НА ЗОБРАЖЕННЯХ АЕРОРОЗВІДКИ

У нашій роботі використовується модель YOLOv8, а саме варіанти YOLOv8s та YOLOv8m. YOLOv8 (You Only Look Once, Version 8) є одним із найсучасніших методів для розпізнавання об'єктів, відомим своєю високою точністю та швидкістю. YOLOv8 обробляє зображення за допомогою згорткових нейронних мереж (CNN), дозволяючи їм «дивитись» на зображення лише один раз, щоб ідентифікувати всі об'єкти на ньому. Це значно прискорює процес розпізнавання в порівнянні з іншими моделями, такими як Faster R-CNN або SSD.

Модель YOLOv8s є меншою за розміром і розрахована на використання у реальному часі. Вона швидко обробляє зображення, роблячи її ефективною для завдань, що вимагають швидкості. YOLOv8m, навпаки, є більшою моделлю, що забезпечує вищу точність на шкоду швидкості. Ця модель (рис. 1) підходить для ситуацій, де важлива не лише швидкість, але і підвищена точність.

В рамках даного проєкту ці моделі використовуються для аналізу аеровідеозйомок з метою виявлення об'єктів військової техніки. Такий підхід дозволяє зберегти баланс між швидкістю обробки даних і точністю виявлення. Використання двох різних варіантів моделі також дає змогу застосувати їх в ансамблюванні (ensambling), що підвищує надійність результатів.

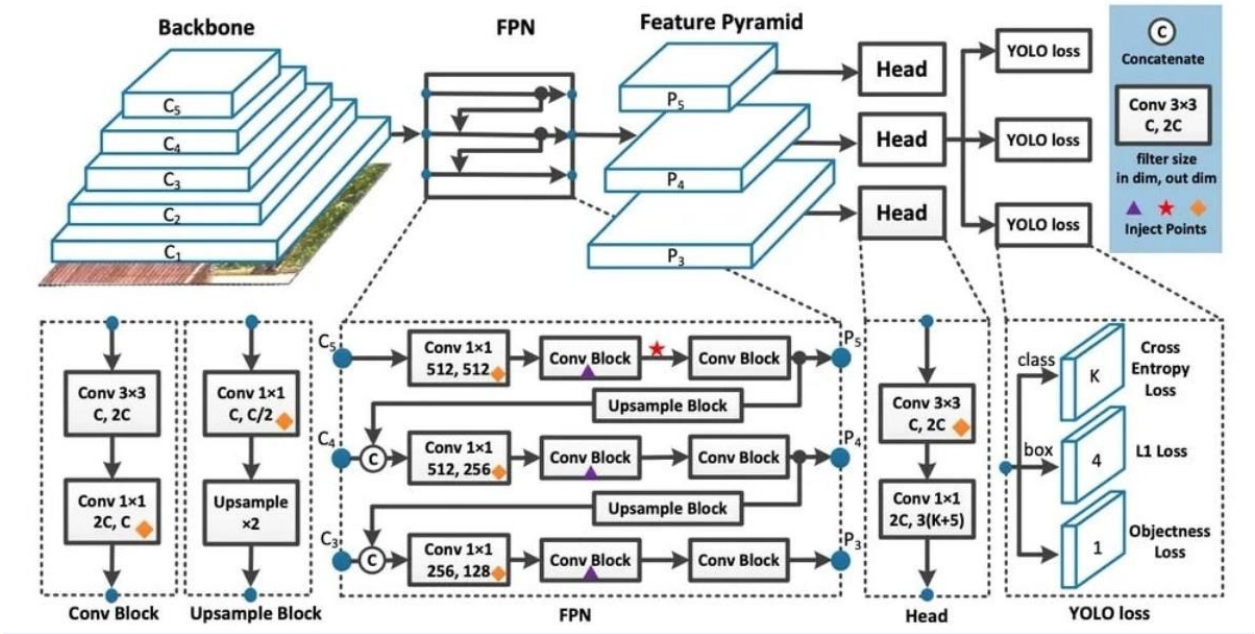

Рисунок 1 - Архітектура YOLOv8

2.2 МЕТОД МАШИННОГО НАВЧАННЯ ДЕТЕКТОРА

Моделі YOLOv8 навчені виявляти об'єкти на зображеннях. Вони використовують свій набір шарів і блоків CNN для обробки вхідного зображення, що дозволяє моделі "дивитися" лише один раз на зображення і передбачити всі об'єкти одночасно.

2.2.1 НАВЧАННЯ МОДЕЛІ YOLOV8 ДЛЯ ДЕТЕКШЕНУ ОБ’ЄКТІВ

Команда !nvidia-smi виконується в середовищі Google Colab (рис. 2) для перевірки доступності та стану графічного процесора (GPU) від NVIDIA. Вона показує інформацію про підключений графічний процесор, включаючи:

- Модель GPU (наприклад, Tesla T4, P100),
- Поточне використання пам’яті GPU,
- Завантажені драйвери та версію CUDA.

Це важливо для навчання YOLOv8, оскільки використання GPU суттєво пришвидшує обробку великих обсягів даних та навчання моделі глибокого навчання, яке потребує значних обчислювальних ресурсів.

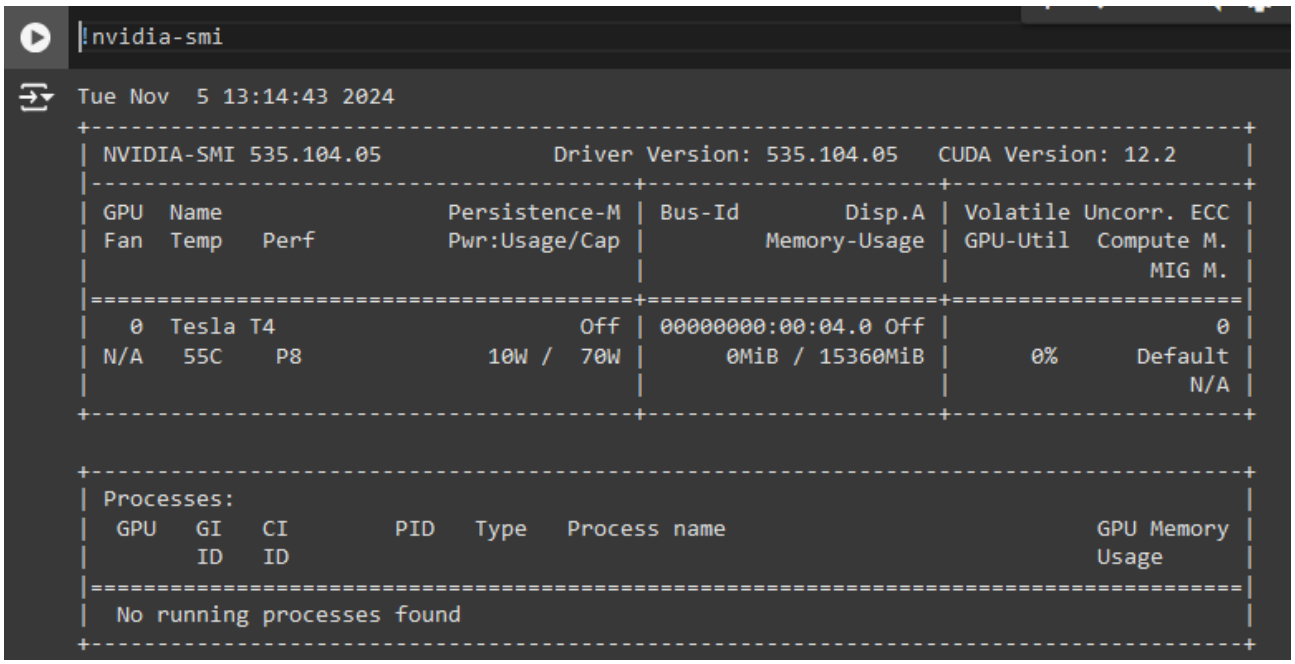

Рисунок 2 - Результат виконання команди !nvidia-smi

Команда import os (рис. 3) імпортує стандартний модуль Python для роботи з операційною системою, а os.getcwd() повертає поточну робочу директорію, в якій виконується Python код.

У даному випадку (рис. 3), команда print(HOME) виведе на екран шлях до поточної директорії (робочої папки), де знаходиться ваш скрипт чи ноутбук. Це корисно для перевірки, чи знаходитеся ви в правильній директорії, де зберігаються файли для навчання або моделі YOLOv8.

Наприклад, якщо ви завантажуєте модель або зберігаєте результати, важливо знати, в якій директорії працюєте, щоб вірно вказувати шляхи до файлів.

Рисунок 3 - Результат виконання команди для встановлення головної директорії

Команда !pip install ultralytics==8.0.196 використовується для встановлення конкретної версії бібліотеки ultralytics (рис. 4), яка містить реалізацію моделі YOLOv8, з PyPI (Python Package Index). Це гарантує, що ви працюєте з потрібною версією бібліотеки, яка підтримує ваші поточні функціональні вимоги.

- Після установки бібліотеки команда from IPython import display імпортує необхідний модуль для очищення виводу в Jupyter/Colab.
- display.clear_output() очищує поточний вивід в середовищі Colab після виконання команди, щоб зробити екран більш охайним.

Команда import ultralytics імпортує саму бібліотеку ultralytics, після чого викликається метод ultralytics.checks() для перевірки конфігурації середовища, сумісності бібліотеки та правильності установки. Це дозволяє переконатися, що всі необхідні компоненти встановлені правильно, і система готова до навчання або виконання інференсу за допомогою YOLOv8.

Ці команди важливі для налаштування середовища та перевірки готовності системи до роботи з YOLOv8.

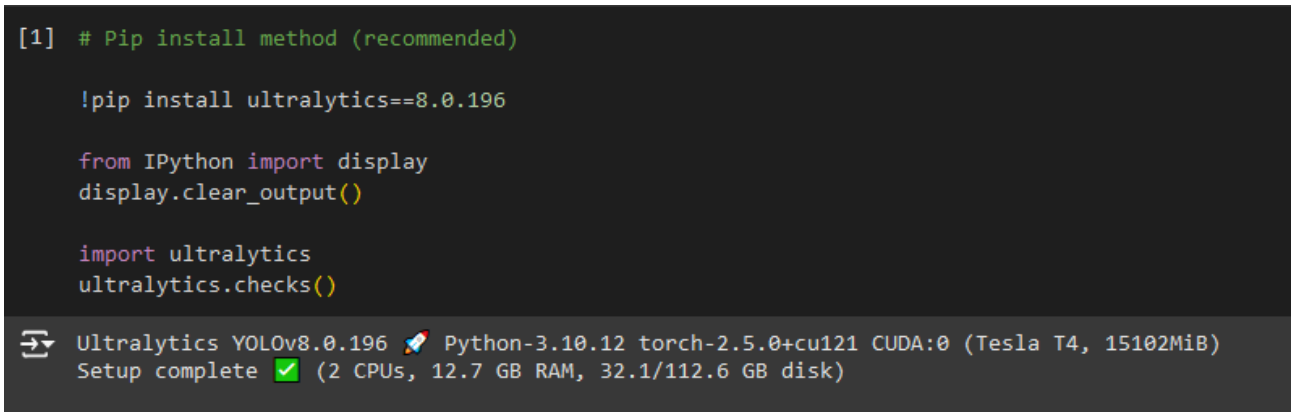

Рисунок 4 - Налаштування середовища пайтон

Команди:

1. from ultralytics import YOLO — імпортує клас YOLO з бібліотеки ultralytics, що дозволяє вам використовувати модель YOLOv8 для завантаження, навчання, інференсу (розпізнавання об'єктів) на зображеннях або відео. Цей

клас надає інтерфейс для взаємодії з моделлю YOLOv8, що включає можливість завантаження моделей, виконання передбачень, налаштування параметрів навчання та виконання інших операцій.

2. from IPython.display import display, Image — імпортує функції для відображення зображень в Jupyter або Google Colab.

   - display() дозволяє показати зображення або інші об'єкти безпосередньо в межах вашої поточної комірки.
   - Image() використовується для завантаження та відображення зображення з файлу або URL.

Ці команди (рис. 5) зручні, якщо ви хочете інтегрувати моделі YOLOv8 в середовище Jupyter/Colab для демонстрації результатів або відображення вихідних зображень безпосередньо в ноутбуці. Вони дозволяють вам не тільки працювати з моделлю, а й виводити зображення для візуального огляду результатів розпізнавання.

```
[ ]  from ultralytics import YOLO

     from IPython.display import display, Image
```

Рисунок 5 - Імпорт важливих бібліотек

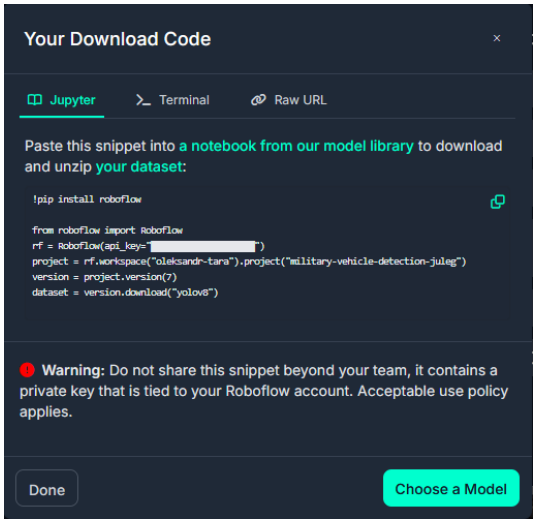

Рисунок 6 - Генерація коду для встановлення сету даних з сайту roboflow

Цей код (рис. 7) організовує середовище для завантаження набору даних, зібраного на платформі Roboflow, і підготовленого для використання з моделями YOLOv8. Процес полягає в наступному:

- Створення папки для зберігання даних.
- Завантаження набору даних безпосередньо з Roboflow в робоче середовище.
- Використання API Roboflow (рис. 6) для завантаження, управління та інтеграції даних у вашому проєкті.

Цей підхід дозволяє безпечно, ефективно та зручно завантажувати дані для вашої моделі комп'ютерного зору, забезпечуючи правильне форматування і підготовку даних для YOLOv8.

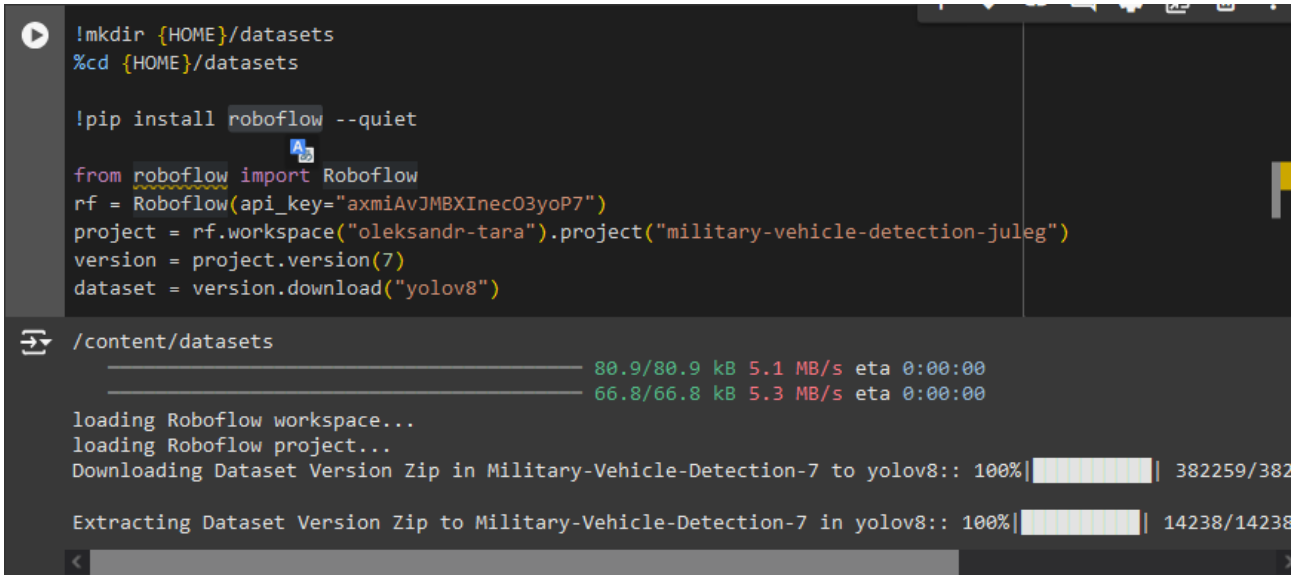

Рисунок 7 - Інсталяція сету даних

Цей код (рис. 8) ініціює процес тренування моделі YOLOv8 для виявлення військової техніки на зображеннях з використанням набору даних, зібраного та підготовленого через платформу Roboflow. Паралельно з тренуванням буде застосовуватися аугментація зображень, щоб покращити здатність моделі адаптуватися до різних варіантів зображень.

- Аугментація зображень дозволяє моделі краще загально виглядати на різних варіантах зображень, тому ваші зображення будуть обертатися, масштабуватися, або змінювати кольори та контраст для кращої універсальності.
- Тренування моделі на основі обраної архітектури yolov8m.pt дозволяє досягти хорошого балансу між швидкістю обробки і точністю. Це середня версія YOLOv8, яка є потужною для завдань з високою точністю, зокрема для розпізнавання військової техніки на аерофото або відео.

Процес тренування розпочнеться після виконання цієї команди, і модель буде покращувати свою здатність виявляти військові об'єкти протягом 20 епох.

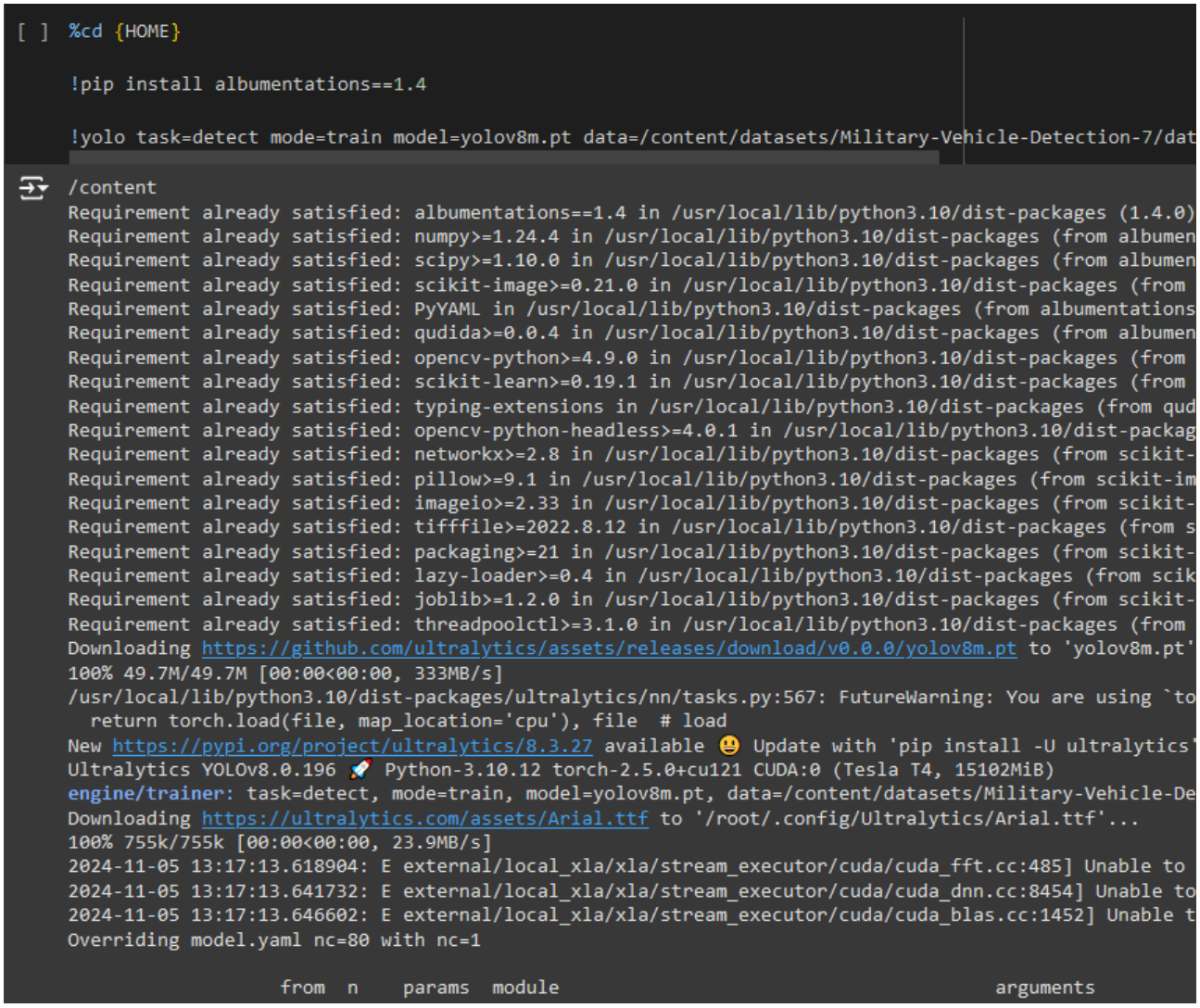

Рисунок 8 - Процес тренування моделі YOLOv8m

Команда !ls {HOME}/runs/detect/train/ виводить список файлів і каталогів у директорії (рис. 9), де зберігаються результати тренування моделі YOLOv8. Це дозволяє вам побачити виведені дані після тренування, включаючи файли, такі як збережені ваги моделі, графіки втрат та точності, а також інші важливі артефакти.

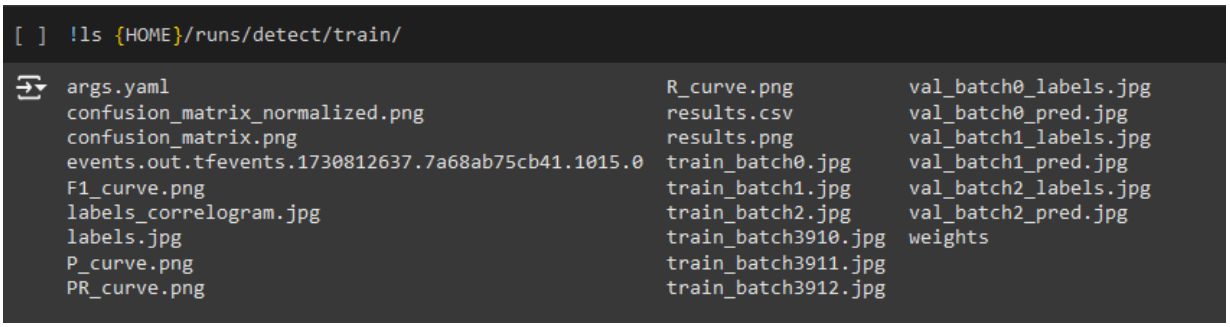

Рисунок 9 - Зміст директорії train

Команда Image(filename=f'{HOME}/runs/detect/train/results.png', width=600) дозволяє відобразити зображення results.png з результатами тренування моделі YOLOv8 (рис. 10). Це зображення зазвичай містить графік, що показує, як змінюються значення метрик, таких як втрата (loss) та точність (accuracy), під час навчання.

Зазвичай на графіку ви побачите кілька ключових компонентів:

1. Втрата (Loss): Це показник того, наскільки добре модель навчається на ваших даних. Якщо значення втрат зменшується протягом епохи, це вказує на те, що модель поліпшується.
2. Точність (Accuracy): Якщо є графіки, що показують точність моделі під час тренування, це допомагає оцінити, наскільки добре модель справляється з класифікацією.

Це зображення надає наочну картину того, як модель навчалася, і чи є необхідність в коригуванні параметрів для подальшого покращення результатів.

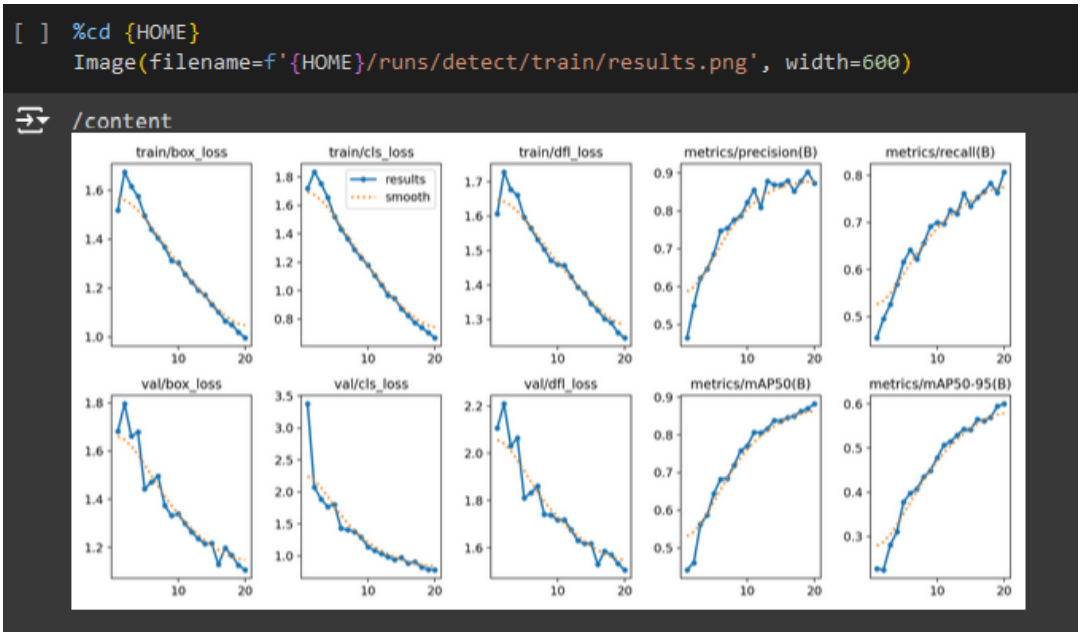

Рисунок 10 - Метрики тренування моделі

Команда !yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt data={dataset.location}/data.yaml (рис. 11) виконує валідацію моделі після її навчання.

Мета команди:

- Валідація: Після тренування важливо перевірити, як добре модель працює на невідомих їй даних. Валідація дозволяє виміряти ефективність моделі за допомогою таких метрик, як мAP (mean Average Precision), IoU (Intersection over Union), precision, recall тощо. Цей процес дає чітке уявлення про те, як модель узагальнює знання на нових даних.

В результаті виконання цієї (рис. 11) команди ви отримаєте метрики для вашої моделі, які вказують на те, наскільки добре вона виконує детекцію на тестових даних.

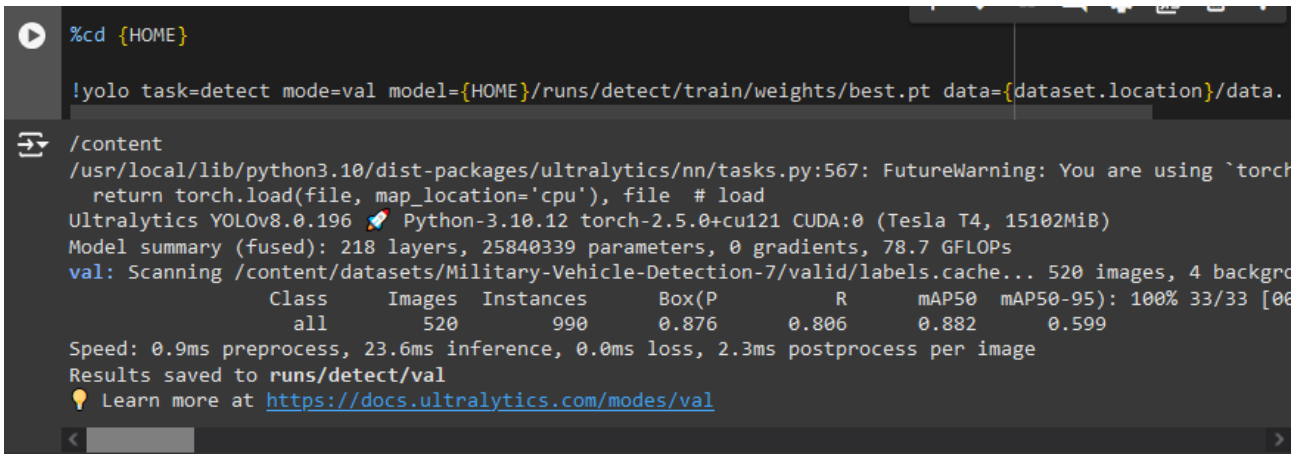

Рисунок 11 - Ілюстрація виводу для алгоритму валідації результатів навчання

Команда !yolo task=detect mode=predict model={HOME}/runs/detect/train/weights/best.pt conf=0.25 source={dataset.location}/test/images save=True augment=True (рис. 12) виконує предикцію (детекцію об'єктів) на тестових зображеннях після завершення тренування моделі.

Мета команди:

- Предсказання (детекція об'єктів): Модель після тренування застосовується до нових зображень і виявляє об'єкти, які відповідають навчальним класам. Це дозволяє оцінити, наскільки добре модель справляється з реальними даними.
- Збереження результатів: Всі результати з предсказаннями (зображення з накладеними коробками навколо об'єктів) будуть збережені, щоб їх можна було проаналізувати.
- Аугментації: Додавання аугментацій під час передбачення може покращити стійкість моделі, забезпечуючи її здатність правильно визначати об'єкти в різних умовах (наприклад, зміни в освітленні, положенні об'єктів тощо).

Результатом виконання цієї команди буде набір зображень, на яких відображатимуться прямокутні рамки навколо виявлених об'єктів, а також збереження цих результатів для подальшого аналізу.

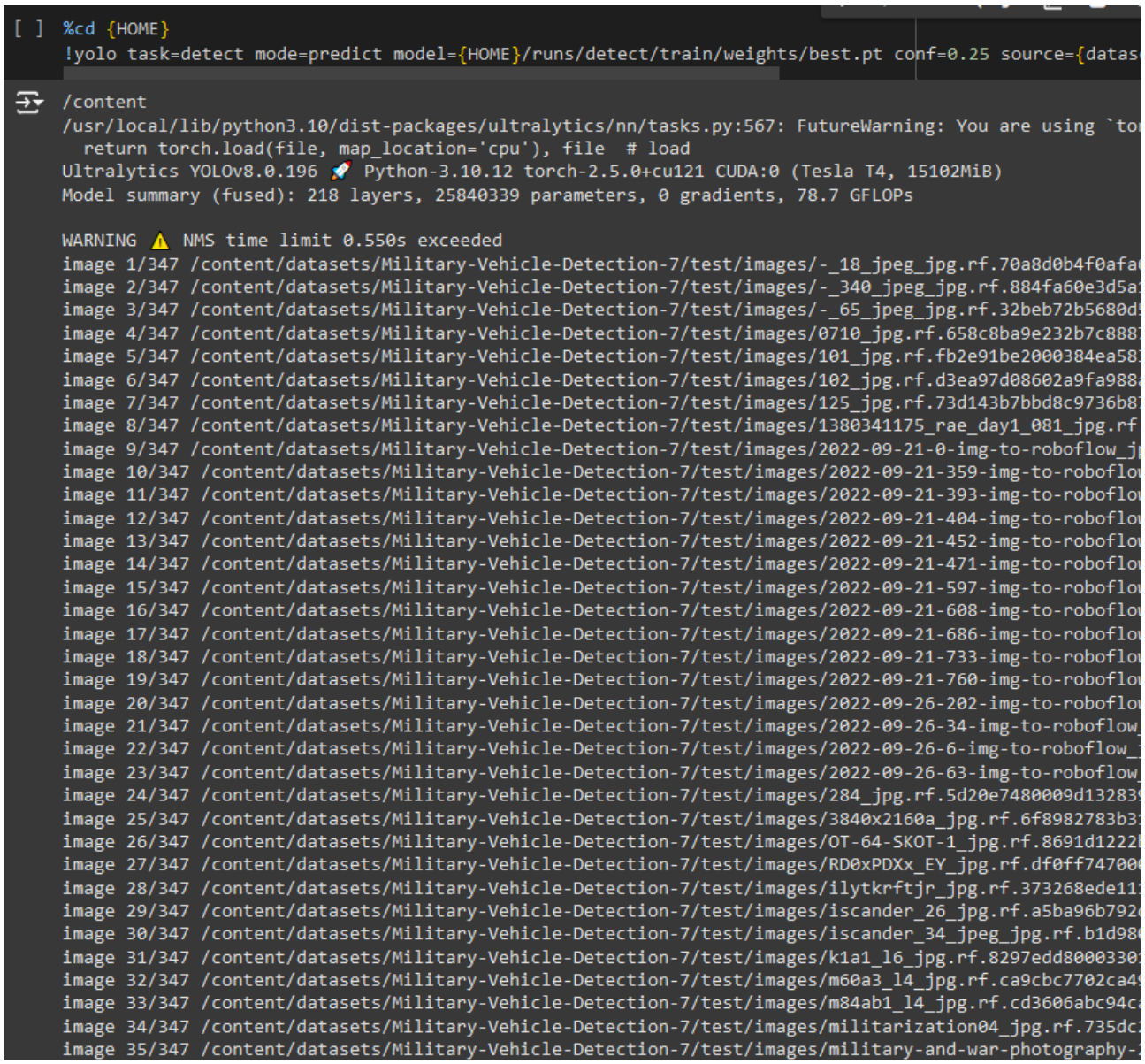

Рисунок 12 - Детекція об'єктів

Пояснення коду на (рис. 13):

- os.path.isdir(): Перевіряє, чи є об'єкт директорією.
- os.path.getmtime(): Повертає час останньої модифікації файлу або директорії, що дозволяє визначити найновіший каталог.
- glob.glob(f'{latest_folder}/*.jpg'): Шукає всі файли .jpg в останньому каталозі.
- Image(filename=image_path, width=600): Виводить кожне зображення в Jupyter notebook з вказаною шириною.

Результатом виконання цього коду буде відображення трьох зображень з найновішого каталогу результатів передбачення (детекції об'єктів) (рис. 14).

```python
import glob
from IPython.display import Image, display

# Define the base path where the folders are located
base_path = '/content/runs/detect/'

# List all directories that start with 'predict' in the base path
subfolders = [os.path.join(base_path, d) for d in os.listdir(base_path)
              if os.path.isdir(os.path.join(base_path, d)) and d.startswith('predict')]

# Find the latest folder by modification time
latest_folder = max(subfolders, key=os.path.getmtime)

image_paths = glob.glob(f'{latest_folder}/*.jpg')[:3]

# Display each image
for image_path in image_paths:
    display(Image(filename=image_path, width=600))
    print("\n")
```

Рисунок 13 - Код для виводу результатів детекції

Рисунок 14 - результат детекції

Ця команда відображає вміст папки weights у директорії runs/detect/train/ (рис. 15). У цій папці зазвичай зберігаються різні версії вагів моделі, отримані під час навчання, зокрема:

- best.pt: файл, що містить найкращу версію збереженої моделі (на основі обраних метрик ефективності).
- last.pt: файл з вагами моделі, збереженими після останньої епохи навчання.

Команда допомагає підтвердити наявність цих файлів і перевірити структуру каталогів у разі потреби.

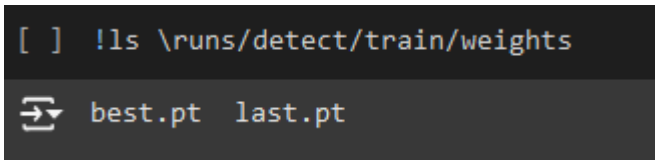

Рисунок 15 - Вміст папки weights

2.3 КРИТЕРІЇ ВАЛІДАЦІЇ ЕФЕКТИВНОСТІ МОДЕЛІ ДЕТЕКТОРА

Для оцінки ефективності системи детекції об’єктів, зокрема при розпізнаванні військової техніки, необхідно враховувати низку важливих показників, які визначають, наскільки добре модель виконує своє завдання. Ці метрики дозволяють глибоко оцінити, наскільки модель відповідає вимогам до точності, надійності та продуктивності, і чи зможе вона використовуватись у реальних умовах. Нижче розглянемо основні критерії оцінки детектора, а також продемонструємо, як вони застосовуються в коді.

Основні критерії оцінки детектора об'єктів

1. Точність (Precision)

   Точність є одним із ключових показників, що визначає якість роботи моделі у випадках, коли важливо уникнути помилкових спрацьовувань. Вона показує частку об'єктів, які були правильно класифіковані як військова техніка серед усіх виявлених об'єктів. Наприклад, якщо модель знаходить об'єкти, які насправді не є військовою технікою, це знижує точність. Висока точність означає, що модель рідко генерує помилкові сигнали, тим самим мінімізуючи кількість хибнопозитивних результатів. У військовій сфері точність має

критичне значення, адже помилкові розпізнавання можуть призвести до хибних рішень. Точність обчислюється за формулою (рис. 16).

```
precision = true_positive / (true_positive + false_positive)
```

Рисунок 16 - Формула точності обчислення

2. Повнота (Recall)

   Повнота відображає здатність моделі знаходити усі об’єкти (рис. 17), що справді належать до класу військової техніки, серед усіх об'єктів на зображенні. Ця метрика є особливо важливою в задачах, де потрібно знайти всі релевантні об'єкти, оскільки непомічені об’єкти можуть призвести до прогалин в аналізі. Висока повнота свідчить, що модель здатна розпізнати більшість наявних на зображенні об'єктів, що є важливим для всебічного огляду місцевості у військових додатках.

```
recall = true_positive / (true_positive + false_negative)
```

Рисунок 17 - Формула обчислення повноти

3. F1-міра

   Це середнє гармонічне між точністю і повнотою, що допомагає знайти збалансовану оцінку, коли потрібно зменшити кількість як хибнопозитивних, так і хибнонегативних спрацьовувань. F1-міра важлива у випадках, де помилки обох типів (недетекція та помилкове розпізнавання) є критичними. Вона дозволяє оцінити модель в комплексі, забезпечуючи більш об'єктивну характеристику її загальної продуктивності (рис. 18).

```
f1_score = 2 * (precision * recall) / (precision + recall)
```

Рисунок 18 - Формула обчислення F1-міри

4. Intersection over Union (IoU)

   IoU використовується для визначення, наскільки близько модель розташувала рамку навколо об'єкта, що є критичним для точного позиціонування об'єктів. Чим більше площа перекриття між рамкою, згенерованою моделлю, і реальною рамкою об'єкта, тим вища IoU, що свідчить про високу точність позиціонування. Для задач військової розвідки цей показник є особливо цінним, адже точність позиціонування об'єктів має прямий вплив на оперативні рішення (рис. 19).

```python
def calculate_iou(box1, box2):
    x1 = max(box1[0], box2[0])
    y1 = max(box1[1], box2[1])
    x2 = min(box1[2], box2[2])
    y2 = min(box1[3], box2[3])
    intersection = max(0, x2 - x1) * max(0, y2 - y1)
    box1_area = (box1[2] - box1[0]) * (box1[3] - box1[1])
    box2_area = (box2[2] - box2[0]) * (box2[3] - box2[1])
    union = box1_area + box2_area - intersection
    return intersection / union if union > 0 else 0
```

Рисунок 19 - Intersection over Union (IoU)

5. Швидкість обробки

   Для роботи у реальному часі, наприклад, у військових умовах або під час аналізу потоку відео з безпілотних апаратів, важливою є не лише точність, але й швидкість обробки відео. Навіть мінімальні затримки можуть значно вплинути на якість прийняття рішень. Тому тестування моделі також включає оцінку її здатності обробляти відео у реальному часі.

Нижче наведено приклад коду для розпізнавання об'єктів за допомогою технології YOLOv8, який використовує дві різні версії моделі — меншу (YOLOv8s) та середню (YOLOv8m). Комбінація цих двох моделей дозволяє

досягти більш високої надійності та точності в розпізнаванні об’єктів, зокрема для задач, де важлива мінімізація помилок.

Код побудований так, що для кожного кадру відео отримуються прогнози від обох моделей. Далі застосовується підхід консенсусного голосування, який об’єднує результати від двох моделей для забезпечення більшої точності. Крім того, для усунення помилкових спрацьовувань застосовується алгоритм Non-Maximum Suppression (NMS), який відсіює дублікати рамок навколо одного й того ж об’єкта.

Основні етапи роботи коду на (рис. 20) та (рис. 21)

1. Ініціалізація моделей YOLOv8
   Для завантаження моделей використовується клас YOLO з бібліотеки ultralytics. Ми завантажуємо дві різні моделі — YOLOv8s (small) та YOLOv8m (medium). Використання різних розмірів моделей допомагає збільшити різноманітність виявлення, забезпечуючи надійні результати.
2. Налаштування відеовходу та відеовиходу
   Ми завантажуємо відеофайл, який потрібно обробити, а також налаштовуємо параметри для збереження обробленого відео. Для запису результатів у форматі H.264 використовується функція cv2.VideoWriter.
3. Обробка кожного кадру відео
   Для кожного кадру відео (в циклі while) виконується кілька послідовних дій:
   - Виконання детекції: обидві моделі (YOLOv8s та YOLOv8m) отримують кадр і повертають результати детекції, які включають координати рамок, ймовірності класів та ідентифікатори класів.
   - Збір результатів детекції: результати з обох моделей обробляються, і ми витягуємо координати рамок (boxes), впевненості (scores) та ідентифікатори класів (classes).

4. Комбінація результатів
   Після отримання результатів від обох моделей ми об'єднуємо всі координати рамок, ймовірності та класи в єдині списки. Це створює повний набір детекцій, отриманих з двох різних моделей, що дозволяє більш точно врахувати всі можливі об’єкти.

5. Застосування консенсусного голосування
   Для підвищення надійності результати проходять через процес консенсусного голосування. Консенсусне голосування дозволяє залишити тільки ті рамки, які були знайдені обома моделями і мають схожі координати (визначені на основі порогу Intersection over Union, IoU). Це знижує ймовірність хибних спрацьовувань, оскільки обидві моделі повинні дійти згоди щодо існування об'єкта.

6. Використання Non-Maximum Suppression (NMS)
   Після консенсусного голосування застосовується метод NMS для усунення дублюючих рамок, які перекриваються. NMS допомагає залишити лише одну найбільш точну рамку для кожного об’єкта, що суттєво підвищує якість остаточного результату.

7. Малювання рамок на кадрі
   Після фільтрації рамок методом NMS ми малюємо фінальні рамки на кадрі, відображаючи об’єкти з достатньою ймовірністю. Це робиться за допомогою функції cv2.rectangle.

8. Запис обробленого кадру
   Кожен оброблений кадр із намальованими рамками записується у вихідне відео, яке зберігається у визначеному форматі.

```python
import os
import cv2
import torch
from ultralytics import YOLO
from torchvision.ops import nms

def yolov8_ensemble_detection_on_video(video_path, file_name, iou_threshold=0.5, score_threshold=0.3):
    # Завантаження двох моделей YOLOv8: меншої (YOLOv8s) та середньої (YOLOv8m)
    model1 = YOLO("YOLO\\YOLOv8s.pt")
    model2 = YOLO("YOLO\\YOLOv8m.pt")

    # Налаштування шляхів для входу та в (function) MEDIA_ROOT: Any
    input_video = os.path.join(settings.MEDIA_ROOT, video_path)
    output_video = os.path.join(settings.MEDIA_ROOT, "output", f"detected_{file_name}")
    cap = cv2.VideoCapture(input_video)
    fourcc = cv2.VideoWriter_fourcc(*'H264')
    out = cv2.VideoWriter(output_video, fourcc, cap.get(cv2.CAP_PROP_FPS),
                          (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)), int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))))
```

Рисунок 20 - Код для розпізнавання об'єктів частина 1

```python
    # Обробка кожного кадру відео
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Виконання детекції з обома моделями з використанням аугментації
        results1 = model1(frame, augment=True)
        results2 = model2(frame, augment=True)

        # Збір прогнозів від кожної моделі
        boxes1, scores1, classes1 = extract_boxes_scores_classes(results1)
        boxes2, scores2, classes2 = extract_boxes_scores_classes(results2)

        # Комбінування рамок та ймовірностей з обох моделей
        combined_boxes = torch.cat((boxes1, boxes2), dim=0)
        combined_scores = torch.cat((scores1, scores2))
        combined_classes = torch.cat((classes1, classes2))

        # Консенсусне голосування: залишаємо лише рамки, що є в обох моделях
        vouted_boxes, vouted_scores, vouted_classes = consensus_voting(combined_boxes, combined_scores, combined_classes, iou_threshold)

        # Застосування NMS для видалення зайвих рамок
        keep_indices = nms(vouted_boxes, vouted_scores, iou_threshold)
        final_boxes = vouted_boxes[keep_indices]
        final_scores = vouted_scores[keep_indices]
        final_classes = vouted_classes[keep_indices]

        # Малювання фінальних рамок на кадрі
        for i, box in enumerate(final_boxes):
            x1, y1, x2, y2 = [int(coord) for coord in box]
            score = final_scores[i].item()
            class_id = int(final_classes[i].item())
            if score > score_threshold:
                cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)

        # Запис кадру з рамками у вихідне відео
        out.write(frame)

    # Завершення обробки
    cap.release()
    out.release()
    return output_video
```

Рисунок 21 - Код для розпізнавання об'єктів частина 2

Переваги методу

Цей підхід забезпечує високу надійність за рахунок використання декількох моделей і застосування фільтрації через консенсусне голосування та NMS, що дозволяє знизити кількість хибних спрацьовувань.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ АНАЛІЗУ ДАНИХ АЕРОВІДЕОСПОСТЕРЕЖЕННЯ

3.1 ФОРМУВАННЯ НАВЧАЛЬНИХ ТА ТЕСТОВИХ ДАНИХ

Вхідні дані для навчання та тестування моделі було отримано з платформи Roboflow, де було створено спеціалізований проєкт для виявлення військової техніки на аерофотознімках. Цей проєкт містив понад 7000 зображень, що охоплюють різні типи військової техніки в різних умовах зйомки. Великий і різноманітний набір зображень є критично важливим для досягнення високої точності розпізнавання об'єктів, оскільки дозволяє моделі навчитися відрізняти об’єкти в різноманітних умовах, наприклад, за різного освітлення, фону, ракурсів та розмірів об’єктів.

Процес підготовки даних включав розподіл на навчальний, тестовий та валідаційний набори у співвідношенні 70:20:10. Це дає змогу забезпечити об’єктивну оцінку точності моделі та її здатності узагальнювати результати на нових, раніше невідомих зображеннях.

Аугментація зображень стала важливим етапом підготовки даних. Під час аугментації до зображень застосовувалися такі методи, як поворот, зміна масштабу, яскравості, контрасту, а також інверсія кольорів і додавання шуму. Ці операції спрямовані на покращення адаптивності моделі до різноманітних сценаріїв у реальних умовах. Наприклад, при зміні умов освітлення, віддаленості об'єкта чи його кута нахилу модель, навчену на аугментованих даних, здатна з більшою точністю визначити об'єкт.

Різноманітність даних була особливо важлива для покращення узагальнювальної здатності моделі. До набору включено зображення, що охоплюють різні погодні умови, типи фону та ракурси, що забезпечує модель надійними зразками для навчання. Це дозволяє моделі краще адаптуватися до реальних умов, оскільки в умовах реального застосування об’єкти можуть з’являтися в найрізноманітніших обставинах.

Анотація даних була здійснена за допомогою інструментів Roboflow, що значно пришвидшило процес підготовки та зменшило трудовитрати. Кожен об'єкт на зображеннях позначався рамкою з визначенням класу, що дозволяло моделі отримати максимально точні координати об’єктів для навчання. Завдяки цьому модель отримувала повну інформацію щодо місцезнаходження об'єктів військової техніки на зображенні, а використання стандартизованих форматів анотацій, таких як COCO та YOLO, спростило інтеграцію набору даних у процес навчання без додаткової обробки.

Переваги використання Roboflow стали очевидними завдяки можливостям платформи для зручного управління набором даних: це стосувалося завантаження, анотацій та організації зображень. Крім того, Roboflow дозволив легко налаштувати процес аугментації для підвищення стійкості моделі, а також надав стандартизований формат даних, що полегшив процес інтеграції з моделями YOLOv8.

Використання набору даних із понад 7000 зображень є важливим кроком для забезпечення стабільної роботи моделі, оскільки для досягнення високої точності та узагальнювальної здатності моделі машинного навчання необхідна велика кількість якісних даних. Це особливо важливо в завданнях розпізнавання об'єктів, де варіативність умов зйомки може сильно вплинути на результати. Залучення великого обсягу даних забезпечує моделі можливість аналізувати патерни в різних обставинах, що, в кінцевому рахунку, підвищує її точність у реальних умовах, які часто бувають непередбачуваними.

3.2 АРХІТЕКТУРА І КОРОТКИЙ ОПИС ВЕБ-ОРІЄНТОВАНОГО ДОДАТКУ/СЕРВІСУ

Цей веб-орієнтований додаток створений за допомогою фреймворку Django та побудований за архітектурною моделлю MVC (Model-View-Controller). Ця архітектура дозволяє чітко розділити функціональність додатка на три ключові компоненти: Model, View, Controller. Такий підхід полегшує підтримку і масштабування проекту.

Архітектура та компоненти

- Model (Модель): Включає всі сервіси, які відповідають за обробку даних і виконання детекції об’єктів за допомогою моделі YOLOv8.
- View (Вигляд): Цей компонент відповідає за взаємодію з користувачем і відображення даних на веб-сторінці.
- Controller (Контролер): Він обробляє запити користувача, взаємодіє з моделями і передає результати у вигляд для подальшого відображення.

Цей веб-додаток дозволяє користувачам завантажувати відеофайли, запускати процес виявлення об’єктів і переглядати результати безпосередньо на сторінці. Процес обробки даних повністю автоматизований і надає користувачеві швидкий результат у вигляді відео з позначеними об'єктами.

Структура сайту

1. Поле для завантаження відео на (рис. 22)

Перше, що бачить користувач на сайті — це поле для завантаження відеофайлу. Користувач вибирає відео для обробки, і це відео буде надіслано на сервер, де воно пройде через всі етапи обробки.

- Поле дозволяє завантажувати відео в форматах MP4 та WebM.
- Вибір файлу обмежується цими типами, щоб уникнути помилок при обробці.

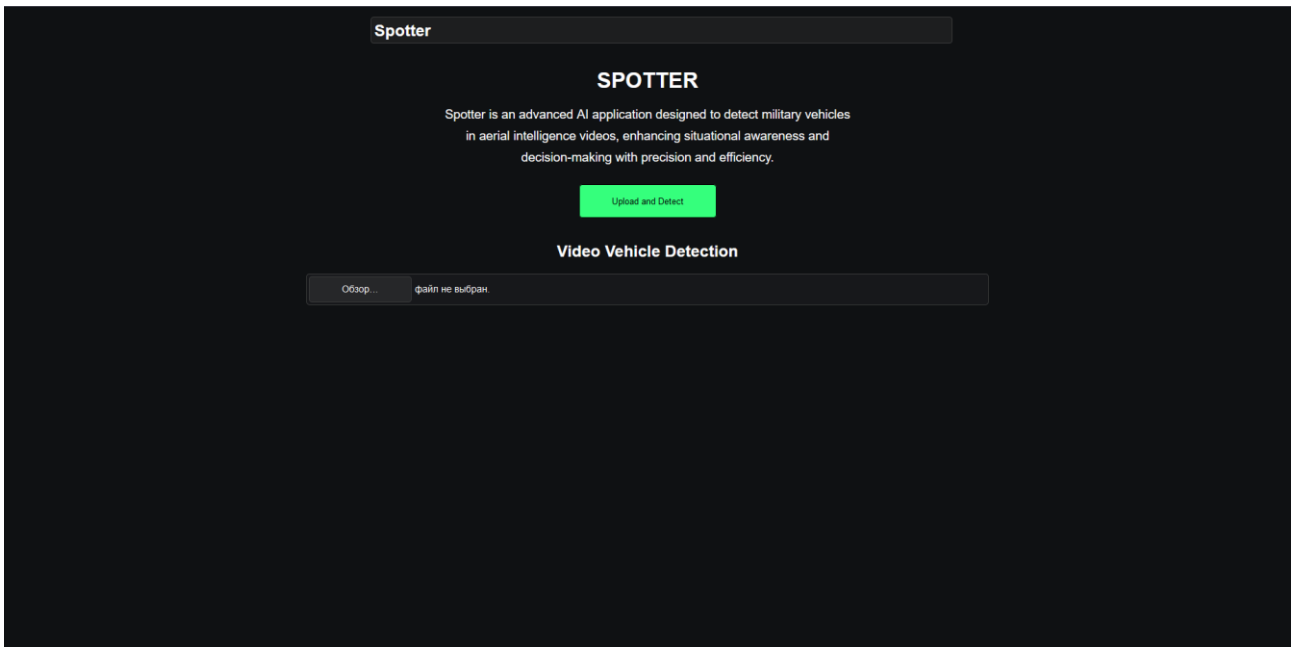

Рисунок 22 - Форма для завантаження відео

2. Кнопка для відправлення форми

Після вибору відеофайлу користувач натискає на кнопку "Upload and Detect" (рис. 23), щоб почати обробку відео. Ця кнопка ініціює процес завантаження та аналізу відео. При натисканні на кнопку сервер отримує відеофайл, починає процес детекції об’єктів, використовуючи дві моделі YOLOv8, і обробляє його.

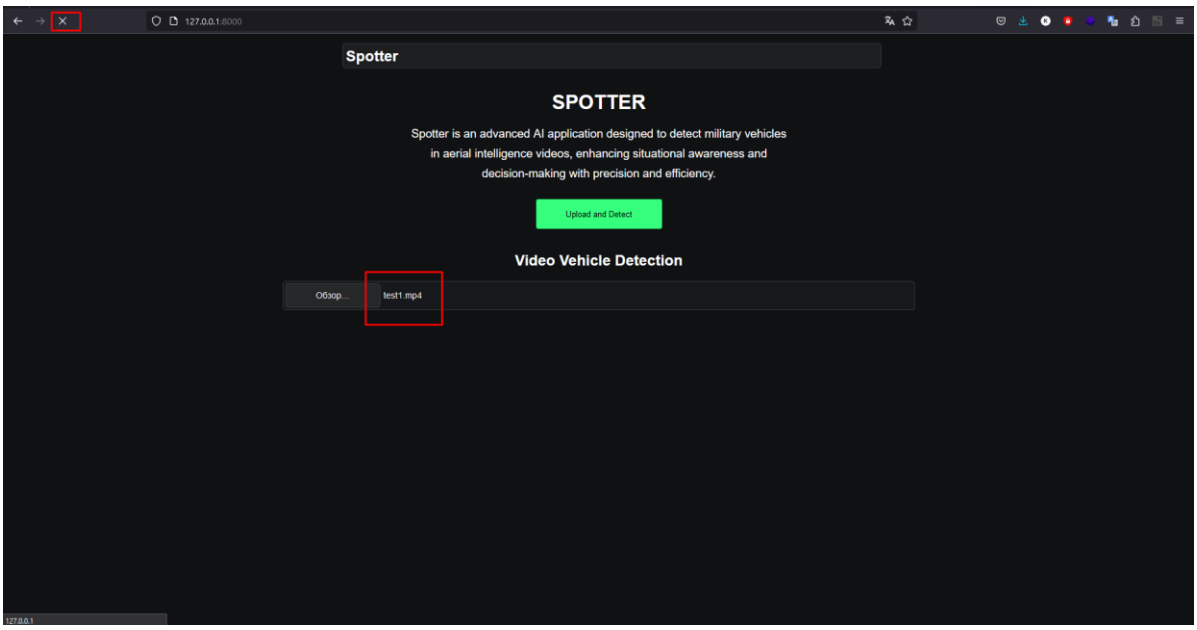

Рисунок 23 - Процес обробки відео

3. Поле для результату

Після завершення процесу обробки відео користувач отримує оброблене відео (рис. 24), яке містить рамки, що позначають виявлені об'єкти — в даному випадку військову техніку. Це відео можна переглянути безпосередньо на веб-сторінці.

- Відео відображається у відеоплеєрі, який дозволяє користувачеві перемотувати та переглядати результат.
- Відображення відео супроводжується інформацією про успішність детекції.

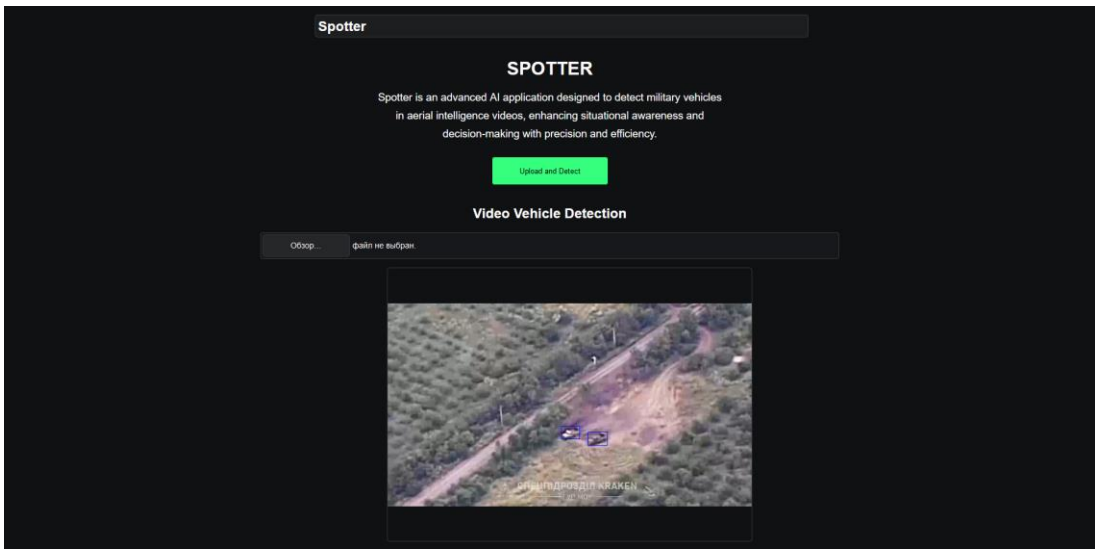

Рисунок 24 - Результат обробки відео

Опис процесу обробки відео

Процес обробки відео розбивається на кілька ключових етапів:

1. Завантаження відео та очищення папок

   Користувач завантажує відеофайл, і цей файл зберігається у директорії uploaded_videos на сервері. Після цього автоматично очищуються всі попередні результати в папці output, щоб уникнути перезапису старих файлів.

2. Запуск YOLOv8 для детекції об'єктів

   Після того, як відео збережено на сервері, система використовує дві моделі YOLOv8:

   - YOLOv8 small model (YOLOv8s).
   - YOLOv8 medium model (YOLOv8m).

   Ці моделі працюють паралельно над кожним кадром відео, виконуючи попередню обробку та визначаючи можливі об'єкти. Потім за допомогою консенсусного голосування (якщо обидві моделі знаходять об'єкт в тій самій області) зберігаються тільки ті рамки, які виявлені обома моделями. Це підвищує точність результатів.

3. Застосування Non-Maximum Suppression (NMS)

   Застосовується метод NMS (Non-Maximum Suppression), який дозволяє позбутися від зайвих, дубльованих рамок. Це важливий крок для отримання чистого результату, в якому кожен об'єкт позначений лише однією рамкою.

4. Збереження результату

   Після обробки всіх кадрів відео результат зберігається у форматі відеофайлу, який містить рамки, що позначають виявлені об'єкти. Користувач може завантажити цей файл і переглянути його через веб-інтерфейс (рис. 25).

Рисунок 25 - Зображення з відміченими об'єктами за навченою моделлю

Інтерфейс користувача

Інтерфейс додатку простий і зручний. Веб-сторінка містить наступні компоненти:

1. Форма для завантаження відео: Користувач вибирає файл відео та натискає кнопку для його завантаження (рис. 26).
2. Опис додатку: Пояснення, що додаток робить, і як він може бути корисний для користувача (рис. 26).
3. Відеоплеєр: Після обробки відео, користувач може переглядати результати детекції об'єктів безпосередньо на сайті (рис. 27).

Цей простий і ефективний інтерфейс дає користувачеві змогу швидко завантажити відео та отримати результати виявлення в реальному часі, не потребуючи додаткових кроків чи налаштувань.

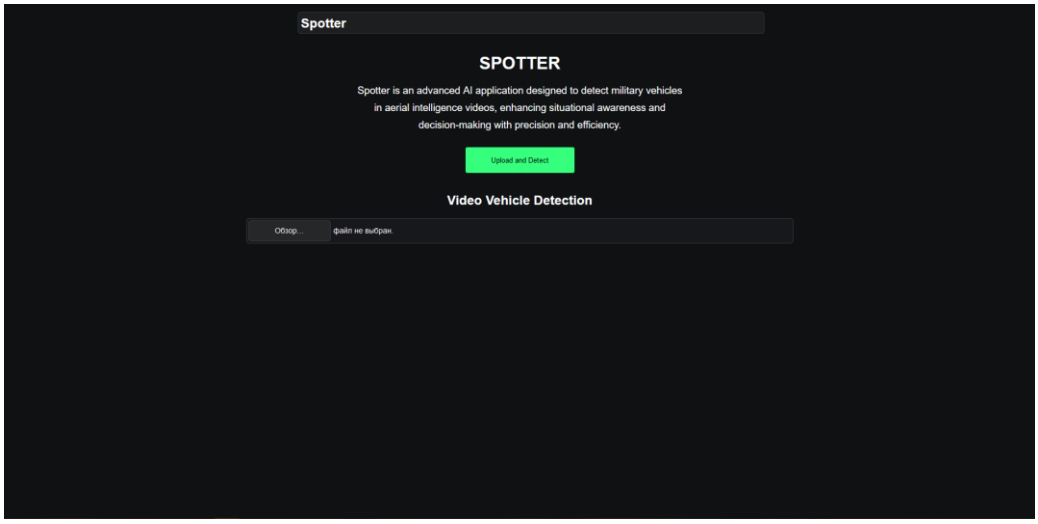

Рисунок 26 - Інтерфейс користувача до відправки відео

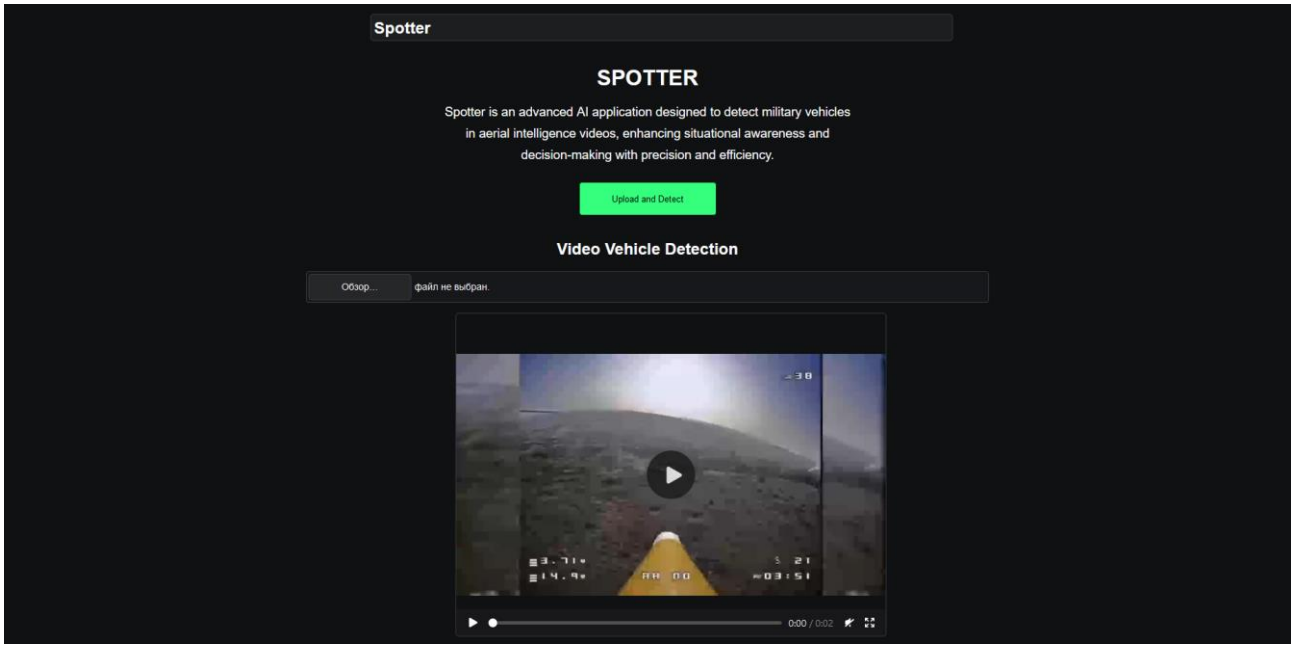

Рисунок 27 - Інтерфейс користувача після відправки відео

ВИСНОВОК

У рамках кваліфікаційної роботи було реалізовано веб-орієнтований додаток для виявлення військової техніки на аерофотознімках за допомогою моделей YOLOv8. Розроблений додаток забезпечує зручний інтерфейс для завантаження відеофайлів, їх обробки та отримання результатів в реальному часі. Основною особливістю реалізованої системи є використання ансамблю моделей YOLOv8 (малого та середнього розміру) для підвищення точності та надійності виявлення об'єктів. Під час роботи системи застосовуються методи консенсусного голосування та Non-Maximum Suppression (NMS) для відсіювання помилкових прогнозів.

Веб-додаток реалізовано за архітектурою MVC в середовищі Django, що дозволяє забезпечити масштабованість та зручність у подальшій підтримці. Всі етапи обробки відео, включаючи зчитування файлів, виявлення об'єктів та запис результатів, виконуються автоматично, що забезпечує високу ефективність та зручність для кінцевих користувачів.

Проведене дослідження та реалізація системи показали, що використання сучасних технологій комп'ютерного зору, таких як YOLOv8, дозволяє досягти високої точності виявлення військової техніки на аерофотознімках. Система демонструє хороший результат при роботі з реальними відеофайлами, що містять різноманітні об'єкти в різних умовах зйомки. Подальший розвиток цієї системи може включати оптимізацію процесу обробки для роботи з більшими наборами даних, а також розширення функціоналу для підтримки інших типів об'єктів та задач розпізнавання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Live object detection and image segmentation with yolov8. Analytics Vidhya. URL: https://www.analyticsvidhya.com/blog/2024/03/live-object-detection-and-image-segmentation-with-yolov8/ (date of access: 08.11.2024).
2. Medium. Medium. URL: https://towardsdatascience.com/test-time-augmentation (date of access: 08.11.2024).
3. Nain A. EfficientDet: scalable and efficient object detection. Medium. URL: https://medium.com/@nainaakash012/efficientdet-scalable-and-efficient-object-detection-ea05ccd28427?source=search_post---------0---------- (date of access: 08.11.2024).
4. Nelson J. Training a yolov3 object detection model with a custom dataset. Roboflow Blog. URL: https://blog.roboflow.com/training-a-yolov3-object-detection-model-with-a-custom-dataset/ (date of access: 08.11.2024).
5. OpenCV: detection of aruco markers. OpenCV documentation index. URL: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html (date of access: 08.11.2024).
6. Ultralytics. Home. Home - Ultralytics YOLO Docs. URL: https://docs.ultralytics.com (date of access: 08.11.2024).

ДОДАТОК А: view.py

```
from django.shortcuts import render
from .services import clean_folder, yolov8_ensemble_detection_on_video
from django.core.files.storage import FileSystemStorage
from video_detector import settings

def index(request):
    if request.method == 'POST' and request.FILES['video']:
        clean_folder(f'{settings.MEDIA_ROOT}\\output')

        video_file = request.FILES['video']
        fs = FileSystemStorage()
        video_path = fs.save(f"uploaded_videos\\{video_file.name}", video_file)

        output_video_url =
fs.url(yolov8_ensemble_detection_on_video(video_path=video_path,
file_name=video_file.name))

        clean_folder(f'{settings.MEDIA_ROOT}\\uploaded_videos')

        return render(request, 'detection\\index.html', {
            'output_video_url': output_video_url,
        })
```

```
    return render(request, 'detection\\index.html')
```

ДОДАТОК В: services.py

```
import os
import cv2
import torch
from ultralytics import YOLO
from django.conf import settings
from torchvision.ops import nms


def yolov8_ensemble_detection_on_video(video_path, file_name, iou_threshold=0.5, score_threshold=0.3):
    # Load two YOLOv8 models
    model1 = YOLO("YOLO\\YOLOv8s.pt")  # YOLOv8 small model
    model2 = YOLO("YOLO\\YOLOv8m.pt")  # YOLOv8 medium model


    # Set up video input and output paths
    input_video = os.path.join(settings.MEDIA_ROOT, video_path)
    output_video = os.path.join(settings.MEDIA_ROOT, "output", f"detected_{file_name}")
    cap = cv2.VideoCapture(input_video)
    fourcc = cv2.VideoWriter_fourcc(*'H264')
```

```
    out = cv2.VideoWriter(output_video, fourcc, cap.get(cv2.CAP_PROP_FPS),
                          (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),
int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))))

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Run detection with both models and augmentation
        results1 = model1(frame, augment=True)
        results2 = model2(frame, augment=True)

        # Gather predictions from both models
        boxes1, scores1, classes1 = extract_boxes_scores_classes(results1)
        boxes2, scores2, classes2 = extract_boxes_scores_classes(results2)

        # Combine boxes and scores from both models
        combined_boxes = torch.cat((boxes1, boxes2), dim=0)  # Ensure 2D shape
        combined_scores = torch.cat((scores1, scores2))
        combined_classes = torch.cat((classes1, classes2))
```

```
        # Consensus voting: Only keep boxes detected by both models
        vouted_boxes, vouted_scores, vouted_classes = consensus_voting(combined_boxes, combined_scores, combined_classes, iou_threshold)

        # Apply NMS to the combined predictions
        keep_indices = nms(vouted_boxes, vouted_scores, iou_threshold)
        final_boxes = vouted_boxes[keep_indices]
        final_scores = vouted_scores[keep_indices]
        final_classes = vouted_classes[keep_indices]

        # Draw final boxes on the frame
        for i, box in enumerate(final_boxes):
            x1, y1, x2, y2 = [int(coord) for coord in box]
            score = final_scores[i].item()
            class_id = int(final_classes[i].item())
            if score > score_threshold:  # Filter based on score threshold
                cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)

        # Write the frame with drawn boxes to the output video
        out.write(frame)

    cap.release()
```

```
    out.release()
    return output_video


def extract_boxes_scores_classes(results):
    """Helper function to extract bounding boxes, scores, and classes from YOLO results."""
    boxes = []
    scores = []
    classes = []
    for result in results:
        for box in result.boxes:
            x1, y1, x2, y2 = box.xyxy[0]
            boxes.append([x1.item(), y1.item(), x2.item(), y2.item()])  # Ensure 2D format
            scores.append(box.conf[0].item())
            classes.append(box.cls[0].item())
    return torch.tensor(boxes).reshape(-1, 4), torch.tensor(scores), torch.tensor(classes)


def consensus_voting(boxes, scores, classes, iou_threshold=0.5):
    """Applies consensus voting to keep only the boxes detected by both models."""
    final_boxes, final_scores, final_classes = [], [], []
    for i in range(len(boxes)):
        for j in range(i + 1, len(boxes)):
```

```
        # Calculate IoU between each pair of boxes
        iou = calculate_iou(boxes[i], boxes[j])
        if iou >= iou_threshold and classes[i] == classes[j]:  # Check if boxes belong to the same class
            final_boxes.append(boxes[i])
            final_scores.append(max(scores[i], scores[j]))  # Use max confidence score
            final_classes.append(classes[i])

    # Check if lists are non-empty before converting to tensors
    if final_boxes:
        return torch.stack(final_boxes), torch.tensor(final_scores), torch.tensor(final_classes)
    else:
        # Return empty tensors if no consensus boxes are found
        return torch.empty((0, 4)), torch.empty(0), torch.empty(0)

def calculate_iou(box1, box2):
    """Helper function to calculate IoU between two boxes."""
    x1 = max(box1[0], box2[0])
    y1 = max(box1[1], box2[1])
    x2 = min(box1[2], box2[2])
    y2 = min(box1[3], box2[3])
    intersection = max(0, x2 - x1) * max(0, y2 - y1)
```

```
    box1_area = (box1[2] - box1[0]) * (box1[3] - box1[1])
    box2_area = (box2[2] - box2[0]) * (box2[3] - box2[1])
    union = box1_area + box2_area - intersection
    return intersection / union if union > 0 else 0

def yolov8_small_detection_on_video(video_path):
    # Load the YOLOv8 model
    model = YOLO("YOLO\\YOLOv8s.pt")  # Assuming YOLOv8n model is being used
    # Run detection on each frame
    input_video = os.path.join(settings.MEDIA_ROOT, video_path)
    output_video = os.path.join(settings.MEDIA_ROOT, "output", "detected_video.mp4")
    os.makedirs(os.path.dirname(output_video), exist_ok=True)
    cap = cv2.VideoCapture(input_video)
    fourcc = cv2.VideoWriter_fourcc(*'H264')
    out = cv2.VideoWriter(output_video, fourcc, cap.get(cv2.CAP_PROP_FPS),
                          (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),
int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))))


    while cap.isOpened():
        ret, frame = cap.read()
```

```
        if not ret:
            break

        # Perform detection
        results = model(frame, augment=True)
        for result in results:
            boxes = result.boxes
            for box in boxes:
                x1, y1, x2, y2 = box.xyxy[0]
                cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (255, 0, 0), 2)


        out.write(frame)

    cap.release()
    out.release()

def clean_folder(directory_path):
    if os.path.exists(directory_path):
        files = os.listdir(directory_path)

        for file in files:
```

```
            file_path = os.path.join(directory_path, file)
            if os.path.isfile(file_path):
                os.remove(file_path)
    else:
        pass
```

ДОДАТОК С: index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="static\css\style.css">
    <meta charset="UTF-8">
    <title>Spotter</title>
</head>
<body>
    <h2 id="logo">Spotter</h2>

    <form method="POST" enctype="multipart/form-data">
        {% csrf_token %}
        <h1>SPOTTER</h1>
```

```
        <h3>Spotter is an advanced AI application designed to detect military
vehicles</h3>
            <h3>in aerial intelligence videos, enhancing situational awareness and</h3>
             <h3> decision-making with precision and efficiency.</h3>


        <button type="submit">Upload and Detect</button>


        <h2>Video Vehicle Detection</h2>


        <input type="file" name="video" accept="video/mp4, video/webm" required>
    </form>


    {% if output_video_url %}
        <video width="640" height="480" controls>
            <source src="{{ output_video_url }}" type="video/mp4">
            <source src="{{ output_video_url }}" type="video/ogg">
            <source src="{{ output_video_url }}" type="video/webm">
            Your browser does not support the video tag.
        </video>
    {% endif %}
</body>
</html>
```