Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розроблення методу ідентифікації вибухонебезпечних об'єктів за їх візуальними ознаками
(тема)

Виконав:
студент 2 курсу, групи КТРСм-22-1

Шестак Д. Д.

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології

Тип програми Освітньо-професійна

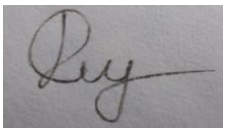
Освітня програма Комп'ютеризовані та робототехнічні системи

Керівник Новоселов С. П.

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

Харків 2023 р.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

«08» 01 2024  Шестак Д. Д.

(підпис) (прізвище, ініціали)

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики і комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти другий (магістерський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми Освітньо-професійна
Освітня програма Комп'ютеризовані та робототехнічні системи
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАР
(підпис)
« 03 » 11 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Шестаку Дмитру Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Розроблення методу ідентифікації вибухонебезпечних об'єктів за їх візуальними ознаками»
Затверджена наказом по університету від 03.11.2023р. №1288 Ст
2. Термін подання студентом роботи до екзаменаційної комісії
3. Вихідні дані до роботи: 3.1 Програмне середовище для програмування мовою Python – Pycharm IDE
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ;
4.2 Аналіз технічного завдання;
4.3 Аналіз існуючих способів ідентифікації об'єктів;
4.4 Підготовка матеріалу для навчання штучного інтелекту;
4.5 Тестування штучного інтелекту;
4.6 Тестування роботи моделі штучного інтелекту;
4.7 Розробка програмного забезпечення для використання моделі штучного інтелекту;
4.8 Тестування програмного забезпечення використовуючи навчену модель штучного інтелекту;
4.9 Висновки;
4.10 Додатки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій Графічний демонстраційний матеріал у форматі Power Point (*.ptt) – 15 сторінок

6. Консультанти розділів роботи

Найменування розділу	Керівник (посада, прізвище, ім'я, по батькові)	Позначка керівника про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз вхідної інформації та вимог ТЗ	04.11.23 р.	Виконано
2	Аналіз літератури за темою кваліфікаційної		
	роботи	06.11.23 р.	Виконано
3	Навчання моделі штучного інтелекту	08.11.23 р.	Виконано
4	Розробка програмного забезпечення для	18.11.23 р.	Виконано
	використання моделі штучного інтелекту		
5	Експериментальне дослідження	24.11.23 р.	Виконано
6	Тест роботи програмного забезпечення	28.11.23 р.	Виконано
7	Оформлення пояснювальної записки	15.12.23 р.	Виконано
8	Подання роботи на перевірку Інтернет-		
	сервісом Unicheck		
9	Подання роботи на рецензію		
10	Подання роботи на підпис зав. кафедри		
11	Подання кваліфікаційної роботи в ЕК		

Дата видачі завдання 03.11.2023 р.

Студент _____ Шестак Д. Д.
(підпис)

Керівник роботи _____ проф. Новоселов С. П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 60 с., 1 табл., 37 рис., 1 дод., 16 джерел.

ІДЕНТИФІКАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, РОЗРОБКА, РОЗПІЗНАННЯ.

Об’єкт дослідження – процес ідентифікації вибухонебезпечних об’єктів з візуальними ознаками.

Предмет дослідження – система ідентифікації вибухонебезпечних об’єктів за візуальними ознаками.

Мета роботи – розробка методу ідентифікації для ефективного виявлення вибухонебезпечних об’єктів, заснованих на аналізі візуальних ознак.

Методи розробки та апаратне забезпечення: навчання моделі штучного інтелекту, дослідження ідентифікації об’єктів за візуальними ознаками, створення дата сету для навчання штучного інтелекту, розробка програмного забезпечення для ідентифікації об’єктів за візуальними ознаками, повернення координат ідентифікованих об’єктів, повернення назви ідентифікованих об’єктів.

Області застосування – розроблене програмне забезпечення може бути використано для ідентифікації вибухонебезпечних об’єктів за візуальними ознаками на виробництві, у польових умовах, побутових тощо.

Також альтернативний варіант використання в якості тестування нових моделей для штучного інтелекту.

ABSTRACT

The explanatory note contains: 60 p., 1 tabl,. 37 fig., 1 app., 16 sources.

IDENTIFICATION, ARTIFICIAL INTELLIGENCE, SOFTWARE, DEVELOPMENT, RECOGNITION.

The object of the study – process of identifying explosive objects with visual signs.

The subject of the study – system of identification of explosive objects by visual signs.

The purpose of the work – development of identification method for effective detection of explosive objects based on the analysis of visual signs..

Development methods and hardware: artificial intelligence model training, object identification research based on visual features, creation of a data set for artificial intelligence training, software development for object identification based on visual features, returning the coordinates of identified objects, returning the name of identified objects objects; the programming language for the software is Python, the software development environment is Pycharm IDE.

Areas of application – the developed software can be used to identify explosive objects by visual signs in production, in the field, at home, etc.

It is also an alternative option for testing new models for artificial intelligence.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – Практичне завдання;

ШІ – Штучний інтелект;

CNN – Convolutional neural network;

CV – Computer vision;

DS – Data Set;

IVA – Intelligent video analytics;

RPN – Regional proposition network.

ВСТУП

В умовах сучасного світу, де загроза життю й безпеці людини збільшена за рахунок бойових дій і мінування житлових будинків, вулиць та лісів, проблема виявлення та ідентифікації вибухонебезпечних об'єктів має величезне значення. Ця проблема виникає в найрізноманітніших контекстах: від військових операцій до робіт із зачистки вибухонебезпечних об'єктів під час рятувальних операцій та робіт із зносу будівель. Важливим аспектом цієї проблеми є можливість ідентифікації цих об'єктів на основі їх візуальних ознак [1].

Оскільки існує дуже багато різних вибухонебезпечних об’єктів, кожний з них також має спеціальне маркування та форму для ефективного застосування. Через це існує необхідність в розробці системи, яка матиме можливість легкого використання, налаштування та ідентифікації таких об’єктів.

Метою даної кваліфікаційної роботи є розробка методу ідентифікації вибухонебезпечних об'єктів, заснованих на аналізі візуальних ознак. Для реалізації поставленої мети необхідно вирішити наступні задачі:

– проаналізувати існуючі методи ідентифікації об’єктів;

– проаналізувати необхідні вимоги для правильної роботи системи;

– обробити зображення додавши до них візуальні дефекти для можливості ідентифікації об’єктів у складних сценах;

– провести навчання моделі штучного інтелекту;

– провести тестування роботи моделі штучного інтелекту;

– провести розробку програмного забезпечення для ідентифікації вибухонебезпечних об’єктів за візуальними ознаками;

– провести експериментальне дослідження;

– оформити пояснювальну записку згідно вимогам ДСТУ 3008:2015 [2] та рекомендацій [3].

1 АНАЛІЗ ВИБУХОНЕБЕЗПЕЧНИХ ОБ'ЄКТІВ ТА МЕТОДІВ ІДЕНТИФІКАЦІЇ

1.1 Аналіз вибухонебезпечних об'єктів, їх візуальних ознак та методів ідентифікації

1.1.1 Аналіз вибухонебезпечних речовин

Вибухові речовини представляють собою хімічні сполуки або суміші, які можуть піддаватися швидким хімічним перетворенням під впливом різних зовнішніх факторів, таких як нагрівання, удар, тертя або взаємодія з іншими вибуховими пристроями. Під час таких перетворень вони виділяють велику кількість енергії та газів, за рахунок чого й виникає вибух.

Снаряди артилерійського типу мають металевий корпус, вибуховий заряд і підривник (головний або донний). Корпус містить вибуховий заряд і може бути забитий вибухівкою та іншими компонентами (рис. 1.1).

Рисунок 1.1 – Артилерійські снаряди знайдені у землі [4]

Візуальні ознаки артилерійських снарядів можуть значно варіюватися залежно від їх типу, калібру, країни виробника та інших факторів.

До загальних візуальних ознак снарядів артилерійського типу відносяться:

– форма артилерійські снарядів є циліндричною або конічною. можуть мати різні форми та розміри, але вони зазвичай є циліндричними або конічними. Розмір снаряду може бути відносно великим, в залежності від калібру;

– снаряди можуть бути фарбованими у різні кольори відповідно до країни виробника або призначення. Типовими кольорами є зелений для наземних дій, коричневий для танків та бронетехніки, сірий або білий для наземні хибо морських та інші. Фарба може бути матовою або глянцевою;

– більшість артилерійських снарядів мають вибуховий заряд, який може бути видимий у вигляді запалювача або інших частин;

– на зовнішній оболонці снаряду може бути індикація його калібру або характеристик. Ця індикація може включати числові або буквені позначення. Це допомагає військовим та службам безпеки легше ідентифікувати та класифікувати боєприпаси. Індикація може включати числові, буквені або комбіновані позначення. Наприклад буквене позначення "HE" може вказувати на снаряд з вибуховою наповнювачем (High Explosive), або комбіноване позначення "M829A4" може вказувати на підтип снаряду або бойового блоку що може буде дуже важливо при ідентифікації автоматизованим роботом;

– деякі снаряди можуть мати символи, лейбли або позначки, що вказують на їхнє призначення або походження, також вони можуть мати вибиті букви або цифри, які вказують на їхнє виробництво, серійний номер або інші важливі деталі. Це може бути корисно як для військових, так і для служб безпеки для ідентифікації та класифікації боєприпасів [5].

Особливу увагу слід звертати на стан зовнішньої оболонки снаряду. Пошкодження, вибоїни, корозія або інші зміни в оболонці можуть бути ознакою небезпеки.

Візуальні ознаки протипіхотних мін можуть бути важливі для визначення їх наявності та безпечного обходження на території, де можуть розміщуватися міни. Важливо зазначити, що виявлення мін є небезпечним завданням і повинно проводитися лише спеціалізованими екіпажами та умовно безпечних умовах.

Багато протипіхотних мін мають видимі вирости або несправності на поверхні землі. Це може бути видимою металевою або пластиковою коробкою, бункером або іншими видимими деталями. Деякі міни можуть мати видимі дротові чи інші видимі конструкції, які призначені для активації та можуть бути нанесені візуальні позначення, такі як кольорові смуги, що позначають їх тип або небезпечність.

В околиці мін може бути помітний палітурний слід або сліди, які свідчать про розміщення мін. Наявність слідів від транспортних засобів, особливо на незвичайних місцях, може свідчити про можливу діяльність військ або груп, які могли розміщувати міни. Об'єкти, які намагаються замаскувати, можуть приховувати міни або інші вибухові пристрої. Місце, де розміщена міна, може мати відсутність рослинності через хімічні речовини, які розпилюються міною, або через фізичний ушкодження коріння рослин. Також, деякі сучасні протипіхотні міни можуть мати світлові індикатори або інші видимі сигнали.

Більшість протипіхотних мін активуються при натисканні на них або витяганні їх з місця розташування. Це означає, що вони можуть вибухнути, коли людина ступає на них або намагається підняти їх, але протипіхотні міни можуть бути оснащені різними видами запалів, які можуть бути відстроченими або негайними. Одним із типів запалу є запал "поштовх", який активується при натиску на корпус об’єкту [6].

Щоб ускладнити виявлення та розмінування, протипіхотні міни можуть бути розташовані в ґрунті або приховані під природніми об'єктами, такими як листя або пісок (рис. 1.2).

Рисунок 1.2 – Протипіхотна міна [7]

Авіаційні міни – це певний тип вибухонебезпечних пристроїв, які призначені для встановлення на борту літаків або вертольотів і для скидання на землю або воду з певної висоти.

Засоби активації авіаційних мін можуть включати в себе датчики тиску, температурні датчики, альтиметри, гіроскопи і інші елементи, що реагують на зміни у середовищі. Через це активувати снаряд може тиск, висота, звукові сигнали, температура та інші фактори. Точний механізм детонації може варіюватися від моделі до моделі. Вони можуть бути масковані або фарбовані відповідно до кольору середовища, в якому вони призначені для використання.

Зазвичай безпечна дистанція це може бути від декількох сотень метрів до кількох кілометрів. Знання безпечної дистанції важливо для забезпечення безпеки. Це особисто важливо при ідентифікації автоматизованим роботом,

бо після ідентифікації типу об’єкту та відмічання на мапі для розмінування сапери матимуть змогу краще підготуватися до розмінування та зменшити ризик для життя [8].

На рисунку 1.3 зображені снаряди авіаційних мін та гранат які були знайдені у польових умовах.

Рисунок 1.3 – Авіаційні міни [9]

Протитанкові міни – різновид технічної міни, спеціально призначеної для знищення бронетехніки, у тому числі танків, бронетранспортерів та іншої броньованої техніки. Вони призначені для пошкодження або знищення бронетехніки шляхом проникнення в броню та викликання вибуху. Безпечна відстань до від них може коливатися від кількох метрів до кількох десятків метрів.

Протитанкові міни призначені для нанесення ушкоджень транспортним засобам, включаючи танки і бронетранспортери. Спосіб нанесення ушкодження може включати вибухи, які порушують броню та наносять

пошкодження техніці, або проникають в неї з великим пошкодженням. Протитанкові міни мають механізм схожий до авіаційних снарядів.

Щоб ускладнити їх виявлення і зруйнування, протитанкові міни можуть бути приховані під землею, після чого вони стають невидимими для ворожої бронетехніки (рис. 1.4). Деякі міни можуть мати проти запобіжний механізм, який робить їх менш надійними для дезактивації або обходу [10].

Рисунок 1.4 – Протитанкова міна замаскована у листві [11]

Важливо зауважити, що використання протитанкових мін, особливо в цивільних конфліктах, може призвести до значних гуманітарних наслідків, так як вони можуть завдати шкоди не тільки військовій техніці, але й цивільному населенню та інфраструктурі. Тому в міжнародних конфліктах існують обмеження на використання протитанкових мін, і проводяться роботи з їх розмінювання та усуненням після закінчення конфлікту.

Протипіхотні міни спеціалізуються на ураженні живих об’єктів та чутливі до тиску. Також вони зазвичай оснащені детонаторами, які активуються навіть при легкому тиску на корпус. Зазвичай призначені для розміщення над або під поверхнею землі, тому їх важко помітити.

Міни можуть мати різні типи спускових механізмів, такі як тиснення, натискання або переміщення, які активуються після певного впливу. Також деякі протипіхотні міни можуть мати захисні механізми для ускладнення їхньої дезактивації або обходу, що робить їх небезпечними для розмінування.

Міни з дистанційним спрацюванням, також відомі як міни з віддаленим вибухом, є спеціальними типами мін, які можуть бути активовані здалеку за допомогою радіо-сигналів, інфрачервоних пристроїв, датчиків руху, або інших подібних методів. Вони призначені для ураження ворожих військ, бронетехніки, або інших об'єктів без необхідності присутності людини в небезпеці ближнього контакту.

Їх основна перевага полягає в тому, що вони можуть бути активовані здалеку оператором або автоматично на підставі певних обставин, що дозволяє стримувати пересування ворожих сил і зменшує ризик для власних військ.

Касетні бомби, снаряди або міни – це вид вибухових пристроїв, які містять внутрішню касету з декількома малими підрозділами, які розсипаються по широкій площі під час вибуху. Касетні бомби призначені для покриття великої території і ураження різних цілей, таких як піхота, бронетехніка або об'єкти інфраструктури.

Вони можуть мати значну руйнівну силу і потенційно становити загрозу для цивільних осіб та мирних цілей, оскільки можуть залишити нерозірвані підрозділи, які можуть вибухнути пізніше після завершення конфлікту. Такий нерозірваний залишок може призвести до травм або смерті цивільних осіб, які ненавмисно стикаються з ним [12].

Касетні бомби мають подовжену циліндричну форму, великі габарити, в основному запускаються з літальних апаратів (рис. 1.5).

Рисунок 1.5 – Касетна бомба з відкритою серединою [13]

1.1.2 Аналіз існуючих методів та засобів ідентифікації

Сьогодні активно розвиваються та використовуються програми, за допомогою яких є можливість розпізнавати символи та образи з маленькою похибкою. Ця технологія дуже активно застосовується у виробництві та навіть у повсякденному житті. Наприклад, автомобілі Tesla мають автопілот, який використовує штучний інтелект та комп’ютерний зір, за допомогою якого ідентифікує об’єкти перед та поряд автомобіля. Такі можливості полегшують працю людини і підвищують точність та надійність різних робочих процесів завдяки виключенню із завдання людського фактору. Але навчити комп'ютер розпізнавати об'єкти не так просто. Щоб навчити комп'ютер бачити та розуміти, що знаходиться на зображенні, люди використовують різні технології.

Розпізнавання образів – це важливе завдання комп’ютерного зору, призначене для розпізнавання окремих класів візуальних об’єктів (таких як люди, тварини, автомобілі, будівлі тощо) у цифрових зображеннях, таких як фотографії чи відеоматеріали. Метою виявлення об’єктів є розробка обчислювальних моделей, які надають інформацію, необхідну для програм комп’ютерного зору.

Для розпізнавання різних типів об'єктів є традиційні та сучасні методи. Традиційні методи ідентифікації та обробки зображень зазвичай не потребують навчальної бази даних. Методи обробки зображень, як правило, не вимагають великих даних для навчання та є неконтрольованими за своєю природою. OpenCV є популярним інструментом для роботи з зображеннями який використовує традиційні методи ідентифікації.

Один із основних методів ідентифікації у OpenCV э метод Хаара (або фільтри Хаара). Він використовується для обробки зображень і відео для виявлення об'єктів на зображеннях. Він використовується в багатьох додатках, зокрема, для розпізнавання обличь на фотографіях, відеоспостереженні, системах безпеки та багатьох інших сферах.

Для ідентифікації об’єкту за допомогою даного методу виконуються наступні кроки:

– створення вектора ознак (або класифікатор), який містить шаблони, які використовуються для оцінки значення пікселів на зображенні. Шаблони можуть розпізнавати основні ознаки об’єкту, такі як границі, контури і текстури;

– побудова інтегрального зображення для оптимізації обчислень, які дозволяють швидше обчислити значення шаблонів для різних областей зображення;

– сканування зображення за допомогою шаблонів. Фільтри переміщаються по зображенню і розпізнають об'єкти;

– класифікація виділених областей за допомогою класифікатора, який може визначати, чи належить об'єкт інтересу до певного класу (наприклад, обличчя чи не обличчя);

– відображення результатів позначаються на зображенні або виводяться як результат в програмі (рис. 1.6).

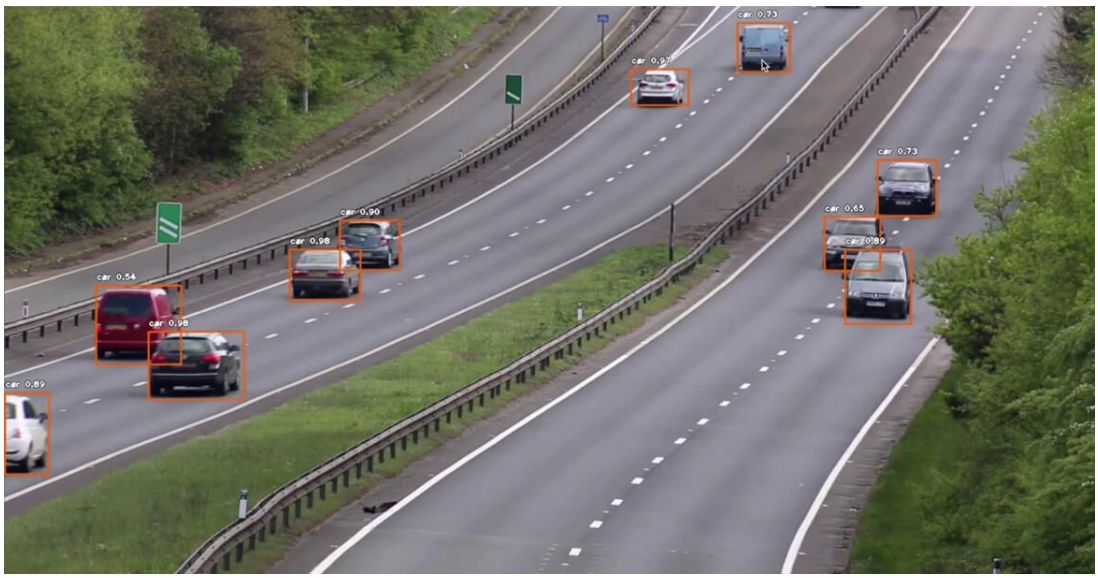


Рисунок 1.6 – Приклад результату роботи методу Хаара використовуючи OpenCV

Метод Хаара широко використовується завдяки своїй швидкості та здатності відокремлювати об'єкти від фону на зображеннях. Він є важливим елементом багатьох систем розпізнавання об’єктів, слідкування за рухом, відеоспостереження та інших додатків у сфері комп'ютерного зору.

До переваг традиційного методу відносяться відсутність необхідності підготовки великої бази зображень в якому на кожному зображенні було промарковане людиною об’єкт для ідентифікації.

До недоліки відносяться обмеження такими факторами, як складні сцени (без суцільного фону), оклюзія (частково приховані об’єкти), освітлення та тіні.

До сучасних методів відносяться штучний інтелект (ШІ) з комп’ютерним зором. Використання такої технології дуже сильно оптимізує та допомагає у дуже багатьох сферах. Наприклад, ШІ допомагає виявити дефекти на платах, корпусі та інших деталях після виробництва перед відправленням на продаж [14].

Для навчання ШІ необхідно створювати дуже великі бази даних, які містять у собі дуже велику кількість зображень. Для навчання ШІ до

середнього рівня необхідно приблизно 500 000 зображень, кожне з яких біло проанотоване в ручному режимі.

Зазвичай, після завантаження кількох наборів даних модель може неправильно розпізнати деякі об’єкти. У цьому випадку модель «перенавчається» на новий набір даних. Наприклад, якщо розглядати сферу відеоспостереження, то основою є аналіз, першим етапом якого є розпізнавання зображення (об'єкта). Потім штучний інтелект використовує машинне навчання для розпізнавання та класифікації дій.

Нейронні мережі також висувають високі вимоги до розміру та якості наборів даних, які використовуються для навчання. Набори даних можна завантажити з відкритих джерел або зібрати окремо. На практиці це означає, що до певної межі, чим більше прихованих шарів у нейронній мережі, тим точніше буде розпізнано зображення.

Для підготовки зображення ділиться на невеликі області до кількох пікселів, кожна з яких стає вхідним нейроном (рис. 1.7).

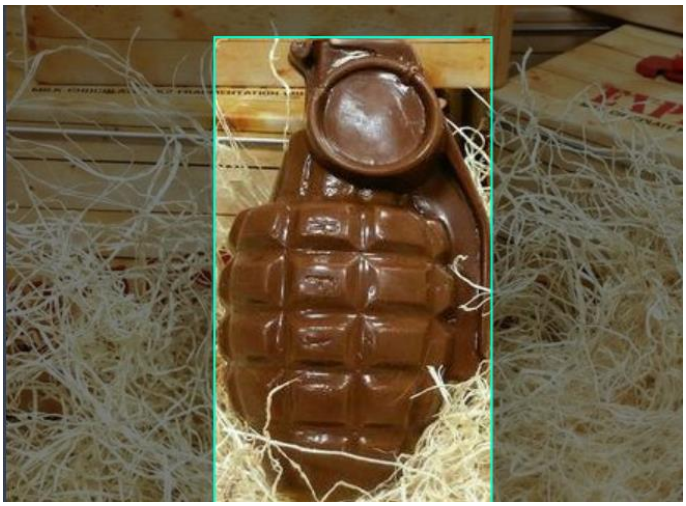


Рисунок 1.7 – Одне із зображень гранати для навчання ШІ [15]

За допомогою синапсів сигнали передаються з одного шару на інший. Сотні тисяч нейронів з мільйонами параметрів порівнюють отриманий сигнал з уже обробленими даними. Іншими словами, якщо ми просимо машину розпізнати фотографію кішки, ми розіб'ємо фото на маленькі шматочки і порівнюватимемо ці шари з мільйонами вже наявних зображень кішок, значення ознак яких мережа вивчила.

Методи глибокого навчання часто покладаються на контрольоване чи неконтрольоване навчання, оскільки контрольовані методи є стандартом для завдань комп’ютерного зору. Продуктивність обмежена обчислювальною потужністю графічних процесорів, яка з кожним роком стрімко зростає.

До плюсів сучасних методів відносяться виявлення об’єктів із глибоким навчанням є більш стійким до оклюзій, складних сцен і складного освітлення.

До недоліків відноситься потреба великої кількості навчальних даних, процес інструкцій із зображень трудомісткий і дорогий. Наприклад, користувач, який позначає 500 000 зображень для навчання алгоритму виявлення об’єктів глибокого навчання, вважається невеликим набором даних. Проте позначені дані доступні в багатьох довідкових наборах даних (MS COCO, Caltech, KITTI, PASCAL VOC, V5).

Сьогодні виявлення об’єктів глибокого навчання широко визнано дослідниками та використовується компаніями комп’ютерного зору для створення комерційних продуктів. Виявлення об'єктів використовується в інтелектуальній аналітиці відео (IVA) скрізь, де в торгових точках є камери відеоспостереження, щоб зрозуміти, як покупці взаємодіють із продуктами. Дані з відео проходять через спеціальну програму, щоб розмити обличчя людей та знеособити їх. Деякі способи використання IVA захищають конфіденційність, дивлячись лише на взуття людей, розміщуючи камери під колінами та гарантуючи, що система фіксує присутність людини без безпосереднього перегляду її особистих ознак. IVA зазвичай

використовується на заводах, в аеропортах і транспортних вузлах для відстеження довжини черг і зон обмеженого доступу [16].

Безпілотні автомобілі використовують виявлення об'єктів, щоб виявляти пішоходів, інші автомобілі та перешкоди на дорозі, щоб безпечно пересуватися. Автономні транспортні засоби, оснащені LIDAR, іноді використовують 3D-виявлення об'єктів, коли навколо об'єктів застосовуються прямокутні форми.

Хірургічне відео містить дані з високим шумом, отримані з ендоскопів під час критичних операцій. Виявлення об’єктів можна використовувати для виявлення об’єктів, які важко побачити, наприклад поліпів або уражень, які потребують негайного хірургічного втручання. Він також використовується для інформування персоналу лікарні про хід операції.

Компанії-виробники можуть використовувати функцію виявлення об’єктів для виявлення дефектів на виробничій лінії. Нейронні мережі можна навчити виявляти найменші недоліки, від зморшок на тканині до нерівностей або спалахів у формованому пластику.

На відміну від традиційних методів машинного навчання, виявлення об’єктів на основі глибокого навчання також може виявляти дефекти в дуже різних об’єктах, наприклад харчових продуктах.

Однак незважаючи на відносно високу продуктивність, ця технологія, як і раніше, стикається з такими проблемами, як різні візуальні ознаки або наявність аксесуарів, що закривають та знижують точність ідентифікації. Для підвищення точності ідентифікації, під час формування бази даних зображень, дуже часто навмисно додають елементи які можуть підвищити важкість ідентифікації об’єкту (рис. 1.8). До таких елементів відносяться програмні пошкодження зображення (візуальні фільтри, шум, плями та інші дефекти) та фізичні (маскування об’єкту або додавання зайвих елементів).

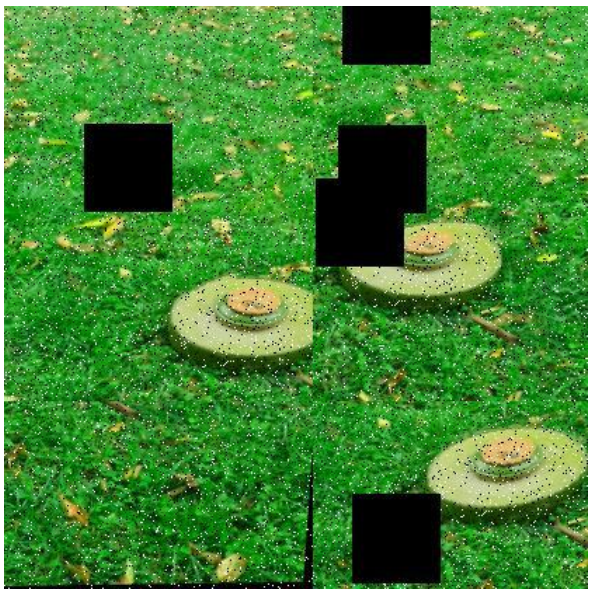


Рисунок 1.8 – Зображення з програмними дефектами із бази даних для навчання ШІ

До сучасних методів відноситься також комп’ютерний зір. Комп'ютерний зір (CV) – це науковий напрям, який визначає, як машини інтерпретують значення зображень та відео. Алгоритми комп'ютерного зору аналізують певні критерії у зображеннях та відео, а потім застосовують інтерпретації до завдань прогнозування чи прийняття рішень. Моделі комп'ютерного зору призначені для перекладу візуальних даних на основі особливостей та контекстної інформації, виявлених під час навчання.

Обробка зображень передбачає зміну або покращення зображень для отримання нових результатів. Це може включати оптимізацію яскравості чи контрасту, масштабування, розмивання конфіденційної інформації або кадрування. Різниця між обробкою зображень і комп’ютерним зором полягає в тому, що обробка зображень не обов’язково потребує розпізнавання вмісту.

Сучасні алгоритми комп’ютерного зору засновані на згорткових нейронних мережах (CNN), які забезпечують суттєве підвищення продуктивності порівняно з традиційними алгоритмами обробки зображень.

CNN – це нейронні мережі з багаторівневою архітектурою, які використовуються для поступового скорочення даних і обчислень до найбільш прийнятного набору. Потім цей набір порівнюється з відомими даними, щоб ідентифікувати або класифікувати вхідні дані. Також вона може виконувати аналіз тексту та аудіо. Однією з перших архітектур CNN була AlexNet, яка виграла конкурс візуального розпізнавання ImageNet у 2012 році.

Коли зображення обробляється CNN, кожен основний колір, який використовується в зображенні, представляється як матриця значень. Ці значення оцінюються та стискаються в 3D-тензори (у випадку кольорових зображень), які є наборами карт об’єктів, пов’язаних із частиною зображення.

Ці тензори створюються шляхом проходження зображення через ряд шарів згортки та об’єднання, які використовуються для вилучення найбільш релевантних даних із сегмента зображення та стиснення їх у репрезентативну матрицю. Цей процес повторюється кілька разів (залежно від кількості згорткових шарів в архітектурі). Остаточні характеристики, витягнуті за допомогою процесу згортки, надсилаються на повністю підключений рівень для створення прогнозів.

Існують такі архітектури глибокого навчання для комп'ютерного зору:

– AlexNet заснована на більш ранній архітектурі LeNet. Він включає в себе п’ять згорткових шарів і три повно зв’язкових шари. AlexNet використовує структуру з двома конвеєрами, що дозволяє використовувати два графічних процесори під час навчання;

– GoogleNet, також відомий як Inception V1, базується на архітектурі LeNet. Він складається з 22 шарів, які складаються з невеликих груп витків, які називаються «основними модулями». Ці модулі використовують пакетну нормалізацію та RMSprop, алгоритм, який використовує методи адаптивної швидкості навчання, щоб зменшити кількість параметрів, якими має керувати GoogleNet;

– VGG-16 є 16-рівнева архітектура (деякі варіанти мають 19 рівнів). VGGNet має згорткові шари, згорткові шари, інші згорткові шари, згорткові шари, інші згорткові шари тощо;

– ResNet (Residual Neural Network) являэться архітектурою, розробленої для великої кількості рівнів. Популярні архітектури варіюються від ResNet-18 (з 18 шарами) до ResNet-1202 (з 1202 шарами). Ці рівні налаштовані з блоками замикання або «пропуску з'єднання», які дозволяють передавати інформацію до наступних згорткових рівнів. ResNet також використовує пакетну нормалізацію для підвищення стабільності мережі;

– архітектура Xception заснована на Inception, яка замінює початкові модулі глибокими декомпозованими згортками (за глибокими згортками слідують точкові згортки). Він працює, спочатку вловлюючи кореляції між об'єктами карти, а потім просторові кореляції. Це дозволяє більш ефективно використовувати параметри моделі;

– ResNeXt-50 є модулем архітектури з 32 паралельними шляхами. Він використовує підрахунок тегів для зменшення помилок перевірки та спрощення початкових модулів, що використовуються в інших архітектурах.

Існує два поширені типи виявлення об'єктів, що виконуються за допомогою методів комп'ютерного зору.

Двоетапне виявлення об'єктів. Для першого кроку потрібна мережа пропозицій регіонів (RPN), що надає ряд регіонів-кандидатів, які можуть містити важливі об'єкти. Другим кроком є передача пропозицій регіонів до архітектури нейронної класифікації, зазвичай це алгоритм ієрархічного угруповання на основі R-CNN або об'єднання областей інтересу у Fast R-CNN. Ці підходи є досить точними, але можуть бути дуже повільними.

Одно етапне виявлення об'єктів Через необхідність виявлення об'єктів у реальному часі з'явилися архітектури одно етапного виявлення об'єктів, такі як YOLO, SSD і RetinaNet. Вони поєднують етап виявлення та

класифікації шляхом регресії передбачень обмежувальної рамки. Кожна обмежувальна рамка представлена лише декількома координатами, що спрощує поєднання етапів виявлення та класифікації та прискорює обробку.

Позиціонування зображення використовується для визначення розташування об’єктів на зображенні. Після ідентифікації об’єкти позначаються обмежувальною рамкою. Розширене виявлення об'єктів і класифікація ідентифікованих об'єктів. Цей процес базується на CNN, таких як AlexNet, Fast RCNN і Faster RCNN.

Семантична сегментація, також відома як сегментація об’єктів, подібна до виявлення об’єктів, за винятком того, що вона базується на конкретних пікселях, пов’язаних з об’єктом. Це дозволяє точніше визначити об’єкти зображення та не потребує обмежувальних рамок. Семантична сегментація зазвичай виконується за допомогою повністю згорткової мережі або мережі U.

Одним із популярних застосувань семантичної сегментації є навчання автономних транспортних засобів. За допомогою цього методу дослідники можуть використовувати зображення вулиць або проїздів із чітко визначеними межами об`єктів.

Оцінка пози – це метод, який використовується для визначення того, де знаходяться суглоби на зображенні людини або об'єкта та на що вказує розташування цих суглобів. Його можна використовувати як із 2D, так і з 3D зображеннями. Основною архітектурою, яка використовується для оцінки пози, є PoseNet, заснована на CNN.

1.2 Постановки задачі дослідження

В результаті проведеного аналізу можна зробити висновок, що використання штучного інтелекту для ідентифікації об’єктів за візуальними ознаками на сьогоднішній день являються кращім рішенням.

Метою роботи є розробка методу ідентифікації об’єкту за візуальними ознаками. Для реалізації поставленої мети необхідно вирішити наступні задачі:

– провести збір зображень для навчання моделі штучного інтелекту;

– обробити зображення додавши до них візуальні дефекти для можливості ідентифікації об’єктів у складних сценах;

– промаркувати зображення для навчання моделі штучного інтелекту;

– провести навчання моделі штучного інтелекту;

– згідно результатів навчання оновити базу зображень для навчання штучного інтелекту;

– провести розробку програмного забезпечення для ідентифікації вибухонебезпечних об’єктів за візуальними ознаками;

– провести експериментальне дослідження;

1.3 Висновки до розділу 1

У першому розділі було проаналізовано технічне завдання, а також існуючи методи ідентифікації об’єктів за візуальними ознаками. Окрім цього було описано плюси та мінуси існуючих методів ідентифікації. Проведено аналіз особливостей та візуальних ознак вибухонебезпечних об’єктів.

2 ПІДГОТОКА ОСНОВНИХ ЕЛЕМЕНТІВ СТВОРЕННЯ ПЗ ДЛЯ ІДЕНТИФІКАЦІЇ ВИБУХОНЕБЕЗПЕЧНИХ ОБ'ЄКТІВ

2.1 Розробка плану та підготовка до написання ПЗ для ідентифікації вибухонебезпечних об'єктів

Відштовхуючись від аналізу матеріалу, було прийнято рішення використовувати комбінацію мови програмування Python, OpenCV та нейронної мережі з відкритим вихідним кодом YOLO для ефективної ідентифікації вибухонебезпечних об'єктів.

OpenCV буде використовуватися для обробки та підготовки зображення для подальшої передачі до нейронної мережі. Також, він буде використовуватися для отримання зображення через підключений модуль камери для Raspberry Pi 8M (рис. 2.1) або іншої подібної моделі.

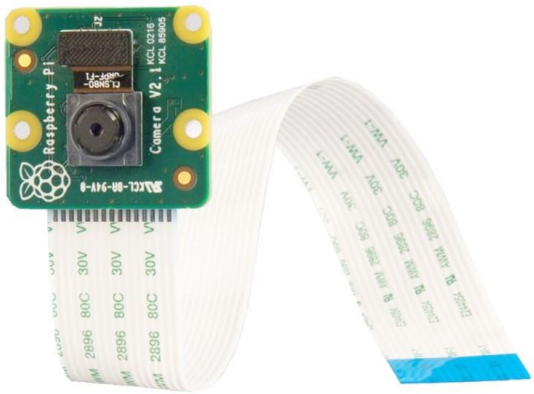


Рисунок 2.1 – Модуль камери Raspberry Pi 8MP (V2) [17]

У таблиці 2.1 наведено технічні характеристики модуля камери Raspberry Pi 8MP (V2)

Таблиця 2.1 – Характеристики модуля камери Raspberry Pi 8MP (V2) [17]

Розмір зображення	8 Мр
Максимальний розмір зображення	3280 x 2464
Режими відео	1080p, 30fps 720p, 60fps 480p, 90fps
Габарити	25 мм x 23 мм x 9 мм
Сумістність з	Raspberry Pi ASUS Tinker Board
Інтерфейс	CSI

Завдяки дуже широким можливостям OpenCV у роботі із зображеннями, при необхідності, можливо підвищувати контрастність зображення, переводити у сірі кольори, змінювати яскравість та інші параметри. Це може бути дуже важливим при умовах поганої видимості, засвітлення через різні джерела світла або погане освітлення.

Основою мого методу ідентифікації являється складний процес навчання моделі. При підготовці матеріалу для навчання, необхідно зібрати дуже велику базу зображень, до якої відносяться зображення замаскованих об’єктів, зображення з поганим освітленням, об’єкти які знаходяться на різному відстані від камери, зображення різної чіткості та зображення на яких об’єкти будуть знаходитися не у прямому зору камери, або під іншими об’єктами. Також необхідно на кожне зображення необхідно додати дефекти, порізати зображення на частини, розмазати тощо.

Також необхідно зазначити, що для ідентифікації буде достатньо зображення розміром 800x800 пікселів. Такий розмір зображення дає непогану чіткість для ідентифікації об’єкту, та за рахунок невеликого

розміру на кожне зображення, при ідентифікації на ній вибухового об’єкту, буде витрачатися небагато часу, що дозволить працювати у режимі реального часу. Але, при необхідності, можна підвищити розмір зображення до 1200x1200 пікселів. Це може підвищити відсоток ймовірності успішної ідентифікації приблизно на 10-15 відсотків, але це негативно відобразиться на оптимізації роботи у реальному часі через збільшення часу ідентифікації на кожному кадру у декілька разів. Це дозволить роботу краще ідентифікувати вибухонебезпечні об’єкти але сильно збільшить час його роботи, через що його ефективність може сильно впасти.

Наступним кроком, після підготовки зображень, є передача їх у нейронну мережу для пошуку та ідентифікації вибухонебезпечних об’єктів. Відштовхуючись від аналізу методів ідентифікації нам відомо, що нейрону мережу необхідно спочатку навчити розпізнавати різні образи та що для цього необхідно мати дуже велику базу зображень відому як Data Set (DS). Для різних нейронних мереж використовуються різні способи створення DS. Оскільки для написання даного ПЗ було вибрано нейрону мережу YOLO, то для неї необхідно підготувати матеріал для навчання.

Сам матеріал повинені містити мінімум три каталоги, перший для навчання, другий для валідації та третій для тесту після навчання, та файл data.yaml у якому будуть указані путі до цих каталогів (рис. 2.2).

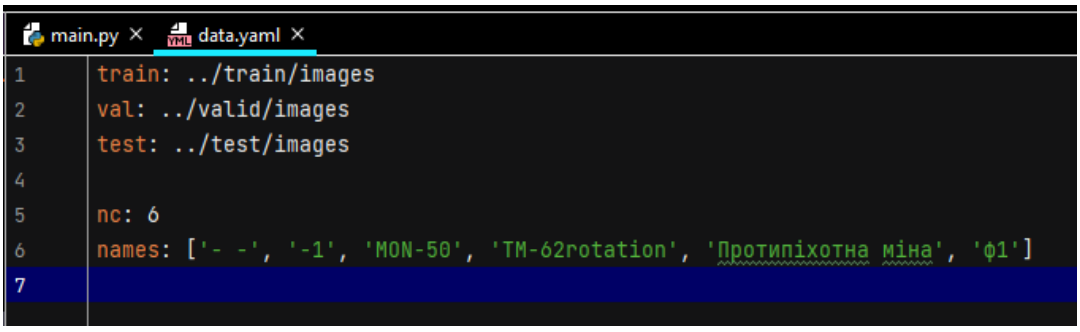


Рисунок 2.2 – Приклад файлу data.yaml для навчання нейронної мережі YOLO

У цьому файлу змінна “nc” має тип цілочисловий тип та вказує на кількість класів об’єктів для ідентифікації які знаходяться у каталогах, змінна “names” містить масив з буквеними значенням, які є назвами класів. Важно зазначити, що довжина масиву має дорівнювати значенню змінної “nc”.

Повернемося до каталогів, у кожному з трьох каталогів має бути ще два каталоги:

– “images”, для маркірованих зображень;

– “labels”, для текстових файлів кожен з котрих має точно таку назву як і зображення до якого він відноситься.

Кожен файл у каталозі labels містить у собі кількість рядків, які дорівнюють кількості об’єктів на зображенні. На початку кожного рядка указується цифрою номер класу об’єкту там чотири його координати.

Наприклад, для маркування об’єкту буде використовуватися такі дані: ”1 0.5024038461538461 0.4639423076923077 0.21995192307692307 0.20673076923076922”, де “1” є номером класу об’єкта ідентифікації, а останні чотири числа за плаваючою комою є координатами чотирьох точок рамки у якій знаходиться об’єкт.

Для маркування таких зображень, як правило, використовуються програми, які дозволяють оптимізувати процес створення маркування. Також, при остаточній генерації DS можна, та рекомендується, задати додаткові параметри, за допомогою яких зображення може бути віддзеркалено, деформовано та мати інші дефекти та особливості.

Приклад такого зображення наведено на рисунку 2.3. На цьому зображенні додано деформації зображення, зернистість та пусті ділянки. Для таких зображусь, у label файлах, буде вже не одна, а декілька строк з координатами.

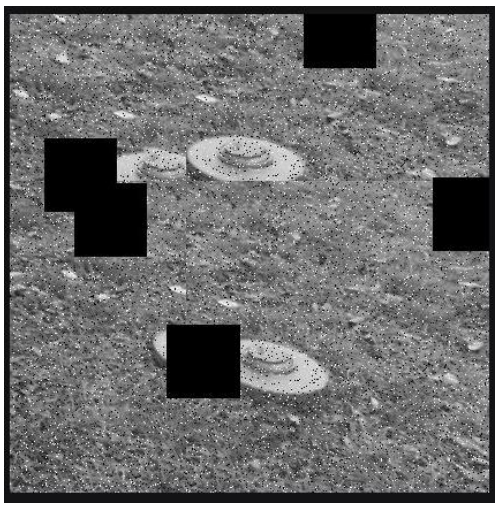


Рисунок 2.3 – Приклад одного з кінцевих зображень DS для навчання нейронної мережі

Виходячи з цієї інформації та поставлених цілей, було зібрано та маркіровано, приблизно, тисяча зображень вибухонебезпечних об’єктів. Серед цих зображень знаходяться зображення гранат, мін, снарядів та інших вибухонебезпечних об’єктів, які знаходяться на відкритих та закритих ділянках, які мають природнє та штучне маскуванням, або і зовсім його не мають. При генерації, до цих зображень було додано додаткові параметр генерації через що, на виході вже буде, приблизно, три тисячі зображень для тренування моделі нейронної мережі.

Також, було підготовлено триста зображень для закріплення навчання, та ще сто п’ятдесят зображень для тестової ідентифікації оцінки навчання.

Наступним кроком буде навчання нейронної мережі, що є дуже трудомістким процесом. При даному DS, такий процес навчання може зайняти більше тридцяти годин. Після завершення процесу навчання, нейрона мережа YOLO поверне графік навчання за яким можна буде побачити як пройшло навчання.

Останнім кроком буде написання, на мові програмування Python, бібліотеки, яка буде приймати у себе необхідні данні для підключення камери робота та датчиків, та повертати вихідні дані у яких буде міститися

інформація отримана при обробці зображення з камери. Якщо при скануванні зображення буде виявлено будь-який вибуховий об’єкт, то у вихідних даних буде міститися інформація стосовно цього. Відштовхуючись від вихідних даних, робот сапер зможе перейти в необхідний сценарій роботи та помітити на карті ділянку де було знайдено вибухонебезпечний об’єкт та, завдяки класифікації об’єктів при навчанні, він зможе ідентифікувати який само вибухонебезпечний було виявлено.

2.2 Навчання нейронної мережі

Оскільки вже було підготовлено DS для навчання, тепер необхідно вибрати модель навчання. У YOLO існує декілька варіантів. Для навчання було вирішено взяти модель “Small”. Модель “Nano” є дуже простою та потенційно матиме більший процент похибки на відмінку від “Small” чи “Medium” моделей. Різниця проценту похибки між “Small” та “Medium” не значна, але значну різницю у навчанні. Саму тому було обрано модель “Small”.

Для запуску навчання необхідно підготувати середу, куди буде встановлено необхідні бібліотеки. Для цього було створено віртуальну середу на основі Python 3.11 та створено файл requirements.txt у якому будуть лежати бібліотеки необхідні для навчання, а саме бібліотеки ultralytics, яка містить образ нейронної мережі YOLO та opencv-python. Усі наступні етапи навчання можуть бути здійснені через модуль python, який буде містити у собі необхідні шляхи до файлу моделі, DS та команди необхідні для навчання, або через термінал, що буде набагато практичніше. Оскільки на кінцевий комп’ютер можна буде передати вже навчену модель, то нема необхідності писати код, який буде навчати модель.

У терміналі, який підключений до віртуального середовища, необхідно ввести наступну команду: “yolo task=detect mode=train model=yolov8s.pt data=data/data.yaml epochs=30 imgsz=800 plots=True”, де:

– “yolo” являється виконавчим файлом;

– “task=detect” вказує но основну задачу для чого вчать модель, а саме на розпізнавання об’єктів;

– “mode=train” є режимом роботи моделі. У даному випадку навчання нейронної мережі;

– “model=yolov8s.pt” являється моделлю, яку будуть використовувати для поставлених задач. У даному випадку вказано модель “Small”. Для ідентифікації у для даної змінної буде вказано кінцеву модель після навчання;

– “data=data/data.yaml” є шляхом до основного файлу DS;

– “epochs=30” вказує на кількість епох необхідних для навчання нейронної мережі;

– “imgsz=800” задає розмір зображень на основі яких буде навчатися нейрона мережа;

– “plots=True” означає, що у кінці навчання будуть сформовані графіки, які будуть збережені у кінцевий каталог.

На рисунках 2.4 зображено кінцеві результати навчання за 30 епох та результати валідації. Час навчання нейронної мережі зайняв 42 години.

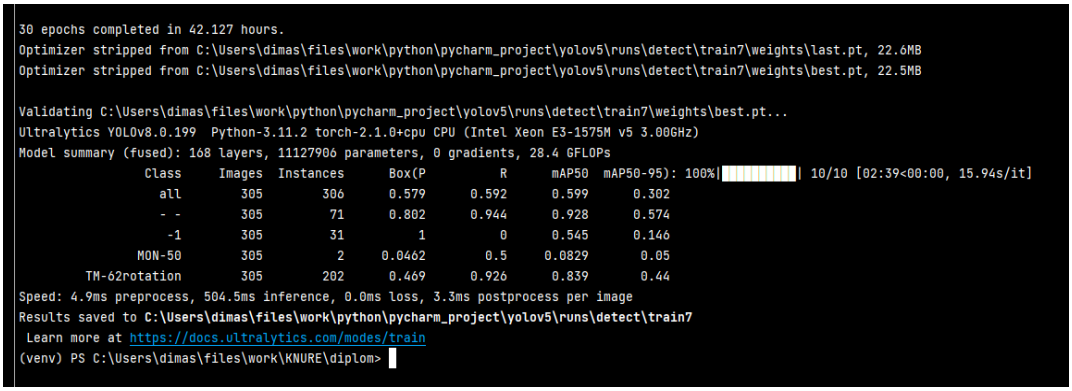


Рисунок 2.4 – Результат навчання нейронної мережі з терміналу

Оскільки у вхідних даних було зазначено, що після закінчення необхідно зберегти графіки навчання. На рисунку 2.5 зображено вихідні графіки, а саме графіки навчання та валідації.

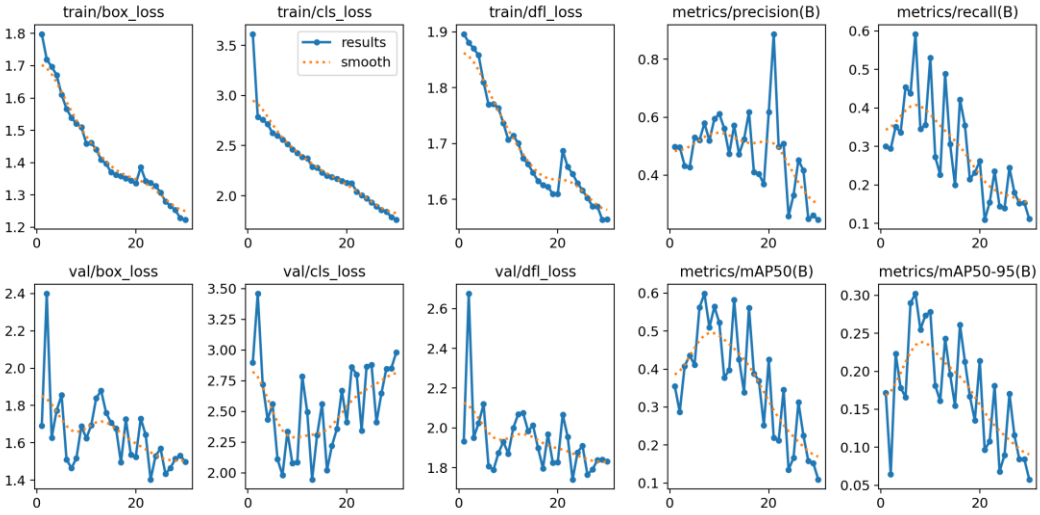


Рисунок 2.5 – Вихідні графіки після навчання нейронної мережі

З цих графіків навчання можна дізнатися, що період навчання нейронної мережі пройшов достатньо спокійно, окрім двадцятої епохи. На двадцятій епосі відбувся невеликий скачок, який міг трішки вплинути на навчання.

З графіків валідації, можна дізнатися, що після навчання, дана модель може мати багато похибок, що може вказувати на проблеми з DS, або на технічні неполадки під час навчання.

На рисунку 2.6 можна побачити зображення та як нейрона мережа їх розпізнавала під час тренування. З нього можна дізнатися, що на деяких знімках, під час навчання, нейрона мережа мала похибки та це може призвести до погіршення результатів під час пошуку вибухонебезпечних об’єктів.

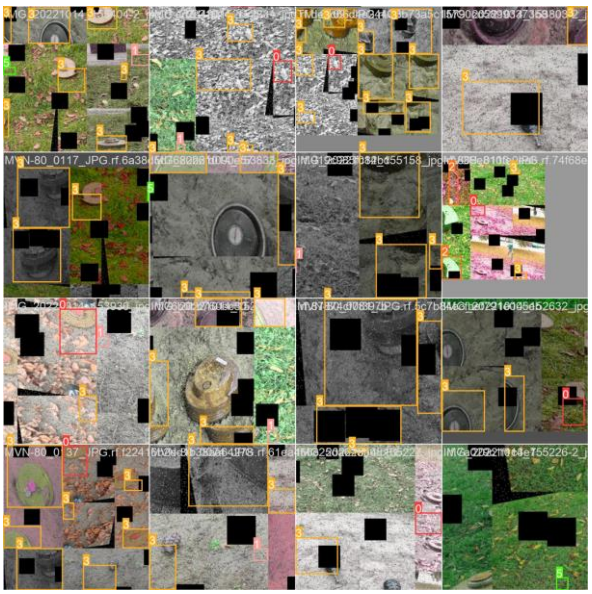


Рисунок 2.6 – Декілька зображень з навчання нейроної мережі

На рисунку 2.7 зображенно результати валідцаії. На першому зображенні продемонстровано дані з файлу label, який зберігає дані маркування. На другому зображені продемонстровано результати після валідації. Виходячи з цих результатів, можна сказати, що нейрона мережа пройшла навчання, але має невеликі похибки, які можуть не сильно вплинути на кінцевий результат ідентифікації.

Рисунок 2.7 – Зображення маркування за файлу label та результати маркування

2.3 Висновки до розділу 2

У другому розділі було проведено збір зображень для навчання моделі штучного інтелекту. Додано візуальні дефекти, порізано зображення тощо. За результатами створення бази зображень було зроблено висновок, що створення такої бази є дуже складним та довгим процесом, який потребує дуже багато матеріалу та зусиль для обробки зображень та маркування кожного з них.

Було проведено навчання штучного інтелекту та проаналізовано результати навчання.

Було проведено тестування роботи моделі штучного інтелекту на тестових зображеннях. Отримані результати даного методу навчання являються задовільними.

3 РОЗРОБКА ПРОГРАММНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВЗАЄМОДІЇ ІЗ ШТУЧНИМ ІНТЕЛЕКТОМ

3.1 Розробка алгоритму роботи

Перед початком розробки програмного забезпечення необхідно розробити його алгоритм роботи.

Оскільки на перед невідомо на якій системі буде використовуватися це програмне забезпечення, на портативних системах чи на стаціонарних машинах, то необхідно розробити бібліотеку, яку можна буде легко встановити та використовувати.

Перш за все, система на якій буде використовуватися дана бібліотека може працювати як автоматизовано, так і у ручному режимі. У обох варіантах необхідно щоб бібліотека повертала наступні данні:

- кількість ідентифікованих об’єкті;
- координати ідентифікованих об’єктів;
- назва ідентифікованого об’єкта;
- код ідентифікації назви об’єкту.

Припустимо, що автоматизована система, де буде використовуватися ця бібліотека, матиме ручне керування. В такій ситуації необхідно щоб також виводилося зображення на якому будуть відображатися області з ідентифікованими об’єктами. Для цього буде необхідно разом з вихідними даними повертати дані зображення в які вже буде додано область з назвою ідентифікованого об’єкту.

Також, треба враховувати один дуже важливий момент. У першому розділі було сказано, що для навчання моделі штучного інтелекту необхідно мати дуже великий збірник маркованих зображень. Оскільки ці моделі необхідно тестувати та за необхідністю оновлювати їх, необхідно додати

можливість передавати модель штучного інтелекту при ініціалізації бібліотеки та додати функцію у яку передаватимуться вхідні дані камери. Сама функція повинна виводити зображення у реальному часі та через термінал відображати вихідні дані про ідентифіковані об’єкти.

3.2 Написання бібліотеки

Перед початком написання бібліотеки, необхідно розробити її структуру враховуючи можливість адаптації або модифікації. Для цього буде використовуватися структура, яка буде мати клас Director, який при ініціалізації буде приймати в себе шлях до моделі, клас (ImageWoker) для роботи із зображенням та додаткові параметри для нього.

Основним класом для роботи з зображенням буде ImageWorker. Цей клас буде ініціалізуватися за допомогою класу Director, ідентифікувати об’єкт за допомогою моделі штучного інтелекту за модифікувати зображення використовуючи додаткові дані.

Як вже було сказано, класи бібліотеки повинні мати можливість модифікації. Тому необхідно створити абстрактний клас від якого успадковуватимуться усі класи.

Для написання цього програмного забезпечення необхідно буде використовувати стороні бібліотеки. За правилами оформлення проектів на мові програмування Python, необхідно створити файл requirements.txt у якому будуть вказані бібліотеки які необхідні для роботи цього програмного забезпечення (рис. 3.1).

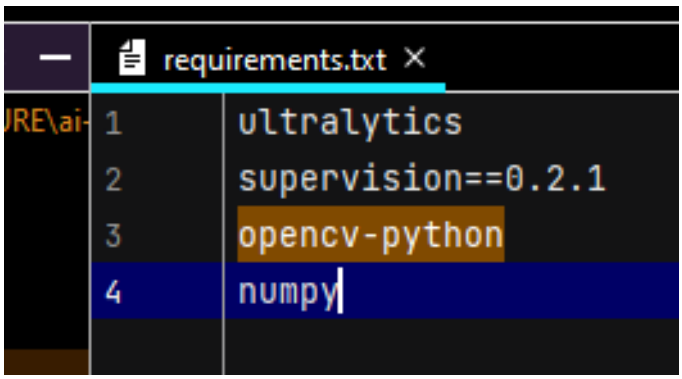


Рисунок 3.1 – Файл requirements.txt зі списком бібліотек

На рисунку 3.2 наведено абстрактний та успадкований від нього клас Director.

```python
class AbsDirector(ABC):
    def __init__(self, yolo_model_path: str, image_worker=ImageWorker, **kwargs):
        self.model = YOLO(yolo_model_path)
        self.image_worker = image_worker(self.model, **kwargs)

    @abstractmethod
    def get_detected_data(self, image: Union[str, np.ndarray]):
        raise NotImplemented()


class Director(AbsDirector):
    def get_detected_data(self, image: Union[str, np.ndarray]):
        return self.image_worker.detect_objects(image)

    def get_detected_data_and_image(self, image: Union[str, np.ndarray]):
        return self.image_worker.get_detected_data_with_image(image)

    def real_time_detect(self, cap):
        self.image_worker.real_time_detect(cap)
```

Рисунок 3.2 – Класс AbsDirector та Director

Клас AbsDirector приймає в себе модель штучного інтелекту yolo та клас ImageWorker.

Клас Director успадковується від класу AbsDirector та має два основних методи:

– get_detected_data використовується для отримання лише вихідних

даних без зображення;

– get_detected_data_and_image використовується для отримання вихідних даних та зображення на якому відображені ідентифіковані об’єкти;

– real_time_detect використовується для тестування роботи моделі або для роботи бібліотеки з відображенням зображення через додаткове вікно використовуючи бібліотеку python-opencv.

Наступним кроком буде написане класу ImageWorker. Цей клас при ініціалізації буде приймати в себе модель, параметр який вказує на необхідність модифікації зображення через погане освітлення, коефіцієнт для підвищення яскравості та роздільна здатність зображення (рис. 3.3).

```
12  class ImageWorker:
13      def __init__(
14          self,
15          model: YOLO,
16          increase_brightness: bool = False,
17          clip_limit: float = 4.0,
18          resolution: Optional[List] = None,
19          **kwargs
20      ):
21          self.model = model
22          self.clip_limit = clip_limit
23          self.increase_brightness = increase_brightness
24          self.resolution = resolution if resolution else [1280, 720]
25
```

Рисунок 3.3 – Клас для ініціалізації класу ImageWorker

Останні два параметри являються не важливими, тому їх необхідно передавати лише при необхідності.

За правилами написання коду не має бути повторного коду. Через це, все спільні елементи коду будуть винесені у функції та викликатимуться у необхідних ділянках коду.

Функція get_detect_result за допомогою моделі буде шукати на зображенні об’єкти для ідентифікації та повертати сирі дані. Для обробки

цих даних буде використовуватися функція get_result_data. В першу чергу ця функція буде перевіряти скільки об’єктів було знайдено та якщо є хоча б один ідентифікованій об’єкт, то почнеться робота циклу для розбору сирих даних у рамках якого буде сформовано словник з такими ключами:

– name для імені ідентифікованого об’єкту;

– name_code для вказання ідентифікатора імені об’єкту;

– xywh_cords для вказання однієї з вершин зони ідентифікованого об’єкту, довжини та висоти сторін зони;

– xyxy_cords для вказання двох вершин зони ідентифікації об’єкту.

На рисунку 3.4 показані ці дві функції.

```
36      def get_detect_result(self, image: Union[str, np.ndarray]):
37          return self.model.predict(image)[0]
38
39      @staticmethod
40      def get_result_data(result):
41          return_data = []
42          detected_objects_num = 0
43          result_len = len(result.boxes)
44
45          if result_len != 0:
46              detected_objects_num = result_len
47
48              for res in result[0]:
49                  res = res.boxes
50                  object_data = {'xywh_cords': [], 'xyxy_cords': []}
51                  name_code = int(res.cls.item())
52                  object_data['name_code'] = name_code
53                  object_data['name'] = result.names[name_code]
54
55                  for xywh_cords in res.xywh:
56                      for cord in xywh_cords:
57                          object_data['xywh_cords'].append(cord.item())
58
59                  for xyxy_cords in res.xyxy:
60                      for cord in xyxy_cords:
61                          object_data['xyxy_cords'].append(cord.item())
62
63                  return_data.append(object_data)
64          return detected_objects_num, return_data
65
```

Рисунок 3.4 – Функції get_detect_result та get_result_data

Наступним кроком необхідно створити функцію для підвищення яскравості зображення. Для цього буде використовуватися функціонал бібліотек python-opencv та numpy.

За допомогою першої бібліотеки зображення буде конвертуватися у інший спектр кольорів, витягуватиметься параметр який відповідає за яскравість та оновлювати его підвищуючи яскравість зображення.

Numpy використовують для роботи з масивом даних та opencv. Оскільки зображення складається пікселів, кожний з яких несе в собі три цифри які відповідають за червоний, зелений та синій кольори, то ця бібліотека буде використовуватися за роботу за цим масивом даних.

На рисунку 3.5 показана функція, яка приймає в себе зображення та, використовуючи класову зміну clip_limit, підвищує її яскравість.

```
149    def increase_image_brightness(self, image: np.ndarray):
150        clahe = cv2.createCLAHE(clipLimit=self.clip_limit, tileGridSize=(8, 8))
151        lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
152        l, a, b = cv2.split(lab)
153        l2 = clahe.apply(l)
154
155        lab = cv2.merge((l2, a, b))
156        return cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
157
```

Рисунок 3.5 – Функція increase_image_brightness для підвищення яскравості зображення

При використані даної бібліотеки може виникнути необхідність ідентифікації об’єкту на зображені. Також, це буде дуже зручно при тестуванні роботи моделі штучного інтелекту. Через це необхідно створити функції для отримання зображення використовуючи шлях до нього. Після отримання зображення необхідно перевірити його стан та викликати помилку якщо масив зображення пустий повернувши у помилці дані шляху за якого програма намагалася завантажити зображення. Також, якщо при ініціалізації

класу було передана параметр що відповідає за підвищення яскравості, необхідно викликати функцію increase_image_brightness.

На рисунку 3.6 наведено код функції get_image, яка отримує зображення використовуючи шлях до нього, викликає помилку та підвищує яскравість за необхідністю.

```
158     def get_image(self, path: str) -> np.ndarray:
159         image = cv2.imread(path)
160         if image is None:
161             raise ValueError(f'Error load {path=}')
162         elif self.increase_brightness:
163             image = self.increase_image_brightness(image)
164 
165         return image
166 
```

Рисунок 3.6 – Код функції get_image для отримання зображення

Оскільки всі основні функції готові, то необхідно створити функцію, яка буде використовувати їх та повертати кількість ідентифікованих об’єктів та список зі словниками у яких будуть дані стосовно імен та зон ідентифікованих об’єктів на зображені (рис. 3.7).

```
114     def detect_objects(self, image: Union[str, np.ndarray]):
115         if isinstance(image, str):
116             image = self.get_image(image)
117 
118         detected_objects_num, return_data = self.get_result_data(self.get_detect_result(image))
119         return {'detected_objects_num': detected_objects_num, 'return_data': return_data}
120 
```

Рисунок 3.7 – Функція detect_object яка повертає дані про ідентифіковані об’єкти на зображені

Наступним кроком буде створення функцій для формування зон ідентифікованих об’єктів на зображенні. Для цього необхідно створити парсер аргументів у який через класову зміну буде отримувати дані роздільної здатності зображення та повертати їх (рис. 3.8).

```
26      def parse_arguments(self) -> argparse.Namespace:
27          parser = argparse.ArgumentParser(description='Live detect')
28          parser.add_argument(
29              '--webcam-resolution',
30              default=self.resolution,
31              nargs=2,
32              type=int
33          )
34          return parser.parse_args()
35
```

Рисунок 3.8 – Функція parse_arguments для створення аргументів

Далі необхідно створити функцію, яка буде створювати анотатор для зображення. Цей анотатор буде створювати одну фіксовану зону та багато динамічних. Статична зона буде заповнювати все зображення та у центрі буде указана кількість ідентифікованих об’єктів у цій зоні. Усі динамічні зони будуть з’являтися тільки при знаходженні об’єкта на зображені. Вони будуть за координатами обводити ділянку зображення з ідентифікованим об’єктом то вказувати ім’я ідентифікованого об’єкту (рис. 3.9). Всі імення беруться з моделі.

```python
@staticmethod
def get_annotator(args):
    zone = sv.PolygonZone(
        polygon=(ZONE_POLYGON * np.array(args.webcam_resolution)).astype(int),
        frame_resolution_wh=tuple(args.webcam_resolution)
    )
    return {
        'box': sv.BoxAnnotator(thickness=2, text_thickness=2, text_scale=1),
        'zone': zone,
        'zone_annotator': sv.PolygonZoneAnnotator(
            zone=zone,
            color=sv.Color.red(),
            thickness=2,
            text_thickness=2,
            text_scale=2
        )
    }
```

Рисунок 3.9 – Функція get_annotator для формування зон на зображенні

Далі необхідно написати функцію, яка буде додавати на зображення ділянки з ідентифікованими об’єктами (рис. 3.10).

```python
def get_annotated_frame(self, result, data, frame):
    detections = sv.Detections.from_yolov8(result)
    labels = [
        f"{self.model.model.names[class_id]} {confidence:0.2f}"
        for _, confidence, class_id, _
        in detections
    ]

    frame = data['box'].annotate(scene=frame, detections=detections, labels=labels)

    data['zone'].trigger(detections=detections)
    return data['zone_annotator'].annotate(scene=frame)
```

Рисунок 3.10 – Функція get_annotated_frame для додавання зон ідентифікованих об’єктів на зображення

Оскільки анотатор для зображення готовий, то тепер необхідно створити метод який повертатиме словник з кількістю ідентифікованих об’єктів, вихідними даними ідентифікованих об’єктів та анотоване зображення (рис. 3.11).

```python
def get_detected_data_with_image(self, image: Union[str, np.ndarray]):
    args = self.parse_arguments()
    if isinstance(image, str):
        image = self.get_image(image)
    elif self.increase_brightness:
        image = self.increase_image_brightness(image)

    box_data = self.get_annotator(args)

    result = self.get_detect_result(image)
    detected_objects_num, return_data = self.get_result_data(result)

    return {
        'detected_objects_num': detected_objects_num,
        'return_data': return_data,
        'frame': self.get_annotated_frame(result, box_data, image)
    }
```

Рисунок 3.11 – Функція get_detected_data_with_image яка повертає вихідні дані та анотоване зображення

Як вже було вказано раніше, для тестування та роботи у портативному режимі, необхідно додати метод, який не буде повертати вихідні дані, а буде відображати їх у терміналі та виводити анотоване зображення через python-opencv у режими реального часу.

Для роботи цього методу, функція буде приймати в себе дані відеокамери, які будуть оброблятися у циклі while. Цей цикл буде зчитувати дані з камери, при необхідності додавати яскравості зображенню, додавати на зображення анотовані ділянки та виводити їх у окреме вікно у режими реального часу. Цикл можна буде перервати за допомогою кнопки “q”.

На рисунку 3.12 показано код функції real_time_detect за циком while.

```python
def real_time_detect(self, cap):
    args = self.parse_arguments()
    frame_width, frame_height = args.webcam_resolution

    cap.set(cv2.CAP_PROP_FRAME_WIDTH, frame_width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, frame_height)

    box_data = self.get_annotator(args)

    while True:
        ret, frame = cap.read()

        if self.increase_brightness:
            frame = self.increase_image_brightness(frame)

        result = self.get_detect_result(frame)
        detected_objects_num, return_data = self.get_result_data(result)

        print(f'{detected_objects_num=}')
        print(f'{return_data=}\n')

        frame = self.get_annotated_frame(result, box_data, frame)

        cv2.imshow('test', frame)

        if cv2.waitKey(1) & 0xff == ord('q'):
            break
```

Рисунок 3.12 – Функція real_time_detect для відображення знайдених об’єктів на зображені у реальному часі

Оскільки всі класи готові для роботи бібліотеки готові, тепер необхідно написати три методу для легкого запуску цієї бібліотеки. Ці методи необхідні для разового запуску бібліотеки без необхідності ініціалізувати класи.

На рисунку 3.13 наведено три функції для запуску бібліотеки. Функція run_detect використовується для разового виводу лише кількості об’єктів та їх вихідних даних. Функція run_detect_with_image виводить вихідні дані та анотованне зображення. Функція run_real_time_detect не повертає дані, а відображає їх через термінал та формує нове вікно з відео зображенням з камери. На цьому зображені також відображається анотація.

```
31  def run_detect(data: dict, image: Union[str, np.ndarray]):
32      return Director(**data).get_detected_data(image)
33
34
35  def run_detect_with_image(data: dict, image: Union[str, np.ndarray]):
36      return Director(**data).get_detected_data_and_image(image)
37
38
39  def run_real_time_detect(data: dict, cap):
40      Director(**data).real_time_detect(cap)
41
```

Рисунок 3.13 – Функції запуску бібліотеки для ідентифікації об’єктів за візуальними ознаками

3.3 Висновки до розділу 3

У третьому розділі було розроблено бібліотеку для використання моделі штучного інтелекту. У рамках розробки було додано можливість:

– передачі модифікованого класу для обробки зображення;

– підвищення яскравості зображення;

– тестування моделей штучного інтелекту;

– отримувати лише інформацію стосовно ідентифікованих об’єктів;

– отримання інформації та їх анотованого на зображенні;

– можливість ідентифікувати об’єкту як у режимі реального часу, так і по зображенням.

4 ПРОВЕДЕННЯ ЕКСПЕРЕМЕНАЛЬНОГО ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО МЕТОДУ ІДЕНТФІКАЦІЇ

4.1 Тестування методів запуску бібліотеки

Для того щоб бути впевненим, що розроблена у минулому розділі бібліотека, потрібно провести тестування написав невелику програму. Ця програма буде симулювати роботу систему на у якій буде використовуватися програма.

В першу чергу необхідно буде провести тестування трьох методів для легкого запуску програми. Для перших двох методів необхідно написати невелику функцію с циклом for для відображення отриманих результатів (рис. 4.1).

```
data = {
    'yolo_model_path': 'C:\\Users\\dimas\\files\\work\\python\\pycharm_project\\yolov5\\runs\\detect\\train7\\weights\\best.pt',
    'resolution': [800, 800]
}


def test_lib_with_get_only_return_data():
    return_data = run_detect(
        data,
        'C:\\Users\\dimas\\files\\work\\KNURE\\diplom\\data\\test\\images\\IMG_20221014_155029_jpg.rf.cb7b7627ff918bea00c1971d78060570.jpg'
    )
    print(f'\nnumber of objects found: {return_data["detected_objects_num"]}\n')
    for r_data in return_data['return_data']:
        print(
            f'name: {r_data["name"]}\n'
            f'name id: {r_data["name_code"]}\n'
            f'xyxy cords: {r_data["xyxy_cords"]}\n'
            f'xywh cords: {r_data["xywh_cords"]}\n'
        )


if __name__ == "__main__":
    test_lib_with_get_only_return_data()
```

Рисунок 4.1 – Код для тестування першого методу запуску

За допомогою змінної data передається словник з шляхом до моделі штучного інтелекту, навчання якої було проведено у другому розділі.

На рисунку 4.2 показано результат роботи програми. З цього

результату можна дізнатися, що було знайдено один об’єкт, а саме вибухівка ТМ-62. Також, було приведено її координати.

```
0: 800x800 1 TM-62rotation, 498.0ms
Speed: 9.0ms preprocess, 498.0ms inference, 2.0ms postprocess per image at shape (1, 3, 800, 800)

number of objects found: 1

name: TM-62rotation
name id: 3
xyxy cords: [118.48002624511719, 226.24383544921875, 296.8326110839844, 329.906005859375]
xywh cords: [207.65631103515625, 278.0749206542969, 178.3525848388672, 103.66217041015625]


Process finished with exit code 0
```

Рисунок 4.2 – Результати виконання методу, який повертає лише вихідні дані

Другий метод запуску повинен повертати вихідні дані та зображення у форматі масиву numpy. Для того щоб відобразити цей масив даних у вигляді зображення, необхідно модифікувати програму додавши метод imshow з python-opencv. На рисунку 4.3 показано модифіковану функцію.

```
def test_lib_with_get_only_return_data():
    return_data = run_detect_with_image(
        data,
        'C:\\Users\\dimas\\files\\work\\KNURE\\diplom\\data\\test\\images\\IMG_20221014_155029_jpg.rf.cb7b7627ff918bea00c1971d78060570.jpg'
    )
    print(f'\nnumber of objects found: {return_data["detected_objects_num"]}\n')
    for r_data in return_data['return_data']:
        print(
            f'name: {r_data["name"]}\n'
            f'name id: {r_data["name_code"]}\n'
            f'xyxy cords: {r_data["xyxy_cords"]}\n'
            f'xywh cords: {r_data["xywh_cords"]}\n'
        )
    cv2.imshow('test_2', return_data['frame'])
    cv2.waitKey(30000)
```

Рисунок 4.3 – Модифікована функція для виконання другого методу запуску

На рисунку 4.4 можна побачити зображення сформоване від час виконання бібліотеки. Оскільки для цього тесту використовувалося зображення з минулого тесту, то вихідні дані ідентифікації будуть як і у першому тесті.

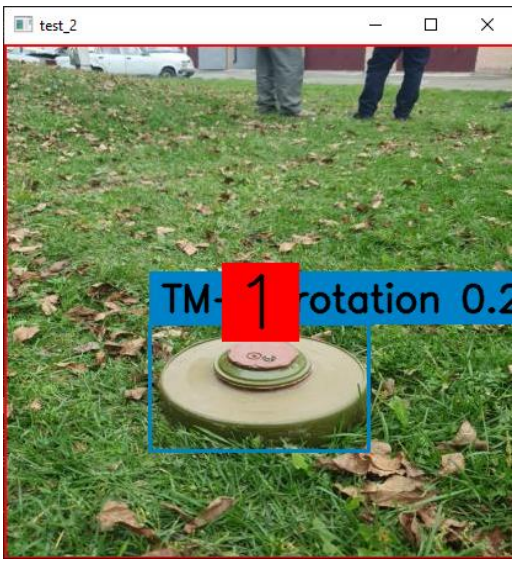


Рисунок 4.4 – Результат виконання другого методу запуску бібліотеки

За результати другого методу, можна побачити, що на кінцевому зображенні ідентифікований об’єкт виділяється синьою рамкою та підписується назва об’єкту. У центрі зображення знаходиться чорна цифра на червоному фоні. Ця цифра відображає кількість ідентифікованих об’єктів у кадрі.

Для тестування третього методу запуску необхідно лише запустити його, передати словник з моделлю та пристрій з якого будуть отримуватися відеодані, бо цей метод використовується для відображення ідентифікації у реальному часі (рис. 4.5).

```
29  def test_third_method():
30      run_real_time_detect(data, cv2.VideoCapture(0))
31
32
33  if __name__ == "__main__":
34      test_third_method()
35
```

Рисунок 4.5 – Функція для виконання третього методу запуску бібліотеки

На рисунку 4.6 можна побачити результат роботи третього методу запуску програми.

Рисунок 4.6 – Результат виконання бібліотеки з відображення ідентифікації у режимі реального часу

З результатів виконання третього методу можна побачити, що бібліотека та модель можуть ідентифікувати зображення у реальному часі використовуючи відеокамеру. Але важливо зазначити, що, через дуже складне навчання моделі штучного інтелекту, під час ідентифікації у реальному часу бувають помилкові спрацьовування системи ідентифікації. Ці помилки можуть виникати через слабку базу даних для навчання штучного інтелекту та через погане зображення. У наступному тесті буде перевірена система додавання яскравості зображення для зменшення випадкових спрацювань

4.2 Тестування роботи бібліотеки при ручному запуску програми

Основною суттю цього тесту буде симуляції роботи програми на роботі, або іншому пристрої.

Для проведення цього тесту, необхідно написати функцію, яка буде передавати у бібліотеку вхідні дані та отримувати результати у режимі реального часу. Для цього необхідно ініціалізувати клас Director. Його ініціалізацію необхідно проводити за межами циклу while, бо у циклі кожна нова ініціалізація буде забирати зайвий час.

Для демонстрації передачі класу обробки зображення та роботи підвищення яскравості, у ініціалізований клас буде передано модель та додаткові параметри. Наступним кроком буде отримання даних з відеокамери. Для цього буде створено цикл while, який буде отримувати зображення та передавати його у бібліотеку, потім вихідний результат буде повертати вихідні дані та анотоване зображення.

На рисунку 4.7 показано тестову функцію, у якій викликається метод для отримання координат та зображення використовуючи бібліотеку.

```python
def hard_test():
    director = Director(
        yolo_model_path='C:\\Users\\dimas\\files\\work\\python\\pycharm_project\\yolov5\\runs\\detect\\train7\\weights\\best.pt',
        increase_brightness=True,
        image_worker=ImageWorker,
        clip_limit=5
    )
    cap = cv2.VideoCapture(0)
    while True:
        _, frame = cap.read()

        return_data = director.get_detected_data_and_image(frame)

        print(f'\nnumber of objects found: {return_data["detected_objects_num"]}\n')
        for r_data in return_data['return_data']:
            print(
                f'name: {r_data["name"]}\n'
                f'name id: {r_data["name_code"]}\n'
                f'xyxy cords: {r_data["xyxy_cords"]}\n'
                f'xywh cords: {r_data["xywh_cords"]}\n'
            )

        cv2.imshow('test', return_data['frame'])

        if cv2.waitKey(1) & 0xff == ord('q'):
            break
```

Рисунок 4.7 – Тестова функція для симуляції роботи програми

На рисунку 4.8 показано результати ідентифікації у терміналі, які будуть виводитися до термінала за допомогою функції print та f-строки для форматування тексту.

```
0: 608x800 (no detections), 358.0ms
Speed: 4.0ms preprocess, 358.0ms inference, 1.0ms postprocess per image at shape (1, 3, 608, 800)

number of objects found: 0


0: 608x800 (no detections), 376.0ms
Speed: 4.0ms preprocess, 376.0ms inference, 1.0ms postprocess per image at shape (1, 3, 608, 800)

number of objects found: 0


0: 608x800 1 TM-62rotation, 365.0ms
Speed: 5.0ms preprocess, 365.0ms inference, 2.0ms postprocess per image at shape (1, 3, 608, 800)

number of objects found: 1

name: TM-62rotation
name id: 3
xyxy cords: [273.877197265625, 141.7575225830078, 511.748046875, 480.0]
xywh cords: [392.8126220703125, 310.8787536621094, 237.870849609375, 338.24249267578125]
```

Рисунок 4.8 – Вихідні результати роботи програми

За цих результатів можна дізнатися, що бібліотека здатна працювати при самостійному запуску програми. Також, для цього тесту було вибрано інше зображення для перевірки роботи на новому зображені.

На рисунку 4.9 показано зображення з відеокамери з підвищеною яскравістю. На цьому зображені можна побачити, що зображення стало світліше через встановленні вхідні параметри.

Рисунок 4.9 – Результат виконання роботи програми

4.3 Висновки до розділу 4

У четвертому розділі було проведено повне тестування програмного забезпечення та його роботи. Було проведено тестування роботи моделі, яку навчали у розділі 2.

У рамках тестування було перевірено:

– роботу трьох методів легкого запуску бібліотеки;

– можливість тестування моделей штучного інтелекту та відображення результаті тесту в режимі реального часу;

– роботу бібліотеки в умовах власної ініціалізації класу.

За результатами тестування, можна сказати що програмне забезпечення працює задовільно. Всі методи легкого запуску, інтеграція у сторонню програму та тестування моделей штучного інтелекту працюють задовільно.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи було обрано використовувати модель штучного інтелекту YOLO. зібрано матеріал для навчання штучного інтелекту; навчено модель штучного інтелекту; розроблено програмне забезпечення для ідентифікації об’єктів за візуальними ознаками використовуючи модель штучного інтелекту.

Під час виконання роботи було виконано наступні завдання:

– проведено аналіз існуючих методів ідентифікації;

– ретельно описано та проаналізовано принцип навчання моделі штучного інтелекту;

– зібрано базу знань для навчання моделі штучного інтелекту;

– навчено модель штучного інтелекту розпізнавати вибухонебезпечні об’єкти за візуальними ознаками;

– проаналізовано основні необхідні функції, які мають бути у програмному забезпечені;

– розроблено програмне забезпечення з усіма необхідними функціями;

– проведено тестування роботи моделі разом з програмним забезпеченням;

– проведено тестування всіх основних функцій програмного забезпечення;

– проведено аналіз результатів тестування.

За результати експериментального дослідження, можна зробити висновок, що даний метод являється працездатним та може ідентифікувати об’єкти на зображенні з дефектами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шестак Д. Д. Класифікація вибухонебезпечних об’єктів, їх візуальні ознаки, методи маскування та ідентифікації / Новоселов С. П., Шестак Д. Д. // M&MS 2023 – 2023. – С. 119-123.

2. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. – Введ. 2015-06-22. – К. Держстандарт України, 2017 – 29 с.

3. Невлюдов І. Ш. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти, спеціальності 151 Автоматизація та комп’ютерно-інтегровані технології, освітньо-професійної програми «Комп’ютеризовані та робототехнічні системи». / І. Ш. Невлюдов, Н. П. Демська, В. В. Євсєєв, Ю. М. Олександров, Р. В. Артюх, Є. А. Разумов-Фризюк, О. О. Чала – Харків: ХНУРЕ, 2021. – 51 с.

4. Артилерійські снаряди [Електронний ресурс]. – Режим доступу: https://05453.com.ua/u-bujvalivtsi-znajshly-artylerijski-snaryady/ – 21.10.2023 р. – Загол. з екрану.

5. ВИБУХОВІ БОЄПРИПАСИ ПОСІБНИК ДЛЯ УКРАЇНИ // SPORTYVG7, 2023. URL: https://sprotyvg7.com.ua/lesson/vibuxovi-boyepripasi-posibnik-dlya-ukraini (дата звернення : 24.08.2023).

6. ВИБУХОВІ БОЄПРИПАСИ ПОСІБНИК ДЛЯ УКРАЇНИ, ДРУГЕ ВИДАННЯ // GICHD, 03.07.2022. – Режим доступу: https://www.gichd.org/fileadmin/uploads/gichd/Publications/GICHD_Ukraine_Guide_2022_Second_Edition_in_Ukrainian.pdf (дата звернення : 24.08.2023).

7. Протипіхотна міна [Електронний ресурс]. – Режим доступу: https://tsn.ua/en/ato/invaders-are-mining-infrastructure-in-kyiv-and-chernihiv-region-during-the-retreat-2026375.html – 21.10.2023 р. – Загол. з екрану.

8. АВІАЦІЙНІ ЗАСОБИ УРАЖЕННЯ // О.Г. Водчиць, С.Н. Єгоров, В.М. Павільч, 2008. – Режим доступу: https://er.nau.edu.ua/bitstream/NAU/26050/3/%D0%90%D0%97%D0%A3.pdf (дата звернення : 24.08.2023).

9. Протипіхотна міна [Електронний ресурс]. – Режим доступу: https://galka.if.ua/miny-ta-aviaczijni-bomby-na-prykarpatti-mynuloyi-doby-znyshhyly-31-vybuhonebezpechnyj-predmet-foto/ – 21.10.2023 р. – Загол. з екрану.

10. МІНИ // посібник військовослужбовця ЗСУ, НГУ, ТРО України, О.Л. Дідур, М.С. Шевченко, 2023. – Режим доступу: https://shron1.chtyvo.org.ua/Didur_Oleksandr/Miny_iaki_vykorystovuiutsia_abo_mozhut_vykorystovuvatysia_viiskamy_rosiiskykh_zaharbnykiv_na_sukhopu.pdf?PHPSESSID=o34f1n93sa09vvp7vg59bf87u4 (дата звернення : 24.08.2023).

11. Протитанкова міна замаскована у листві [Електронний ресурс]. – Режим доступу: https://unn.ua/en/news/a-man-exploded-on-a-russian-mine-in-a-field-in-kherson-region – 21.10.2023 р. – Загол. з екрану.

12. МІНИ ПОСІБНИК ВІЙСЬКОВОСЛУЖБОВЦЮ ЗСУ, НГУ, ТрО УКРАЇНИ, 2-е видання, Доповнене та розширене // О.Л. Дідур, М.С. Шевченко, 2022. – Режим доступу: https://sprotyvg7.com.ua/wp-content/uploads/2023/01/%D0%9C%D1%96%D0%BD%D0%B8_%D0%9F%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%BA_%D0%B2%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%BE%D0%B2%D0%BE%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%BE%D0%B2%D1%86%D1%8E.pdf (дата звернення : 24.08.2023).

13. Касетна бомба з відкритою серединою [Електронний ресурс]. – Режим доступу: https://uanews.net/en/post/16228 – 21.10.2023 р. – Загол. з екрану.

14. OpenCV // OpenCV team, 2023. – Режим доступу: https://opencv.org/ (дата звернення : 24.08.2023).

15. Зображення гранати для навчання ШІ [Електронний ресурс]. – Режим доступу: https://www.desertcart.in/products/22978164-chocolate-grenade-full-size-mkii-solid-chocolate-hand-grenade-in-tin – 21.10.2023 р. – Загол. з екрану.

16. Deep Learning for Computer Vision // Run:ai, 2023. – Режим доступу: https://www.run.ai/guides/deep-learning-for-computer-vision (дата звернення : 24.08.2023).

17. Модуль камери Raspberry Pi 8MP (V2) [Електронний ресурс]. – Режим доступу: https://miniboard.com.ua/kamery/310-raspberry-pi-kamera-8mp-v2.html – 21.10.2023 р. – Загол. з екрану.