КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Роботу розглянуто та допущено до захисту на
засіданні кафедри математичної інформатики
«15» травня 2023 р.
протокол № 10
Завідувач кафедри
Терещенко Василь Миколайович _____
(підпис)

Кваліфікаційна робота

на здобуття ступеня магістра
за спеціальністю 122 Комп'ютерні науки
на тему:

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ПОШУКУ ВІЙСЬКОВОЇ ТЕХНІКИ ТА ВІЙСЬКОВИХ НА ВІДЕО

Виконав студент 2-го курсу
Тара Олександр Миколайович
(підпис)

Науковий керівник:
асистент,
кандидат фізико-математичних наук
Терещенко Ярослав Васильович
_____
(підпис)

Науковий консультант:
доцент,
кандидат фізико-математичних наук
Деревянченко Олександр Валерійович
_____
(підпис)

Засвідчую, що в цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент
(підпис)

Київ - 2023

РЕФЕРАТ

Обсяг роботи 69 сторінок, 28 ілюстрацій, 23 джерел посилань.

НЕЙРОННА МЕРЕЖА, ПОШУК ОБ'ЄКТІВ, ДЕТЕКЦІЯ, ПОВНІСТЮ ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, АНДРОЇД, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, ДАТАСЕТ, АРХІТЕКТУРА, TENSORFLOW LITE, MULTIK, OPENCV, ТЕНЗОР

Об'єктом розв'язання задачі створення застосунку для знаходження військової техніки та військових на відео є застосунок та детекції на зображенні

Метою дипломної роботи є розробка застосунку, який дозволить знаходити військову техніку та військових на відео, а також навчити існуючу модель розрізняти військові об'єкти з дронів.

Підстави для виконання: задача пошуку військових об'єктів є критичною у нинішній війні, а автоматизація цього процесу може сильно допомогти українським військовим. Розробка застосунку, який дозволяє використовувати навчену модель для пошуку військової техніки та військових надасть зручний інтерфейс для навченої моделі.

Результати роботи: виконано загальний огляд відомих підходів до розв'язання поставленої задачі, проаналізовано переваги та недоліки використання різних архітектур нейронних мереж, запропоновано використання моделі YOLOv8, яка є найновішою та найкращою моделлю для пошуку об'єктів на зображенні. Проведено навчання моделі на власному датасеті, збір якого описано в розділі 2. Модель навчено з використанням методу навчання з підкріпленням, що дозволяє зберегти надбання YOLOv8 у задачі пошуку об'єктів і водночас змінити модель під наші потреби. Створено додаток, який слугує інтерфейсом для роботи з навченою моделлю.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта дослідження або розробки.

Розпізнавання об'єктів на відео - це одна з головних задач комп'ютерного зору, яка включає в себе виявлення, класифікацію та відстеження об'єктів на відео. Сучасні алгоритми розпізнавання об'єктів на відео використовують нейронні мережі, які забезпечують високу точність розпізнавання.

Одним із найпопулярніших алгоритмів розпізнавання об'єктів на відео є YOLO (You Only Look Once). Він дозволяє виявляти об'єкти на відео в режимі реального часу та має високу швидкість роботи. YOLO використовує глибокі нейронні мережі для виявлення об'єктів та розділення їх на класи. Більш детальний опис алгоритму можна знайти у статті авторів алгоритму. [1]

Ще одним ефективним алгоритмом розпізнавання об'єктів на відео є Mask R-CNN, який використовується для сегментації об'єктів на відео. Цей алгоритм забезпечує точність розпізнавання об'єктів та високу якість сегментації. Більш детальний опис алгоритму можна знайти у статті авторів. [2]

Також на ринку існують готові продукти для розпізнавання об'єктів на відео, такі як Amazon Rekognition [3], Google Cloud Video Intelligence [4], Microsoft Azure Video Indexer [5], тощо.

У загальному, сучасний стан розпізнавання об'єктів на відео відображає високу точність та ефективність алгоритмів за рахунок використання нейронних мереж та інших технологій машинного навчання. Однак, ця галузь постійно розвивається та вдосконалюється.

Одним з напрямків розвитку розпізнавання об'єктів на відео є розробка алгоритмів, які враховують зміну орієнтації та відстані між об'єктами на відео. Наприклад, алгоритмів, які використовують стереокамери та глибинні карти для визначення розташування об'єктів у тривимірному просторі. Це дозволить більш точно відслідковувати рух об'єктів на відео.

Також, з поширенням відеонагляду та камер з високим роздільним здатністю, з'являється проблема аналізу великих об'ємів даних, що генеруються

відеопотоками. Тому активно розвивається аналіз відео в режимі реального часу та аналіз відео на віддаленому сервері за допомогою хмарних технологій.

У загальному, розпізнавання об'єктів на відео - це широка галузь комп'ютерного зору, що постійно розвивається та яка має багато прикладних застосувань, таких як відеонагляд, робототехніка, автоматична транспортна система тощо.

Використання нейронних мереж для розпізнавання об'єктів у військових додатках є одним з напрямів розвитку військової техніки та технологій. Нейронні мережі можуть бути застосовані для розпізнавання різних об'єктів, таких як техніка, бойові пости, люди та ін.

Одним з прикладів використання нейронних мереж для розпізнавання об'єктів у військових додатках є проект Maven, який був запущений Пентагоном. Цей проект передбачає створення системи штучного інтелекту, яка буде здатна розпізнавати техніку та інші об'єкти на відео з висоти пташиного польоту.

Проект Maven - це спільний проект Міністерства оборони США та консорціуму підприємств зі створення системи штучного інтелекту для автоматичного розпізнавання об'єктів на відео з висоти пташиного польоту. Проект був запущений в 2018 році і передбачає створення системи, яка зможе автоматично розпізнавати техніку та інші об'єкти на відео з дронів.

У рамках проекту Maven використовуються найсучасніші технології машинного навчання та глибинного навчання, що дозволяє системі розпізнавати об'єкти з високою точністю та швидкістю. Одним з ключових компонентів системи є згорткові нейронні мережі, які навчаються на великих масштабах даних з різних джерел.

Проект Maven є частиною загальної стратегії Міністерства оборони США щодо використання штучного інтелекту та машинного навчання для поліпшення військових технологій та забезпечення безпеки. За словами представників Міністерства оборони, система розпізнавання об'єктів з висоти пташиного

польоту може забезпечити значний внесок у покращення військових операцій та зменшення загроз безпеці.[6] [7]

Специфіка розробок у військовій сфері не дозволяє знаходити деталі нових технологій у відкритому доступі.

Актуальність роботи та підстави для її виконання.

В умовах нинішньої війни проти росії надзвичайно велику роль відіграє збір інформації. В той же час комерційні дрони та інші безпілотні літальні засоби використовуються у всіх підрозділах Збройних Сил України для виявлення та враження ворога з висоти. Зображення отримані з дронів та інших безпілотних літальних засобів мають типові особливості, наприклад це відео з висоти кількасот метрів можливих позицій росіян та їх техніки. Знімки також сильно залежать від пори року - взимку переважає білий та темно-коричневий колір, а влітку - зелений та коричневий. Такий вузький спектр отриманих зображень вказує на те, що використання нейронних мереж для розпізнавання об’єктів є дуже перспективним.

Одна з особливостей сучасної української армії - це насиченість мобільними обчислювальними пристроями (смартфони, планшети). Таким чином андроїд застосунок може легко бути використаний пілотами дронів. Також в ЗСУ вже використовують українське програмного забезпечення для розширення бойових можливостей: наприклад “Кропива” допомагає артилеристам з прицілюванням. Отже, не важко собі уявити, що вдалий андроїд застосунок стане справді поширеним у війську.

Застосунок, який використовує нейронні мережі для автоматичного розпізнавання військової техніки та військових на відео, може бути дуже корисним для військових підрозділів, які відповідають за моніторинг великих територій. Застосунок може допомогти збільшити швидкість та ефективність

пошуку військової техніки та військових, що є особливо важливим у воєнних умовах.

Крім чисто воєнного використання, такий застосунок може бути використаним і так званими OSINT журналістами, що опрацьовують величезні обсяги даних з відкритого доступу.

Крім того, розробка застосунку , який використовує нейронні мережі для розпізнавання об'єктів на відео, є важливим напрямком розвитку сучасної технології машинного навчання та штучного інтелекту. Використання нейронних мереж для розпізнавання об'єктів може знайти застосування в багатьох галузях, таких як безпека, медицина, транспорт, промисловість та інші.

В процесі огляду сучасного стану об’єкту дослідження не було виявлено хороших підходів до розпізнавання саме військових об’єктів. Навіть найкращі моделі зазвичай не тренують на військових даних, а отже звичайні моделі навіть не мають окремих класів для військової техніки. Ті роботи, що фокусувались на військових зображеннях використовували старі моделі для розпізнавання об’єктів і не мали великого успіху.

Мета й завдання роботи. Метою дипломної роботи є розробка додатку для пошуку військової техніки та військових на відео з використанням нейронних мереж. Для досягнення цієї мети поставлено наступні завдання:

- Огляд і аналіз існуючих методів та технологій розпізнавання об'єктів на відео, зокрема нейронних мереж.
- Описати ідею власного розв’язку задачі
- Зібрати тренувальні дані для моделі.
- Виконати навчання запропонованої моделі.
- Розробка алгоритму для автоматичного розпізнавання військової техніки та військових на відео з використанням нейронних мереж.
- Реалізація додатку для пошуку військової техніки та військових на відео з використанням розробленого алгоритму.

- Тестування та оцінка ефективності розробленого додатку на реальних відеоданих.

Об’єкт, методи й засоби розроблення. Об’єктом розв’язання задачі розробки є застосунок для пошуку військової техніки та військових на відео з використанням нейронних мереж є застосунок, що використовує навчену нейронну мережу для розпізнавання об’єктів на відео.. Робота виконується за допомогою Python, Pytorch для навчання моделі та Android Studio та Kotlin для створення застосунку. Серед допоміжних засобів можна згадати бібліотеки numpy, matplotlib, PIL, scipy.

Можливі сфери застосування.

Є три основні сфери, де можна використати розроблений застосунок:

1. Військовими. Застосунок допоможе пілотам дронів у спостереженні за великими територіями. Людське око пристосоване помічати рухомі об’єкти, а статичні об’єкти часто залишаються непоміченими. [8] Запропонована нейронна мережа працює саме зі статичними зображеннями, а отже працює у симбіозі з людиною.
2. OSINT спеціалістами. В процесі своєї роботи ці журналісти повинні знайти, опрацювати та задокументувати величезну кількість інформації. Одним з основних напрямків їх роботи є документування втрат техніки. Додаток, який помічає військову техніку допоможе OSINT спеціалістам підвищити свою ефективність.
3. Наукова сфера. У відкритому доступі не було знайдено ефективних моделей, які в навчанні використовували зображення військових об’єктів.

Взаємозв’язок з іншими роботами. В цій роботі було використано модель YOLO 8 [1] [9] для знаходження об’єктів на зображеннях. Для навчання, окрім власного датасету, було використано [10] [11].

РОЗДІЛ 1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Загальні означення.

Методом пошуку об’єктів на зображенні є нейронна мережа, яка повертає координати знайдених об’єктів.

Штучний алгоритм навчання нейронної мережі, або просто нейронна мережа - це обчислювальна система навчання, яка використовує мережу функцій для розуміння та переведення введених даних однієї форми у бажаний результат, як правило, в іншій формі. Концепція штучної нейронної мережі була натхненна людською біологією та тим, як нейрони людського мозку функціонують разом, щоб зрозуміти вхідні дані отримані від людських органів чуття.

Нейронні мережі - лише один із багатьох інструментів та підходів, що використовуються в алгоритмах машинного навчання. Сама нейронна мережа може використовуватися як одна частина в багатьох різних алгоритмах машинного навчання для переведення складних вхідних даних у простір, який можуть зрозуміти комп'ютери.

Нейронні мережі сьогодні застосовуються до багатьох реальних проблем, включаючи розпізнавання мови та зображень, фільтрування спаму, електронну пошту, фінанси та медичну діагностику.

Алгоритми машинного навчання, які використовують нейронні мережі, як правило, не потребують програмування з певними правилами, які визначають, чого очікувати від вхідних даних. Натомість алгоритм навчання нейронних мереж навчається на обробці багатьох маркованих прикладів (тобто даних із «відповідями»), які подаються під час навчання, і використовуючи цей ключ відповіді, щоб дізнатись, які характеристики вхідних даних необхідні для побудови правильного результату, це так зване навчання з вчителем. Після обробки достатньої кількості прикладів нейронна мережа може почати обробляти нові, невідомі вхідні дані та успішно повертати точні результати. Чим

більше прикладів та різноманітності вхідних даних бачить програма, тим точнішими стають результати, оскільки програма навчається з досвідом. [12]

1.2. Згорткові нейронні мережі

Архітектури нейронних мереж, що показали найкращі результати в пошуку об’єктів на зображенні базувались на згорткових нейронних мережах (CNN, ConvNet), тому важливо детально розуміти це поняття.

CNNs – це категорія нейронних мереж, що зарекомедувала себе як ефективна нейронна мережа в сфері розпізнавання та класифікації зображень. ConvNets успішно проявила себе в ідентифікації облич, об’єктів і дорожніх знаків не кажучи про забезпечення зору роботів та безпілотників.

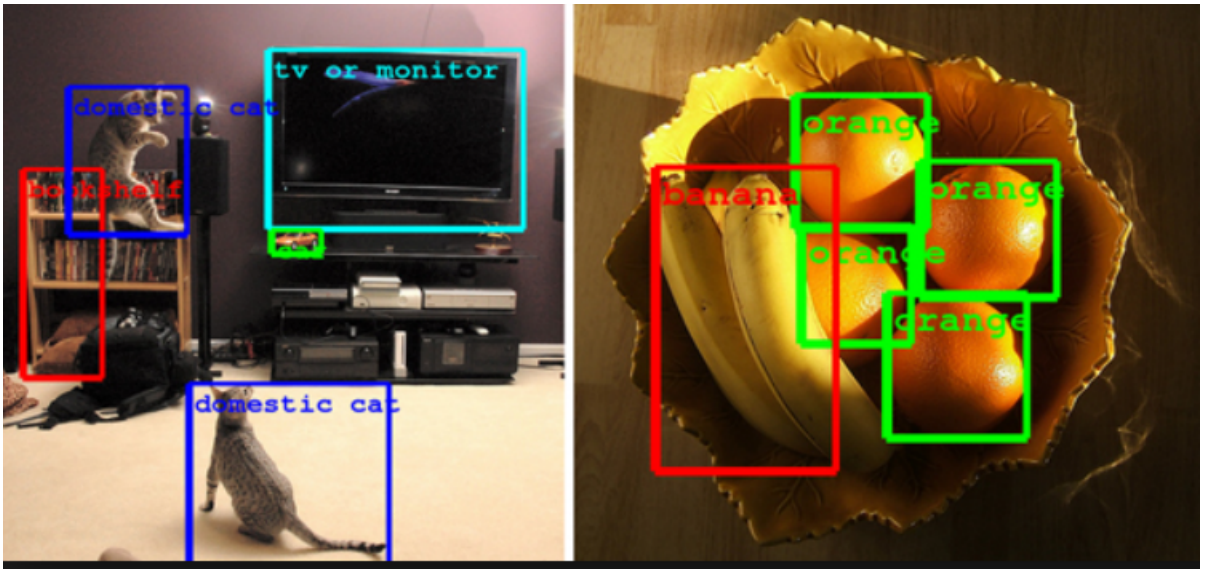

Рис 1.

LeNet, розроблена Yann LeCun, була однією з перших згорткових нейронних мереж, яка допомогла розвинути сферу глибинного навчання [13]. Згорткова нейронна мережа на рис. 1 схожа за архітектурою до LeNet і класифікує вхідне зображення за певними сталими категоріями. Коли нейронна мережа отримує на вхід зображення, то на виході повертає ймовірність попадання зображення в певну категорію. Найбільшу ймовірність отримує та

категорія, яка на думку нейронної мережі є представлена на вхідному зображенні.

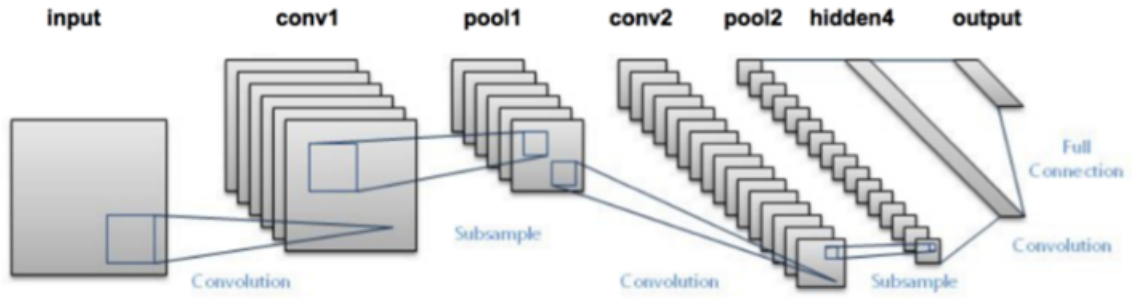

Рис 2.

Є чотири основні операції в ConvNet, які показані на рис. 3

- Convolution
- Non Linearity (ReLU)
- Pooling or Sub Sampling
- Classification (Fully Connected Layer)

Ці операції є фундаментом будь-якої згорткової нейронної мережі.

Канал [14] – це загальноприйнятий термін, який використовується щоб звернутись до певного елементу зображення. Зображення зі стандартної цифової камери матиме три канали – червоний, зелений і синій.

Крок згортки [15]

Основна мета згортки у випадку ConvNet – виділити особливі риси з вхідного зображення. Згортка зберігає просторові зв’язки між пікселями за допомогою вивчення особливих рис зображення, використовуючи невеликі квадрати вхідних даних.

Як згадано вище, кожне зображення можна вважати матрицею значень пікселів. Розглянемо зображення 5 * 5, чиї пікселі можуть набувати значень 0 і 1 (хоча для чорно-білого зображення значення бувають від 0 до 255, зелена матриця внизу є особливим випадком, де пікселі набувають значень 0 та 1):

1	1	1	0	0

0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

Таблиця 1.

Також, розглянемо іншу матрицю 3 * 3, як показано вищеmatrix. Тоді згортка матриць зображень 5 * 5 та 3 * 3 може бути обчислене як почергове знаходження суми по профілю (поелементне множення елементів матриць 3 * 3 і 5 * 5 і подальше знаходження суми по-елементних добутків, яка формує значення 1 клітинки у вихідній матриці) матриці 3 * 3 у матриці 5 * 5. Кожного разу профіль матриці 3 * 3 зсувається вправо на одну позицію. В даному випадку результатом згортки буде матриця 3 * 3, яка показана у Таблиці 2.

4	3	4
2	4	3
2	3	4

Таблиця 2

В термінології CNN, матриця 3 * 3 називається фільтром, ядром або детектор особливостей, а матриця, утворена ковзанням фільтру по зображенню і обчисленням точкового результату, називаєтсья згорнена особливість або картою активації. На Рис. 4 показано результат застосування згортки для реальної картинки із певним фільтром.

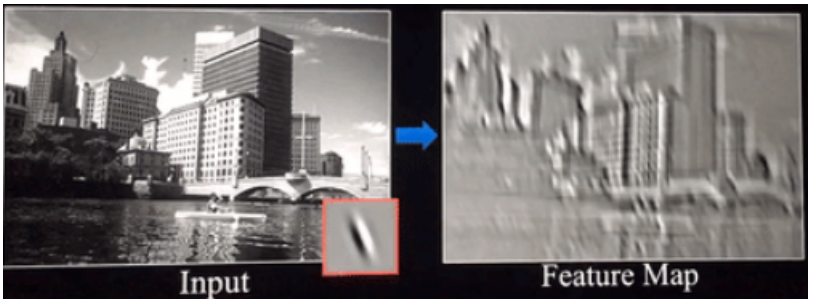

Рис. 3

Важливо відзначити, що етап згортки охоплює локальні залежності в оригінальному зображенні. На практиці CNN засвоює значення цих фільтрів самостійно в процесі навчання (хоча нам як і раніше необхідно вказати такі параметри, як кількість фільтрів, розмір фільтра, архітектура мережі і т.д. до початку процесу навчання). Чим більше фільтрів, тим більше особливостей зображень витягуються і тим краще наша мережа стає у розпізнаванні образів в довільних зображеннях. При роботі з багатовимірними вхідними даними, такими як зображення розмірності входів, таких як зображення, як видно вище, непрактично з'єднувати нейрони зі всіма нейронами попереднього рівня. Замість цього ми будемо підключати кожен нейрон тільки в локальній області вхідного рівня. Просторова протяжність цього зв'язку є гіперпараметр, що називається сприйнятливе поле.

Розмір карти особливостей контролюється трьома параметрами, які ми визначаємо перед етапом згортки:

- Глибина: Глибина відповідає кількості фільтрів, які ми використоуємо для операції згортки. У мережі, показаній на рис 4., ми виконуємо згортку зображення човна з використанням трьох різних фільтрів, таким чином, створюючи три карти особливостей, як показано на малюнку. Можна вважати ці три карти особливостей stacked двовимірними матрицями, а тому, глибина карти особливостей буде три.

Рис. 4

- Крок: крок– це кількість пікселів, на яку ми ковзаємо фільтром по вхідній матриці. Якщо крок рівний 1, фільтр пересувається на один піксель. Коли крок рівний 2, фільтр пересувається на 2 пікселя за одне ковзання. Більший крок буде створювати менші карти особливостей.
- Доповнення нулями: вхідна матриця доповнюється нулями на межі зображення, щоб можна було використати фільтр для елементів на межі матриці. Хорошою властивістю zero padding є можливість контролю розміру матриці особливостей. Додавання zero-padding також називається широкою згорткою, а не додавання – вузька згортка.

Розширені згортки.

Останні розробки включають в себе введення ще одного гіперпараметру, що називається розширення [16]. Досі ми розглядали тільки суміжні фільтри CONV. Проте можливо мати фільтри, в яких є вільне місце між кожною клітинкою. Вони називаються розширення. Як приклад, в одному вимірі фільтр w розміру 3 для входу x застосується так:

w[0]*x[0] + w[1]*x[1] + w[2]*x[2].

Це розширення для 0. Для розширення 1 фільтр працюватиме так:

w[0]*x[0] + w[1]*x[2] + w[2]*x[4];

Іншими словами, існує розрив розміру 1 між використаннями. Це може бути дуже корисно в деяких випадках для використання в поєднанні з 0-розширеними фільтрами, оскільки вона дозволяє об'єднати просторову інформацію вхідних даних набагато агресивніше з меншою кількістю шарів. Наприклад, якщо скласти два 3x3 CONV шари, то ви можете бути впевнені, що

нейрони на 2-му шарі є функцією 5x5 (ефективне сприйнятливе поле цих нейронів - 5x5). Якщо ми будемо використовувати розширені згортки, то це ефективне сприйнятливе поле буде збільшуватись набагато швидше.

Нелінійність

Нелінійність – це поелементна інформація (застосована для кожного пікселя), що заміняє значення кожного пікселя на інше згідно з вибраною нелінійною функцією. Зараз найбільш часто вживана нелінійність для згорткових шарів є ReLU [17] та її модифікації. Метою нелінійності є введення нелінійності в нашому ConvNet, так як більша частина реальних даних, на яких ConvNet повинна навчатись буде нелінійним (згортка є лінійної операцією - поелементно матричне множення і додавання, тому ми враховуємо нелінійність, вводячи нелінійну функцію ReLU).

Пулинг (Pooling)

Пулинг зменшує розмірність кожної карти особливостей, але зберігає найважливішу інформацію. Пулинг може бути різних типів: Max, Average, Sum і т.д.

У випадку Max Pooling, ми визначаємо просторову околицю (наприклад область 2×2) і вибираємо найбільший елемент з матриці особливостей в цій області. Замість того, щоб вибирати найбільший елемент, можна також вибрати середній (Average Pooling) або суму всіх елементів в цій області. На практиці, Max Pooling зарекомендувало себе найкраще. На таб. 3 показано приклад роботи Max Pooling на Rectified Feature map (отриманої після операції згортки та операції ReLU), використовуючи область 2 × 2.

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Таб. 3

Функція пулингу використовується для прогресивного зменшення просторового розміру представлення вхідних даних. Зокрема, pooling:

Робить представлення вхідних даних меншим і легшим в керуванні.

Зменшує кількість параметрів і обчислень в мережі, а тому контролює перенавчання [18] .

Робить мережу стійкою до малих трансформацій, спотворень та переміщень у вхідному зображенні (невелике спотворення у вхідному зображенні не змінить вивід функції пулингу, оскільки ми вибираємо максимальне/середнє значення в околиці).

допомагає перейти до практично стійкого до зміни розширення представлення зображення. Це дуже важливо, оскільки ми можемо виявити об'єкти в зображенні незалежно від того, де вони знаходяться [19].

Відмова від пулингу.

Деякі дослідники вважають, що без операції пулингу можна обійтись. Наприклад Striving for Simplicity: The All Convolutional Net пропонує відкидати шар пулингуна користь архітектури, що складається лише з шарів згортки. Для того, щоб зменшити розмір представлення, вони пропонують використовувати більший крок в CONV шарі. Відкидання pooling шарів також є важливими для навчання генеративних моделей, такі як варіаційні автоассоціатори (VAEs) або generative adversarial networks (Gans).

1.3. Заморозка шарів моделі

В процесі навчання YOLO v8 можна заморозити частину шарів, щоб зберегти результати попереднього навчання.

Замороження шарів моделі є важливим кроком у процесі навчання глибоких нейронних мереж, особливо якщо ви використовуєте передварительно навчену модель, наприклад, VGG16, Inception або ResNet, для вирішення своєї задачі.

При замороженні шарів моделі ви блокуєте ваги та зміщення шарів, що знаходяться вище від заморожених шарів, від оновлення під час проходження зворотного поширення помилки під час навчання. Це означає, що ваги та зміщення цих шарів залишаються незмінними під час тренування.

Замороження шарів може бути корисним тоді, коли ви хочете використовувати передній кінець (фронтенд) вже навченої моделі, який може виконувати високо рівні завдання, наприклад, розпізнавання обличчя або класифікацію зображень, але у вас є новий датасет, який відрізняється від того, на якому була навчена вихідна модель. Замороження шарів дозволяє досягти найкращих результатів за менший час, використовуючи вже навчений фронтенд.

Після того, як шари були заморожені, вам слід додатково навчити ваги та зміщення заморожених шарів, щоб підлаштувати їх до нового датасету. Цей процес називається налаштуванням (fine-tuning) моделі. Зазвичай, налаштування моделі виконують шляхом розмороження кількох верхніх шарів та відновлення навчання всієї моделі на новому датасеті.

Замороження шарів - це ефективний спосіб скоротити час тренування та зменшити кількість потрібних для навчання даних

1.4. YOLO v8

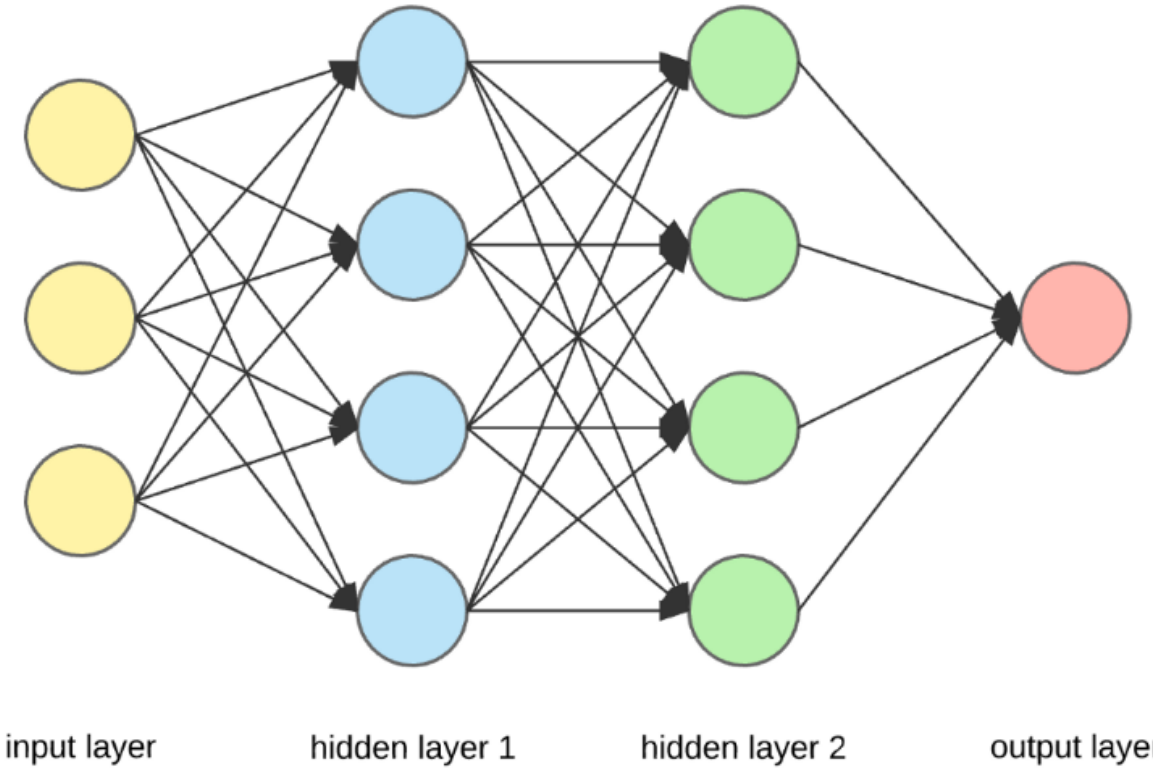

Рис. 5. Приклад мережі прямої передачі.

Нейронні мережі - це складні структури, які складаються зі штучних нейронів, які можуть приймати кілька входів для отримання єдиного виходу. Це основна робота нейронної мережі - перетворити вхід у значущий результат. Зазвичай нейронна мережа складається з вхідного та вихідного рівня з одним або декількома прихованими шарами всередині.

У нейронній мережі всі нейрони впливають один на одного, а отже, всі вони пов’язані. Мережа може визнавати та спостерігати кожен аспект набору даних, що знаходиться під рукою, і те, як різні частини даних можуть стосуватися чи не співвідноситися між собою. Таким чином нейронні мережі здатні знаходити надзвичайно складні шаблони у величезних обсягах даних.

У нейронній мережі потік інформації відбувається двома шляхами.

Мережі прямої передачі: у цій моделі сигнали рухаються лише в одному напрямку, до вихідного шару. Мережі Feedforward мають вхідний рівень і один

вихідний рівень з нулем або декількома прихованими шарами (Рис. 2.). Вони широко використовуються при розпізнаванні зразків.

Мережі зворотного зв’язку: у цій моделі рекурентні або інтерактивні мережі використовують свій внутрішній стан (пам’ять) для обробки послідовності входів. У них сигнали можуть рухатися в обох напрямках через петлі (прихований шар / шари) в мережі. Вони зазвичай використовуються в часових рядах та послідовних завданнях.

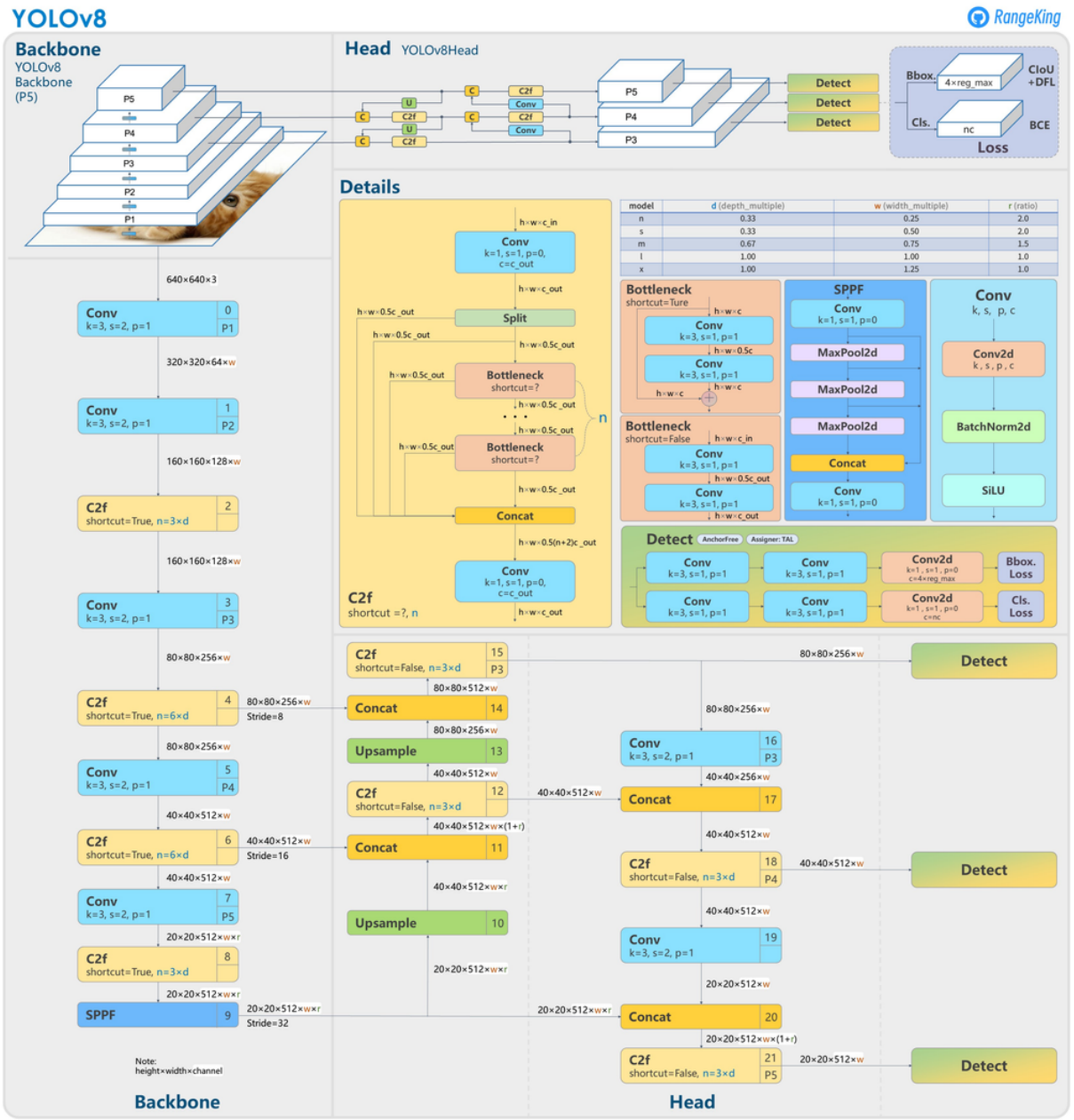

Рис. 6. Архітектура моделі YOLO v8

В цій дипломній роботі для пошуку військових об’єктів на зображеннях було використано SOTA модель для пошуку зображень - YOLO, а саме її найновіша версія - v8.

YOLO (You Only Look Once) - це алгоритм розпізнавання об'єктів, який працює на основі нейронної мережі з використанням конволюційних шарів. Архітектура YOLO v8 є найновішою версією алгоритму, яка була представлена в 2021 році.

Архітектура YOLO v8 має наступні етапи роботи:

- Введення зображення (image input): Вхідне зображення передається в нейронну мережу з попередньо визначеним розміром, наприклад 640x640 пікселів.
- Передпроцесинг зображення (image preprocessing): Зображення піддається передпроцесингу, включаючи зміну розміру та нормалізацію пікселів.
- Конволюційні шари (convolutional layers): Мережа містить багато конволюційних шарів, що дозволяє виконувати виявлення ознак на різних рівнях зображення.
- Згортання (downsampling): Після кожних кількох конволюційних шарів виконується згортання з метою зменшення розміру зображення та збільшення числа функцій активації.
- З'єднання (concatenation): На кінці кожного блоку конволюційних шарів виконується з'єднання функцій активації з різних рівнів.
- Повторне згортання (upsampling): Після з'єднання виконується повторне згортання з метою збільшення розміру зображення.
- Повно з'єднані шари (fully connected layers): На останньому етапі мережа використовує повно з'єднані шари, що дозволяє згенерувати прогнозовані координати та класи об'єктів.
- Виведення результатів (output): Остаточний вектор результатів містить інформацію про класи об'єктів та їхні координати на зображенні.

У YOLOv8 використовуються декілька інновацій, що покращують точність та швидкість розпізнавання об'єктів. Зокрема, використовується адаптивний шар, який дозволяє змінювати розмір фільтра згортки в залежності від розміру об'єкта на зображенні. Також використовуються механізми аугментації даних та зменшення впливу шуму на зображенні.

Архітектура YOLOv8 дозволяє досягти високої точності розпізнавання об'єктів при високій швидкості обробки зображень. Вона використовується в багатьох застосунках, включаючи системи безпеки, автономні транспортні засоби та робототехніку. [9] [1]

Інформація про кожен з цих шарів міститься в оригінальній статті про YOLOv8 та може бути складною для сприйняття без спеціального математичного підгрунтя. Проте, загальну ідею можна зрозуміти таким чином: YOLOv8 використовує нейронну мережу для обробки відео та пошуку об'єктів у кадрах. Ця мережа складається з багатьох шарів, кожен з яких виконує певну операцію з вхідними даними. Результати обробки передаються на вихід шару, який відповідає за розпізнавання об'єктів та їх класифікацію.

У загальному випадку, YOLOv8 може бути поділений на три частини:

Вхідний шар: в якому вхідне відео розбивається на окремі кадри та змінює свою форму, щоб можна було передати його до наступних шарів мережі.

Основна частина: складається з декількох повторюваних блоків, кожен з яких містить кілька сверточних та підвибіркових шарів. Ці блоки роблять вищеописану обробку даних та допомагають виявляти об'єкти на кадрах.

Вихідний шар: отримує результати обробки відео з основної частини мережі та повертає відповідність об'єктів та їх класифікацію у вигляді чисел.

Така архітектура дозволяє YOLOv8 працювати досить швидко та ефективно, що робить його ідеальним кандидатом для застосування в додатках для пошуку військової техніки та військових на відео.

1.5. Навчання YOLO v8

Навчання YOLO v8 - це процес, що вимагає значної кількості обчислювальних ресурсів, знань з глибинного навчання та експериментування з параметрами моделі. Основна особливість навчання YOLO v8 полягає в тому, що використовується архітектура мережі з великою кількістю шарів, що має багато параметрів. Тому, навчання моделі може займати від декількох годин до кількох днів, в залежності від розміру датасету та наявності необхідних ресурсів.

Для навчання YOLO v8 потрібно мати достатню кількість даних для тренування. Основна особливість даних для навчання - це мітки, що відповідають реальним об'єктам на зображеннях. Ці мітки повинні бути визначені точно та повністю, щоб модель могла правильно розпізнавати об'єкти на зображеннях. Навчальний датасет поділяється на тренувальний, валідаційний та тестовий набори для ефективної оцінки результатів.

Навчання YOLO v8 також потребує наявності обчислювального обладнання з високою продуктивністю, так як велика кількість даних потребує багато ресурсів для обробки. Для навчання моделі зазвичай використовуються графічні процесори (GPU) або спеціальні послідовні обчислювальні пристрої, такі як Tensor Processing Unit (TPU), що забезпечують високу продуктивність обчислень.

Навчання YOLO v8 також потребує ретельного налаштування гіперпараметрів моделі, таких як швидкість навчання (learning rate), кількість епох (epochs), розмір пакетів даних (batch size), з якими модель буде навчатися. Швидкість навчання - це крок, на який ваги моделі оновлюються під час навчання. Якщо швидкість навчання вибрана занадто високою, модель може пропустити оптимальну точку і не зможе знайти оптимальних ваг. З іншого боку, якщо швидкість навчання занадто низька, то навчання може займати надто багато часу.

Кількість епох визначає, скільки разів модель буде проходити через весь навчальний набір даних. Якщо кількість епох занадто мала, то модель не матиме достатньої інформації, щоб визначити патерни та залежності між даними. З іншого боку, якщо кількість епох занадто велика, то це може призвести до перенавчання моделі.

Розмір пакетів даних впливає на швидкість навчання та точність моделі. Більші пакети даних можуть зменшити кількість кроків, потрібних для навчання моделі, але можуть також зменшити точність моделі, оскільки ваги можуть бути оновлені занадто швидко з великою кількістю даних. З іншого боку, менші пакети даних забезпечують більш точне оновлення ваг, але можуть збільшити кількість кроків, потрібних для навчання моделі.

Навчання YOLO v8 вимагає значних обчислювальних ресурсів, тому можливо потрібна глибокої навчання на високопотужних графічних процесорах (GPU) або на хмарних платформах, таких як Amazon AWS або Microsoft Azure. Навчання може займати багато часу в залежності від розміру датасету та обсягу обчислювальних ресурсів. Однак, з правильно підібраними гіперпараметрами та відповідно налаштованою архітектурою моделі, YOLO v8 може дати досить точні результати для різноманітних завдань в області комп'ютерного зору, таких як розпізнавання об'єктів та їх класифікація, виявлення дій та прогнозування майбутніх подій.

Додатково, однією з особливостей YOLO v8 є можливість використовувати техніку transfer learning, що дозволяє дошкільно навчити модель на великому датасеті, а потім використовувати її для вирішення завдань на меншому датасеті, що дозволяє економити час та ресурси на навчання моделі з нуля.

У цілому, YOLO v8 є потужним інструментом для розв'язання задач комп'ютерного зору, що вимагають розпізнавання об'єктів та їх класифікації. Проте, для досягнення точних результатів, важливо мати достатньо великий та

репрезентативний датасет, налаштувати гіперпараметри та використовувати потужні обчислювальні ресурси.

1.6. Android Studio

Для розробки застосунку для пошуку військової техніки та військових на відео ми використаємо Android Studio, яка є інтегрованою середовищем розробки (IDE) для розробки мобільних застосунків під операційну систему Android. Ми також використаємо мову програмування Kotlin, яка є однією з найбільш популярних мов програмування для розробки застосунків для Android.

Android Studio - це інтегроване середовище розробки (IDE) для створення застосунків для операційної системи Android. Воно базується на платформі IntelliJ IDEA від компанії JetBrains і містить в собі набір інструментів для розробки, тестування та налагодження Android-додатків.

Android Studio надає розробникам широкі можливості для створення різноманітних додатків. У редакторі коду можна писати код на Java або Kotlin, створювати графічний інтерфейс користувача, працювати з базами даних та використовувати різноманітні бібліотеки та фреймворки.

Основні переваги Android Studio:

- Інтегровані засоби для розробки та тестування Android-додатків.
- Зручний інтерфейс, який дозволяє швидко створювати та налаштовувати проекти.
- Підтримка Java та Kotlin, що дає можливість розробникам вибирати мову програмування, яку вони найкраще володіють.
- Велика кількість плагінів та бібліотек, які розширюють можливості розробки.
- Підтримка системи контролю версій Git.

Android Studio - це основний інструмент для розробки Android-додатків та надає розробникам всі необхідні інструменти для успішної роботи.

Загальна структура Android Studio складається з наступних компонентів:

- Редактор коду: Android Studio має вбудований редактор коду, який підтримує різні мови програмування, включаючи Java, Kotlin та C++. Редактор коду містить ряд інструментів для автоматичного завершення кодування, перевірки синтаксису та аналізу коду.
- Дизайнер інтерфейсу: Android Studio має вбудований дизайнер інтерфейсу, який дозволяє розробникам створювати графічний інтерфейс користувача за допомогою перетягування та розміщення різних елементів інтерфейсу.
- Gradle Build System: Gradle є системою збірки, яка використовується для компіляції та збирання додатків Android. Android Studio має вбудований Gradle Build System, який дозволяє розробникам налаштувати та збирати свої додатки.
- Android Virtual Device Manager: Android Studio має вбудований Android Virtual Device Manager, який дозволяє розробникам створювати та керувати віртуальними пристроями Android, щоб перевіряти свої додатки на різних конфігураціях пристроїв.
- Debugger: Android Studio має вбудований debugger, який дозволяє розробникам налагоджувати свої додатки в режимі реального часу та знаходити помилки.
- Profiler: Android Studio має вбудований Profiler, який дозволяє розробникам відстежувати та аналізувати роботу своїх додатків, включаючи використання пам'яті та процесора.

Android Studio надає розробникам можливість створювати різноманітні додатки для Android, включаючи ігри, мультимедійні додатки, соціальні мережі, додатки для бізнесу та інші. Розробники можуть використовувати Android

1.7. Kotlin

Kotlin - це мова програмування, що розроблена компанією JetBrains в 2011 році. Вона є статично типізованою, об'єктно-орієнтованою мовою програмування, яка компілюється в байт-код JVM та може бути використана для розробки програмного забезпечення на різних платформах, включаючи Android, серверні додатки та веб-розробку.

Основні особливості Kotlin:

- Короткість та читабельність: Kotlin має багато коротких синтаксичних конструкцій, які дозволяють розробникам писати менше коду та зробити його більш читабельним.
- Безпечність типів: Kotlin має статичну типізацію, що дозволяє розробникам виявляти помилки в процесі компіляції, зменшуючи кількість помилок в час виконання.
- Null-безпека: Kotlin має вбудовану підтримку null-безпеки, що дозволяє зменшити кількість помилок, пов'язаних з нульовими посиланнями.
- Розширення функціональності Java: Kotlin може бути використана з Java-бібліотеками та фреймворками, що дозволяє розробникам зберігати свій наявний Java-код та додавати нову функціональність.
- Мультиплатформеність: Kotlin може бути використана для розробки додатків на різних платформах, таких як Android, сервери, браузери, мобільні пристрої та інші.
- Компіляція до байт-коду JVM: Kotlin може бути компільована до байт-коду JVM, що дозволяє використовувати її на будь-якій платформі, що підтримує JVM.
- Функціональне програмування: Kotlin має підтримку функціонального програмування, що дозволяє розробникам використовувати функції вищого порядку, замикання та інші концепції функціонального програмування.

- Розширення: Kotlin дозволяє розширювати класи, не вносячи змін в самі класи. Це забезпечує більшу гнучкість та дозволяє розробникам збільшувати функціональність існуючих класів.
- Інтероперабельність з Java: Kotlin може інтегруватися з Java-кодом, що дозволяє розробникам використовувати бібліотеки та фреймворки, написані на Java.
- Асинхронне програмування: Kotlin має вбудовану підтримку асинхронного програмування з використанням корутин.

Kotlin стає все більш популярною серед розробників, оскільки вона дозволяє писати менше коду, зменшує кількість помилок та підвищує продуктивність розробників. Вона має чудову інтеграцію з Android Studio та дозволяє розробникам писати більш безпечний та ефективний код для Android-додатків.

1.8 Використані бібліотеки

TensorFlow Lite є спеціальною версією бібліотеки TensorFlow для мобільних та вбудованих пристроїв, яка була оптимізована для прискорення роботи машинного навчання на мобільних пристроях та забезпечення швидкої та ефективної роботи на пристроях з обмеженими ресурсами. Переваги:

- Швидкість та продуктивність: TensorFlow Lite був розроблений з метою підвищення продуктивності та зменшення витрат енергії для машинного навчання на мобільних та вбудованих пристроях. TensorFlow Lite дозволяє виконувати обчислення на графічних процесорах (GPU) та інших прискорювачах, що забезпечує значне прискорення роботи моделей машинного навчання.
- Легкість використання та інтеграції: TensorFlow Lite забезпечує зручний інтерфейс програмування додатків (API) для розробників мобільних додатків, що дозволяє легко включати машинне навчання в свої додатки. TensorFlow Lite підтримує багато мов програмування, включаючи Java,

C++, Python та Swift, що дозволяє розробникам використовувати ту мову програмування, яку вони воліють.

- Підтримка різних форматів моделей: TensorFlow Lite підтримує різні формати моделей, включаючи TensorFlow SavedModel, Keras Model та TensorFlow Lite Model. Це дозволяє розробникам легко інтегрувати моделі, які були навчені за допомогою різних фреймворків машинного навчання, в TensorFlow Lite.
- Розмір: TensorFlow Lite має дуже маленький розмір, що дозволяє легко включати його в мобільні додатки. TensorFlow Lite може бути дуже корисним для додатків, які працюють з нейронними мережами.

TensorFlow Lite може працювати на різних мобільних платформах, таких як Android, iOS та деякі мікроконтролери, тому вона є універсальним інструментом для розробки мобільних додатків з машинним навчанням. Крім того, TensorFlow Lite підтримує кілька форматів моделей, включаючи TensorFlow, Keras та інші, що дозволяє легко переносити моделі між платформами.

Також варто відзначити, що TensorFlow Lite має декілька функцій, які забезпечують максимальну продуктивність на мобільних пристроях, таких як квантизація моделей, оптимізація виконання графів, а також спеціальні обчислювальні бібліотеки, які оптимізовані для роботи з певними типами процесорів.

Узагалі, TensorFlow Lite є потужним інструментом для розробки мобільних додатків з машинним навчанням, який дозволяє легко і швидко впроваджувати моделі машинного навчання на мобільних пристроях та забезпечувати ефективну роботу на пристроях з обмеженими ресурсами. [21] [22]

Multik - це бібліотека для мови програмування Kotlin, яка надає масивні обчислення, підтримує багатовимірні масиви і пропонує різноманітні операції над ними. Бібліотека multik була розроблена з урахуванням вимог до швидкодії і масштабовності, що дозволяє розробникам легко та швидко реалізовувати складні обчислення в Kotlin.

Однією з переваг бібліотеки Multik є те, що вона надає багато функцій для обробки масивів, такі як транспонування, зведення масиву до вектора, розбиття на підмасиви тощо. Крім того, бібліотека підтримує індексацію, що дозволяє вибирати елементи з масиву за допомогою індексів.

Іншою перевагою бібліотеки Multik є те, що вона може працювати з масивами, що містять дані різних типів. Бібліотека підтримує різні типи даних, такі як цілі числа, числа з плаваючою комою, булеві значення та інші.

Крім того, бібліотека Multik дозволяє розробникам працювати з масивами різних розмірностей. Вона може підтримувати масиви від одновимірних до багатовимірних, що дозволяє розробникам створювати складні обчислювальні моделі та алгоритми.

Окрім цього, бібліотека Multik підтримує зручний спосіб інтерполяції, що дозволяє розробникам побудувати функції на основі заданих точок і значень. Це дуже корисна функція для роботи з даними, де потрібно оцінити значення функції у певній точці. Бібліотека multik містить реалізації базових алгоритмів лінійної алгебри, таких як обчислення векторних та матричних операцій, зведення матриць до симетричних форм, обернення матриць та розв'язання систем лінійних рівнянь. multik була спроектована з метою полегшити роботу з матрицями в Kotlin та забезпечити гнучкий та швидкий інструментарій для вирішення задач, пов'язаних з аналізом даних.

Однією з ключових переваг бібліотеки multik є її спрощена синтаксис, що зроблює програмування більш зрозумілим і менш помітним.

multik також підтримує роботу з різними типами даних, такими як Float, Int, Long та Boolean, та здатна працювати з многовимірними масивами будь-якої

форми та розміру. Це дає можливість більш гнучко працювати з даними та заощаджувати час, який витрачається на перетворення даних в потрібний формат.
multik також надає розширення для Kotlin, які дозволяють зручно використовувати функції бібліотеки в будь-якому місці коду. [23]

OpenCV 4 Android - це бібліотека комп'ютерного зору з відкритим кодом, яка надає програмістам можливість розробляти застосунки для платформи Android з використанням функцій комп'ютерного зору, таких як розпізнавання обличчя, зіставлення зразків, детекція руху, виявлення контурів і зведення зображення.

Бібліотека OpenCV для Android має наступні переваги:

- Багатофункціональність: OpenCV 4 Android надає велику кількість функцій комп'ютерного зору, які можна використовувати для розробки різноманітних застосунків.
- Відкритий код: OpenCV 4 Android є бібліотекою з відкритим кодом, що дозволяє програмістам налаштовувати її функції та використовувати їх у своїх застосунках.
- Швидкодія: OpenCV 4 Android має високу швидкодію, що дозволяє виконувати розрахунки і обробку зображень в реальному часі.
- Підтримка відомих платформ: OpenCV 4 Android підтримує відомі платформи, такі як Android, iOS, Windows і Linux.
- Легка інтеграція: OpenCV 4 Android легко інтегрується з іншими бібліотеками, такими як TensorFlow Lite, що робить його дуже популярним для розробки застосунків зі штучним інтелектом.
- Розширюваність: OpenCV 4 Android дозволяє розширювати функціональність застосунку за допомогою додаткових модулів, таких як opencv_contrib.

- Наявність документації: OpenCV 4 Android має детальну документацію та підтримку спільноти, що дозволяє легко знайти рішення на будь-які питання стосовно розробки застосунків з використанням OpenCV.

РОЗДІЛ 2 НАВЧАННЯ МОДЕЛІ ДЛЯ ПОШУКУ ВІЙСЬКОВИХ ОБ’ЄКТІВ НА ЗОБРАЖЕННІ

Для розробки застосунку для пошуку військової техніки та військових на відео ми будемо використовувати YOLOv8 як основну архітектуру для розпізнавання об'єктів. Ми будемо навчати модель на великому наборі зображень військової техніки та військових, що дозволить нам розпізнавати їх на відео.
Для побудови додатку ми будемо використовувати мову програмування Kotlin, IDE Android Studio а також бібліотеки OpenCV для обробки відео та TensorFlow для використання навченої моделі YOLO v8. Ми також будемо використовувати сервіс Google Colab Pro (Python) для навчання та обробки великої кількості зображень.

Загальна архітектура нашого додатку буде включати в себе такі компоненти:

1. Збір відео - ми зберігатимемо відео з камери або дрона.
2. Обробка відео - ми будемо використовувати OpenCV для обробки відео та виявлення кадрів з військовою технікою та військовими.
3. Розпізнавання об'єктів - ми будемо використовувати модель YOLOv8 для розпізнавання військової техніки та військових на кадрах з відео.

4. Підсвічування об'єктів - ми будемо виділяти знайдені об'єкти на відео за допомогою рамок або інших візуальних елементів.
5. Відображення результатів - ми будемо відображати результати на відео та зберігати їх для подальшого аналізу.
6. В результаті розробки нашого додатку, користувачі зможуть швидко та легко знайти військову техніку та військових на відео, що дозволить їм ефективно використовувати цю інформацію для різ

2.1. Підготовка даних

Машинне навчання допомагає нам знаходити закономірності в даних - шаблони, які ми потім використовуємо для прогнозування нових даних. Щоб ці прогнози були правильними, ми повинні побудувати набір даних і правильно перетворити дані. Якщо є багато недоречної та надлишкової інформації, або наявні дані ненадійні чи зашумлені, то виявлення закономірностей на етапі навчання є складнішим. Етапи підготовки та фільтрації даних можуть зайняти значну кількість часу на обробку. Попередня обробка даних включає очищення, вибір екземпляра, нормалізацію, трансформацію, вилучення та виділення особливостей тощо. Результат попередньої обробки даних є остаточним навчальним набором.

Попередня обробка даних може вплинути на те, як ми інтерпретуємо остаточні дані. [30] Цей аспект слід ретельно розглянути, коли інтерпретація результатів є ключовим моментом.

У нашому випадку всі зображення приводяться до розміру 640x640 пікселів. Це дозволяє використовувати YOLOv8 з параметрами за замовчуванням.

Набір даних можна розглядати як сукупність об'єктів даних, які часто також називають записами, точками, векторами, шаблонами, подіями, випадками, зразками, спостереженнями або сутностями.

Об'єкти даних описуються низкою ознак, що охоплюють основні характеристики об'єкта, такі як маса фізичного об'єкта або час, коли сталася подія тощо. Функції часто називають змінними, характеристиками, полями, атрибути або розміри.

Особливості можуть бути:

Категоричні: Функції, значення яких беруться з певного набору значень. Наприклад, дні в тижні: {понеділок, вівторок, середа, четвер, п’ятниця, субота, неділя} - це категорія, оскільки її значення завжди береться з цього набору. Іншим прикладом може бути логічний набір: {True, False}

Числові: Функції, значення яких є безперервними або цілочисельними. Вони представлені числами і володіють більшістю властивостей чисел. Наприклад, кількість кроків, які ви проходите за день, або швидкість руху вашої машини.

Оцінка якості даних

Оскільки дані часто беруться з безлічі джерел, які зазвичай не надто надійні, а також у різних форматах, більше половини нашого часу витрачається на вирішення питань якості даних під час роботи над проблемою машинного навчання. Очікувати, що дані будуть ідеальними, просто нереально. Можуть виникнути проблеми через людські помилки, обмеження вимірювальних приладів або недоліки в процесі збору даних. Давайте розглянемо декілька з них та методи боротьби з ними:

Відсутні значення:

Дуже звично мати відсутні дані у наборі даних. Можливо, це сталося під час збору даних, або, можливо, через якесь правило перевірки даних, але незалежно від цього відсутність значень необхідно враховувати.

Виключити зображення з відсутніми даними:

Проста і часом ефективна стратегія. Якщо в об’єкті здебільшого відсутні значення, то сам цей об'єкт також може бути усунений.

Оцінка відсутніх значень:

Якщо бракує лише розумного відсотка значень, ми також можемо запустити прості методи інтерполяції для заповнення цих значень. Однак найпоширенішим методом роботи з відсутніми значеннями є заповнення їх середнім, медіанним або значенням режиму відповідної ознаки.

В процесі збору власних даних, ми отримали певну кількість зображень (близько 10%), які не містили ніяких військових об’єктів. Було вирішено залишити такі зображення. Причина - навчити модель того, що на зображенні не завжди повинні бути об’єкти. Насправді, ми передбачаємо, що більшість зображень з дронів не містять військової техніки.

Рис. 7. Приклад зображення без об’єктів.

На рис. 7 видно приклад зображення, на якому немає військової техніки чи військових. Цей кадр отриманий одразу після влучання снаряду у бронетехніку і, через задимлення, її не видно.

Невідповідні значення:

Ми знаємо, що дані можуть містити суперечливі значення. Наприклад, поле «Адреса» містить «Номер телефону». Можливо, це пов’язано з людською

помилкою або, можливо, інформація була прочитана неправильно під час сканування з рукописного документу. Тому завжди рекомендується проводити оцінку даних, наприклад, знаючи, яким має бути тип даних об’єктів і чи однаковий він для всіх об’єктів даних.

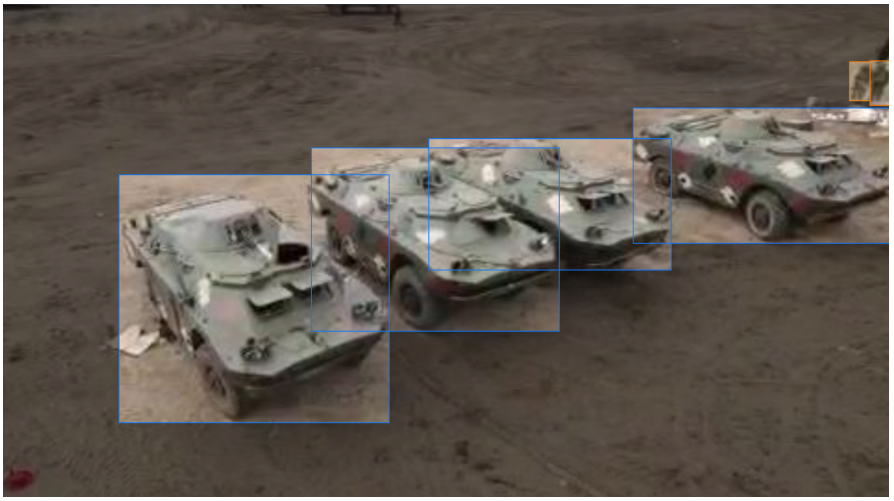

Рис. 8 Приклад зображення з відкритого датасету. Розмічена техніка позначена як “tank”, що не відповідає нашому класу “vehicle”.

Для розширення власного датасету було додано датасети з відкритого доступу. [11] [10] Ці датасети були перевірені на відповідність. Загалом розмітка відповідала якості нашого датасету, проте містила значно більшу кількість класів для розпізнавання. Наприклад типи військової техніки. Ідея нашою моделі - знаходження техніки на фото, а не розпізнавати її модель. Для приведення датасету до нашої потреби було додано крок передпроцесингу - ми створили відповідність між класами з нашого датасету та доданих. Отже в нашому датасеті використовуються лише два класи - техніка та люди.

Повторювані значення:

Набір даних може включати об’єкти даних, які є дублікатами один одного. Це може статися, коли, скажімо, одна і та ж особа подає форму неодноразово. Термін дедуплікація часто використовується для позначення процесу роботи з дублікатами. У більшості випадків дублікати видаляються, щоб не дати конкретному об'єкту даних перевагу або упередження під час запуску алгоритмів машинного навчання.

В нашому випадку повторів не було виявлено, проте деякі зображення дуже схожі оскільки є кадрами з одного відео. В такому випадку ми додавали лише кадри, які досить сильно різняться.

Агрегація даних

Агрегації даних виконується таким чином, щоб взяти агреговані значення наявних даних для їх покращення. Прикладом цього є дані транзакцій, припустимо, у нас є повсякденні операції з продуктом, що реєструють щоденні продажі цього товару в різних магазинах протягом року. Агрегація транзакцій до єдиних щомісячних або річних транзакцій магазину допоможе нам скоротити сотні або потенційно тисячі транзакцій, які відбуваються щодня в певному магазині, тим самим зменшуючи кількість об'єктів даних.

Це призводить до зменшення споживання пам'яті та часу обробки. Агрегації забезпечують нам високий рівень даних, оскільки поведінка груп або агрегатів стабільніша, ніж окремі об'єкти даних.

Збір даних.

Однією з ключових переваг YOLO v8 є можливість навчання моделі на власних даних. Це означає, що ми можемо зібрати власний датасет військової техніки та військових на відео, а потім навчити модель розпізнавати ці об'єкти на нових відео.

Для цього потрібно ретельно промаркувати кожен об'єкт на відео, тобто визначити його клас та координати на кадрі. Для покращення точності розпізнавання можна також використовувати техніки аугментації даних, такі як зміна розміру, повороти, зміщення тощо.

Після того, як датасет буде готовий, ми можемо використати інструменти YOLO для навчання моделі на цих даних. Цей процес може зайняти деякий час, залежно від розміру датасету та складності задачі. Однак, після навчання моделі

буде можливість використовувати її для розпізнавання військової техніки та військових на нових відео з точністю, яка перевищує можливості більш традиційних методів.

Використання власного датасету та навчання моделі на ньому є важливим кроком у забезпеченні максимально точного та ефективного розпізнавання військової техніки та військових на відео.

Отримати якісні дані про військову техніку та військових є складним завданням, оскільки доступ до відео з дронів зазвичай відкривається тільки після їх редагування. Більшість з таких відео, які потрапляють у вільний доступ, мають низьку якість зображення. Це може бути викликано різними факторами, наприклад погані погодні умови, або методи кодування та передачі файлів, які використовуються при їх створенні.

Однак, основним джерелом даних для нашого проекту став месенджер Telegram. Telegram має багато каналів, де поширюються відео з дронів, що можуть бути корисними для наших цілей. Незважаючи на те, що вони зазвичай відредаговані та стиснуті, вони все ще можуть бути використані для навчання.

Використання Telegram як джерела даних має свої переваги.

По-перше, Telegram є широко поширеним месенджером, що означає, що ми маємо доступ до великої кількості каналів, які поширюють відео з дронів. Це дає нам можливість зібрати великий обсяг даних для навчання.

По-друге, Telegram має зручний інтерфейс та добре організовану структуру каналів, що спрощує пошук відео, пов'язаних з військовою технікою та військовими.

Незважаючи на те, що багато відео, які ми знайдемо в Telegram, можуть бути відредаговані та стиснуті, ми все ще можемо використати їх як початкові дані для нашого застосунку.

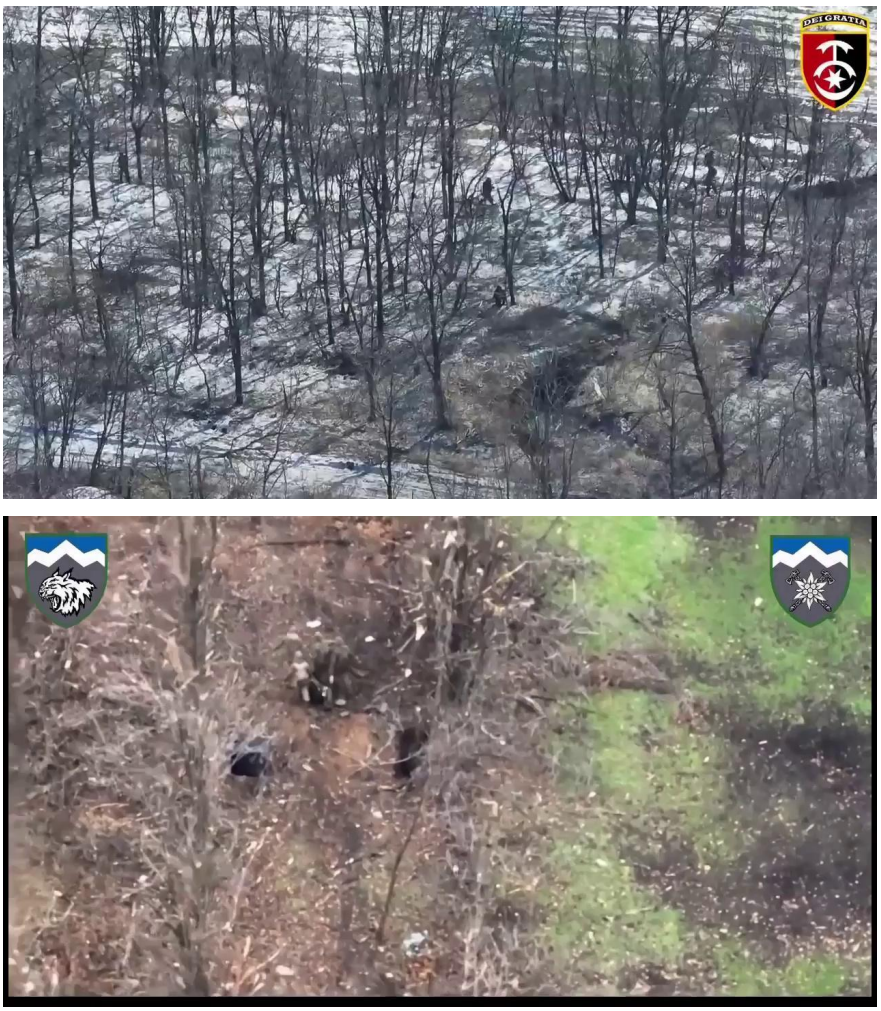

Рис. 9. Приклад зображень з військовими

Рис. 10. Приклад зображень з військовою технікою

Крім зображень зібраних з телеграму, в датасет було додано вже розмічені зображення з інших датасетів: [11] [10].

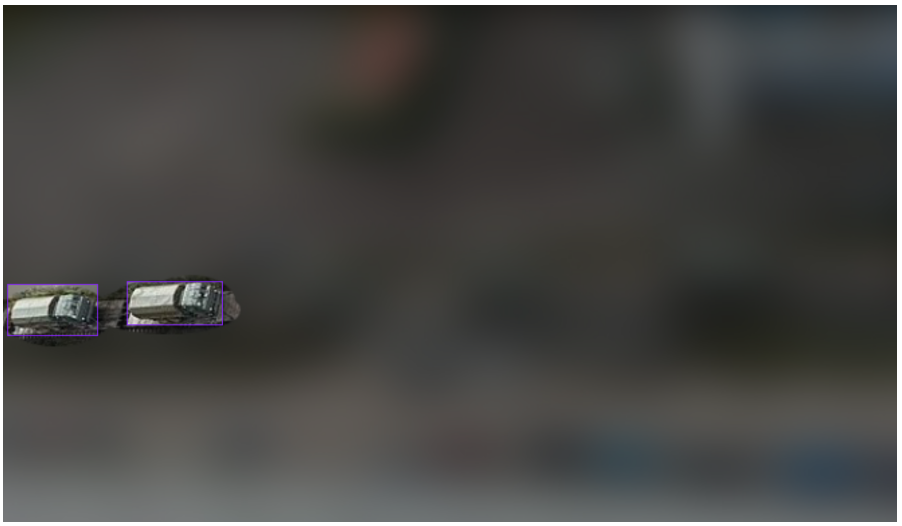

Рис. 11Приклад розміченого зображення з датасету [10]

Рис. 12. Приклад розміченого зображення з датасету [11]

Готові датасети уже розмічені, але очевидно, що наш датасет потрібно розмітити. Для цього завдання було використано сайт Roboflow.com.

Рис 13. Приклад розміченого зобаження з власного датасету.

Розмір зібраного датасету - близько трьох з половиною тисяч зображень. Цю кількість можна збільшувати, що покращить якість нетренованої моделі. Також було використано метод доповнення даних. Серед перетворень були такі як поворот зображення чи випадкова обрізка. Датасет після доповнення збільшився до семи тисяч зображень.

При створенні моделі машинного навчання, набір даних зазвичай розбивається на три частини: тренувальні, валідаційні та тестові дані.

Тренувальні дані - це набір даних, який використовується для навчання моделі, щоб вона могла навчитися розпізнавати об'єкти на відео. Зазвичай ця частина даних складається з більшої кількості прикладів, бо саме на цій стадії модель навчається визначати загальні ознаки і шаблони у зображеннях.

Валідаційні дані - це набір даних, який використовується для оцінки продуктивності моделі під час навчання, а саме - для оцінки точності і запобігання перенавчанню. На цій стадії важливо перевірити, як добре модель працює з новими зображеннями. Зазвичай валідаційна вибірка складає близько 20% від усіх доступних даних.

Тестові дані - це окремий набір даних, який використовується для оцінки кінцевої продуктивності моделі. Ця частина даних зазвичай збільшується до 30% від усіх доступних даних, оскільки це дозволяє забезпечити більш точну оцінку продуктивності моделі.

Загальні пропорції тренувальної, валідаційної та тестової вибірок можуть змінюватись в залежності від розміру даних, а також від складності задачі. Наприклад, у випадку, коли набір даних досить великий, а завдання неважке, можна виділити більшу кількість даних для тестування. Однак, якщо ми маємо дуже обмежений набір даних, то можна зменшити відсоток даних, призначених для валідації та тестування, і збільшити розмір навчального датасету. Таким чином, ми зможемо максимально ефективно використовувати наявні дані для навчання нашої моделі. Однак, це може призвести до перенавчання моделі, тому потрібно забезпечити, щоб розміри кожної частини датасету були відповідні для забезпечення рівноваги між точністю та загальною прохідністю моделі. Також, важливо забезпечити, щоб дані в кожній частині датасету були репрезентативні для повної генералізації моделі. У нашому випадку будемо дотримуватись пропорції 90%-10%-10%

Roboflow дозволяє завантажити датасет у потрібному форматі для навчання моделі YOLO v8 -

- папка test містить розмічені зображення для тестування (10%)

- папка train містить розмічені зображення для тренування (90%)
- папка valid містить розмічені зображення для валідації (10%)
- файл data.yaml деякі необхідні дані, наприклад класи для розпізнавання і шлях до папок з даними

2.2. Навчання

Задача навчання моделі YOLO v8 на власному датасеті є важкою і вимагає значних обчислювальних ресурсів. Однак, використовуючи Google Colab Pro, можна вирішити цю проблему.

Для початку, необхідно завантажити власний датасет та підготувати його до навчання. Наступним кроком є налаштування середовища для навчання моделі. У Google Colab Pro можна використовувати безкоштовні графічні процесори (GPU) або сплатити за використання більш потужних ресурсів, таких як TPU.

Після цього можна починати навчання моделі. Для цього необхідно налаштувати конфігураційний файл, який визначає параметри моделі та шлях до датасету, та запустити навчання.

Під час навчання необхідно слідкувати за метриками, такими як середня помилка локалізації та класифікації, та візуалізувати результати за допомогою відповідного програмного забезпечення. Після закінчення навчання можна оцінити якість моделі на тестовому наборі даних.

Навчання моделі YOLO v8 на власному датасеті за допомогою Google Colab Pro може зайняти від кількох годин до кількох днів, залежно від розміру датасету та обсягу обчислювальних ресурсів. Однак, після успішного навчання, отримана модель може бути використана для розв'язання практичних задач, таких як детекція об'єктів на зображеннях або відео.

YOLOv8 має різні розміри, кожен з яких має свої переваги і недоліки.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU (ms)	Speed T4 GPU (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	-	-	3.2	8.7
YOLOv8s	640	44.9	-	-	11.2	28.6
YOLOv8m	640	50.2	-	-	25.9	78.9
YOLOv8l	640	52.9	-	-	43.7	165.2
YOLOv8x	640	53.9	-	-	68.2	257.8

Рис. 14. Порівняння показників моделі різних розмірів.

YOLOv8n

Для початкового навчання було обрано найменшу модель - YOLOv8n (n означає nano). Як видно, її точність найменша серед запропонованих, але навчання було досить швидким, щоб виконати його на поцесорі власного пк. Датасет на цей момент був зібраним власноруч і дуже маленьким - 250 зображень. На цьому етапі нас цікавило чи взагалі виконувана поставлена задача. Далі наведемо результати:

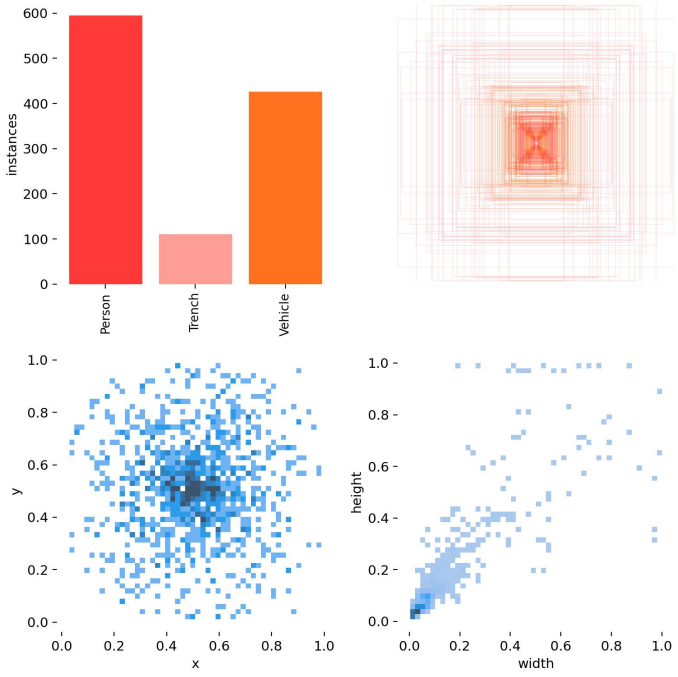

Рис. 15. Розмітка в початковому датасеті.

На рис 15видно перший датасет використаний для навчання. Як видно, крім техніки і людей тут є третій клас - окоп. Спочатку ми хотіли розрізняти також окопи, проте як видно на рис 14, прикладів цих об’єктів дуже мало, а їх природа не дозволяла розмітити їх за допомогою прямокутників. Зазвичай окопи вузькі, довгі і хвилясті. Таким чином площа прямокутника, в який вміщався окоп зазвичай займала 80% зображення.

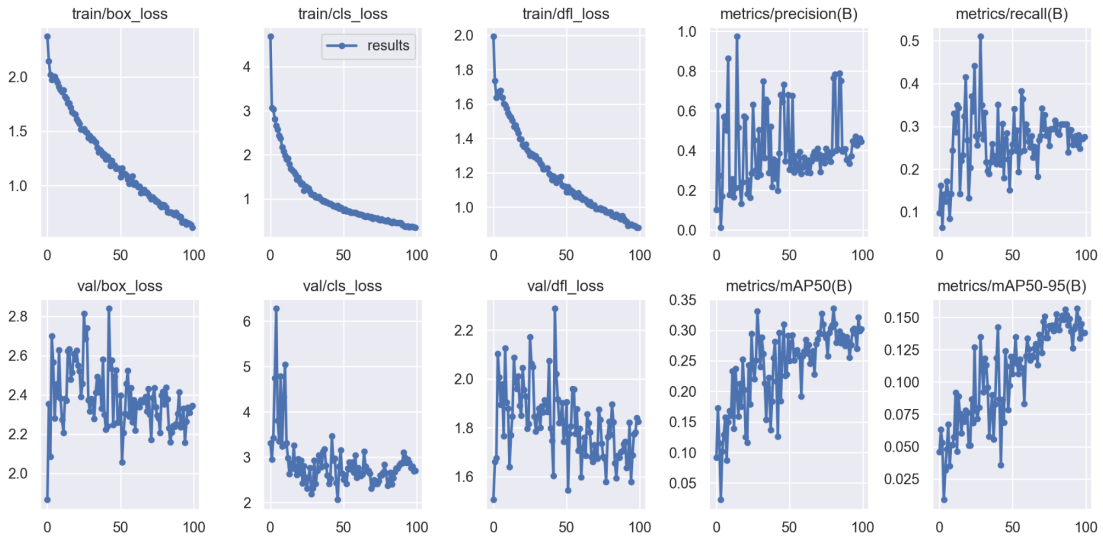

Рис. 16. Результати навчання після ста епох.

На рис 16 видно результати навчання. Як видно, було досягнено mAP близько 35%, що не є задовільним.

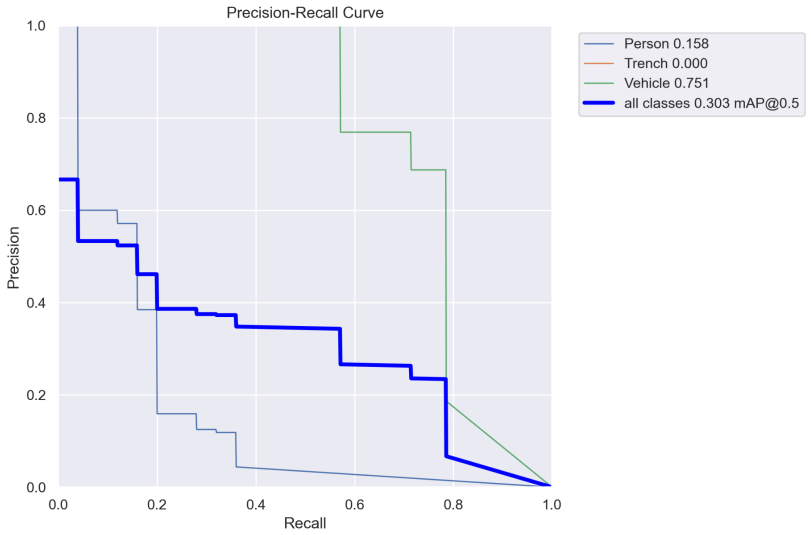

Рис 17. Precision-Recall curve

На рис 17. видно, що результати на окопах повністю незадовільні. На цьому етапі було вирішено видалити цей клас з датасету.

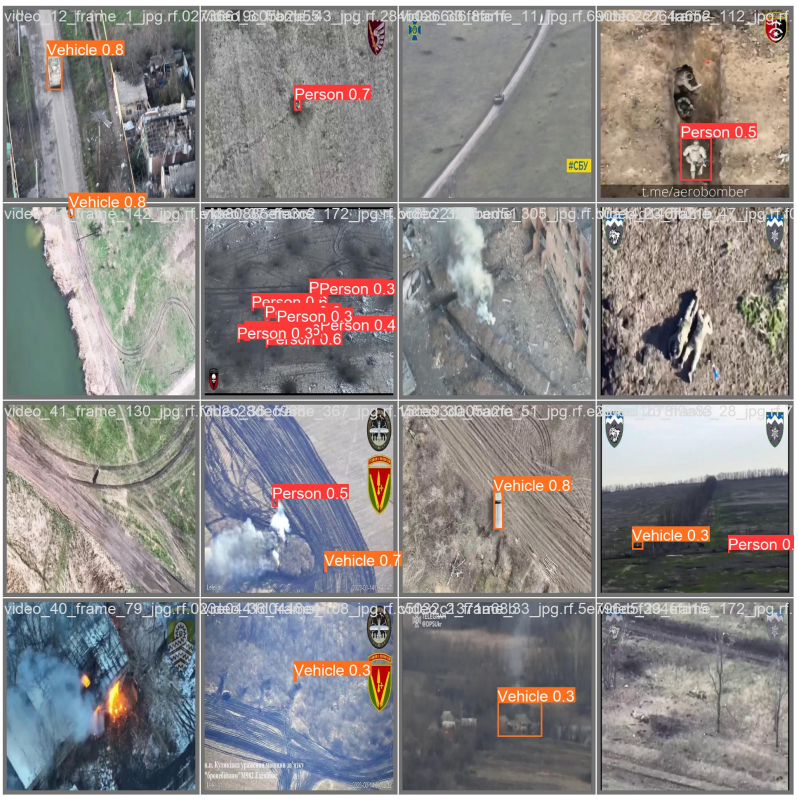

Рис. 18. Результати на валідаційній частині датасету

YOLOv8s

Наступним кроком в навчанні було використання більшої моделі - small На цьому етапі було вирішено додати датасети з відкритого доступу. Також для прискорення навчання, тепер навчання проводилось в Google Colab Pro. Pro версія дає доступ до потужних відеокарт, які значно прискорили навчання.

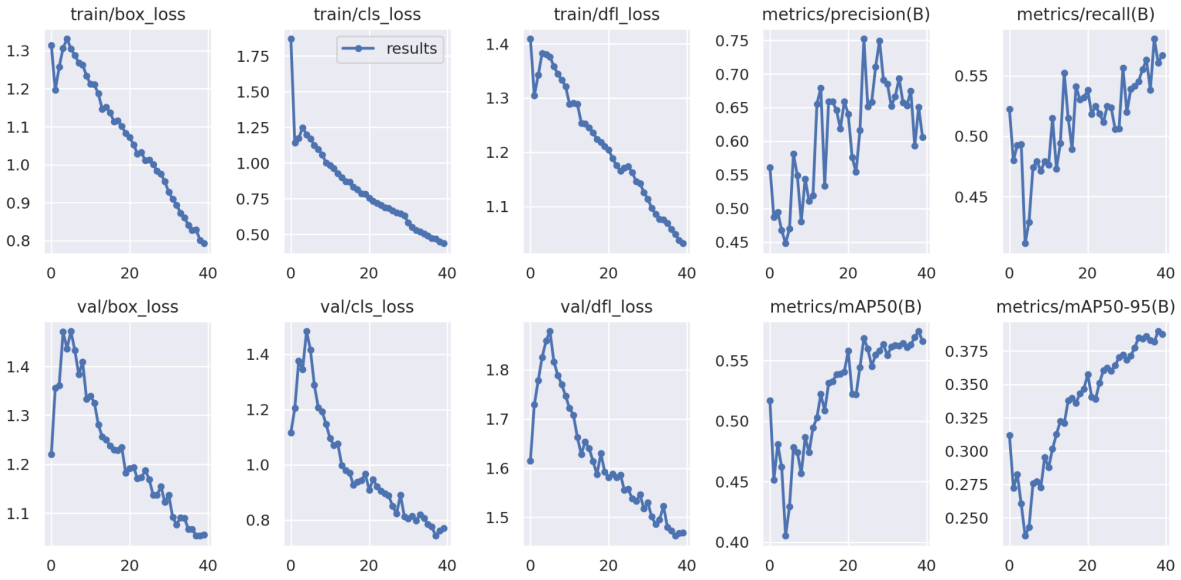

Рис. 19. Результати навчання після сорока епох.

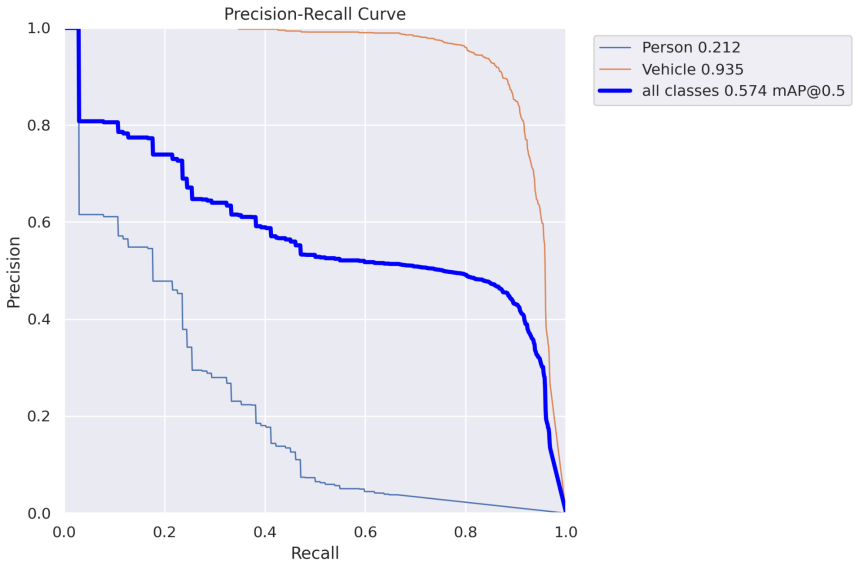

Рис. 20. PR-curve.

На рис. 19, що результати моделі значно покращились навіть після 40 епох. Також на PR-curve видно, що точність виявлення техніки набагато краща, ніж людей. Це може бути зв’язано з кількома причинами:

- Кількість об’єктів в датасеті.
- Розмір людей з висоти польоту дрона. Люди значно менші за техніку, а тому з однакової висоти помітити людину значно важче ніж техніку.
- Камуфляж. Військові ефективно використовують камуфляж і можуть пристосуватися до середовища. Змінити камуфляж техніки значно важче.

Через наведені причини, розглядався варіант відмовитись від класу людей в моделі.

Рис. 21. Результати на валідаційній частині датасету.

YOLOv8m

Очевидно, що збільшення розміру моделі та датасету принесли користь, тому можемо продовжити в тому ж дусі.

Цього разу використовуємо той самий датасет, але додамо етап аугментації. Після аугментації розмір датасету зріс до майже семи тисяч. Таке зростання датасету та розміру моделі сповільнило процес навчання, проте цього разу ми заморозили backbone моделі, а тому прискорили навчання. Крім прискорення навчання, цей крок зберіг вже навчену частину моделі, що відповідає за розпізнавання простих об’єктів (наприклад контури об’єктів).

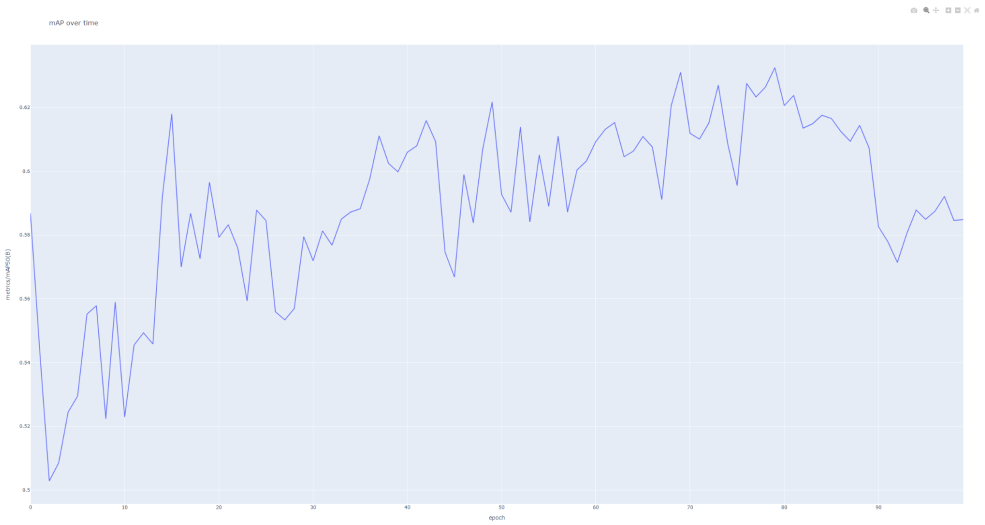

Рис. 22. mAP50 протягом ста епох

На рис. 22 показано результат навчання описаної моделі. Найкращий результат - 63.2%. Варто розуміти, що результати розпізнавання техніки все ще значно переважають результати розпізнавання людей.

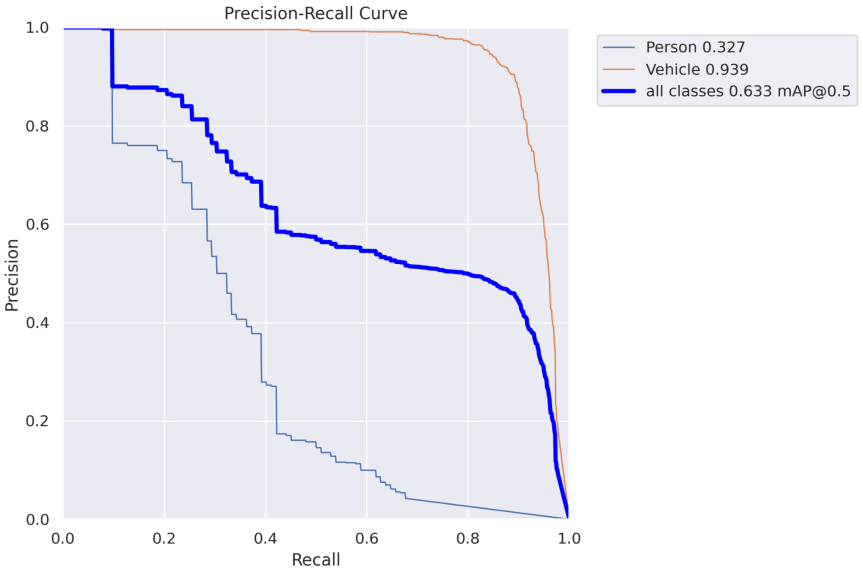

Рис. 23. PR-curve.

На рис. 23бачимо, що результати розпізнавання техніки приблизно втричі кращі, ніж розпізнавання людей. Проте бачимо покращення в розпізнаванні людей на цій ітерації моделі.

Рис. 24. Результати на валідаційному датасеті.

ВИСНОВКИ ДО РОЗДІЛУ 2

В розділі 2 було описано процес навчання моделі YOLOv8 на власному датасеті. YOLOv8 має різні розміри, які впливають на точність передбачень, розмір на диску і швидкість навчання. Було досліджено 3 розміри архітектури - nano, small, medium. Результати nano показали, що розпізнавання техніки і людей в нашому випадку є перспективними, а розпізнавання окопів - ні. Таким чином в наступній ітерації - small ми відмовились від класу “trench”. Після збільшення датасету та з використанням більшої моделі, покращення були очевидні, а тому було швидко прийнято рішення збільшити датасет та модель знову. Після доповнення датасету за допомогою трансформацій, використання моделі YOLOv8m та заморозки backbone моделі, було проведено навчання протягом ста епох. Результати розпізнавання значно покращились, проте точність розпізнавання техніки значно переважає точність розпізнавання людей, що може бути зв’язано з кількість розмічених військових в датасеті, їх розмірі в порівняння з технікою та кращим маскуванням люедй порівняно з технікою. Очевидним є те, що варто покращити точність розпізнавання військових, що буде зроблено в майбутньому.

Майбутні покращення можна отримати наступним шляхом:

- Доповнити датасет більшою кількістю зображень з телеграму, бажано з великою кількістю людей.
- Доповнити датасет зображеннями в літню пору року (зелений фон)
- Видалити з датасету зображення техніки, зроблені не з висоти
- Провести навчання на більшій кількості епох
- Отримати доступ до відео з дронів без обробки та стискання
- Додати більшу кількість трансформацій на етапі доповнення

Серед майбутніх покращень відсутня опція використання більшої моделі. Тренування моделі розміру large було неможливим на Google Colab через свій розмір. Також варто пам’ятати, що навчена модель буде використана на

мобільних пристроях, а отже розмір та швидкість моделі є дуже важливими. Large варіант моделі дає дуже невелику перевагу у точності в порівняння з моделлю medium, а тому ми будемо спиратись в майбутньому саме на модель medium.

РОЗДІЛ 3 РОЗРОБКА ЗАСТОСУНКУ З ВИКОРИСТАННЯМ НАВЧЕНОЇ МОДЕЛІ

З навченою моделлю, що може розпізнавати військову техніку та військових, залишилось помістити її у зручний інтерфейс для використання у війську. Для початку визначимо вимоги до застосунку.

3.1 Вимоги до застосунку

Цільовою аудиторією застосунку є пілоти дронів та в другу чергу OSINT спеціалісти. Ці дві групи мають дуже різні вимоги до інструментів, які вони використовують. Насправді, OSINT журналістам було б зручніше користуватись веб-застосунком або програмою для пк, проте це неможливо у випадку пілотів дронів. Для використання застосунку в польових умовах важливим є можливість детекції об’єктів в реальному часі. Тож визначимо наступні вимоги:

1. Опрацювання відео в реальному часі
    a. Модель повинна виконуватись досить швидко, щоб опрацьовувати як мінімум 20 кадрів в секунду, тобто 0.05 секунди на опрацювання зображення.
    b. Таке обмеження швидше за все не дозволить хостинг моделі як сервіса.
2. Опрацювання великої кількості відео в режимі офлайн
    a. Застосунок повинен підтримувати режим опрацювання декількох відео за раз. В такому режимі важливо використовувати методи паралельного програмування
    b. В режимі опрацювання відео офлайн необхідно підтримувати виконання застосунку у фоновому режимі, оскільки опрацювання великої кількості відео вимагає часу.
3. Отримання відеопотоку з комерційного дрону
    a. Для використання застосунку в реальному часі, необхідно отримувати відеопотік в реальному часі. В ідеальному

випадку, можна отримати відеопотік з дрона напряму, проте якщо це неможливо, потрібно знайти можливість швидко передавати відеопотік з дрону в застосунок.

4. Завантаження відео чи фото з файлової системи
    a. Для опрацювання відео в режимі офлайн, необхідно мати можливість передачі цих відео.
5. Захищеність
    a. Застосунок може опрацьовувати чутливу інформацію (позиції військових), тому захищеність опрацювання даних є дуже важливою.
    b. Можна вважати, що дані, які знаходяться в оперативній пам’яті мобільного пристрою знаходиться в безпеці.
    c. Дані, які знаходяться на диску мобільного пристрою можуть бути в небезпеці. Якщо пристрій втрачено, ці дані можна можна відновити.
    d. Найнебезпечніший етап опрацювання даних - передача їх між мобільним пристроєм і зовнішнім світом. Варто мінімізувати таку передачу. Найбезпечнішим варіантом опрацювання даних є локальна обробка відео. Ми будемо використовувати модель на пристрої. Таким чином єдина комунікації застосунку за межами пристрою - це отримання відеопотоку з дрону.
6. Швидкість виконання
    a. Окрім опрацювання відео в реальному часі, варто мінімізувати складні обчислення в застосунку, щоб його можна було швидко використати.
7. Можливість запуску на малопотужних мобільних пристроях
    a. Варто намагатись зробити застосунок доступним на якомога більшій кількості пристроїв, оскільки мобільні пристрої у операторів дронів можуть бути малопотужними.

8. Оптимізація витрати енергії
    a. Складні обчислення використовують багато енергії, що значно зменшує час автономної роботи мобільного пристрою.
9. Простота інтерфейсу
    a. Чим легше почати працювати з застосунком, тим більше людей почнуть його використовувати.
10. Використання сучасних шаблонів дизайну, якщо це не зменшує ефективність застосунку

3.2. Створення застосунку у Android Studio

Для створення застосунку будемо використовувати Android Studio 2022 та Kotlin 1.8. Для підтримки якомога більшої кількості пристроїв обираємо мінімальний SDK 21, що підтримуватиме 99% пристроїв.

Рис. 25. Емулятор Pixel 4

Вибір емулятора Андроїд є важливою складовою процесу розробки додатків для цієї платформи. Одним з ключових факторів, що впливає на вибір емулятора, є доступність відповідного SDK (набору інструментів розробника), який необхідний для розробки і тестування додатків.

Крім того, доступ до Play Market є важливим для розробників, оскільки це офіційний магазин додатків для платформи Android. Якщо додаток повинен бути доступний для загального користування, то доступ до Play Market є необхідним.

Проте, в деяких випадках, розробники можуть обрати емулятор без доступу до Play Market з метою обмежити застосунок до локального використання. Наприклад, якщо додаток призначений для внутрішнього використання в компанії або в особистих цілях, то доступ до Play Market не є необхідним і може бути обмеженим.

Крім того, що доступ до Play Market вимагає наявності відповідного емулятора та SDK, слід відзначити, що наявність Play Market також вимагає наявності Google акаунту. Це може обмежити аудиторію застосунку, оскільки деякі користувачі можуть не бажати створювати Google акаунт або не мати доступу до нього.

Крім того, залежність від Play Market може також обмежувати деякі функції нашого застосунку, якщо вони не відповідають політиці Google Play. Наприклад, Google Play може забороняти розміщення застосунків, які використовують у військових цілях.

Отже, при виборі емулятора Android та SDK для розробки додатку, необхідно враховувати потреби вашого проекту та аудиторії, яка буде користуватися вашим додатком. Якщо доступ до Play Market не є необхідним, то можливо розглянути емулятор без доступу до цього магазину, але слід врахувати, що це може обмежити аудиторію додатку.

У вашому випадку, обрання Pixel 4 без доступу до Play Market може бути обґрунтованим, тим що застосунок призначений для локального використання.

Проте, якщо ми захочемо розмістити додаток в загальний доступ, то необхідно буде забезпечити доступ до Play Market для користувачів, які хочуть завантажити і використовувати застосунок.

Виконуючи поставлені вимоги, потрібно створити View для опрацювання відео в реальному часі та в режимі офлайн. Для демонстрації також додамо окремий View “Demo”. Наведемо діаграму навігації всередині застосунку:

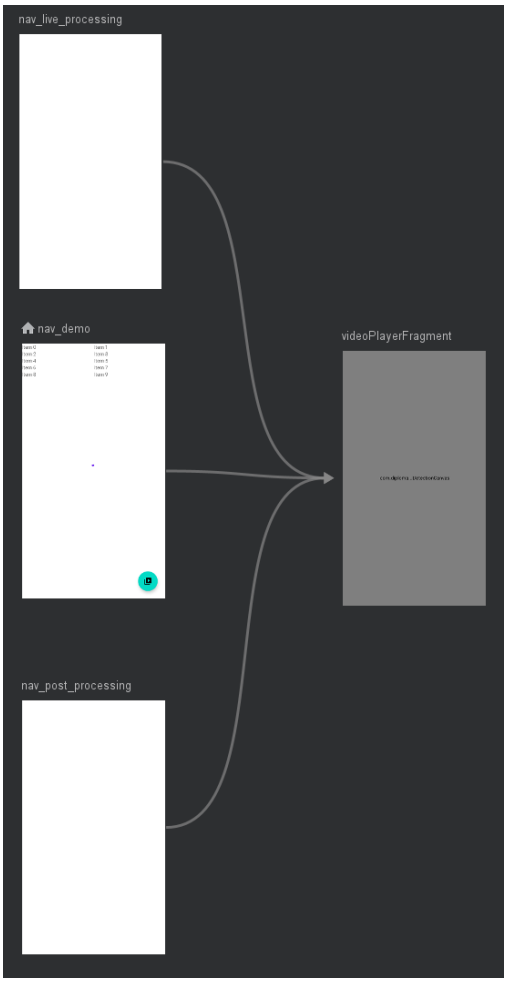

Рис. 26. Діаграма навігації.

Як бачимо окрім згаданих фрагментів Live, Demo, Postprocessing, маємо ще один фрагмент VideoPlayer, який відповідає за програвання оброблених відео. Навігація між основними фрагментами відбувається за допомогою висувного меню:

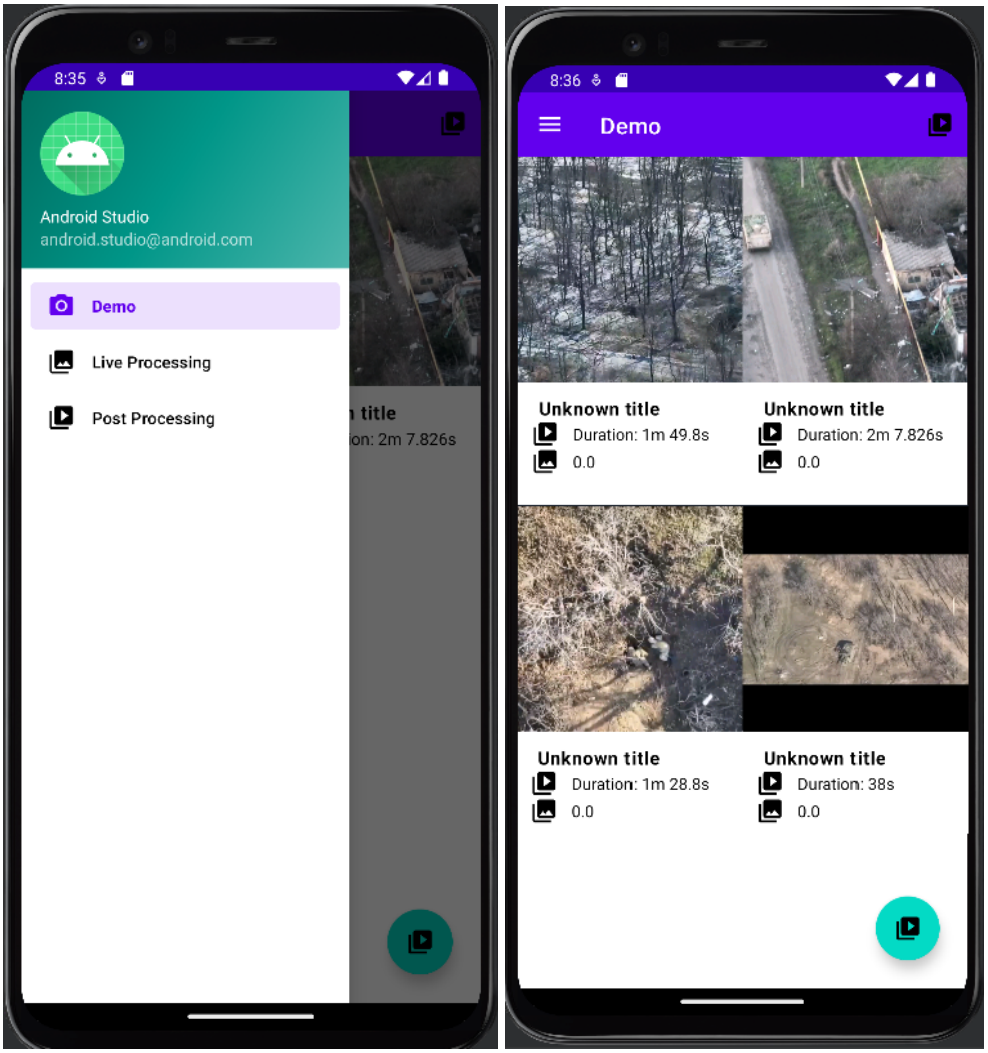

Рис. 27. Вигляд висувного меню для навігації та фрагменту демо

Першим екраном після запуску застосунку є екран демо. На цьому фрагменті розташований RecyclerView з GridLayoutManager, який дозволяє розташувати динамічні дані у рухомій таблиці з двома стовпчиками.

RecyclerView - це важлива компонента в Android, яка дозволяє відтворювати список елементів у вигляді прокручуваного списку або сітки. Серед переваг RecyclerView є наступні:

- Ефективність пам'яті: RecyclerView забезпечує ефективне використання пам'яті, оскільки елементи списку відтворюються в міру їх відображення, а не всі одночасно. Це зменшує навантаження

на пам'ять пристрою та дозволяє ефективніше використовувати ресурси.

- Гнучкість: RecyclerView надає багато можливостей для налаштування відображення елементів списку, таких як зміна розмірів, кольорів, анімації та інших параметрів. Також можна налаштувати різні типи макетів, включаючи лінійний, сітковий та грід.
- Підтримка анімації: RecyclerView надає підтримку анімації при додаванні, видаленні та переміщенні елементів списку. Це дозволяє створювати більш динамічні та привабливі інтерфейси користувача.
- Підтримка віджетів: RecyclerView надає можливість вбудовувати віджети в елементи списку, такі як кнопки, тексти та зображення.
- Кешування даних: RecyclerView підтримує кешування даних для елементів списку. Це означає, що при повторному відображенні елементів списку RecyclerView може використовувати раніше завантажені дані, що зменшує час завантаження та покращує продуктивність.

Ці властивості є важливими в нашому випадку, оскільки, за задумкою, дотик до елементів списку повинен запускати фрагмент VideoPlayer.

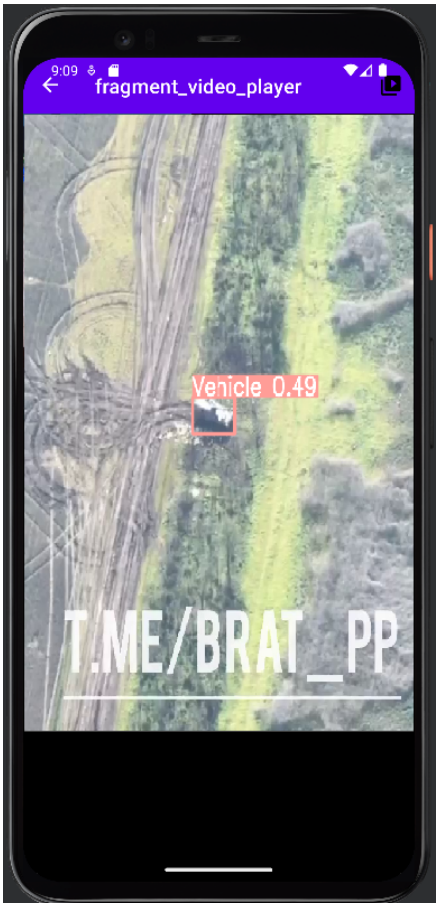

Рис. 28. Вигляд фрагменту відео плеєра зі знайденою технікою.

3.3. Використання навченої моделі у застосунку

Неочікуваним ускладненням у розробці застосунку стало використання навченої моделі YOLOv8 для пошуку військової техніки. Оскільки YOLOv8 є найновішою моделлю компанії, підтримка для неї поки залишається значно гіршою ніж для старих моделей. Таким чином YOLOv8 не має бібліотеки для Java чи Kotlin, які необхідні для розробки Андроїд застосунку. Оригінальна бібліотека написана на Python і пропонує всі необхідні функції для легкого використання навченої моделі - завантаження ваг з файлу .pt, передача параметрів таких як мінімальна впевненість у детекції, а також, що найважливіше, перетворення виводу моделі у зручний формат. Без цих можливостей використання моделі на Андроїд неможливе.

Для Андроїд створено декілька бібліотек для використання нейронних мереж, наприклад TensorFlowLite, Android Pytorch, Ml Kit API, TensorFlow Vision API та ін, проте жодна з них не надає бажаних функцій. До прикладу, TensorFlow Vision API дозволяє використовувати моделі з визначеними вхідними та вихідними даними, які не відповідають моделі YOLOv8.

Після пошуку готового рішення для цієї проблеми було прийнято рішення реалізувати всі функції оригінальної Python бібліотеки самостійно на Kotlin з використання TensorFlow Lite.

Першим етапом є завантаження моделі в застосунок. Цей крок є досить простим, оскільки підтримується бібліотекою TensorFlowLite, а саме класом InterpreterApi. Бібліотека дозволяє завантажувати файли типу .tflite з директорії assets. Єдина відмінність від оригінальної бібліотеки - це тип файлу з вагами. В оригінальній Python бібліотеці використовується розширення .pt (PyTorch). Отже потрібно було конвертувати цей файл у файл .tflite. На щастя ця функція підтримується оригінальною бібліотекою.

Після конвертування файлу, його можна завантажувати у застосунок:

```
    private fun loadModelFromAssets(context: Context, fileName: String): InterpreterApi {
        val tfliteModel = FileUtil.loadMappedFile(context, fileName)
        val options = InterpreterApi.Options()
        return InterpreterApi.create(tfliteModel, options)
    }
```

Наступним етапом є передача зображення на вхід моделі. Модель тренувалась на вхідних зображенням 640x640 пікселів (стандартний розмір), а отже і вхідні дані мають бути нормалізовані до такого розміру. Додатково, модель очікує триканальне зображення (RGB) з нормалізованими пікселями (значення від 0 до 1). Така нормалізація зображення підтримується бібліотекою TensorFlowLite:

```
val modelInputShape = Size(640, 640)
```

```
val resizeOp = ResizeOp(modelInputShape.height, modelInputShape.width, ResizeOp.ResizeMethod.BILINEAR)
val modelInputPreprocessor: ImageProcessor = ImageProcessor.Builder()
        .add(resizeOp)
        .add(NormalizeOp(0f, 255f))
        .build()
```

Отже ми маємо нормалізоване зображення, яке можна подавати на вхід моделі:

```
interpreter.run(normalizedImage.buffer, outputTensor.buffer)
```

Виходом моделі є тензор вигляду (1, 6, 8400), де 1 - розмір batch, 8400 - максимальна кількість детекцій, 6 містить інформацію про кожну детекцію: перші чотири значення вказують на координати детекції, а наступні два значення на confidence score двох можливих класів (Person, Vehicle). Ефективно використовувати такий вихід неможливо, тому його потрібно привести до нормалізованого вигляду. Нашою метою є отримання списку результатів у вигляді Result(x1, y1, x2, y2, score, class). Перші чотири значення вказуватимуть на координати прямокутника, де x1, y1 - координати верхньої лівої точки, а x2, y2 - координати нижньої правої точки, score - впевненість моделі у результаті та class - клас детекції.

На жаль TensorFlow Lite не надає жодних функцій для роботи з тензорами, через це розглядались два варіанти - пошук бібліотеки на Kotlin чи Java для роботи з тензорами, або використання звичайних масивів. Використовувати масиви для ефективної роботи з тензорами дуже важко. Єдиною бібліотекою, що підтримується є multik від JetBrain. Ідея multik - надати можливості схожі до numpy і хоча в загальному, це й справді так, numpy набагато легший у використанні та потужніший.

Отже для приведення виходу моделі в нормальний вигляд було проведено такі кроки:

1. Для кожного результату з 8400 обираємо найкращий клас. Для цього у вимірі (1, 4:6) обираємо максимальне значення. Отримуємо тензор (8400, 1)
2. Далі прибираємо вимір 1 у початковому тензорі для зручності, маємо (6, 8400)
3. Транспонуємо його у (8400, 6)
4. За допомогою тензору найкращих класів відкридаємо з транспонованого тензору результати з confidence < threshold (0.25)
5. Залишається тензор (кількість результатів, 6)
6. Якщо кількість результатів == 0, закінчуємо обробку. На зображення нічого не знайдено
7. У початковому тензорі координати мають вигляд (координати центру), ширина, висота. Приводимо це у вигляд (x1, y1, x2, y2) - координати протилежних точок.
8. Також виділяємо координати в окремий тензор (кількість результатів, 4)
9. Те ж саме з класами - тензор (кількість результатів, 2)
10. Обираємо тензор найкращих класів - (кількість результатів, 1)
11. За допомогою нього обираємо тензор confidence scores
12. Об’єднуємо отримані тензори координат, scores та класів у тензор кандидатів (кількість результатів, 6)
13. Сортуємо тензор за виміром (:, 4) - scores.
14. Якщо кандидатів більше threshold - відкидаємо їх
15. Створюємо тензор зсуву координат наступним чином: для кожного кандидату обчислюємо class_index*зсув. Зсув обираємо такий, що точно більний за максимальний розмір вхідного зображення. Тензор (кількість кандидатів, 1)

16. Створюємо тензор зсунутих координат наступним чином: кандидати[:, :4] + зсув[:]. Таким чином результати, які відповідають різним класам ніколи не перетинаються.
17. Далі проводимо фільтрацію кандидатів за алгоритмом non maximum suppression. Це вбудована функція у PyTorch, але для Котліну відповідника не було знайдено. Тому цю функцію також було реалізовано:
    a. Для кожного кандидату обчислюємо Intersection over Union (це легше робити з точками у форматі x1, y1, x2, y2) з іншими кандидатами. Якщо це значення більше threshold (0.7) - відкидаємо кандидата з меншим confidence. Оскільки наші кандидати відсортовані, то залишаємо лише кандидата з меншим індексом
18. Тепер серед наших кандидатів немає таких, що вказують на один об’єкт. Якщо таких більше threshold, відкидаємо надлишок.
19. Останнім кроком є обернена нормалізація координат детекцій. Оскільки на вхід моделі подавалось зображення зміненого розміру - координати на ньому не відповідають координатам на початковому зображенні. Цей крок можна можна зробити під час остаточного перетворення результатів.

Тепер ми маємо тензор результатів, які ми будемо відображати. Залишилось створити клас для зберігання цих результатів:

```
data class DetectionResult(
    val boundingBox: RectF,
    val confidence: Float,
    val classLabel: ClassLabel
)
```

Для кожного результату у тензорі створюємо наведений об’єкт і проводимо обернену нормалізацію для координат за допомогою resizeOp.inverseTransform() з бібліотеки TensorFlow Lite.

Для візуалізації результатів будемо користуватись бібліотекою OpenCV, яка дозволяє завантажувати відео з кешу застосунку, читати його покадрово і малювати прямокутники за координатами.

ВИСНОВКИ ДО РОЗДІЛУ 3

Розробка застосунку, який використовує навчену модель для пошуку об’єктів на зображенні виявилась складнішою задачею, ніж очікувалось. Розробка під Андроїд дуже сильно відрізняється від розробки звичайних програм для віндовс чи Backend. Це був перший Андроїд застосунок автора, що дуже сильно заповільнило розробку та обмежило кінцеві можливості додатку. Серед поставлених завдань для застосунку були безпека та швидкість. Якщо безпека була забезпечена шляхом виконання всіх операцій локально, то швидкість залишила бажати кращого через недосвідченість у роботі з мобільними пристроями. Для хороших результатів необхідне використання швидких мобільних пристроїв.

Неочікуваним ускладненням у розробці стало використання навченої моделі. Виявилось, що формат вихідних даних моделі не підтримується жодною бібліотекою для пошуку об’єктів на відео. Через було реалізовано післяобробку вихідних даних моделі, у формат, який можна легко і ефективно використовувати для відображення детекцій. Для відображення було використано потужну бібліотеку OpenCV, яка дає широкі можливості для обробки зображень, проте можливо варто було знайти більш легку та швидку бібліотеку.

Загалом вважаємо, що розробка застосунку була успішною, хоча і не ідеальною.

ВИСНОВКИ

Перед початком виконання роботи були поставлена мета розробка додатку для пошуку військової техніки та військових на відео з використанням нейронних мереж. Для досягнення цієї мети поставлено наступні завдання:

- Огляд і аналіз існуючих методів та технологій розпізнавання об'єктів на відео, зокрема нейронних мереж.
- Описати ідею власного розв’язку задачі
- Зібрати тренувальні дані для моделі.
- Виконати навчання запропонованої моделі.
- Розробка алгоритму для автоматичного розпізнавання військової техніки та військових на відео з використанням нейронних мереж.
- Реалізація додатку для пошуку військової техніки та військових на відео з використанням розробленого алгоритму.
- Тестування та оцінка ефективності розробленого додатку на реальних відеоданих.

У першому розділі проведено огляд існуючих досліджень за темою та наведено необхідні теоретичні дані для подального виконання поставлених завдань.

У другому розділі описано процес збирання даних для навчання моделі, а також описано процес навчання. Наведено графіки ефективності навчання, які показують як хороші результати вже зараз, так і потенціал подальшого навчання на більшій кількості даних. Серед можливих покращень можна виділити такі:

- зібрати більше власних даних, щоб краще відповідати поставленій цілі - пошуку техніки та людей з дронів.
- Більш агресивно відфільтрувати дані, які не відповідають поставленій задачі, наприклад зображення техніки зблизька
- Продовжити навчання

У третьому розділі було описано розробку мобільного застосунку за допомогою Android Studio та Kotlin. Було розроблено бібліотеку для післяобробки виходу моделі YOLOv8 у зручний формат. Також було створено мінімальний функціонуючий застосунок. Серед очевидних покращень застосунку є оптимізація його роботи для роботи в реальному часі, покращення дизайну та розширення функцій до тих, що були наведені у розділі 3.

Загалом вважаємо, що результати дослідження та розробки є багатообіцяючі. Модель YOLOv8 дуже гарно показала себе у задачі знаходження військової техніки і має потенціал у задачі пошуку військових. Розроблений застосунок надає простий інтерфейс, що дозволяє використовувати модель на реальних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection.” arXiv, May 09, 2016. Accessed: Apr. 25, 2023. [Online]. Available: http://arxiv.org/abs/1506.02640

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN.” arXiv, Jan. 24, 2018. Accessed: Apr. 25, 2023. [Online]. Available: http://arxiv.org/abs/1703.06870

[3] “Image Recognition Software - ML Image & Video Analysis - Amazon Rekognition - AWS,” *Amazon Web Services, Inc.* https://aws.amazon.com/rekognition/ (accessed Apr. 25, 2023).

[4] “Video AI - Video Content Analysis | Cloud Video Intelligence API,” *Google Cloud*. https://cloud.google.com/video-intelligence (accessed Apr. 25, 2023).

[5] “Azure Video Indexer – Video Analyzer for Media | Microsoft Azure.” https://azure.microsoft.com/en-au/products/video-indexer (accessed Apr. 25, 2023).

[6] “The Pentagon’s New Algorithmic Warfare Cell Gets Its First Mission: Hunt ISIS,” *Defense One*, May 14, 2017. https://www.defenseone.com/technology/2017/05/pentagons-new-algorithmic-warfare-cell-gets-its-first-mission-hunt-isis/137833/ (accessed Apr. 25, 2023).

[7] “Artificial intelligence (AI) becomes the latest arms race as adversaries seek to perfect machine learning,” *Military Aerospace*, Dec. 22, 2020. https://www.militaryaerospace.com/computers/article/14189468/artificial-intelligence-ai-project-maven-arms-race (accessed Apr. 25, 2023).

[8] J. S. H. S. Writer, “Harvard scientists find vision relates to movement,” *Harvard Gazette*, Aug. 11, 2020. https://news.harvard.edu/gazette/story/2020/08/harvard-scientists-find-vision-relates-to-movement/ (accessed Apr. 25, 2023).

[9] “YOLOv8,” *Ultralytics*. https://ultralytics.com/yolov8 (accessed Apr. 25, 2023).

[10] “YOLOXX Dataset > Overview,” *Roboflow*. https://universe.roboflow.com/yolox-1usxm/yoloxx-eewzp (accessed Apr. 25, 2023).

[11] “ukr soldier Dataset > Overview,” *Roboflow*. https://universe.roboflow.com/dataset3/ukr-soldier (accessed Apr. 25, 2023).

[12] “Neural Network,” *DeepAI*, May 17, 2019. https://deepai.org/machine-learning-glossary-and-terms/neural-network (accessed Apr. 25, 2023).

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha, “Gradient-Based Learning Applied to Document Recognition,” 1998.

[14] “Channel (digital image),” *Wikipedia*. Sep. 21, 2022. Accessed: Apr. 26, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Channel_(digital_image)&oldid=1111471216

[15] “Convolution,” *Wikipedia*. Apr. 10, 2023. Accessed: Apr. 26, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Convolution&oldid=1149150985

[16] F. Yu and V. Koltun, “Multi-Scale Context Aggregation by Dilated Convolutions.” arXiv, Apr. 30, 2016. doi: 10.48550/arXiv.1511.07122.

[17] L. Ladicky, J. Shi, and M. Pollefeys, “Pulling Things out of Perspective,” Jun. 2014, pp. 89–96. doi: 10.1109/CVPR.2014.19.

[18] H. Qin and Q. Dai, “DEPT: Depth Estimation by Parameter Transfer with a Lightweight Model for Single Still Images”.

[19] A. Oliva and A. Torralba, “Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope,” *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, May 2001, doi: 10.1023/A:1011139631724.

[20] “CS231n Convolutional Neural Networks for Visual Recognition.” https://cs231n.github.io/transfer-learning/#tf (accessed Apr. 26, 2023).

[21] “Object detection with Android | TensorFlow Lite,” *TensorFlow*. https://www.tensorflow.org/lite/android/tutorials/object_detection (accessed May 02, 2023).

[22] “TensorFlow Lite for Android.” https://www.tensorflow.org/lite/android (accessed May 02, 2023).

[23] “Multik.” Kotlin, Apr. 28, 2023. Accessed: May 02, 2023. [Online]. Available: https://github.com/Kotlin/multik