МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

БАРАНОВСЬКИЙ ГЛІБ ЄВГЕНОВИЧ

Допускається до захисту:

завідувач кафедри інформаційних технологій,

кандидат технічних наук

_____ О.В. Зелінська

«__» _____ 2024р.

РОЗПІЗНАВАННЯ ЗБРОЇ ТА ВІЙСЬКОВОЇ ТЕХНІКИ У ВІДЕОПОТОЦІ ЗА ДОПОМОГОЮ БІБЛІОТЕК КОМП'ЮТЕРНОГО ЗОРУ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Спеціальність 122 «Комп’ютерні науки»

Кваліфікаційна (бакалаврська) робота

Науковий керівник:

Антонов Ю.С., доцент кафедри

інформаційних технологій,

к. ф.-м. н., доцент

Оцінка __/__/_____

Голова ЕК:_____

Вінниця - 2024

АНОТАЦІЯ

Барановський Г.Є. Розпізнавання зброї та військової техніки у відеопотоці за допомогою бібліотек комп’ютерного зору у режимі реального часу.

У цій кваліфікаційній роботі було здійснено аналіз ринку фреймворків та бібліотек у сферах глибокого навчання та комп’ютерного зору. Було розглянуто методи розпізнавання об’єктів, як на основі нейронних мереж, так і без, виділено найбільш успішні. Була натренована нейронна мережа та створена програма для розпізнавання об’єктів у відео потоці в режимі реального часу.

Ключові слова: ONNX, YOLO, OpenCV, комп’ютерний зір, глибоке навчання, згорткові нейронні мережі.

Рис. 13 Бібліограф.: 43 найм.

ABSTRACT

Baranovskyi H. E. Recognition of Weapons and Military Equipment in a Video Stream Using Computer Vision Libraries in Real Time

The analysis of the market for frameworks and libraries in the fields of deep learning and computer vision was carried out. Methods of object recognition were considered, both based on neural networks and without, and the most successful ones were identified. A neural network was trained and a program was created to recognize objects in the video stream in real time.

Fig. 13, Bibliography.: 43 items.

ЗМІСТ

ВСТУП

Актуальність роботи: У сучасному світі збільшення конфліктів та терористичних загроз робить розпізнавання зброї та військової техніки в реальному часі надзвичайно важливим для забезпечення безпеки та оперативного реагування на можливі загрози. Під час військових дій чи миротворчих операцій, своєчасне і точне ідентифікування військової техніки дозволяє швидко змінювати стратегію і приймати рішення, що підвищує ефективність дій збройних сил та зменшує ризик втрат.

У контексті боротьби з тероризмом, системи розпізнавання зброї можуть використовуватися для моніторингу громадських місць, таких як аеропорти, вокзали чи масові заходи, що допомагає запобігти можливим атакам та швидко реагувати на підозрілі ситуації.

Автоматизація цього процесу за допомогою бібліотек комп'ютерного зору дозволяє значно підвищити точність і швидкість розпізнавання. Використання таких технологій також звільняє ресурси, знижуючи навантаження на персонал та покращуючи ефективність роботи систем безпеки.

Мета дослідження: розробка системи розпізнавання зброї та військової техніки у відеопотоці в режимі реального часу, використовуючи бібліотеки комп’ютерного зору.

Завдання дослідження:

- аналіз предметної області;
- аналіз бібліотек комп’ютерного зору та суміжних по предметній області;
- підбір технологічного стеку та впровадження системи розпізнавання зброї та військової техніки у відеопотоці в режимі реального часу;

Об’єкт дослідження: процеси роботи з бібліотеками комп’ютерного зору.

Предмет дослідження: системи та методи розпізнавання об’єктів комп’ютерним зором, зокрема зброї та військової техніки.

Практичне значення одержаних результатів: створення системи розпізнавання зброї та військової техніки у відеопотоці в режимі реального часу.

Структура кваліфікаційної роботи: Бакалаврська робота складається із вступу, трьох розділів та висновків до них, списку використаних джерел та одного додатку.

У першому розділі бакалаврської роботи було зроблено огляд предметної області, розглянуто різні підходи для розпізнавання об’єктів за допомогою комп’ютерного зору.

У другому розділі було поставлено вимоги та визначено технології, що будуть використовуватись при розробці системи.

У третьому розділі описано процес розробки системи, створення датасету, тренування моделі, та написання програми, що дозволить моделі обробляти відео в режимі реального часу.

Бакалаврська робота включає в себе 45 сторінок, 13 рисунків і список літератури із 43 джерел.

РОЗДІЛ 1

ОГЛЯД ТЕОРІЇ ТА МЕТОДІВ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ЗА ДОПОМОГОЮ КОМП'ЮТЕРНОГО ЗОРУ

1.1 Введення в розпізнавання об’єктів

Розпізнавання об’єктів – це задача з дисципліни комп’ютерного зору, що полягає в ідентифікації примірників семантичних об'єктів певного класу (таких як люди, будівлі чи автомобілі) у цифрових зображеннях та відео. Розпізнавання об'єктів має широкий спектр застосувань у різних сферах комп'ютерного бачення.

Наприклад, у системах безпеки та відеоспостереження ця технологія використовується для автоматичного моніторингу та аналізу відеопотоку, що дозволяє виявляти та реагувати на підозрілі дії чи об'єкти. У галузі пошуку зображень виявляння об'єктів допомагає покращити точність та швидкість знаходження необхідних зображень у великих базах даних. Крім того, ця технологія застосовується в автономних транспортних засобах для виявлення пішоходів, інших автомобілів та перешкод на дорозі, що підвищує безпеку руху. Інші сфери застосування включають робототехніку, де виявляння об'єктів допомагає роботам орієнтуватися в навколишньому середовищі та виконувати складні завдання, а також медичні діагностичні системи, де ця технологія використовується для виявлення патологій на медичних зображеннях, таких як рентгенівські знімки або МРТ.

Кожен клас об'єктів має свої ознаки, які допомагають у його класифікуванні. Наприклад, усі кола є круглими. Виявлення класу об'єктів використовує ці характерні ознаки. Наприклад, при пошуку кіл шукають об'єкти, що мають всі точки рівновіддалені від центра. Подібним чином, при пошуку квадратів шукають об'єкти з перпендикулярними кутами та однаковою довжиною сторін.

1.2 Класифікація методів розпізнавання об’єктів

Методи розпізнавання об’єктів переважно розділяють на дві категорії – з використанням нейронних мереж, та без. Розвиток нейронних мереж дав

особливо значний поштовх для цієї галузі комп’ютерного зору, але і ненейроні підходи досі використовуються у деяких випадках.

До ненейроних підходів належать: ознаки гістограм напрямлених градієнтів [1], система Віоли — Джонса виявляння об'єктів [2] та масштабо-інваріантне ознакове перетворення [3]. Варто зазначити, що дані методи не є самостійними, вони лише виявляють ознаки, ці ознаки в свою чергу використовуються для класифікації. Найпоширенішим методом класифікації на даний момент є опорновекторні машини [4, 30]. Серед нейромережних підходів також можна виділити дві категорії підходів: двоступеневі та одноступеневі, двоступеневі методи є як за звичай старішими. Прикладами двоступеневих методів є різноманітні методи пропозиції областей (R-CNN) [5], прикладом одноступеневих моделей є одноходовий багаторамковий виявляч [6]. Далі розглянемо ці методи детальніше.

1.3 Методи розпізнавання ознак для ненейроних підходів

1.3.1 Гістограми напрямлених градієнтів

Гістограма напрямлених градієнтів є одним із дескрипторів ознак, що використовується у сфері комп'ютерного зору та обробки зображень з метою виявлення об'єктів. Ця методика підраховує напрямки градієнта в локалізованих частинах зображення. Алгоритм створення гістограми напрямлених градієнтів зазвичай починається з необов'язкової глобальної нормалізації зображення, призначеної для пом'якшення ефектів освітлення. Це досягається за допомогою стиснення гамми, що передбачає обчислення квадратного кореня або логарифму для кожного кольорового каналу. Оскільки інтенсивність текстури зображення зазвичай пропорційна локальній освітленості поверхні, таке стиснення ефективно зменшує вплив локальних затінень і змін освітленості.

Після цього обчислюються градієнти зображення першого порядку для отримання контурів, силуетів і деякої інформації про текстуру, що забезпечує додаткову стійкість до змін освітленості. У процесі використовується локально домінуючий колірний канал, що забезпечує

значну колірну інваріантність. Деякі варіації методу також включають похідні зображення другого порядку, які функціонують як примітивні детектори стрижнів, корисні для ідентифікації таких структур, як стрижні в велосипедах або кінцівки у людей.

Для точного кодування вмісту зображення, зберігаючи при цьому стійкість до невеликих змін пози або зовнішнього вигляду, інформація про орієнтацію градієнта об'єднується локально, аналогічно функції SIFT, про яку буде згадано в розділі 1.3.3. Вікно зображення розділене на невеликі просторові області, відомі як "клітинки". У кожній клітинці по всіх пікселях накопичується локальна 1-мірна гістограма орієнтації градієнта або країв. Ця 1-мірна гістограма на рівні комірки формує базове представлення "гістограми орієнтації", в якому діапазон кутів градієнта ділиться на фіксовану кількість заздалегідь визначених комірок. Значення градієнта пікселів всередині комірки використовуються для відображення на гістограмі орієнтації.

Потім застосовується нормалізація шляхом відбору локальних груп клітин і нормалізації їх загальної реакції на контраст, перш ніж переходити до наступного етапу. Цей етап підвищує інваріантність до освітленості, затінення і контрасту по краях. Він передбачає накопичення "енергії" локальної гістограми в групах клітин, які називаються "блоками". Цей результат використовується для нормалізації кожної комірки в блоці. Як правило, кожна комірка розподіляється між кількома блоками, що призводить до різних результатів нормалізації для однієї комірки. Хоча така надмірність може здатися надмірною, вона значно підвищує продуктивність. Ці нормалізовані дескриптори блоків відомі як гістограма орієнтованих градієнтних дескрипторів.

Під кінець, дескриптори з усіх блоків у щільній перекриваючій сітці, що охоплює вікно виявлення, збираються в об'єднаний вектор ознак.

1.3.2 Система Віоли-Джонса виявлення об’єктів

Алгоритм виявлення об'єктів Віоли-Джонса був запропонований П. Віолою та М. Джонсом 2001 році [2]. Він переважно використовується для розпізнавання облич [7], але підходить і для інших типів об’єктів.

Алгоритм Віоли-Джонса починається з вибору гаароподібних об'єктів. Ідея гаароподібних об'єктів заснована на гаарових вейвлетах з теорії обробки сигналів [7, 8]. Ці функції є основоположними при аналізі інтенсивності зображення, що дозволяє розрізняти світлі і темні області пікселів. Процес передбачає використання вікна виявлення, як правило, прямокутного, для обчислення суми значень пікселів у певних областях зображення. Віднімаючи суму пікселів у білих областях від суми пікселів у чорних областях, алгоритм генерує значення, які підкреслюють контраст у вікні виявлення. Цей метод ефективно фіксує основні властивості виявлених об'єктів, використовуючи узгоджені характеристики інтенсивності для полегшення точної класифікації.

Для підвищення ефективності обчислень в алгоритмі використовується концепція інтегрального зображення, також відома як таблиця сумарних площ. Інтегральне зображення надає швидкий і простий спосіб обчислення значення будь-якої гаарової ознаки на зображенні. Кожна точка на інтегральному зображенні являє собою сукупну суму всіх пікселів над і зліва від даної точки на вихідному зображенні. Це попереднє обчислення дозволяє швидко розрахувати суму пікселів для будь-якої прямокутної області, значно скорочуючи кількість операцій, необхідних при виділенні об'єктів.

Алгоритм додатково підвищує точність виявлення за допомогою процесу навчання з використанням AdaBoost, або адаптивного підвищення. AdaBoost-це алгоритм машинного навчання, призначений для визначення найбільш значущих ознак із заданого набору даних. Він працює, створюючи безліч слабких класифікаторів, кожен з яких працює трохи краще, ніж випадкове вгадування. Потім ці слабкі класифікатори об'єднуються, утворюючи сильний класифікатор, який є більш точним і

надійним. На етапі навчання AdaBoost коригує вагові коефіцієнти кожного слабкого класифікатора залежно від їх продуктивності, гарантуючи, що неправильно класифікованим прикладам буде надано більше значення. Цей ітеративний процес триває до тих пір, поки не буде створено надійний класифікатор [9].

Завершальним компонентом алгоритму Віоли-Джонса є створення каскадів класифікаторів. Каскад-це послідовність класифікаторів, організованих таким чином, щоб поступово фільтрувати негативні вибірки, зберігаючи при цьому потенційні позитивні вибірки. Кожен етап каскаду містить слабкий класифікатор, навчений розпізнавати специфічні характеристики об'єкта. Коли область зображення проходить усі етапи, вона або класифікується як та, що потенційно містить об'єкт (і, таким чином, переходить до наступного етапу), або відкидається, якщо вона не відповідає критеріям на будь-якому етапі. Така ієрархічна структура дозволяє алгоритму швидко усувати області, які не містять об'єкт, концентруючи обчислювальні ресурси на областях, які з більшою ймовірністю представляють інтерес [9].

1.3.3 Масштабо-інваріантне ознакове перетворення

Масштабо-інваріантне ознакове перетворення (SIFT) – це один з алгоритмів, що використовується для виявлення та опису локальних особливостей на цифрових зображеннях [3]. Він визначає розташування певних ключових точок і потім надає їм кількісну інформацію (так звані дескриптори), яка може бути використана, наприклад, для розпізнавання об'єктів. Дескриптор SIFT інваріантний до переміщень, поворотів і масштабних перетворень в області зображення і стійкий до помірних перспективних перетворень і змін освітленості. Експериментально доведено, що дескриптор SIFT дуже корисний на практиці для зіставлення зображень і розпізнавання об'єктів в реальних умовах.

Процес роботи алгоритму такий:

Спочатку алгоритм подвоює ширину і висоту, використовуючи білінійну інтерполяцію, а потім розмиває по Гаусу.

Потім ми виконуємо серію згорток зі збільшенням стандартного відхилення. Кожне отримане зображення додатково розмивається, а потім зменшується дискретизація, починаючи новий ряд витків. Цей процес повторюється, поки зображення не стануть занадто малими. Кожен рядок, відомий як октава, представляє різні масштаби, імітуючи різні рівні спостереження та пригнічуючи дрібномасштабні структури.

Ця послідовність створює масштабний простір. Для визначення ключових точок ми використовуємо метод гауссових різниць, обчислюючи різницю в пікселях між сусідніми зображеннями. Яскраві плями вказують на збільшення яскравості, тоді як темні плями вказують на зменшення. Екстремуми цих відмінностей є потенційними ключовими точками, визначеними, коли значення пікселя більше або менше, ніж у його 26 сусідів.

Ми уточнюємо ці ключові точки, апроксимуючи квадратичне розкладання масштабно-просторової функції за Тейлором. Цей ітеративний процес підвищує точність розташування або відкидає точку, якщо уточнення не вдається. Точки на ребрах, які є менш стійкими, також відкидаються за допомогою основних кривизн масштабно-просторової функції.

Решта ключових точок отримують базову орієнтацію на основі градієнтів у їх околицях. Кожен градієнт вносить свій внесок у гістограму, яка згладжується повторним розмиттям. Екстремуми на гістограмі використовуються для визначення орієнтації. Ключові точки без достатньої кількості сусідів або домінуючої орієнтації відкидаються. Деякі ключові моменти можуть з'являтися кілька разів, і кожен з них має різну орієнтацію.

Нарешті, обчислюються дескриптори для кожної ключової точки. Аналогічно попередньому кроку, але в оберненій системі координат, ми

обчислюємо гістограми для градієнтів в круговій околиці навколо кожної ключової точки. Ці гістограми формують дескриптор, що зберігається у вигляді 128 8-розрядних цілих чисел [10].

1.4 Нейромережеві підходи до задачі розпізнавання об’єктів

1.4.1 Згорткові нейронні мережі

В основі абсолютної більшості нейромережевих підходів до розпізнавання об’єктів лежить один тип нейронних мереж, а саме згорткові нейроні мережі [15]. Згорткові нейронні мережі (ЗНМ, англ. convolutional neural networks, CNN) - це підвид глибоких нейронних мереж, побудований на основі матричної операції згортки (англ. convolution) [16]. Архітектурно, згорткові нейронні мережі складаються з трьох типів шарів, згорткових шарів, шарів вибірки (також відомих як агрегувальні шари, англ. Pool layers) та повністю з’єднані шари [18]. Хоча формально лише шари першого типу є обов’язковими, переважно, нейронні мережі містять всі три типи шарів. Згорткові нейроні мережі сприймають і обробляють дані у вигляді чотиривимірних тензорів. З вхідного зображення береться в якості параметрів ширина та довжина, та кількість каналів в якості параметра глибини [18]. Найбільш поширеним із них є RGB-кодування, в якому колір кожного пікселя на зображенні кодується за допомогою значень для трьох кольорів – червоного, зеленого та синього, але нічого в загальній архітектурі згорткової нейронної мережі не заважає використовувати інше кодування. Четвертий вимір тензора в свою чергу утворюється в ході роботи мережі і містить карти виявлених ознак зображення [20].

Розглянемо алгоритм роботи даного типу мереж. Вхідними даними для згорткового шару, як було зазначено вище, є тривимірний тензор, зчитаний з вхідного зображення, що має формат N x N x C, де N - ширина та висота зображення, а C - кількість каналів, для простору RGB параметр C має значення 3. Згортковий шар також містить K фільтрів (або ядер), розміром M x M x Q де M строго менше розміру зображення, а Q може

бути менше або бути рівним кількості каналів, значення Q досить часто варіюється в залежності від характеру фільтра. Розмір фільтрів призводить до створення локально пов'язаної структури, що є згорнута із зображенням для отримання F карт об’єктів розміром N – M + 1. Далі ці карти, як зазвичай, передаються до шару вибірки. Шари вибірки як зазвичай використовують або середнє, або максимальне значення. Кожна карта піддається вибірці на основі об’єднання безперервної області з розміром T x T, значення T переважно дорівнює 2 для відносно малих розмірів вхідних даних, та рідко переходить за 5 для великих вхідних даних. Також або до, або після шару підвибірки до кожної карти об'єктів застосовується адитивне зміщення та сигмоїдальна нелінійність [17].

Після згорткових шарів може бути будь-яка кількість повністю зв’язаних шарів. Повністю зв’язані шари є ідентичними до шарів у стандартній багатошаровій нейронній мережі.

1.4.2 Методи пропозиції областей (R-CNN)

Нейронні підходи до вирішення задачі знаходження об’єктів условно ділять на дві категорії, двоступеневі та одноступеневі. Сімейство Регіональних Згорткових Нейронних Мереж (англ. Region-Based Convolutional Neural Network, скорочено R-CNN) [5] належить саме до першої категорії і є основним серед двоступеневих підходів [11]. На основі початкового алгоритму R-CNN було зроблено багато похідних архітектур з кращою швидкодією, та витривалістю до незвичних умов (кутів, розмірів, оклюзії то що), а саме Fast R-CNN [12], Faster R-CNN [13] та Mask R-CNN [14]. Також на основі R-CNN побудована окрема гілка деформованих згортокових мереж [21, 22].

Розглянемо загальну архітектуру R-CNN. Архітектура R-CNN складається з трьох основних компонентів:

Регіональна пропозиція: R-CNN починається зі створення регіональних пропозицій, які є прямокутними обмежувальними рамками, що потенційно містять об'єкти на зображенні. Зазвичай використовується

алгоритм вибіркового пошуку, що дозволяє отримати близько 2000 пропозицій по регіонах;

Виділення ознак: для кожної пропозиції по регіону витягується відповідна область зображення і нормалізується до фіксованого розміру. Потім ці області пропускаються через згорткову нейронну мережу (CNN) для вилучення відповідних ознак;

Класифікація та локалізація: витягнуті функції з кожної регіональної пропозиції передаються через дві окремі гілки: гілку класифікації та гілку локалізації. Гілка класифікації передбачає клас об'єкта (наприклад, людина, автомобіль, кішка) для кожної пропозиції за регіоном, тоді як гілка локалізації уточнює координати обмежувального прямокутника, щоб краще охопити об'єкт;

Деталі імплементації кожного з кроків варіюються від моделі до моделі. Також варто розглянуті вище згадані напрацювання на основі початкової моделі.

Fast R-CNN: Fast R-CNN усуває вузьке місце в обчисленнях R-CNN, надаючи згорткові функції для всіх регіональних пропозицій [12]. Замість того, щоб пересилати кожну пропозицію по всьому каналу CNN, характеристики обчислюються один раз для всього зображення, а потім виділяються характеристики, що стосуються конкретного регіону, за допомогою об'єднуючого шару “регіону, що цікавить” (англ. Region of interest, скорочено ROI).

Faster R-CNN: Faster R-CNN ще більше підвищує ефективність, інтегруючи формування регіональних пропозицій в архітектуру CNN [13]. Він використовує мережу регіональних пропозицій (англ. Region proposal network, скорочено RPN), яка спільно використовує згорткові функції з мережею виявлення об'єктів, дозволяючи одночасно генерувати регіональні пропозиції та класифікувати об'єкти.

Mask R-CNN: R-FCN та Mask R-CNN розширюють рамки R-CNN для сегментації екземплярів, метою якої є ідентифікація та сегментація

окремих об'єктів на зображенні [14]. R-FCN замінює метод розсувного вікна повністю згортковою мережею (англ. Fully convolutional network, скорочено FCN) [23] для ефективного прогнозування класів об'єктів та обмежувальних рамок. Mask R-CNN надбудовується на основі R-FCN шляхом додавання гілки, яка передбачає маски об'єктів на рівні пікселів, забезпечуючи дрібнозернисту сегментацію об'єктів [14].

1.4.3 Одноходовий багаторамковий виявляч (SSD)

Одноходовий багаторамковий виявляч (англ. Single Shot MultiBox Detector, скорочено SSD) є одним з перших прикладів одноступеневих згорткових нейронних мереж [6]. На момент створення цієї архітектури, використання єдиного проходу крізь одну згорткову мережу було одним з її ключових досягнень, що відображено у назві (Single Shot). Архітектура даної згорткової нейронної мережі може бути розділена на три основні компоненти:

Базова мережа, в даній архітектурі, це зазвичай попередньо навчена згорткова нейронна мережа, подібна до VGG-16, яка служить для вилучення ознак. На ранніх шарах базової мережі відображаються об'єкти нижчого рівня, такі як краї та текстури, тоді як на глибших шарах відображаються об'єкти вищого рівня, такі як частини об'єктів та цілі об'єкти.

Допоміжні згорткові шари, ці шари додаються поверх базової мережі для створення карт об'єктів у різних масштабах. Вони поступово зменшують просторові розміри карт об'єктів, одночасно збільшуючи поле сприйняття, дозволяючи виявляти об'єкти більшого розміру.

Шари прогнозування, ці шари застосовуються до кожної карти об'єктів для прогнозування наявності об'єктів та їх розташування. Шари прогнозування діляться на шари прогнозування локації, що зміщують стандартні обмежувальні рамки таким чином, щоб вони покривали об’єкт, та шари прогнозування впевненості, що оцінюють вірогідність того, що

об’єкт знаходиться в межах рамки. Схему архітектури згорткової мережі збражено на рисунку 1.1

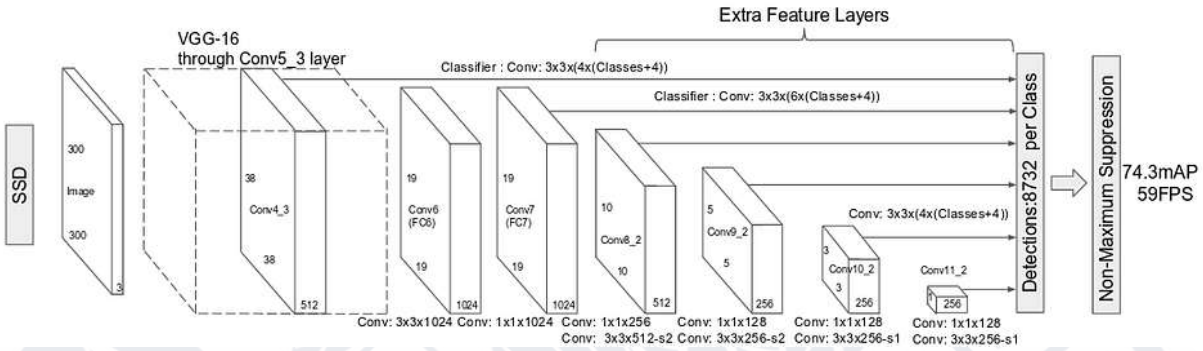

Рисунок 1.1 Схема архітектури згорткової нейронної мережі SSD [6]

Загальний алгоритм роботи моделі проходить наступним чином:

Начальна обробка вхідного зображення, що включає в себе зміну розміру вхідного зображення до стандартного, наприклад 512x512, та пропускання зображення через базову мережу, для генерації карт об’єктів.

Генерація стандартних рамок, для кожної комірки карти об'єктів визначається стандартний набір рамок з різними співвідношеннями сторін і масштабами.

Прогнозування зсувів та оцінок, за рахунок шарів прогнозування, що були описані вище.

Функція Non-Max Supression, що дозволяє усунути всі зайві рамки навколо об’єкту, і залишити лише ту, що має найвищий показник впевненості.

Також, під час навчання використовуються два додаткові кроки - розрахунок функції втрат, та обрахування відповідності рамок до справжніх рамок. Функція втрат у архітектурі SSD складається з двох компонентів - втрати локації та втрати впевненості. Для обрахування відповідності рамок використовується Intersection over Union, (скорочено IoU, також відомий як коефіцієнт Жаккара [24]).

1.4.4 RetinaNet

RetinaNet – це варіант одноступеневої згорткової нейронної мережі, що використовує фокусну функцію втрат (англ. Focal loss) та пірамідальну

архітектуру для поліпшення результатів порівняно з конкурентами [25]. Ця мережа відома хорошою роботою з дрібними об’єктами та непоганим співвідношенням точності до швидкодії. Архітектура згорткової нейронної мережі RetinaNet складається з 4 основних частин:

Backbone – основна (базова) мережа, що служить для вилучення ознак з надходящого на вхід зображення. Дана частина мережі є варіативною і в її основу можуть також входити класифікаційні нейронні мережі, такі як ResNet, VG, EfficientNet та інші;

Feature Pyramid Net (FPN) - згорткова нейронна мережа, побудована у вигляді піраміди, що служить для об'єднання достоїнств карт ознак нижніх і верхніх рівнів мережі. Нижні рівні мережі мають високу роздільну здатність, але низьку семантичну, узагальнюючу здатність, верхні ж навпаки, мають високу семантичну здатність.

Класифікаційна підмережа - підмережа, яка витягує з FPN інформацію про класи об'єктів, вирішуючи задачу класифікації.

Регресійна підмережа – підмережа, яка витягує з FPN інформацію про координати об'єктів на зображенні, вирішуючи задачу регресії.

Схема згорткової нейронної мережі зображена на рисунку 1.2

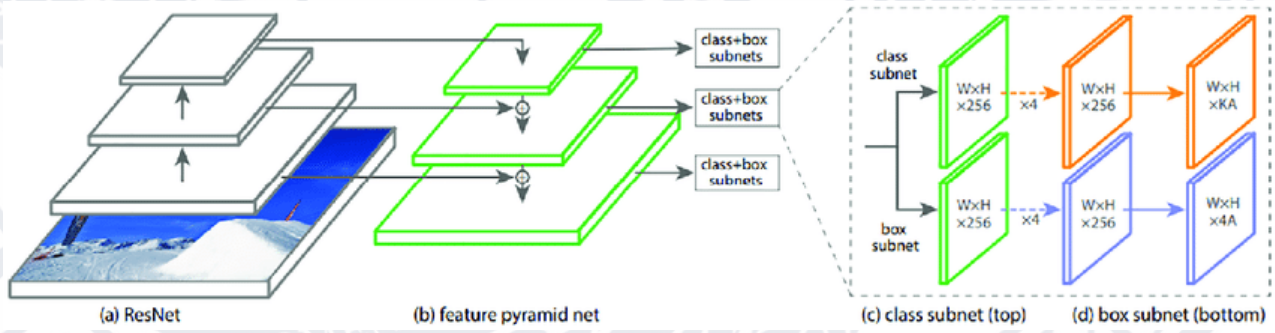

Рисунок 1.2 Схема згорткової нейронної мережі RetinaNet [25].

Найцікавішою частиною архітектури RetinaNet, тою що виділяє її фоні інших підходів є Feature Pyramid Net, тому варто зупинитись на її будові.

Feature Pyramid Network складається з трьох основних частин: висхідний шлях (bottom-up pathway), низхідний шлях (top-down pathway) і бічні з'єднання (lateral connections).

Висхідний шлях являє собою умовну ієрархічну "піраміду" – послідовність згорткових шарів з зменшуюшуюся розмірністю, які створюють backbone мережу. Верхні шари згорткової мережі мають більше семантичне значення, але менше розширення, а нижні навпаки. Bottom-up pathway має вразливість при вилученні ознак - втрата важливої інформації про об'єкт, наприклад через зашумлення невеликого, але значущого, об'єкта фоном, так як ближче до кінця мережі інформація сильно стиснута і узагальнена.

Низхідний шлях також являє собою “піраміду”. Карти ознак верхнього шару цієї піраміди мають розмір карт ознак верхнього шару bottom-up піраміди і збільшуються вдвічі методом найближчого сусіда у напрямку вниз.

Таким чином в top-down мережі кожна карта ознак вищерозміщеного шару збільшується до розмірів карти нижчого. Крім цього, в FPN присутні бічні з'єднання, це означає, що карти ознак відповідних шарів bottom-up і top-down пірамід поелементно складаються, причому карти з bottom-up проходять згортку один до одного.

Бічні з'єднання вирішують проблему загасання важливих сигналів в процесі проходу по шарах, поєднуючи семантично важливу інформацію, отриману до кінця першої піраміди і більш детальну інформацію, отриману в ній раніше.

Далі, кожен з отриманих шарів в top-down піраміді обробляється класифікаційною та регресійними мережами.

1.4.5 You Only Look Once

You Only Look Once (скр. YOLO) – це ще одне сімейство одноступеневих згортокових нейронних мереж [27, 28]. За алгоритмом роботи, You Only Look Once є подібною до SSD, але вони мають різну архітектуру мережі і тому всеж вважаються різними сімействами.

Архітектура YOLO заснована на GoogLeNet [29], вона складається з базової згорткової нейронної мережі, подібної до тих що були описанні у

розділах вище, у випадку першої версії, ця базова згорткова мережа містить 24 слої. Ці слої виділяють високорівневі ознаки. Після більшості таких шарів зазвичай ідуть шари вибірки за максимальним значенням, що збільшують глибину, але зменшують площу виводу для наступного слоя. Далі, вивід цих згорткових шарів потрапляє у повністю зв’язані шари, у першій версії таких шарів є два. При цьому останній шар має чітко визначений розмір, а саме S x S x (5B + C), де S - розмір сітки, що використовується для розподілу зображення, B - кількість рамок на одну комірку сітки, а C - кількість можливих класів об’єктів. На рисунку 1.1 зображена схема першої версії цієї згорткової нейронної мережі

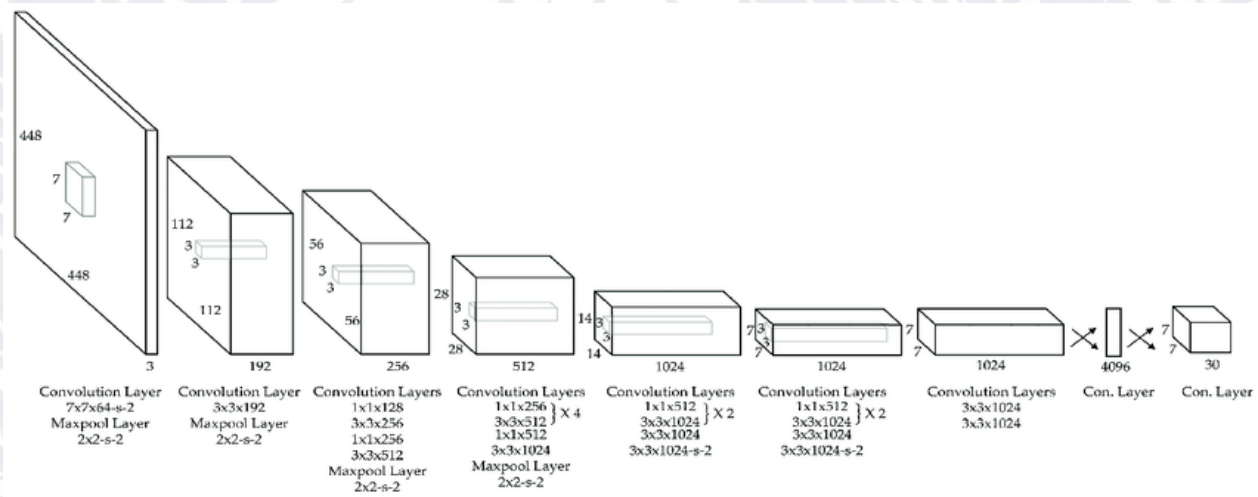

Рисунок 1.3 Схема згорткової нейронної мережі YOLO [29]

Алгоритм роботи YOLO, як було сказано вище, є подібним до SSD, але має достатню кількість відмінностей, щоб описати його окремо. Алгоритм складається з таких етапів:

Начальна обробка вхідного зображення, що включає в себе зміну розміру вхідного зображення до стандартного, та розподіл зображення за допомогою сітки S x S. Кожна комірка в цій сітці відповідає за виявлення об'єктів, центр яких знаходиться всередині комірки.

Передбачення обмежувальних рамок, кожна комірка сітки передбачає фіксовану кількість (B) обмежувальних рамок. Для кожного обмежувального прямокутника мережа передбачає: X та Y: координати центру обмежувального прямокутника щодо меж комірки сітки; W і H:

ширину та висоту обмежувальної рамки, нормалізовані по ширині і висоті зображення.

Коефіцієнт впевненності, що є мірою ймовірності наявності прямокутника, що містить об'єкт, так і точності самого обмежувального прямокутника.

Передбачення класу, кожна комірка обраховує імовірність того, що об’єкт в ній належить до класу, при чому це обраховується для кожного класу.

Об’єднання передбачень, для кожного обмежувального прямокутника остаточний прогноз передбачає поєднання ймовірностей умовного класу з оцінками достовірності, що дозволяє отримати підсумкову оцінку для кожного класу в кожному обмежувальному полі, що вказує на ймовірність присутності певного класу в прогнозованому обмежувальному полі.

Non-Max Supression, як і Single Shot Multi Box Detector, YOLO виводить більше ніж одну рамку для об’єкту і ця функція дозволяє усунути всі надлишкові рамки.

Серед особливостей архітектури You Only Look Once варто також відзначити спеціалізовану функцію втрати. Ця функція складається з трьох компонентів:

Втрата локалізації, частина функції що відповідає помилці між передбаченою обмежувальною рамкою та справжньою рамкою, заданою у навчальних даних. Формула функції виглядає наступним чином:

$$LocalizationLoss = \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(x_i - x_{ip}\right)^2 + \left(y_i - y_{ip}\right)^2 + \left(w_i - w_{ip}\right)^2 + \left(h_i - h_{ip}\right)^2 \right]$$

В цій формулі 1_{ij}^{obj} це функція індикатор, що дорівнює 1 якщо рамка j у комірці і відповідальна за передбачення об’єкта, також x_i це x координата справжньої рамки, а x_{ip} це x координата передбаченої рамки.

Втрата впевненості, частина функції, у якій вимірюється помилка в прогнозованому показнику достовірності присутності об'єкта в комірці таблиці. Формула функції виглядає наступним чином:

$$ConfidenceLoss = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left[1_{ij}^{obj} \left(C_{ip} - C_i \right)^2 + 1_{ij}^{noobj} \left(C_{ip} - C_i \right)^2 \right]$$

В цій формулі 1_{ij}^{noobj} це функція індикатор, що дорівнює 1 якщо в рамці j немає присутніх об’єктів.

Втрата ймовірності класу, частина функції, у якій вимірюється помилка в прогнозованих ймовірностях класів для кожної комірки сітки, що містить об'єкт. Формула функції виглядає наступним чином:

$$Class\ \Pr o\ babilityLoss = \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} \left(p_i(c) - p_{ip}(c) \right)^2$$

В цій формулі 1_i^{obj} це функція індикатор, що дорівнює 1 якщо комірка і містить об’єкт.

Загальна втрата є сумою цих трьох компонентів з додатковими ваговими коефіцієнтами λ_{coord} і λ_{noobj}, щоб збалансувати внесок помилок локалізації і достовірності для комірок, в яких немає об'єктів [29].

РОЗДІЛ 2
ПОСТАНОВКА ЗАДАЧІ ТА ПІДБІР ТЕХНОЛОГІЙ

2.1 Постановка задачі та вимоги до майбутньої системи

Необхідно розробити програму, що буде вирішувати задачу розпізнавання зброї та військової техніки у режимі реального часу, використовуючи один з підходів означених раніше. Загальні вимоги до програми є наступними:

- Розпізнавання потрібних об’єктів в відеопотоці довільного розміру;
- Можливість використання “живого” відео з камери пристрою, на якому запущена програма;
- Класифікація виявленої зброї та воєнної техніки;
- Створення рамок та підписів навколо знайдених об’єктів;
- Робота в режимі реального часу в умовах віртуальної машини з обмеженими ресурсами;
- Виведення моніторингу у вигляді лічильнику кадрів;

Розглянемо детальніше ці вимоги.

Робота з потоком довільного розміру є необхідною для можливості використання моделі в реальних умовах, адже підганяти розмір виводимого зображення для відеокамер може бути проблематично. Також робота з відеофайлами з довільними шириною та висотою дозволяє використовувати програму у ситуаціях, де контроль над розміром зображень не є можливий, наприклад при аналізі відео у соціальних мережах. Звісно, навіть для моделі що працює з одним єдиним розміром зображень можливо створити відповідні умови, змінюючи розмір зображення, але це в одночас є ще одним процесом, що негативно вплине на швидкодію, а також за умови зміни відношення сторін деталі можуть бути сильно спотворені, що негативно вплине на якість розпізнавання об’єктів.

Можливість використання відео з камери пристрою є необхідною, адже основною платформою для моєї програми мають бути саме прості вбудовані прилади. В таких умовах, використання камери, що є частиною пристрою значно збільшує потенційну сферу використання.

Виведення рамок та підписів на екран в першу чергу потрібно для взаємодії людини з програмою, адже без цієї функціональності значення роботи програми для людини немає. Хоча створення рамок не є потрібним для інтеграції з повністю автоматизованими системами, якщо люди присутні на будь-якій стадії, то візуалізація розпізнавання та класифікації у ввигляді рамок є необхідним. Також, для людей рамки можуть дати додаткову інформацію, яку б автоматизованому механізму було б отримати складно. Беручи приклад з минулого пункту, рамки навколо об’єктів можуть допомогти зрозуміти де саме було приховано зброю коли вона пропала з кадру.

Класифікація об’єктів є другою по важливості задачею після власне розпізнавання об’єктів для даної програми. Різні класи об’єктів вимагають різної реакції, і для більшості можливих пристосувань даної програми це є критично важливою задачею. Наприклад, інтеграція даної програми в бортовий комп’ютер дрона може як допомагати оператору для виявлення більш пріоритетних цілей, так і бути інтегрованою з системою автопілота, що може керувати дроном-камікадзе, що дозволить йому завжди обирати найкращу ціль та прямувати саме до неї. Якщо говорити про зброю, то тут класифікація також є надважливою, адже людина з гвинтівкою в недозволеному місці, наприклад аєропорті, вимагає більш інтенсивної реакції аніж людина з пістолетом. Також люди з вогнепальною зброєю та вибуховою зброєю можуть вимагати дуже різних дій, з боку, наприклад, служби безпеки аеропорта.

Віртуальна машина з обмеженими ресурсами у даній роботі це віртуальна машина з двома виділеними процесорними потоками на частоті від 3ГГц 3.5ГГц, та двома гігабайтами оперативної пам’яті. Такі обмеження було обрано для можливості використання існуючого обладнання, та щоб отримати значення швидкодії максимально наближені до реальних вбудованих систем. Це необхідно з точки зору практичного значення моделі, адже як за звичай для роботи нейромережевого розпізнавання об’єктів використовуються промислові кластери з великою кількістю відеокарт або спеціалізованих обчислювальних

модулів. Розроблена програма має здійснювати розпізнавання об’єктів на місці, без доступу до мережі.

В якості вимоги роботи в режимі реального часу було виставлено 16 кадрів на секунду, тобто програма має опрацьовувати в середньому більше ніж 16 кадрів за секунду, або рівно стільки. Таке значення частоти кадрів було обрано через те, що в середньому для сприйняття неперервності руху необхідно близько 16 кадрів на секунду [43], хоча варто зазначити, що це значення є індивідуальним, і наприклад в сфері анімації часто використовуються і нижчі значення частоти кадрів.

Для можливості моніторингу та оцінки швидкодії моделі необхідно впровадити лічильник кадрів. Лічильник кадрів має оновлятись мінімум раз в секунду, його принцип роботи має бути елементарно простим та не добавляти зайвого навантаження на систему.

Оскільки однією з основних цілей є висока швидкодія в обмежених умовах, архітектура програми має бути максимально мінімалістичною, та мати мінімум надлишкових деталей. Програма має розбивати відео на кадри, загружати модель, опрацьовувати вихідні дані з моделі, та виводити кадр з додатковою графікою. На рисунку 2.1 зображена орієнтовна схема роботи програми.

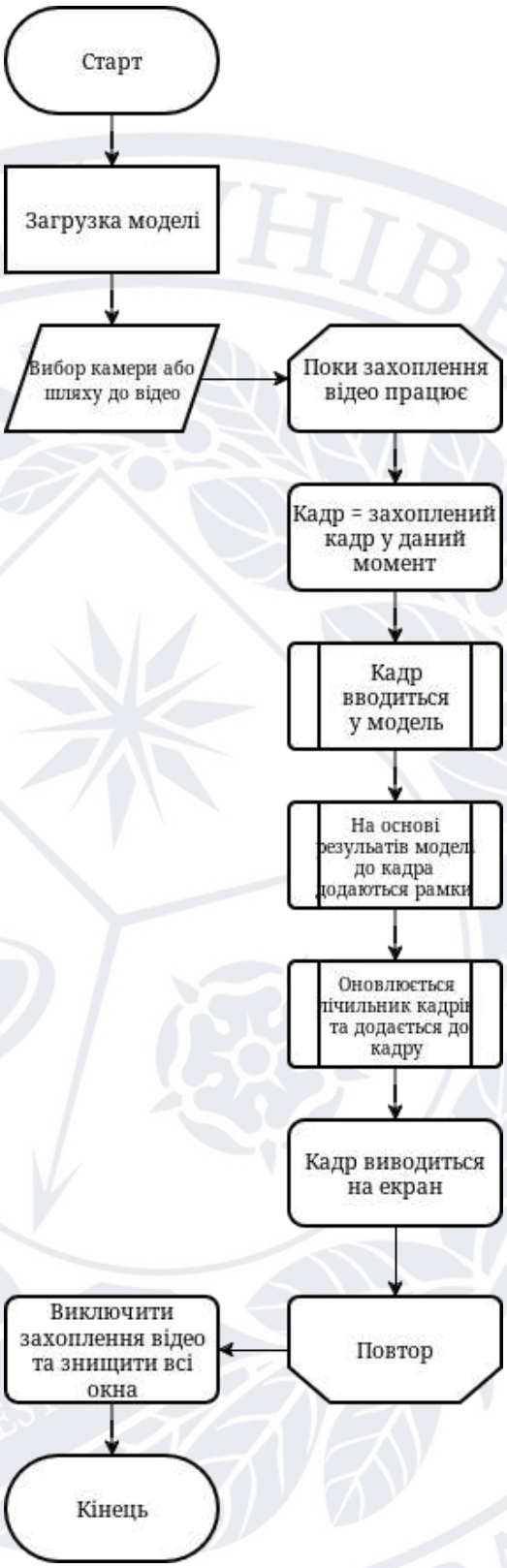

Рисунок 2.1 Орієнтовна блок-схема програми

2.2 Огляд методів виявлення об’єктів, що задовольняють вимогам

Розглянемо детальніше постановку задачі, та відсіємо методи що не задовольняють вимогам. В першу чергу варто відсіяти методи, що відомі високою обчислювальною важкістю. Серед ненейроних методів такими є метод

гістограм напрямлених градієнтів, та масштабо-інваріантне ознакове перетворення. Обчислення градієнтів, фільтрів Гауса та функцій Тейлора є загальновідомими важкими обчислювальними задачами. Серед нейромережевих підходів варто відкинути двоступеневі методи, адже статистично, вони показують гіршу швидкодію ніж одноступеневі [26]. Також на мою думку, варто відкинути систему Віоли-Джонса, адже вона слабко пристосована для роботи з усіма об’єктами, що не є обличчями, та має низьку точність якщо об’єкт є малим відносно фону. Тому серед залишившихся методів вибір стоїть між RetinaNet, You Only Look Once та Single Shot MultiBox Detector.

2.3 Огляд фреймворків, що можуть бути використані для вирішення задачі

Оскільки важливою частиною постановки задачі є робота в режимі реального часу в умовах віртуальної машини з обмеженими ресурсами, підбір фреймворка для вирішення цієї задачі є дуже важливим. Розглянемо потенційні варіанти.

PyTorch - фреймворк для глибокого навчання, розроблений дослідницькою лабораторією штучного інтелекту Facebook (FAIR) [35]. Відрізняється високою гнучкістю, є популярним вибором для досліджень та експериментів. Особливо добре підходить для навчання та створення нейронних мереж, але потребує велику кількість ресурсів для ефективної роботи.

TensorFlow - фреймворк для глибокого навчання, розроблений Google Brain [36]. Має чудову екосистему інструментів та бібліотек, досить добре підходить для навчання та створення нейронних мереж. Окремо варто відзначити існування Lite-версії, оптимізованої для роботи з низькою кількістю ресурсів.

OpenVINO - (Open Visual Inference and Neural Network Optimization) фреймворк для глибокого навчання, розроблений Intel [37]. Призначений в першу чергу для оптимізації та розгортання нейронних мереж, особливу увагу приділено апаратному забезпеченню Intel.

Caffe - (Convolutional Architecture for Fast Feature Embedding) фреймворк для глибокого навчання, розроблений Університетом Каліфорнії Берклі [38]. На

відміну від інших, цей фреймворк сфокусований на створенні та роботі зі згортковими нейронними мережами. Що правда, працездатність на слабшому залізі залишає бажати кращого, особливо в порівнянні з конкурентами.

NCNN - фреймворк для глибокого навчання, розроблений китайською компанією Tencent [39]. Основною ціллю розробки даного фреймворку є поліпшення роботи нейронних мереж на мобільних та вбудованих пристроях. Він є дуже популярним вибором при розробці мобільних додатків. З недоліків варто зазначити проблеми з документацією, адже цей фреймворк орієнтований в першу чергу на китайський ринок.

ONNX - (Open Neural Network Exchange), на відміну від рішень, які були роглянуті вище, не є повноцінним фреймворком. ONNX - це формат для представлення моделей, розроблений для портативності та конвертації між фреймворками [40]. Але з часом, цей формат отримав досить велику кількість спеціалізованого інструментарію і найголовніше, власне середовище та добре оптимізований двигун для обробки та виведення даних, що робить цей формат конкурентноздатним, навіть у порівнянні з повноцінними фреймворками.

З непов’язаних з глибоким навчанням фреймворків та бібліотек варто відзначити OpenCV, як стандарт індустрії комп’ютерного зору для роботи з зображеннями та відео. Він також містить імплементації ненейроних методів виявлення ознак розглянутих в першому розділі.

РОЗДІЛ 3
ПРОЦЕС РОЗРОБКИ СИСТЕМИ РОЗПІЗНАВАННЯ ЗБРОЇ ТА ВІЙСЬКОВОЇ ТЕХНІКИ У ВІДЕОПОТОЦІ

3.1 Обґрунтування вибору типу нейронних мереж / фреймворків

По результатам дослідження, описаного в попередніх розділах, було прийнято рішення зупинитися на даних фреймворках та моделях:

- You Only Look Once в якості загальної архітектури згорткої нейронної мережі.
- YOLOv8 Nano від ultralytics якості базової моделі, та пакет ultralytics для використання модуля Non-Max Supression.
- PyTorch як фреймворк для навчання моделі на нових даних.
- ONNX як формат збереження моделі
- ONNXRuntime як середовище та рушій для обробки даних нейронною мережею
- OpenCV для розбиття відео на кадри, та додавання графіки до зображення
- ClearML для кращих графіків моніторингу навчання моделі.

Також було використано дистрибутив мови програмування Python Anaconda та входящий до нього менеджер пакетів Conda для створення середовища виконання та тренування моделей.

3.2 Підхід до формування датасетів

Датасет було сформовано на основі 4 публічних джерел, розміщених на платформі Roboflow [31, 32, 33, 34], по два окремих джерела на зброю та техніку. Зображення з джерел були випадковим чином обрані та розділенні на три виборки, навчальну, валідаційну та тестову, у співвідношенні 75/15/10. До датасету було застосовано аугументацію даних - кожне зображення було продубльовано три рази, з випадковим обертанням в межах від -20° до +20°, випадковим нахилом від -15° до +15°, та обмежувальною рамкою. Також деякі частини близько 10% зображень було перекрито, для кращого розуміння оклюзії, та ще 25% було переведено в чорно-білу кольорову гамму, методом освітленості.

Рисунок 3.1 Приклад зображеннь використаних у датасеті

Після чого датасет було зменшено до 9872 зображень, та розмір зображень було зменшено з 640 на 640 пікселів, до 320 на 320 пікселів. Це було зроблено в цілях збільшення швидкодії моделі, а також прискорення тренування. Використання зменшених зображень в датасеті дає майже двократний прирост швидкодії, що є особливо критичним для постановки задачі [42, 26].

3.3 Створення ПЗ

Для роботи з моделлю, мною було створено програму на мові Python. Програма складається з трьох частин: менеджер детектора, допоміжних графічних засобів, та основної частини. Менеджер детектора підключає необхідні бібліотеки, загружає модель за вказаним шляхом, виконує задачу

знаходження об’єкту на зображенні, та проводить обгортку для зручнішого користування детектором. Також у цій частині містяться назви класів об’єктів, адже вони не є збереженими в моделі формату ONNX. У допоміжних графічних засобах знаходяться функції для анотації (обведенні знайдених об’єктів на зображенні та підписанні класу об’єктів), та засіб моніторингу швидкодії в реальному часі у вигляді лічильнику кадрів. В основній частині програми іде розбивка відео на кадри, викликання детектору та допоміжних графічних засобів, та виведення на екран обробленого кадру. Через аргументи командного рядка реалізована можливість тестування програми на різних відеофайлах, або підключення камери. Вихід з програми можливий при натисканні кнопки “c” на клавіатурі.

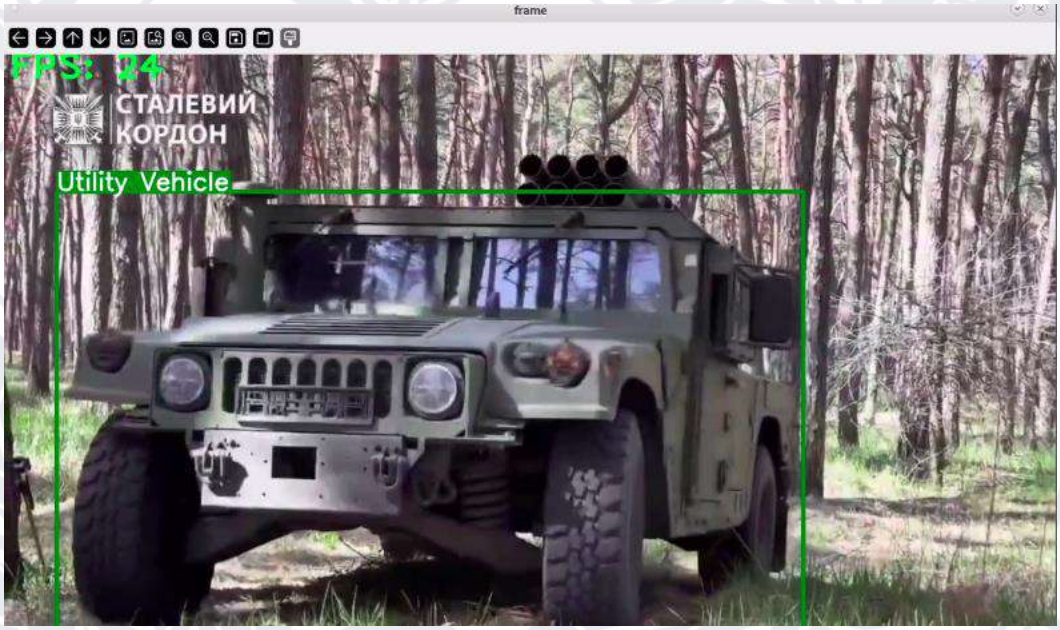

Рисунок 3.2 Вікно та результат роботи програми

3.4 Навчання та верифікація

Процес навчання моделі зайняв близько 1 годину 49 хвилин на локальному комп’ютері з відеокартою Nvidia RTX 3060, що має 12 гігабайт відеопам’яті, та 16 гігабайт оперативної пам’яті . При навчанні були використані такі параметри: кількість епох - 50, кількість семплів опрацьованих за раз - 8, та відключено кешування.

Ітогова модель має 3013383 параметрів, обчислювальна важкість її оцінена в 8.207 GFLOPs, Середня точність ітогової моделі становить 0.854932056776563, відклік моделі - 0.7863317201867596, метріка mAP50 (Average Precision з значенням Intersection over Union більше 50%) становить 0.8451325293084223.

Model summary (fused): 168 layers, 3008183 parameters, 0 gradients, 8.1 GFLOPs

Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 51%
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
all	1481	2599	0.855	0.786	0.845	0.665
Artilery	1481	76	0.93	0.947	0.945	0.833
Grenade	1481	434	0.836	0.495	0.627	0.412
Gun	1481	406	0.728	0.719	0.763	0.547
Handgun	1481	89	0.903	0.888	0.934	0.755
Heavy Tank	1481	131	0.933	0.855	0.914	0.77
Knife	1481	328	0.687	0.549	0.617	0.388
Military Truck	1481	92	0.875	0.761	0.835	0.602
Mine-Resistant Utility Vehicle	1481	66	0.811	0.894	0.925	0.776
Personnel Carrier	1481	167	0.984	0.916	0.973	0.823
Pistol	1481	345	0.752	0.702	0.739	0.543
Rifle	1481	153	0.846	0.686	0.83	0.658
Tank	1481	187	0.94	0.93	0.968	0.81
Utility Vehicle	1481	125	0.89	0.88	0.916	0.73

Рисунок 3.3 Консольне підбиття ітогів навчання

По графіку на рисунку 3.4 ми можемо бачити, що середній час тренування на епоху становив ~128.97, і за винятком двох викидів залишався досить близько до цього значення.

Згідно за графіком, вони відбулись у першу, та сорокову епохи. Якщо перша епоха зазвичай триває трохи довше аніж всі інші, то викид на сороковій епосі є цікавим, адже він також співпадає з значним падінням функцій втрат, що ми можемо спостерігати на рисунку 3.5

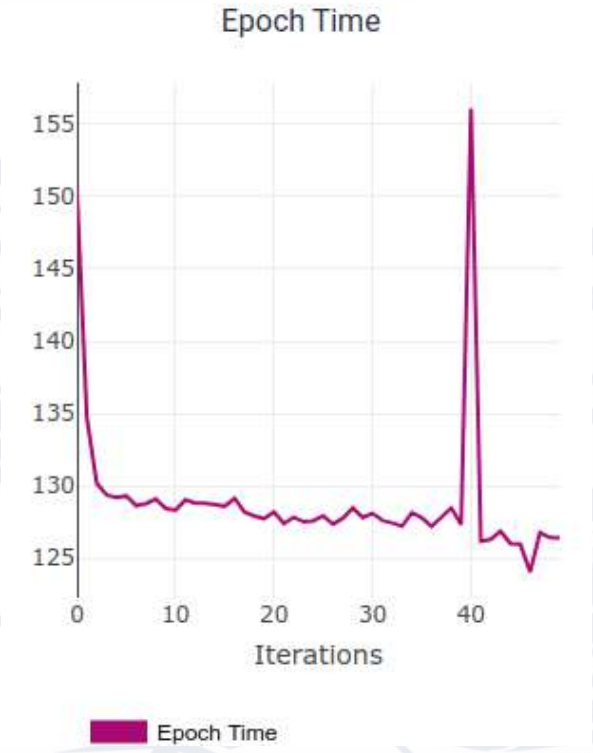

Рисунок 3.4 Графік часу проходження однієї епохи

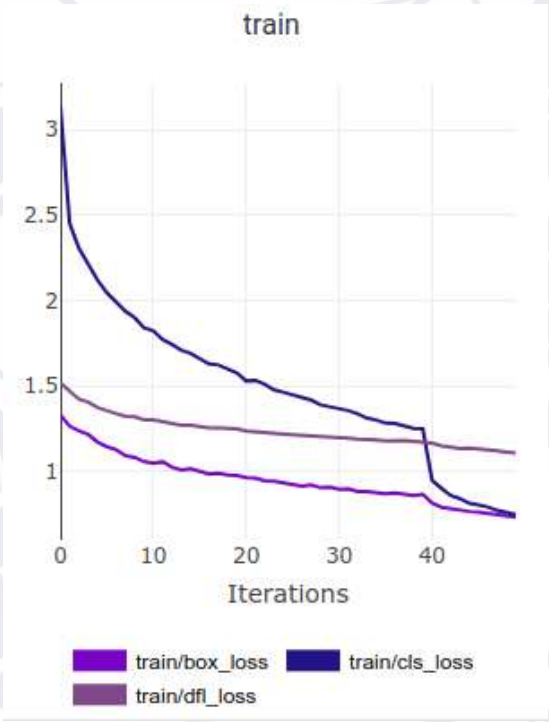

Рисунок 3.5 Графік функцій втрат

Але розглянувши більш детальні графіки на рисунках 3.6 та 3.7, ми можемо бачити що значне падіння на сороковій епосі відбулось лише відносно тренувального датасету, тобто на роботу в реальних умовах це падіння вплине

слабо.

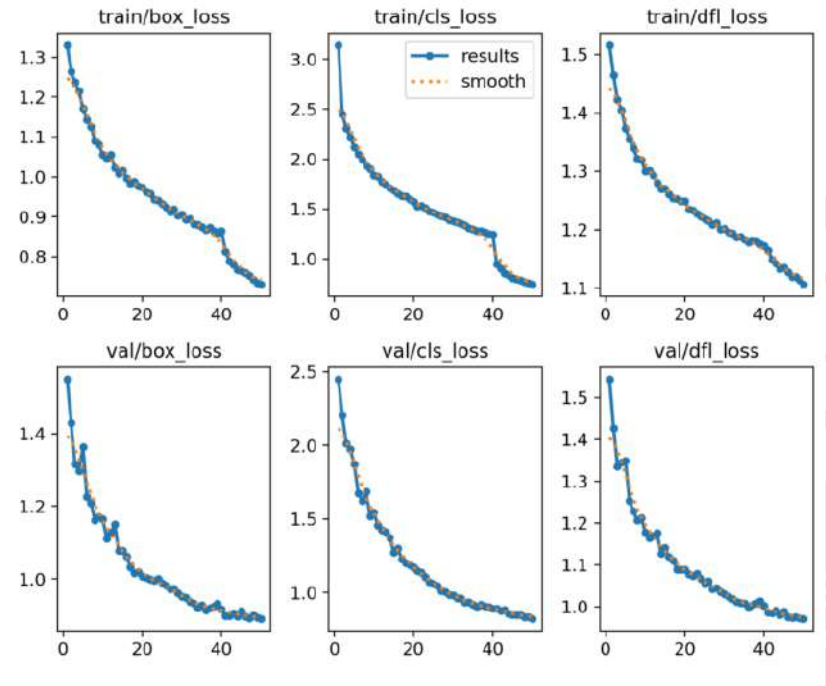

Рисунок 3.6 Графіки функцій втрат для тренувальної та оціночної вибірки.

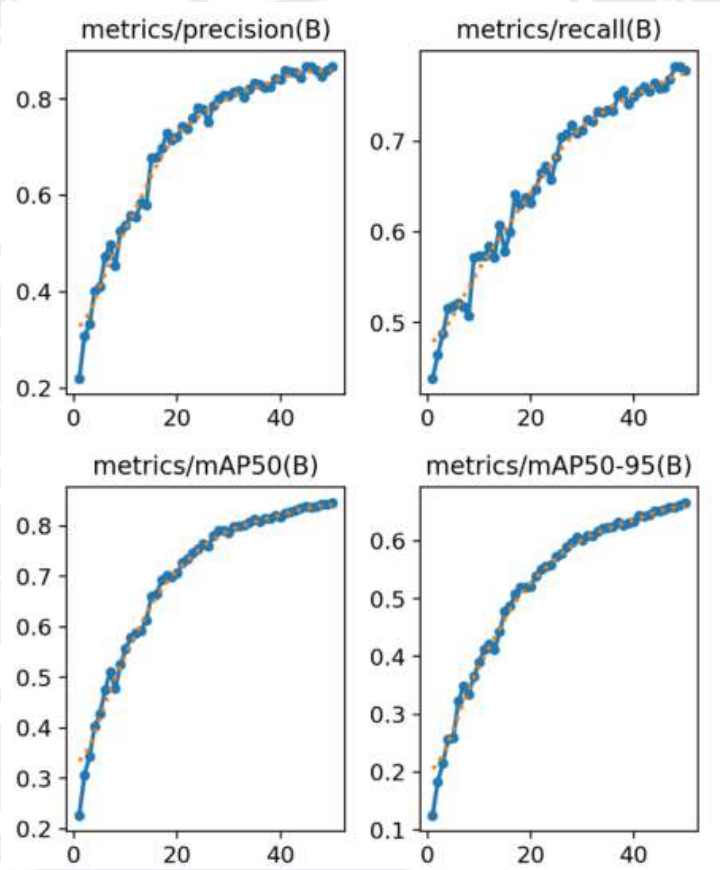

Рисунок 3.7 Детальні графіки тренувальних метрік.

Далі варто розглянути матрицю помилок, адже вона може багато сказати про характер натренованої моделі. Варто зазначити, що матриця помилок, що зображена на рисунку 3.8 є нормалізованою.

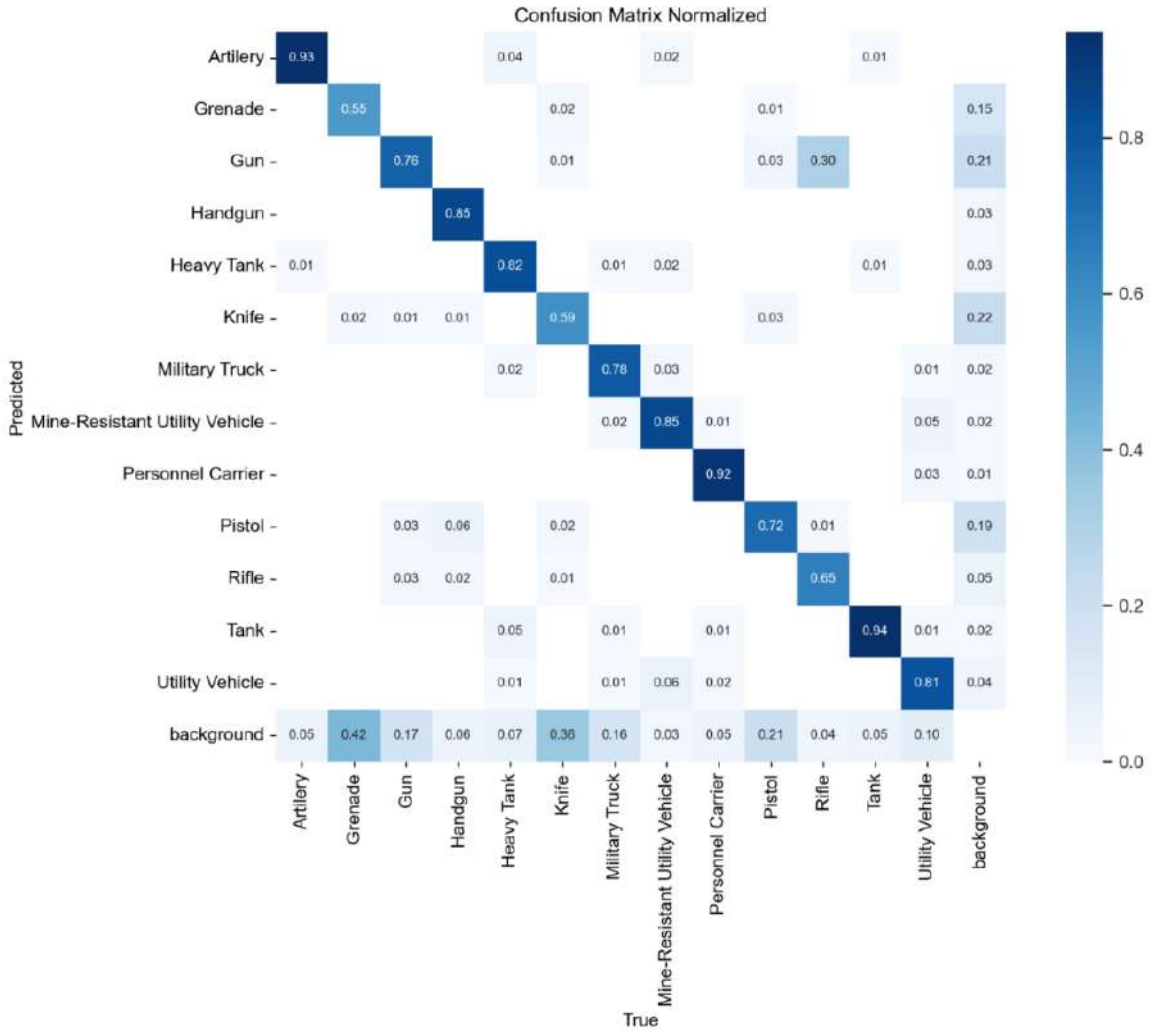

Рисунок 3.8 Матриця помилок

Аналізуючи матрицю помилок, можна помітити, що гранати є найбільш проблематичним класом для моделі, її точність у знаходженні гранат слабо відрізняється від підкидання монети. Також схожу проблему мають ножі, отже має сенс зробити висновок, що отримана модель гірше працює з дрібними об’єктами. Також, може здатись, що модель має проблеми і з класифікацією гвинтівок, адже значення розпізнавання в матриці доволі низьке - 0.65. Але це є меншою проблемою, аніж могло би здаватись на перший погляд, адже основний клас з яким плутаються гвинтівки це рушниці, а не фон, як було у випадку з

гранатами та ножами. Відносно високе значення плутанини між рушницями та гвинтівками може бути пояснено також недостатньою якістю тренувальної розмітки та класифікації, адже і люди також досить часто не до кінця розрізняють ці категорії. Останній аргумент також може бути використаний з іншої сторони, бо раз класифікація рушниць та гвинтівок не до кінця зрозуміла навіть багатьом людям, то чого чекати від нейронної мережі?

Що до решти категорій, модель показує себе справно. Відсоток правильних відповідей у багатьох інших категорій становить близька 90%.

Також цікавим графіком є графік відношення F1-Confidence. Міра F1 відображає гармонічне середнє між Precision та Recall (точність та відкликання) [41], її співвідношення до впевненості може багато показати про модель. Цей графік показано на рисунку 3.9.

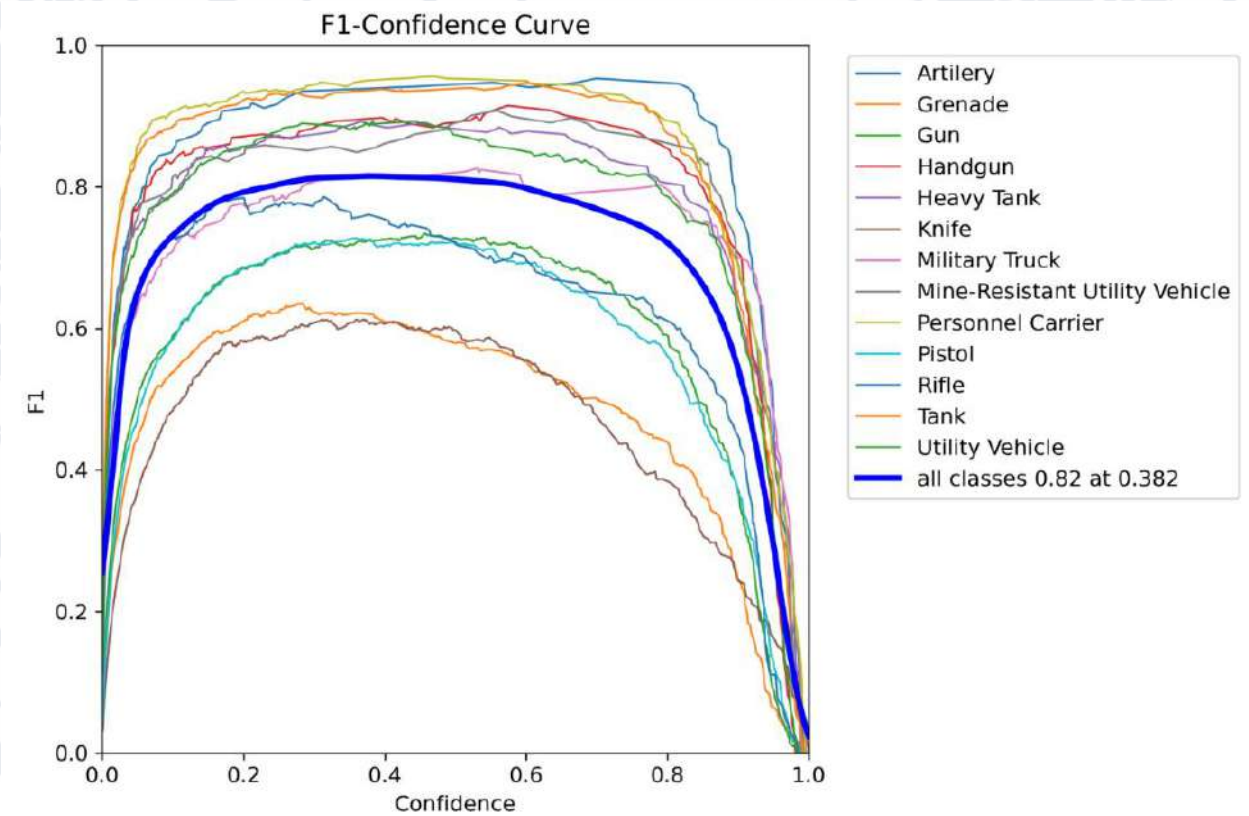

Рисунок 3.9 Графік співвідношення F1-Confidence

Значно нижче середнього ми можемо бачити категорії гранат та ножів, що підтверджує виводи, зроблені вище. Але також ми можемо побачити що

абсолютна більшість категорій ручної зброї знаходиться нижче середньої лінії, що дає нам можливість зробити припущення про низьку якість датасетів, що стосуються зброї. Позитивним викидом є категорія короткоствольної зброї.

Графік середнього значення виглядає доволі здоровим, ми бачимо досить високі значення F1 навіть при низьких значеннях впевненості, що є хорошою ознакою.

Також мною було проведене тестування програми на швидкодію. Модель справляється з обробкою в середньому 23.8 кадрів на секунду, в умовах віртуального середовища KVM з двома виділеними процесорними потоками на частоті 3.4ГГц та двома гігабайтами оперативної пам’яті, що дає можливість припускати, що подібна програма може працювати на “вбудованих” пристроях, таких як дрони, розумні камери тощо.

Практичне тестування програми також показало, що навчена модель не є чутливою до кутів і справляється з знаходженням об’єктів під різноманітними нахилами. Також модель досить добре справляється з частковою оклюзією об’єктів в кадрі, до близько половини площі об’єкта може бути перекрито іншим об’єктом і модель не буде його втрачати. Ключовою проблемою для моєї моделі став розмір об’єктів. Об’єкти що займають 1/25 частину площі або меньше мають значно меншу точність розпізнавання, або можуть бути зовсім втраченими. Це накладає деякі рамки що до можливого практичного використання моделі у реальних умовах, але для деяких сфер застосувань це не є критичним.

ВИСНОВКИ

В ході виконання бакалаврської роботи було досліджено частину дисципліни комп’ютерного зору що стосується знаходження та класифікації об’єктів. Було здійснено аналіз ринку фреймворків та бібліотек у сферах глибокого навчання та комп’ютерного зору. Було також розглянуто методи розпізнавання об’єктів, як на основі нейронних мереж, так і без, виділено найбільш успішні.

На основі отриманих результатів було підібрано технологічний стек, та впроваджено систему розпізнавання зброї та військової техніки у відеопотоці за допомогою бібліотек комп'ютерного зору у режимі реального часу.

Були вивченні принципи роботи з бібліотекою комп’ютерного зору OpenCV, фреймворком PyTorch, рушієм та середовищем виконання ONNXRuntime, а також інструментами Conda та ClearML.

Створена під час виконання роботи може бути в подальшому інтегрована до інших систем, або може працювати як самостійний додаток.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Navneet Dalal, Bill Triggs. Histograms of Oriented Gradients for Human Detection. International Conference on Computer Vision & Pattern Recognition (CVPR ’05), Jun 2005, San Diego, United States. pp.886–893, ff10.1109/CVPR.2005.177ff. ffinria-00548512f
2. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.
3. David G. Lowe. Object Recognition from Local Scale-Invariant Features. Computer Science Department University of British Columbia Vancouver, B.C., V6T 1Z4, Canada
4. Lauren Barghout. Spatial-Taxon Information Granules as Used in Iterative Fuzzy-Decision-Making for Image Segmentation Springer International Publishing Switzerland 2015 W. Pedrycz and S.-M. Chen (eds.), Granular Computing and Decision-Making Studies in Big Data 10, doi: 10.1007/978-3-319-16829-6_12
5. Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE. pp. 580–587. doi:10.1109/CVPR.2014.81
6. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector. Computer Vision – ECCV 2016. Lecture Notes in Computer Science. Vol. 9905. pp. 21–37
7. Viola, P., Jones, M.J. Robust Real-Time Face Detection. International Journal of Computer Vision 57, 137–154 (2004). doi:10.1023/B:VISI.0000013087.49260
8. Papageorgiou, Oren and Poggio, "A general framework for object detection", International Conference on Computer Vision, 1998.

9. Yi-Qing Wang, An Analysis of the Viola-Jones Face Detection Algorithm, Image Processing On Line, 4 (2014), pp. 128–148. doi: 10.5201/ipol.2014.104
10. Ives Rey Otero, and Mauricio Delbracio, Anatomy of the SIFT Method, Image Processing On Line, 4 (2014), pp. 370–396. doi:/10.5201/ipol.2014.82
11. Comprehensive Review of R-CNN and its Variant Architectures . (2024). International Research Journal on Advanced Engineering Hub (IRJAEH), 2(04), 959-966.
12. Girshick, R. (2015). Fast R-CNN. In Proceedings of the IEEE interna-tional conference on computer vision (pp. 1440-1448).
13. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.
14. Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. Mask R-CNN
15. Valentina E. Balas, Raghvendra Kumar, Rajshree Srivastava. Recent Trends and Advances in Artificial Intelligence and Internet of Things.
16. Steven W. Smith. The Scientist and Engineer's Guide to Digital Signal Processing.
17. Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015
18. Ljubisa Stankovic, Danilo Mandic. Convolutional Neural Networks Demystified: A Matched Filtering Perspective Based Tutorial.
19. Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. Parallel distributed processing model with local space-invariant interconnections and its optical architecture.
20. Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton. ImageNet Classification with Deep Convolutional Neural Networks. University of Toronto, Canada.
21. Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, Yichen Wei. Deformable Convolutional Networks.
22. Xizhou Zhu, Han Hu, Stephen Lin, Jifeng Dai. Deformable ConvNets v2: More Deformable, Better Results.

23. Evan Shelhamer, Jonathan Long, Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation.
24. Jaccard Paul. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bulletin de la Société vaudoise des sciences naturelles.
25. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár. Focal Loss for Dense Object Detection.
26. Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors.
27. Joseph Redmon, Ali Farhadi. YOLO9000: Better, Faster, Stronger. University of Washington, Allen Institute for AI.
28. Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. University of Washington, Allen Institute for AI.
29. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. University of Washington, Allen Institute for AI, Facebook AI Research.
30. Dmitriy Fradkin and Ilya Muchnik. Support Vector Machines for Classification. DIMACS Series in Discrete Mathematics and Theoretical Computer Science.
31. MCV2 Dataset. Baskent Univercity. URL: https://universe.roboflow.com/baskent-univercity-5cybx/mcv2 (дата звернення: 11.05.2024)
32. weapon detection Dataset. ENSPY. URL: https://universe.roboflow.com/enspy/weapon-detection-yyv12 (дата звернення: 11.05.2024)
33. MIR 1 Dataset. Military Image Recognition. URL: https://universe.roboflow.com/military-image-recognition/mir-1 (дата звернення: 11.05.2024)

34. Weapon Detection3 Dataset. Fend Tech. URL: https://universe.roboflow.com/fend-tech/weapon-detection3 (дата звернення: 11.05.2024)
35. InfoWorld. Facebook brings GPU-powered machine learning to Python. URL: https://www.infoworld.com/article/3159120/facebook-brings-gpu-powered-machine-learning-to-python.html (дата звернення: 23.05.2024)
36. Why Tensorflow. URL: http://tensorflow.org/about (дата звернення: 23.05.2024)
37. Intel® Distribution of OpenVINO™ Toolkit, Overview. URL: https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html (дата звернення: 23.05.2024)
38. Edge AI + Vision. The Caffe Deep Learning Framework: An Interview with the Core Developers. URL: https://www.edge-ai-vision.com/2016/01/the-caffe-deep-learning-framework-an-interview-with-the-core-developers/ (дата звернення: 23.05.2024)
39. NCNN on Github. URL: https://github.com/Tencent/ncnn (дата звернення: 23.05.2024)
40. Engadget. Microsoft and Facebook's open AI ecosystem gains more support. URL: https://www.engadget.com/2017-10-11-microsoft-facebooks-ai-onxx-partners.html (дата звернення: 23.05.2024)
41. Aziz Taha, Abdel. Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. BMC Medical Imaging.
42. Andrew G. Howard Menglong Zhu Bo Chen Dmitry Kalenichenko Weijun Wang Tobias Weyand Marco Andreetto Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications
43. Andrew B. Watson. Sensory Processes and Perception, chapter 6, “Temporal Sensitivity”. Nasa Ames Research Center, Moffett Field, California.

ДОДАТОК А.
ЛІСТІНГ КОДУ

```
# detection_manager.py
import cv2
import onnxruntime
from ultralytics import YOLO
from ultralytics.engine.results import Results
from pathlib import Path

label_names = {0: 'Artilery', 1: 'Grenade', 2:'Gun', 3:'Handgun',
4:'Heavy Tank', 5:'Knife', 6:'Military Truck', 7:'Mine-Resistant Utility
Vehicle', 8:'Personnel Carrier', 9:'Pistol', 10:'Rifle', 11:'Tank',
12:'Utility Vehicle'}

class Detector:
    def __init__(self, model_path, task='detect'):
        self.model = YOLO(model_path, task=task)
        self.imgsz = 320

    def predict(self, frame, conf):
        return self.model.predict(source = frame, imgsz=self.imgsz,
save=False, save_txt=False, show=False, verbose=False, conf=conf)

    def get_label_names(self):
        if self.model.names is None or len(self.model.names) == 0:
            return label_names
        return self.model.names

class DetectorWrapper:
    def __init__(self, model_path):
        model_path = Path(model_path)
        self.detector = Detector(model_path)

    def predict(self, frame, conf=0.5):
        return self.detector.predict(frame, conf=conf)

    def get_label_names(self):
        return self.detector.get_label_names()
# draw_tools.py
import cv2
import time
from ultralytics.utils.plotting import Annotator

class FrameCounter:
    def __init__(self):
        self.start_time = time.time()
```

```
        self.display_time = 1
        self.frame_count = 0
        self.fps = 0
        self.is_fps_updated = False

    def get_fps(self):
        elapsed_time = time.time() - self.start_time
        self.frame_count += 1
        is_fps_updated = False
        if elapsed_time > self.display_time:
            self.fps = self.frame_count / elapsed_time
            self.frame_count = 0
            self.start_time = time.time()
            is_fps_updated = True
        return int(self.fps), is_fps_updated

def draw_fps(frame, fps):
    cv2.putText(frame, f'FPS: {fps}', (10, 30), cv2.FONT_HERSHEY_PLAIN,
3, (64, 255, 64), 3, cv2.LINE_AA)

def draw_boxes(frame, label_names, results):
    annotator = None
    for r in results:
        annotator = Annotator(frame)
        boxes = r.boxes
        for box in boxes:
            box_coord = box.xyxy[0]
            label_class = box.cls
            annotator.box_label(box_coord, label_names[int(label_class)],
color=(16, 128, 32))

    if annotator is not None:
        annotated_frame = annotator.result()
    else:
        annotated_frame = frame.copy()

    return annotated_frame
# main.py
import cv2
import sys
from draw_tools import FrameCounter, draw_fps, draw_boxes
from detection_manager import DetectorWrapper

if __name__ == '__main__':
    model_path = 'yolov8n.onnx'
    detector = DetectorWrapper(model_path)
    label_names = detector.get_label_names()
    model = DetectorWrapper(model_path)
```

```
    if len(sys.argv) > 1:
        if sys.argv[1] == 'camera':
            cap = cv2.VideoCapture(0)
        else:
            cap = cv2.VideoCapture(sys.argv[1])
    else:
        print('Введіть шлях до відео, або введіть "camera" для аналізу
відео з камери пристрою')
        sys.exit(1)

    frame_counter = FrameCounter()

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        fps, is_updated = frame_counter.get_fps()
        frame = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)
        results = model.predict(frame)
        display_image = draw_boxes(frame, label_names, results)
        draw_fps(display_image, fps)

        cv2.imshow('frame', display_image)
        if cv2.waitKey(1) & 0xFF == ord('c'):
            break

    cap.release()
    cv2.destroyAllWindows()
```