МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

МАРЧУК МИХАЙЛО БОРИСОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
д. т. н., доцент
_____ Т.В. Нескородєва
«_____» _____ 2022 р.

ВИКОРИСТАННЯ АЛГОРИТМІВ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ КЛАСИФІКАЦІЇ ВІЙСЬКОВОЇ ТЕХНІКИ НА ЗОБРАЖЕННЯХ

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
Бабаков Р.М, д.т.н., доцент,
доцент кафедри
інформаційних технологій

_____
(підпис)

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2022

АНОТАЦІЯ

Марчук М. Б. Використання алгоритмів комп’ютерного зору для класифікації військової техніки на зображеннях. Спеціальність 122 «Комп’ютерні науки». Освітня програма «Комп'ютерні технології обробки даних (Data Science)». Донецький національний університет імені Василя Стуса, Вінниця, 2022.

У кваліфікаційній роботі досліджено використання алгоритмів комп’ютерного зору для класифікації військової техніки на зображеннях, а також збір набору даних у вигляді зображень військової техніки та навчання моделі комп’ютерного зору для вирішення задачі класифікації.

Перед початком розробки програмного забезпечення, за допомогою якого можна класифікувати військову техніку на зображеннях, був проведений аналіз вимог для програмного продукту, в рамках якого були визначені потенційні користувачі програмного забезпечення, з якими проблемами вони можуть зіштовхнутись та як програмне забезпечення допоможе вирішити ці проблеми. Також було вивчено питання, як саме штучний інтелект може використовувати в програмному забезпеченні для вирішення проблеми.

Також в роботі описані історія розвитку сфери комп’ютерного зору та про його застосування в журналістиці та військовій аналітиці.

В результаті виконання кваліфікаційної роботи було створено програмне забезпечення, за допомогою якого можна обробляти великі масиви зображень та розпізнавати військову техніку на них.

У кваліфікаційній роботі описані деякі найпопулярніші алгоритми комп’ютерного зору, а також зроблений їхній порівняльний аналіз та опис вибору алгоритму, який підійде найкраще для виконання завдання. Також були описані процеси збору даних та тренування моделі машинного навчання. По результатам виконання зроблено аналітичні графіки.

Ключові слова: комп’ютерний зір, машинне навчання, військова техніка, класифікація.

Загальний об’єм роботи: 70 сторінок, 30 рисунків, 24 джерел.

SUMMARY

Marchuk M.B. Using computer vision algorithms to classify military equipment in images. Specialty 122 «Computer Science», Educational Program «Computer data processing technologies (Data Science)». Vasyl Stus Donetsk National University, Vinnytsia, 2022.

The qualification work examines the usage of computer vision algorithms for classification of military equipment in images and gathering of data in the form of images of military equipment that can be used to train machine learning models for solving classification tasks.

The software product requirements were analysed before the beginning of development of software application, that helps to identify military equipment on images. During the analysis the possible users of software application, their possible problems and ways, how software application can solve that problems, were defined.

As a result of qualification work, the software application that is able to process large amounts of images and classify military equipment on it was developed.

In qualification work some most popular computer vision algorithms are described and compared with analysis for which algorithm is best suited for accomplishing the military equipment classification task. Also the process of gathering of data and training of machine learning models are described.

Keywords: computer vision, machine learning, military equipment, classification.

Total volume of work: 70 pages, 30 images, 24 sources.

СПИСОК ТЕРМІНІВ

- **Комп’ютерний зір** - це напрямок в сфері штучного інтелекту, який вивчає, як отримувати та обробляти дані з зображень, відео та інших візуальних способів подачі інформації.
- **Машинне навчання -** це напрямок в сфері штучного інтелекту, який вивчає здатність комп’ютерних систем навчатись робити певні задачі без чітких інструкцій, натомість шукаючи патерни в наборах даних.
- **Штучна нейронна мережа** - це обчислювальна система, яка імітує нейронні мережі, що є складовими мозку в живих істот. *Також можуть використовуватись терміни* ***нейронна мережа*** *та* ***нейромережа****.*
- **Згорткова нейронна мережа -** це класс штучних нейронних мереж, який зазвичай застосовується для аналізу візуальних зображень. *Також може використовуватись термін* ***CNN****.*
- **Датасет -** це набір даних, який використовується для навчання моделі машинного навчання.
- **Задача класифікації** - це формалізована задача, яка містить множину об’єктів чи ситуацій, поділених на класи, також певну скінченну множину об’єктів чи ситуацій, для яких відомі їх класи, і для вирішення якої необхідно створити алгоритм, який визначить належність об’єкту чи ситуації до їх класу чи множини класів..
- **Задача розпізнавання -** в сфері комп’ютерного зору це задача, яка містить множину об’єктів чи ситуацій, поділених на класи, а також візуальне представлення цих об’єктів чи ситуацій, для вирішення якої необхідно ідентифікувати клас об’єктів чи ситуацій і їх місцерозташування на зображенні.
- **Функція втрат -** це обчислювальна функція, яка представляє ціну за невірність передбачень в задачі класифікації.
- **Автоматизація** - це така поведінка програмного забезпечення з використанням штучного інтелекту, коли задача вирішується без втручання користувача.

- **Аугментація** - це розширення можливостей здібностей чи потенціалу користувача за допомогою програмного забезпечення з використанням штучного інтелекту.
- **Коефіцієнт Жаккара** - це міра подібності двох множин, яка визначається у кількості спільних частин, поділених на міру об’єднання множин.
- **Ознака** - (від англійського *feature*) в розпізнаванні об’єктів та машинному навчанні це характеристика або окрема властивість спостережуваного явища, яку можливо виміряти.

ВСТУП

Актуальність теми

Все частіше журналісти в намаганнях розслідувати та висвітлювати військові події в своїх роботах спираються на відкриті джерела даних, як от соцмережі. Це можливо завдяки поширеному використання гаджетів з фотокамерами, як от смартфони, поширенням доступу до мережі Інтернет та загальної культури людей ділитися тим, що їх оточує, на загал, в тому числі і моменти з війни. Саме такі дані можуть використовувати журналісти.

Але проблема в тому, що обсяг даних може бути настільки великим, що журналісти просто не в змозі переглянути його весь. Це призводить до того, що обробляється лише частина матеріалу, внаслідок чого важлива інформація може бути не виявлена. Цю проблему можна вирішити через автоматизацію перевірки зображень на наявність певних об’єктів за допомогою алгоритмів комп’ютерного зору.

Мета дослідження

Створити програмне забезпечення, що здатне обробляти великі масиви зображень та класифікувати військову техніку на них.

Завдання дослідження

Для досягнення мети дослідження необхідно виконати наступні завдання:

1) Аналіз предметної області, а саме, які бувають види військової техніки та як вони розрізняються.

2) Аналіз інших розробок на основі машинного навчання та що використовують алгоритми комп’ютерного зору, які використовуються в журналістських розслідуваннях та військовій аналітиці.

3) Дослідження та порівняльний аналіз різноманітних алгоритмів комп’ютерного зору.

4) Збір даних, необхідних для тренування моделі машинного навчання.

5) Тренування моделі машинного навчання.

Структура роботи

Магістерська робота складається зі вступу, трьох розділів, висновків

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Обґрунтування використання

Світ сучасної цивілізації прагне миру, чому неабияк мають сприяти прогрес та розвиток технологій. Але вторгнення російської федерації в Україну показало, що навіть за такого розвитку технологій місце війни в світі ще залишається. Прогрес сприяє розвитку не тільки мирних технологій, але й зброї. Авжеж, такий розвиток подій вимагає розвитку й оборонних технологій.

Зараз в армій на озброєнні знаходяться ракети, артилерійські установки та різноманітні літальні апарати, для протистояння яким існують протиповітряні установки, безпілотники та інші засоби. Але всіма цими засобами керують люди, в природі яких є можливість допускати помилки. А коли на кону стоять людські життя, ціна помилки стає дуже високою.

Завдяки розвитку інформаційних технологій люди навчились автоматизовувати рутинні та небезпечні аспекти своєї діяльності. Одним зі стовпів автоматизації є комп’ютерний зір.

Комп’ютерний зір - це різновид штучного інтелекту, який дозволяє комп’ютерам отримувати інформацію з зображень і на основі неї приймати рішення, виконувати дії та/або надавати рекомендації.

В цивільному житті комп’ютерний зір використовується всюди, починаючи з смартфонів, в яких є функція розпізнавання обличчя, закінчуючи фабриками та заводами, де комп’ютерний зір використовується для контролю виробництва. Велику роль комп’ютерний зір відіграє в створенні безпілотних автомобілів.

Але, незважаючи на великий успіх в розвитку комп’ютерного зору, його використання в військових та військово-аналітичних цілях поки доволі обмеженне. Про причини цього описано в підрозділі 1.3.

1.1.1 Використання комп’ютерного зору в військовій журналістиці

У наш час гаджети, як от смартфони чи планшети, з доступом у Інтернет є у багатьох людей. Це призводить до того, що багато моментів життя людей у цифровому вигляді (фотографії, відео, аудіозаписи), потрапляють в мережу. Але іноді серед таких моментів може бути війна.

Контент, що відтворює події війни, може бути корисним для журналістів та розслідувачів, які висвітлюють ці події. Але проблеми виникають тоді, коли контенту стає настільки багато, що у людей не вистачає часу, щоб перевірити його повністю. Наприклад, після війни у Сирії залишилось 350 000 годин відео, що зображують події того часу. Це більше, ніж тривала сама війна. І оскільки багато годин відео залишаються непереглянутими, це може залишати нерозкритими військові злочини, які потрапили на ті відео.

Щоб вирішити цю проблему, організація “Mnemonic”, що опікується сайтом “Syrian Archive”, використовує штучний інтелект, щоб виявляти на зображеннях кластерну бомбу RBK-250 [1]. Дана бомба може залишатись вибухонебезпечною навіть через десятиліття після закінчення війни, саме тому важливо ідентифікувати всі місця, де бомби можуть знаходитись.

Схоже рішення використовувались і для доведення перебування російських військових на Донбасі в 2014 році. Цим займалась “Forensic Architecture” - дослідницька організація, що розслідує порушення прав людини, використовуючи сучасні технології [2]. Вони зібрали та каталогізували докази участі російських військових в боях за Іловайськ.

Докази включали числені зображення з супутників, де були зафіксовані колони з російською військовою технікою. Також дослідники зібрали близько 2500 годин відео з сервісу “YouTube”, які були записані в тій локації і в той час. Для знаходження потрібних матеріалів вони використали машинне навчання, щоб знайти відео, де з високою ймовірністю були танки.

Але попит на використання і, відповідно, на розробку подібних рішень все ще існує. Так, наприклад, подібні рішення збирається використовувати журналістська некомерційна організація Bellingcat, відома своїми розслідуваннями з використанням відкритих джерел даних, одним з яких, наприклад, є розслідування збиття Boeing 777 біля Донецька. Про це в своїй книзі “Ми - Bellingcat” пише засновник організації Еліот Гіґґінс [3]. Він наводить приклад, що при розслідуванні збитого літака двоє розслідувачів витратили цілий рік на дослідження акаунтів в соціальних мережах, щоб виявити осіб, які служили в підрозділі, причетному до збиття літака. Цю роботу можна було б автоматизувати за допомогою програмного забезпечення, що за допомогою штучного інтелекту могло б вивчати фотографії, виявляти людей в військовій формі чи камуфляжному одязі та шукати їх профілі в соцмережах, одночасно аналізуючи їх зв’язки з іншими людьми. Але проблема в тому, що такий інструмент може використовуватись не тільки журналістами для розслідування військових злочинів, але й, наприклад, службам розвідки чи диктаторським режимам.

1.2 Стан досліджень за обраною темою

1.2.1 Історія розвитку комп’ютерного зору

Вчені та інженери вже близько 60 років намагаються розробити способи, за допомогою яких машини зможуть бачити та розуміти візуальні дані. Експерименти почалися в 1959 році, коли нейрофізіологи показали піддослідному котові низку зображень, намагаючись співвіднести реакцію в його мозку. Вони виявили, що він спочатку реагує на жорсткі краї або лінії, і з наукової точки зору це означає, що обробка зображень починається з простих форм, таких як прямі краї.

Приблизно в той же час була розроблена перша технологія комп’ютерного сканування зображень, яка дозволила комп’ютерам оцифровувати та отримувати зображення. Ще одна віха була досягнута в 1963 році, коли комп’ютери змогли перетворити двовимірні зображення в

тривимірні форми. У 1960-х роках штучний інтелект став академічною галуззю дослідження, і це також поклало початок пошукам штучного інтелекту щодо вирішення проблеми людського зору.

У 1974 році була представлена технологія оптичного розпізнавання символів (OCR), яка могла розпізнавати текст, надрукований будь-яким шрифтом або гарнітурою. Подібним чином інтелектуальне розпізнавання символів (ICR) могло розшифрувати рукописний текст за допомогою нейронних мереж. Відтоді , OCR та ICR знайшли свій шлях до обробки документів і рахунків-фактур, розпізнавання номерних знаків транспортних засобів, мобільних платежів, машинного перекладу та інших поширених програм.

У 1982 році нейробіолог Девід Марр встановив, що зір працює ієрархічно, і запровадив алгоритми для машин, щоб виявляти краї, кути, криві та подібні основні форми. Паралельно комп'ютерний вчений Куніхіко Фукусіма розробив мережу клітин, які можуть розпізнавати шаблони. Мережа під назвою Neocognitron включала згорткові шари в нейронну мережу.

До 2000 року основна увага приділялася розпізнаванню об'єктів, а до 2001 року з'явилися перші програми для розпізнавання обличчя в реальному часі. Стандартизація того, як набори візуальних даних позначаються тегами та анотуються, з'явилася в 2000-х роках. У 2010 році набір даних ImageNet став доступним. Він містив мільйони позначених тегами зображень у тисячі класів об'єктів і забезпечує основу для CNN і моделей глибокого навчання, які використовуються сьогодні. У 2012 році команда з Університету Торонто взяла участь у конкурсі CNN на розпізнавання зображень. Модель під назвою AlexNet значно знизила частоту помилок при розпізнаванні зображень. Після цього прориву рівень помилок впав лише до кількох відсотків.

1.3 Невивчені або недостатньо вивчені питання

Незважаючи на великі переваги у використанні комп'ютерного зору, державні військові структури не поспішають використовувати його в обороні. Однією з причин є бюрократизованість таких структур. Інша, яка відноситься до теми даної роботи, є недовіра до комп'ютерних систем.

Оскільки комп'ютерний зір в оборонній сфері є відносно новим явищем, він є недостатньо вивченим, через що комп'ютерні системи можуть допускати помилки. Це нівелює всю суть використання таких систем, адже, як було зазначено в цій роботі вище, їх призначенням є якраз нівелювати людські помилки.

Однією з основних причин помилок є недостатня наповненість наборів даних, на яких системи комп'ютерного зору мають навчатись. В контексті машинного навчання наповненість набору даних - це характеристика, яка визначає, наскільки добре набір даних представляє предметну область, з якою модель машинного навчання має працювати. Наприклад, уявімо, що необхідно створити модель комп'ютерного зору, яка має розрізняти породи собак. Якщо в наборі, на основі якого вчиться модель, будуть лише зображення вівчарки, то модель зможе розрізняти лише вівчарок, але не інші породи. Теж саме відноситься і до військової техніки, але хороший датасет має містити зображення не лише різних видів озброєнь та транспорту, але й під різними кутами, з різним освітленням, в різні пори року, дня й ночі. Враховуючи, що така техніка приймає участь в бойових діях, сюди потрібно додати техніку в різних станах (починаючи від техніки, що тільки зійшла з конвеєра, закінчуючи станом після потрапляння снаряду), різними текстурами (бруд, пісок, вода, вогонь), різними станами самої камери (бруд, тріщини, різна роздільна здатність) і так далі.

Раніше збір такого датасету був проблемним, оскільки більшість зображень з поля бою, де техніка приймала участь, або були зробленні через тривалий час після закінчення бою журналістами, або засекречені. Війна в

Україні частково це змінила, оскільки завдяки повсюдному використанню соціальних мереж та використанню комбатантами смартфонів зображення з бойових дій (до того ж, з точки зору обидвох сторін) легко поширюються.

Саме цей фактор може стати поштовхом до розвитку в цій сфері, де потрібна технологія вже існує певний час, але не було потрібних даних. На даний момент в Інтернеті вже існує набір даних з зображеннями техніки, що приймала участь в російсько-українській війні, але він складається лише з зображень без поміток та пояснень, які могли б значно допомогти при тренуванні моделі.

Для того, щоб тренувати якісну модель, потрібні не тільки зображення, але й помітки, які позначають, де і яка техніка знаходиться на зображенні. Зображення можна зібрати з різноманітних відкритих джерел в Інтернеті, як от соціальні мережі, сайти новин та звіти від державних та гуманітарних організацій. Для попередньої ідентифікації техніки на зображеннях можна використовувати бази даних з зафіксованими записами про російські війська, які збирали журналісти та аналітики. Наприклад, таку базу має інформаційне агенство InformNapalm, яке підтримує цю базу від початку російської агресії з 2014 року. Також можна скористатись послугами спеціаліста, що здатен визначати військову техніку, як це робили в своєму розслідуванні журналісти The New York Times [4].

Ще одна проблема, яку необхідно подолати перед повною автоматизацією розпізнавання військового обладнання, це навчитись визначати автентичність зображень. Від початку російського вторгнення в 2022 році мережу заполонили фотографії, які ніби як висвітлювали війну, хоча такими не були. Так, за зображення сьогоднішньої війни видавались фото з Криму за 2014 рік, відео вибуху в Сирії та навіть запис відеогри.

РОЗДІЛ 2. ЗБІР ВИМОГ

2.1 Визначення проблеми

Для того, щоб будь-яким програмним забезпеченням на основі штучного інтелекту” могли користуватись, у нього має бути так званий людиноцентричний інтерфейс, тобто інтерфейс, який дозволяє людині розуміти результати роботи штучного інтелекту[5].

Рисунок 1. Людиноцентричний інтерфейс. Туду вставити номер

І для того, щоб створити людиноцентричний інтерфейс для програмного забезпечення, розробникам необхідно знати відповіді на наступні питання:

- Хто буде користуватись даним програмним забезпеченням?
- Які цінності у потенційних користувачів?
- Які проблеми можуть виникнути у потенційних користувачів?
- Як програмне забезпечення може вирішити проблеми користувачів?
- Як програмне забезпечення має працювати у випадку вирішення проблеми?

Для отримання відповідей на ці питання необхідно спочатку скласти гіпотезу про те, якою може бути проблема та як програмне забезпечення може її вирішити. Після цього потрібно перевірити гіпотезу, використовуючи один чи декілька з перелічених методів:

- Проведення опитувань серед потенційних користувачів.
- Проведення інтерв’ю з потенційними користувачами.
- Аналіз даних, статей, книжок та інших джерел інформації, що стосуються гіпотетичної проблеми.

При розробці програмного забезпечення для кваліфікаційної роботи джерелом такої інформації виступали статті в мережі Інтернет, а також книги. Потенційними користувачами вважаються журналісти та військові аналітики, які в своїй повсякденній роботі використовують медіадані, що зображають бойові дії, як от фотографії та відео. Потенційною проблемою може бути те, що медіаданих може бути настільки багато, що журналісти та аналітики не зможуть встигнути переглянути та обробити всі файли, зокрема фотографії та відео, які можуть містити зображення військової техніки.

Підтвердження наявності такої проблеми можна знайти в книзі Еліота Гіґґінза “Ми - Bellingcat”. У ній автор наводить приклад, що двом журналістам-розслідувачам служби Bellingcat знадобився рік, щоб досліджувати фотографії в соціальних мережах, щоб визначити, хто служив у підрозділі, який причетний до збиття літака рейсу MH17. Гіґґінз стверджує, що використання штучного інтелекту, який міг би розпізнавати людей в військовій формі та шукати їх в соціальних мережах, змогло б допомогти їм та іншим журналістам пришвидшити їх роботу.

Рисунок 2. Обкладинка українського видання “Ми - Bellingcat” туду вставити номер

На фінальному етапі збору вимог розробники мають вирішити, як саме програмне забезпечення має вирішувати проблему користувачів. Також необхідно визначити, чи потрібно використовувати штучний інтелект і якщо так, то яким чином. Штучний інтелект варто використовувати в таких випадках:

- Коли потрібно рекомендувати різні варіанти виконання роботи для різних користувачів;
- Коли потрібно розпізнавати певні сутності;
- Коли потрібно передбачити події в майбутньому;
- Коли використання динамічно-змінюваного контенту краще за використання статичного інтерфейсу;

Використання штучного інтелекту недоцільне в наступних випадках:

- Коли від програмного забезпечення вимагається статична передбачувана поведінка;

- Коли необхідно надавати статичну чи обмежену інформацію;
- Коли помилки в роботі програмного забезпечення недопустимі або їх ціна занадто висока;
- Коли необхідна так звана “прозорість” у роботі програмного забезпечення або повне розуміння користувачів про те, як воно працює;

2.2 Задачі автоматизації та аугментації

Коли розробники визначились з проблемою, яку має вирішувати програмне забезпечення, та маючи переконання, що для вирішення проблеми необхідно застосовувати штучний інтелект, необхідно вирішити, яким саме шляхом штучний інтелект буде вирішувати проблему. Існують два способи, якими штучний інтелект зазвичай виконує задачі, це автоматизація та аугментація [8].

Деякі задачі користувачі воліють віддати повністю для виконання штучному інтелекту, а деякі хочуть виконувати самі, але не проти того, щоб їх продуктивність була підвищена за рахунок штучного інтелекту.

2.2.1 Задача автоматизації

Автоматизація - це така поведінка програмного забезпечення з використанням штучного інтелекту, коли задача вирішується без втручання користувача.

Автоматизація зазвичай застосовується, коли користувачі не бажають виконувати певні задачі або коли час, гроші чи інші ресурси не вартують того, щоб витрачати їх на виконання задачі. Вдало імплементована автоматизація зазвичай відповідає наступним пунктам:

- Підвищення ефективності;
- Підвищення рівня безпеки користувачів;
- Зменшення кількості нудних для користувачів задач;

- Досягнення нових рівнів показників, які не були б можливими без використання автоматизації;

Автоматизація зазвичай є найкращим варіантом для виконання задач, де людські слабкості поєднуються з перевагами штучного інтелекту. Найкраще автоматизацію застосовувати в наступних випадках:

- Коли користувачі недостатньо спроможні або не мають достатньо знань для виконання задачі;
- Коли задача для користувачів нудна, неприйнятна або небезпечна;

2.2.2 Задача аугментації

Аугментація - це розширення можливостей здібностей чи потенціалу користувача за допомогою програмного забезпечення з використанням штучного інтелекту.

Зазвичай розробники можуть вважати, що при розробці програмного забезпечення з використанням штучного інтелекту найкращим варіантом вирішення проблеми – це автоматизація. Але існує ситуацій, коли користувачі воліють продовжити виконувати свої завдання, замість того, щоб їх повністю автоматизовувати. В таких випадках варто застосовувати аугментацію, що дозволяє покращити якість виконання завдання користувачем.

Вдало імплементована аугментація має відповідати одному чи кільком з наведених пунктів:

- Підвищений рівень задоволення користувача від виконання завдання;
- Рівень контролю користувача над виконанням завдання вищий, ніж якби виконання було автоматизоване;
- Збільшення можливостей користувача для виконання завдання;
- Підвищений рівень креативності користувача;

Аугментацію не завжди легко відрізнити від автоматизації, але зазвичай її імплементація більш складна та інтерфейс реалізації більш

спрямований на взаємодію з користувачем. Найкраще аугментацію застосовувати в наступних випадках:

- Коли користувачі отримують задоволення від виконання завдання або ж завдання не здається їм занадто нудним або час, гроші чи інші ресурси, які витрачаються на виконання завдання відповідають цінності результату виконання завдання;
- Коли відповідальність користувачів за результат виконання завдання необхідна або висока;
- Коли ціна помилки при неправильному виконанні завдання занадто висока;
- Коли розробники не можуть врахувати усі уподобання чи побажання користувачів при проектуванні інтерфейсу програмного забезпечення;

2.2.3. Вибір між автоматизацією та аугментацією для розпізнавання військової техніки

У ході виконання магістерської роботи має бути розроблено програмне забезпечення, яке дозволить обробляти великі масиви медіафайлів та розпізнавати військову техніку на них. Це має допомогти журналістам та військовим аналітикам, які використовують медіафайли для висвітлювання військових подій.

Враховуючи характер задачі, що має бути вирішена, спосіб вирішення має відповідати наступним пунктам:

1. ***Зменшення виконання рутинної та нудної роботи.*** Припускається, що медіафайлів, які обробляють журналісти та аналітики, може бути дуже багато, тому їх перегляд перетворюється на монотону роботу, яку було б бажано зменшити та пришвидшити. Це також має підвищити рівень ефективності користувачів при виконанні роботи.

2. ***Досягнення результату роботи, який був би неможливим без використання програмного забезпечення.***
   Якщо знову взяти гіпотезу про великий об’єм медіаданих, які потрібно обробити, то можна також припустити, що не всі медіадані будуть переглянуті журналістами та аналітиками, внаслідок чого може бути пропущена важлива інформація. Наприклад, організація “Mnemonic” вивчає фотографії та відеозаписи з війни у Сирії, щоб знаходити касетні бомби, які можуть залишатись небезпечними навіть після закінчення війни, тому важливо переглянути всі медіафайли. Штучний інтелект міг ідентифіковувати бомби на фотографіях набагато швидше за людей.
3. ***Програмне забезпечення має бути розраховане на користувачів з потрібними компетенціями.***
   Припускається, що користувачами будуть лише журналісти та військові аналітики, тому можна ускладнити інтерфейс, щоб надати користувачам з відповідними компетенціями більше можливостей та контролю за виконання завдання.
4. ***На користувачах лежить відповідальність за виконання завдання.***
   За допомогою фотографій та відеозаписів можна лише задокументувати, яка військова техніка перебувала в якій локації в який момент часу. Але сама по собі ця інформація нічого не значить. Кінцевим завданням журналістів є визначити хронологію подій та описати її, і в кожному окремому випадку спосіб досягнення результату буде відрізнятись, через що неможливо створити інтерфейс для такої задачі, на відміну від розпізнавання об’єктів на фотографіях, що є рутинною задачею.
5. ***Висока ціна помилки.***
   Неправильно інтерпретовані дані і, відповідно, хибне висвітлення подій може завдати репутаційного удару журналістам та військовим аналітикам.

І хоча пункти 1 і 2 відповідають тому, що завдання має вирішуватись шляхом автоматизації, пункти 3, 4 і 5 по своїй значущості переважають їх і вказують на те, що завдання має вирішуватись шляхом аугментації.

2.3. Функція втрат

Кожне програмне забезпечення, що використовує для виконання завдання штучний інтелект, має використовувати для направлення своєї роботи так звану функцію втрат. У машинному навчанні та математичній оптимізації функція втрат - це обчислювальна функція, яка представляє ціну за невірність передбачень в задачі класифікації. Саме ця функція визначає дії та/або поведінку штучного інтелекту, а також їх модифікацію та корекцію.

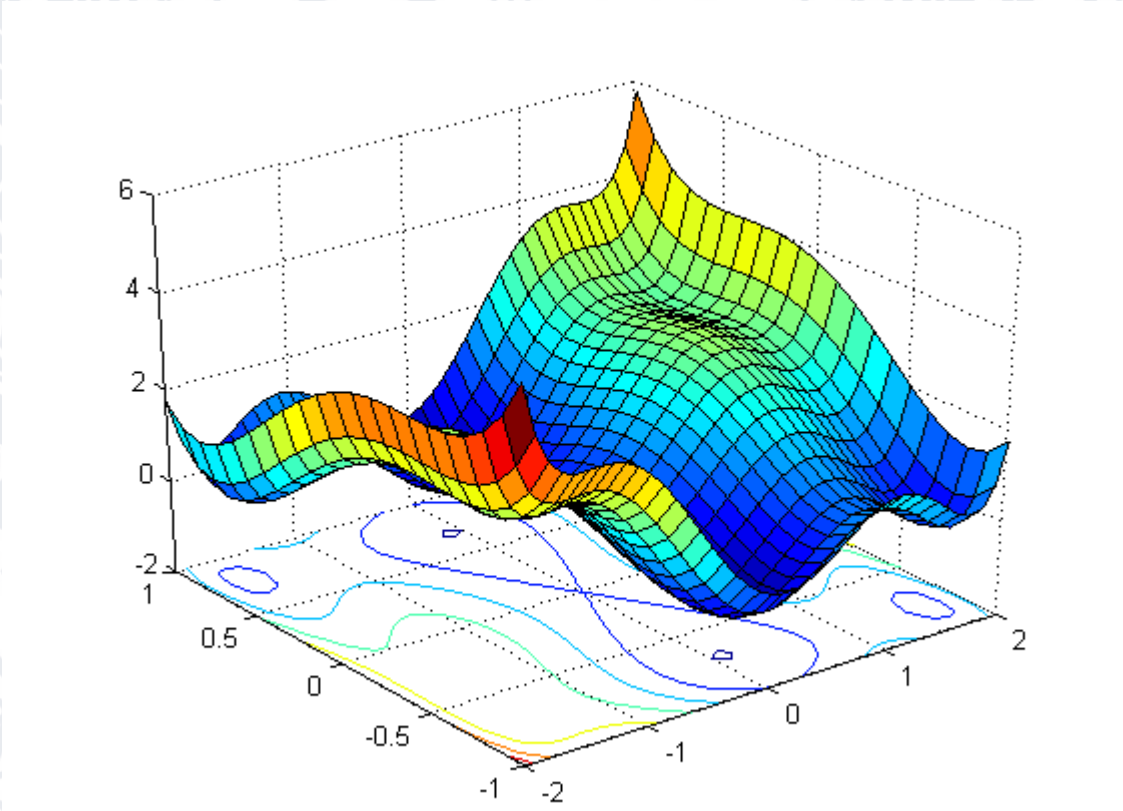


Рисунок 3. Графічне представлення функції втрат туду вставити номер

Виведення функції втрат є важливим етапом, оскільки це безпосередньо впливає на досвід користувачів користуванням програмного забезпечення.

2.3.1 Позитивні та негативні передбачення

Об’єктом дослідження в кваліфікаційній роботі є розпізнавання військової техніки, але заради пояснення в цьому підрозділі задача буде спрощена до бінарної кваліфікації, тобто в приклада модель машинного навчання має передбачити, чи є на зображенні військова техніка або ж нема. Таким чином в прикладі всього два варіанти відповіді, стверджувальна та заперечувальна, і чотири можливі ситуації на одне передбачення:

1. Модель дала стверджувальну відповідь і це коректне передбачення;
2. Модель дала заперечувальну відповідь і це коректне передбачення;
3. Модель дала стверджувальну відповідь і це не коректне передбачення;
4. Модель дала заперечувальну відповідь і це не коректне передбачення;

При виведенні функції втрат потрібно передбачити так звані “ваги” кожної ситуації, щоб у користувачів був якомога кращий досвід від використання програмного забезпечення. Наприклад, якщо штучний інтелект використовується в автономному автомобілі, то варіант 4 (система класифікувала, що на дорозі є пішохід в той час, коли там нікого немає) є більш бажаним, ніж варіант 3 (система класифікувала, що на дорозі немає пішохода в той час, коли той є). Тому потрібно надавати перевагу певним варіантам над іншими.

2.3.2. Компроміси з точністю та відкликанням

Точність та відкликання є метриками, які описують глибину та ширину результатів, які надає програмне забезпечення з використанням штучного інтелекту, а також видів помилок щодо результату [9].

Точність - це частка стверджувальних коректних передбачень серед усіх стверджувальних та заперечувальних коректних передбачень. Чим більше точність, тим більше вірогідність, результат роботи програмного забезпечення буде коректним, але в такому випадку деяка частка коректних передбачень може вважатись некоректною.

Відкликання - це частка стверджувальних коректних передбачень серед стверджувальних коректних та заперечувальних некоректних передбачень. Чим більше відкликання, тим більше вірогідність, що всі коректні результаті будуть визначені як такі, але тоді збільшується ймовірність, що некоректні передбачення будуть позначенні як коректні.

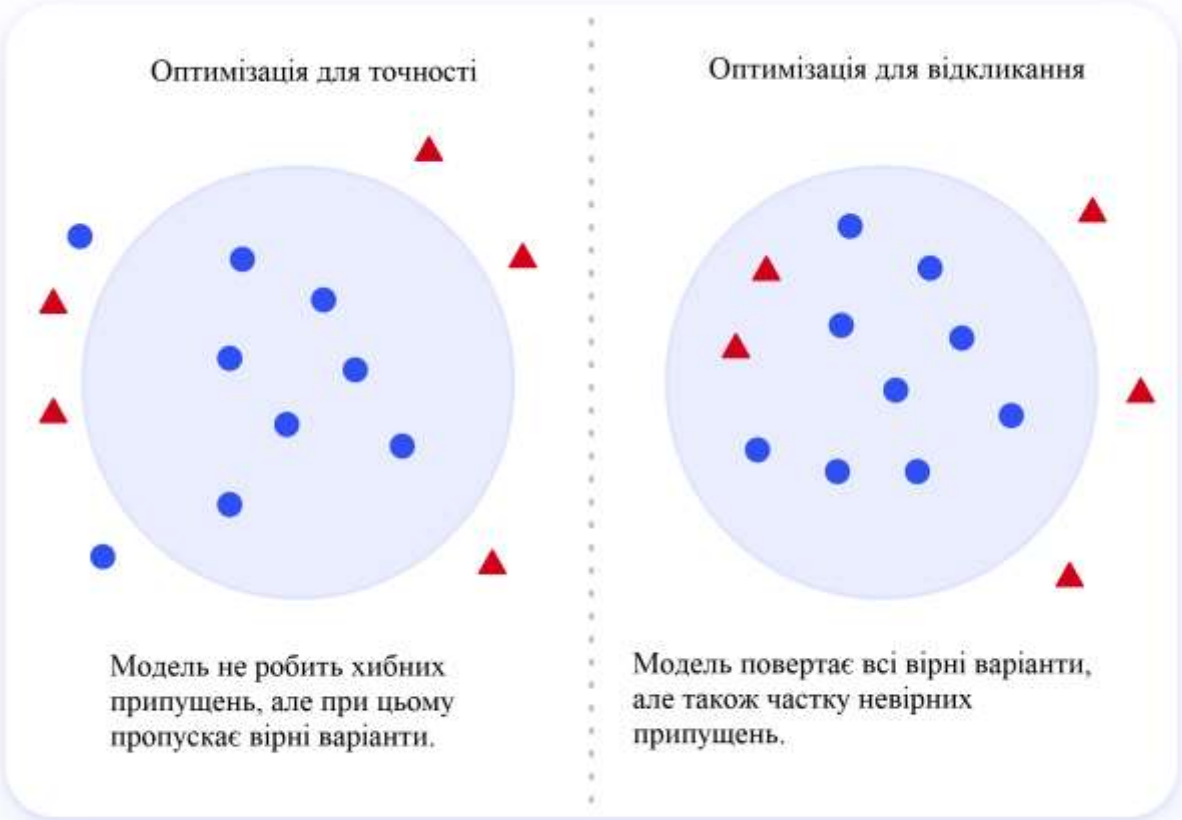


Рисунок 4. Візуалізація точності та відкликання. Туду змінити номер

При розробці будь-якого програмного забезпечення, що використовує штучний інтелект, необхідно враховувати компроміси у вигляді точності та відкликання і обирати те, що найбільше підходить для виконання задачі.

У випадку з аугментацією задачі обробки зображень, щоб визначати на них військову техніку, важливо знаходити всі вірні варіанти, навіть якщо при цьому система буде видавати частину не коректних припущень. Отож в випадку даної роботи буде використовуватись стратегія відкликання.

Висновок до розділу 2

В даному розділі були описані вимоги програмного забезпечення, що використовує штучний інтелект, яке розробляється в межах даної кваліфікаційної роботи. Були визначені основні вимоги, а саме:

- Програмне забезпечення має вирішувати проблему обробки великого об’єму медіаданих журналістами та військовими аналітиками;
- Відповідальність за результат виконання завдання лежить на кінцевих користувачах;
- Важливо отримати всі вірні результати, навіть якщо при цьому програмне забезпечення допустить декілька помилок;
- Програмне забезпечення має зменшити кількість рутинної роботи для користувачів;

Для штучного інтелекту, який буде використовуватись в програмному забезпеченні, необхідно були визначити стратегію взаємодії з користувачем (автоматизація чи аугментація), а також стратегія оптимізації роботи штучного інтелекту для якості передбачень (точність чи відкликання).

У випадку з стратегіями автоматизації та аугментації розглядалися наступні пункти:

- Наскільки стратегія зменшить виконання рутинної та нудної роботи для користувачів?
- Якого результату можна досягти, якщо використовувати штучний інтелект?
- На чиїй стороні лежить відповідальність за результат виконання роботи?
- Наскільки висока ціна помилки?

З огляду на це було обрано стратегію аугментації.

У випадку з стратегією оптимізації якості передбачень було обрано стратегію відкликання, оскільки для користувачів важливо отримати всі вірні результати, навіть якщо при цьому будуть показані декілька невірних передбачень.

РОЗДІЛ 3 ЗБІР ДАНИХ

3.1 Збір та анотація даних

3.1.1 Критерії якості датасету

Для тренування моделі необхідний датасет, який складається з зображень, що містять об’єкти для розпізнавання, та позначенннями, за якими координатами на зображеннях знаходяться об’єкти. Якість та наповненість датасету напряму впливає на точність роботи нейронних мереж.

Те, чи підходить датасет для навчання моделі, визначається наступними характеристиками:

- Різноманітність: зображення повинні містити різні типи об’єктів, що відносяться до тематики проблеми, яку необхідно вирішити. Наприклад, для мережі, що розпізнає танки, необхідно зібрати зображення танків різних моделей, різних модифікацій та різних років випуску.
- Якість: чим більші роздільна здатність, глибина кольору та розмір зображення, тим більша точність досягається у тренованої моделі.
- Кількість: кількість різноманітних зображень так само впливає на точність передбачень моделі.
- Щільність: зображення мають містити різну кількість об’єктів з різними умовами. Наприклад, на одному зображення може бути один танк на близькій відстані, а на іншому декілька танків на далекій відстані.

Рисунок 5. Приклад зображення, що підходить для датасету

3.1.2 Збір даних

Процесс створення датасету починається за збору даних. Тип даних залежить від проблеми, яку потрібно вирішити методами алгоритмів машинного навчання. У випадку проблеми, яка розглядається в даній роботі, необхідно зібрати набір зображень військової техніки.

Для збору даних існують наступні джерела:

- Готові набори зображень.

Переваги:

  - Не вимагають часу на збір
  - Зазвичай безкоштовні або коштують невелику ціну

Недоліки:

- Може підходити по тематиці, але не підходити по вимогам
- Може вимагати валідації та переробки

- Зазвичай не підходить для промислового користування

Приклади:

- https://registry.opendata.aws/: Регістр датасетів від Amazon Web Services
- https://datasetsearch.research.google.com/: Пошук по датасетах від Google
- https://www.kaggle.com/datasets: Даний сайт пропонує близько 19 000 датасетів.

- Створення власного набору зображень: дані можна зібрати самотужки або ж використовуючи спеціальне програмне забезпечення, парсери, що роблять запити по різноманітних адресах в мережі Інтернет та колекціонують зображення з отриманих сторінок. Також дані можуть бути зібрані за допомогою різноманітних пристроїв, як от камери, сенсори, дрони, супутники і так далі.

Переваги:

- Можливість створити датасет, який точно відповідає вимогам
- Творець датасету може використовувати його як інтелектуальну власність

Недоліки:

- Вимагає часу та ресурсів

- Найм третіх осіб для створення датасету.

Переваги:

- Можливість створити датасет, який точно відповідає вимогам
- Наявність експертизи третьої сторони

Недоліки:

- Вимагає значних вкладень коштів.

Як вже було виявлено в розділі 1.3, набору даних, який би відповідав вимогам роботи, поки не існує. Щоб зібрати його, потрібні не тільки зображення військової техніки, але й інформація про їх характеристики та стан. Для цього можуть використовуватись наступні джерела:

- Офіційні звіти від державних та правозахисних організацій
- Сайти новин
- Профілі в соціальних мережах комбатантів, журналістів, аналітиків та спеціалістів з OSINT (в основному Telegram та Twitter)

3.1.3 Анотація даних

Для аннотації даних використовувався сервіс MakeSense.ai - це безкоштовний для використання інструмент для анотації фотографій. Щоб розмітити зображення, потрібно виконати наступні кроки:

- Відкрити сайт makesense.ai
- Завантажити фотографії та обрати пункт Object Detection, як позначено на рисунку 6

Рисунок 6. Інтерфейс програми makesense.ai

- Ввести позначки, які модель має розпізнавати (в даному випадку це назви військової техніки) і натиснути Start project

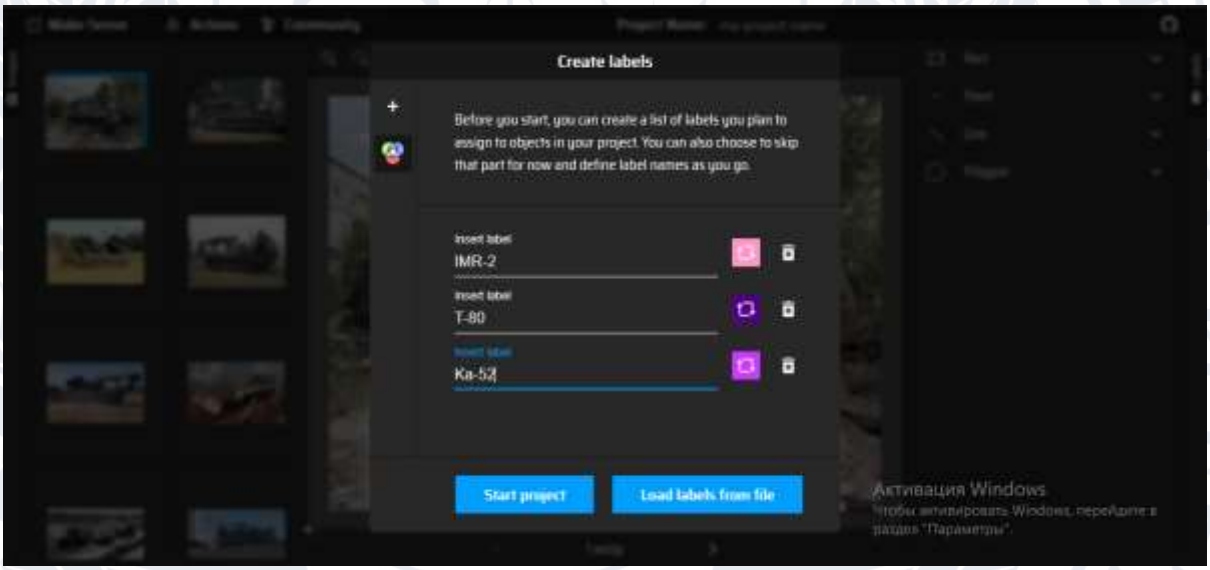


Рисунок 7. Введення позначень для датасету

- Розмітити зображення

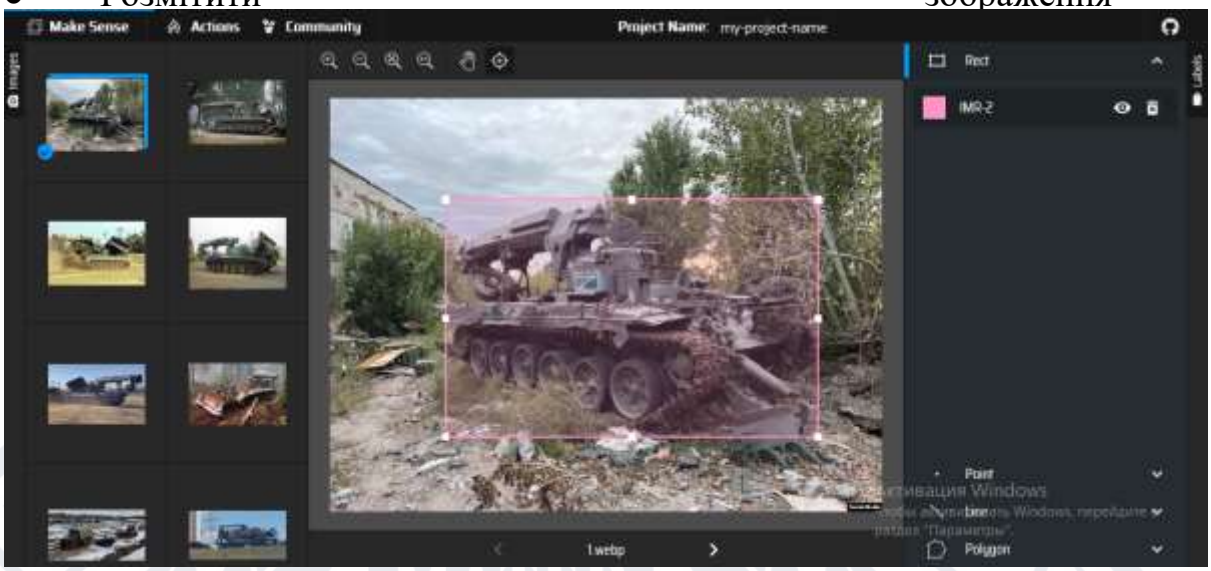


Рисунок 8. Розмітка зображень

Варто зазначити, що при розмітці на кожному зображенні виділялись не об’єкти повністю, а лише їх основні частини (наприклад, корпус). Це пов’язано з тим, що більшість екземплярів військової техніки мають елементи, що випирають з корпусу (антени, башти, тощо) і тому на зображеннях в прямокутниках, в які обводяться об’єкти для позначення їх місцерозташування на зображеннях, буде багато артефактів, які не є частиною військової техніки (фон, небо, будівлі, трава, інші текстури), але потрапляють в датасет через те, що вони знаходяться на фоні. Щоб уникнути артефактів, усі випираючі елементи обрізаються. Наприклад, у танка є башта, яка випирає з основного корпусу. При розмітці зображень з танками башти не беруться до уваги.

Рисунок 9. Приклад неправильної розмітки танка. Не дивлячись на те, що танк виділяється повністю, в прямокутник потрапляють зайві артефакти.

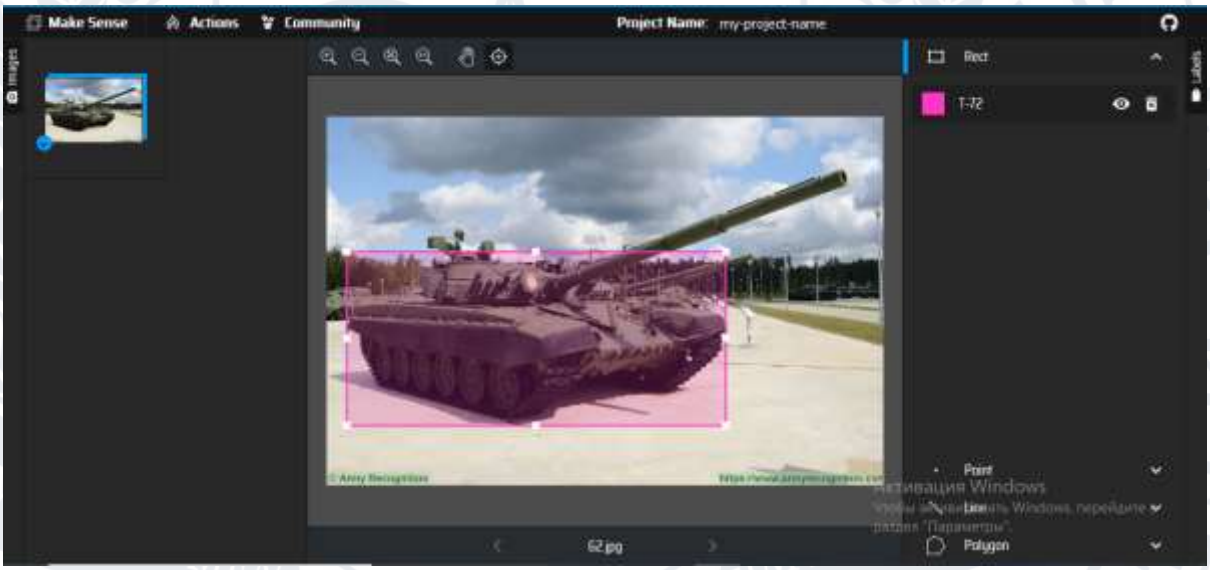


Рисунок 10. Приклад правильної розмітки танка.

- Зберегти результати, вибравши в меню Actions пункт Export Annotations, обрати пункт A .zip package containing files in YOLO format та натиснути Export

В кінці мають вийти дві директорії - одна містить зображення, інша містить файли з розміткою для зображень.

3.2 Вибір мови та бібліотек

Для розробки моделі буде використовуватись мова програмування Python.Це одна з найпопулярніших мов програмування з багатим набором бібліотек для машинного навчання.

Також для розробки будуть використовуватися наступні бібліотека PyTorch - це бібліотека з відкритим вихідним кодом для машиного навчання, розроблена компанією Meta (ex-Facebook). Дана бібліотека є однією з найпопулярніших і має детальну документацію.

РОЗДІЛ 4. НАВЧАННЯ МОДЕЛІ

4.1 Задачі з класифікації та розпізнавання

В другому розділі даної роботи вже було визначено, що для програмного забезпечення, що розробляється в рамках даної роботи, будуть використовуватись стратегії аугментації та відкликання. Також в третьому розділі був описаний збір даних, необхідних для тренування моделі машинного навчання. Але перед тим, як почати тренувати модель, необхідно обрати алгоритм комп'ютерного зору, який буде використовувати для програмного забезпечення. Для цього спочатку необхідно визначити, яку задачу з ідентифікації військової техніки алгоритм має вирішувати - класифікації чи розпізнавання.

Задача класифікації - це формалізована задача, яка містить множину об'єктів чи ситуацій, поділених на класи, також певну скінченну множину об'єктів чи ситуацій, для яких відомі їх класи, і для вирішення якої необхідно створити алгоритм, який визначить належність об'єкту чи ситуації до їх класу чи множини класів.

Задача розпізнавання - в сфері комп'ютерного зору це задача, яка містить множину об'єктів чи ситуацій, поділених на класи, а також візуальне представлення цих об'єктів чи ситуацій, для вирішення якої необхідно ідентифікувати клас об'єктів чи ситуацій і їх місцерозташування на зображенні.

Програмне забезпечення має відповідати наступним пунктам:

- Інтерфейс програмного забезпечення має дати змогу користувачам ідентифікувати військову техніку навіть в тому випадку, якщо користувачі певну модель бачать вперше. Наприклад, якщо зображення містить декілька зразків військової техніки і користувачі жодну з моделей раніше не бачили, то інтерфейс програмного забезпечення

має позначити, яка модель військової техніки знаходить на зображенні і де саме.

- Інтерфейс програмного забезпечення має дати змогу користувачам ідентифікувати військову техніку навіть в тому випадку, навіть якщо її не видно неозброєним оком. Фотографії можуть містити зображення військової техніки на задньому плані і в далекій перспективі, отож пересічній людині важко буде ідентифікувати модель техніки в такому випадку. Тому цю задачу має виконати штучний інтелект а інтерфейс має позначити, де саме на зображені знаходиться військова техніка.

Основна різниця між класифікацією та розпізнаванням полягає в тому, що в задачі класифікації одне зображення представляє один об’єкт, в той час як в задачі розпізнавання одне зображення може містити декілька об’єктів.

Враховуючи вищеописані пункти, алгоритм комп’ютерного зору, що буде використовуватись в програмному забезпеченні, має вирішувати задачу розпізнавання.

4.2 Принципи роботи алгоритмів комп’ютерного зору

4.2.1 Рамки обмеження

В сфері комп’ютерного зору, зокрема в розпізнаванні об’єктів, зазвичай використовуються так звані рамки обмеження, щоб позначати об’єкти на зображенні. Зазвичай це прямокутник, в межах якого знаходиться об’єкт для розпізнавання [10]. Є два способи описати рамки обмеження на зображенні:

1. Позначення на зображенні координат двох точок: лівого верхнього кута прямокутника та нижнього правого кута прямокутника.
2. Позначення на зображенні координат центра довжини об’єкта з вказуванням довжини та ширини для нього.

Для прикладу використаєм код на Python з використанням бібліотек PyTorch і matplotlib. Спочатку виведемо зображення, на якому буде показуватись приклад:

```
%matplotlib inline
import torch


plt.set_figsize()
img = plt.imread('../img/catdog.jpg')
plt.imshow(img);
```

Результат показано на рисунку 11.

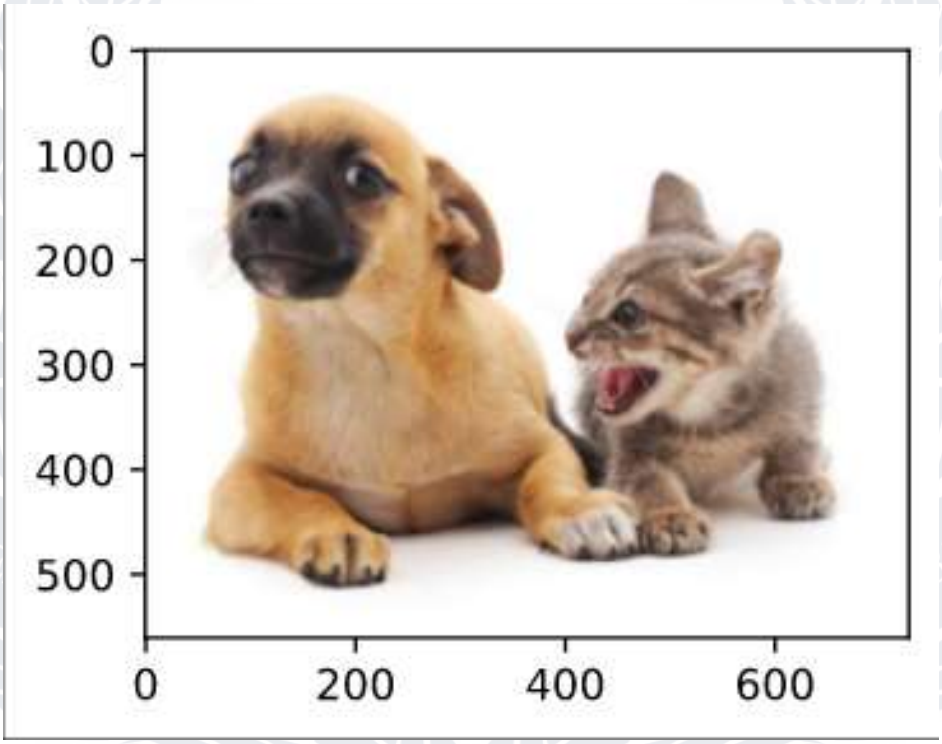


Рисунок 11. Результат виконання коду

Далі для прикладу потрібно написати дві функції, одна конвертує координати лівого верхнього кута та правого нижнього кута рамки обмеження в координати центру об’єкта з шириною та висотою, а друга робить зворотній процесс:

```
#@save
def box_corner_to_center(boxes):
```

```
    """Convert from (upper-left, lower-right) to (center, width, height)."""
    x1, y1, x2, y2 = boxes[:, 0], boxes[:, 1], boxes[:, 2], boxes[:, 3]
    cx = (x1 + x2) / 2
    cy = (y1 + y2) / 2
    w = x2 - x1
    h = y2 - y1
    boxes = torch.stack((cx, cy, w, h), axis=-1)
    return boxes

#@save
def box_center_to_corner(boxes):
    """Convert from (center, width, height) to (upper-left, lower-right)."""
    cx, cy, w, h = boxes[:, 0], boxes[:, 1], boxes[:, 2], boxes[:, 3]
    x1 = cx - 0.5 * w
    y1 = cy - 0.5 * h
    x2 = cx + 0.5 * w
    y2 = cy + 0.5 * h
    boxes = torch.stack((x1, y1, x2, y2), axis=-1)
    return boxes
```

Далі позначимо координати для рамок обмеження для кота та собаки:

```
dog_bbox, cat_bbox = [60.0, 45.0, 378.0, 516.0], [400.0, 112.0, 655.0, 493.0]
```

Після цього виведемо на зображення рамки обмеження:

```
#@save
def bbox_to_rect(bbox, color):
    """Convert bounding box to matplotlib format."""
    # Convert the bounding box (upper-left x, upper-left y, lower-right x,
    # lower-right y) format to the matplotlib format: ((upper-left x,
    # upper-left y), width, height)
    return d2l.plt.Rectangle(
```

```
    xy=(bbox[0], bbox[1]), width=bbox[2]-bbox[0], height=bbox[3]-bbox[1],
    fill=False, edgecolor=color, linewidth=2)

fig = plt.imshow(img)
fig.axes.add_patch(bbox_to_rect(dog_bbox, 'blue'))
fig.axes.add_patch(bbox_to_rect(cat_bbox, 'red'));
```

Результат показано на рисунку 12.

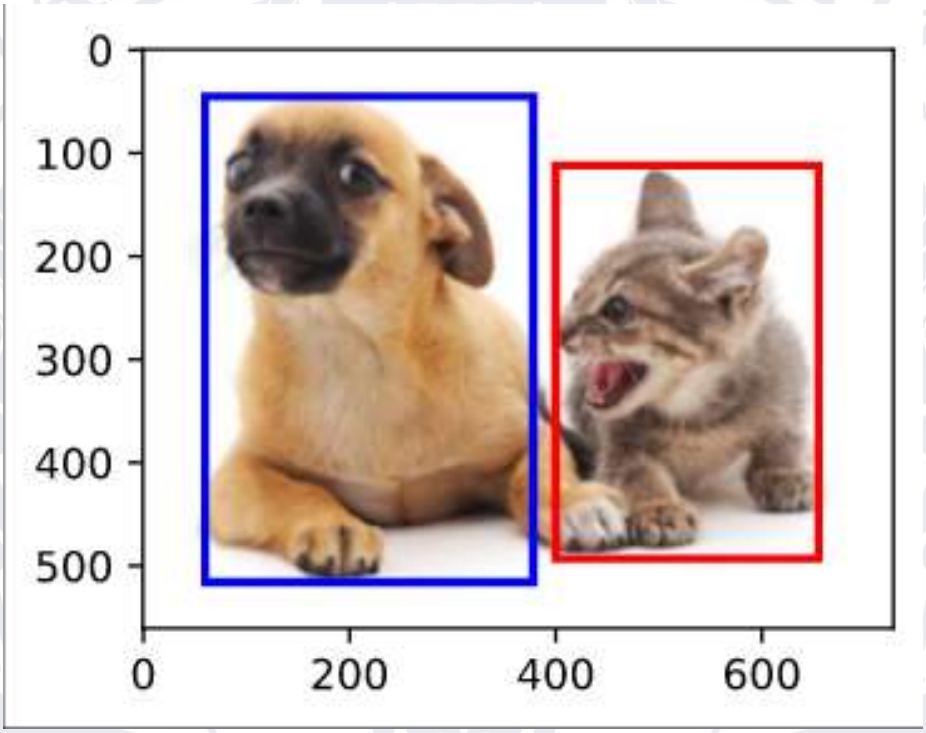


Рисунок 12. Результат виконання коду

4.2.2 Якірні рамки

У минулому підрозділі було описано, як саме алгоритмами комп’ютерного зору позначаються об’єкти. Цей формат підходить як для розмітки датасету, так і для виведення результатів роботи алгоритму. Але сам по собі формат представлення не є частиною процесу ідентифікації об’єктів. Алгоритми розпізнавання об’єктів зазвичай розбивають зображення на багато фрагментів, які називаються регіонами, потім певним чином вибираються масиви регіонів та алгоритми намагаються передбачити, наскільки певний

набір регіонів відповідає певному класу. Такі набори регіонів називаються якірними рамками [11].

В даному прикладі для більш детального опису якірних рамок буде використовуватись алгоритм, який випадково розташовує якірні рамки різного розміру на зображенні.

У вхідного зображення є висота h та ширина w. Також є розмір s та співвідношення ширини до висоти r, області визначення яких:

$$s \in (0, 1]$$

$$r > 0.$$

Тоді висота та ширина кожних якірних рамок будуть:

$$ws\sqrt{r}$$

$$hs/\sqrt{r},$$

Щоб створити якірні рамки різних розмірів, потрібно спочатку задати масиви різних значень розмірів та різних значень співвідношень ширини до висоти:

$$s_1, \ldots, s_n$$

$$r_1, \ldots, r_m.$$

Поєднання цих значень будуть давати розташування та розмір кожних якірних рамок:

$$(s_1, r_1), (s_1, r_2), \ldots, (s_1, r_m), (s_2, r_1), (s_3, r_1), \ldots, (s_n, r_1).$$

Для генерації даних якірних рамок буде використовуватись наступний код:

```
#@save
def multibox_prior(data, sizes, ratios):
    """Generate anchor boxes with different shapes centered on each pixel."""
    in_height, in_width = data.shape[-2:]
    device, num_sizes, num_ratios = data.device, len(sizes), len(ratios)
    boxes_per_pixel = (num_sizes + num_ratios - 1)
    size_tensor = torch.tensor(sizes, device=device)
    ratio_tensor = torch.tensor(ratios, device=device)
```

```
    # Offsets are required to move the anchor to the center of a pixel. Since
    # a pixel has height=1 and width=1, we choose to offset our centers by 0.5
    offset_h, offset_w = 0.5, 0.5
    steps_h = 1.0 / in_height  # Scaled steps in y axis
    steps_w = 1.0 / in_width  # Scaled steps in x axis

    # Generate all center points for the anchor boxes
    center_h = (torch.arange(in_height, device=device) + offset_h) * steps_h
    center_w = (torch.arange(in_width, device=device) + offset_w) * steps_w
    shift_y, shift_x = torch.meshgrid(center_h, center_w, indexing='ij')
    shift_y, shift_x = shift_y.reshape(-1), shift_x.reshape(-1)

    # Generate `boxes_per_pixel` number of heights and widths that are later
    # used to create anchor box corner coordinates (xmin, xmax, ymin, ymax)
    w = torch.cat((size_tensor * torch.sqrt(ratio_tensor[0]),
                   sizes[0] * torch.sqrt(ratio_tensor[1:])))\
                   * in_height / in_width  # Handle rectangular inputs
    h = torch.cat((size_tensor / torch.sqrt(ratio_tensor[0]),
                   sizes[0] / torch.sqrt(ratio_tensor[1:])))
    # Divide by 2 to get half height and half width
    anchor_manipulations = torch.stack((-w, -h, w, h)).T.repeat(
                                        in_height * in_width, 1) / 2

    # Each center point will have `boxes_per_pixel` number of anchor boxes,
so
    # generate a grid of all anchor box centers with `boxes_per_pixel` repeats
    out_grid = torch.stack([shift_x, shift_y, shift_x, shift_y],
                           dim=1).repeat_interleave(boxes_per_pixel, dim=0)
    output = out_grid + anchor_manipulations
    return output.unsqueeze(0)
```

Для візуалізації якірних рамок на зображенні потрібно використати наступний код:

```
#@save
def show_bboxes(axes, bboxes, labels=None, colors=None):
    """Show bounding boxes."""

    def make_list(obj, default_values=None):
    if obj is None:
    obj = default_values
    elif not isinstance(obj, (list, tuple)):
    obj = [obj]
    return obj

    labels = make_list(labels)
    colors = make_list(colors, ['b', 'g', 'r', 'm', 'c'])
    for i, bbox in enumerate(bboxes):
    color = colors[i % len(colors)]
    rect = d2l.bbox_to_rect(bbox.detach().numpy(), color)
    axes.add_patch(rect)
    if labels and len(labels) > i:
    text_color = 'k' if color == 'w' else 'w'
    axes.text(rect.xy[0], rect.xy[1], labels[i],
            va='center', ha='center', fontsize=9, color=text_color,
            bbox=dict(facecolor=color, lw=0))

plt.set_figsize()
bbox_scale = torch.tensor((w, h, w, h))
fig = d2l.plt.imshow(img)
show_bboxes(fig.axes, boxes[250, 250, :, :] * bbox_scale,
    ['s=0.75, r=1', 's=0.5, r=1', 's=0.25, r=1', 's=0.75, r=2',
    's=0.75, r=0.5'])
```

Результат виконання коду показано на рисунку 13.

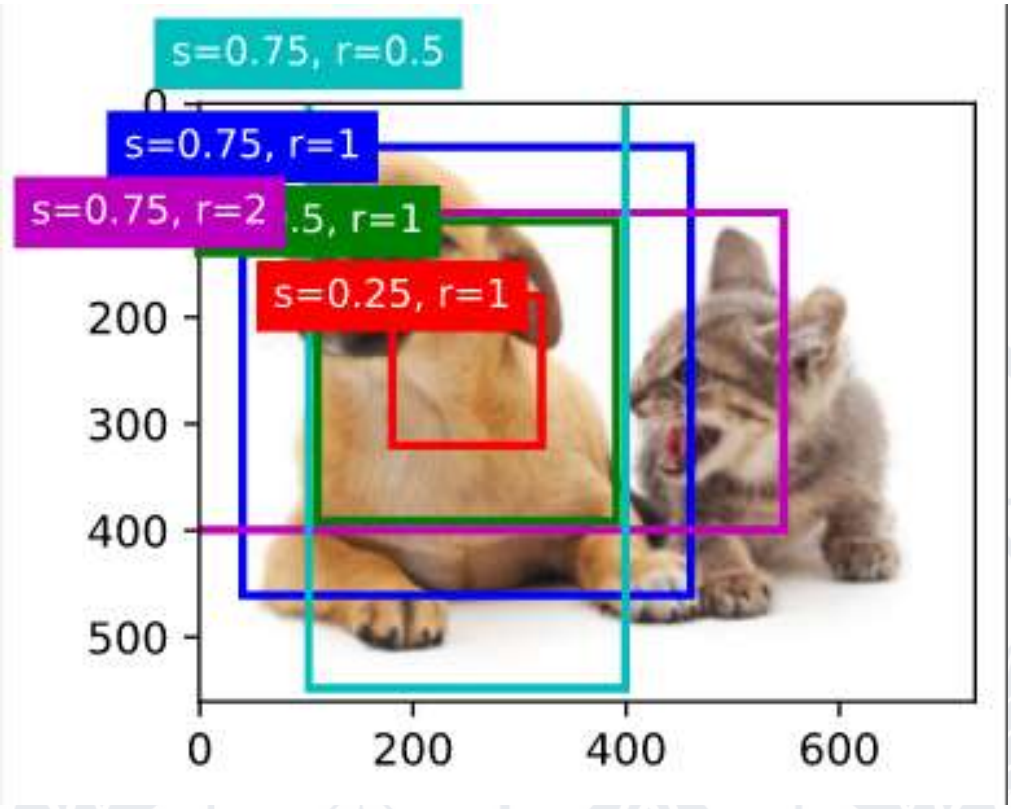


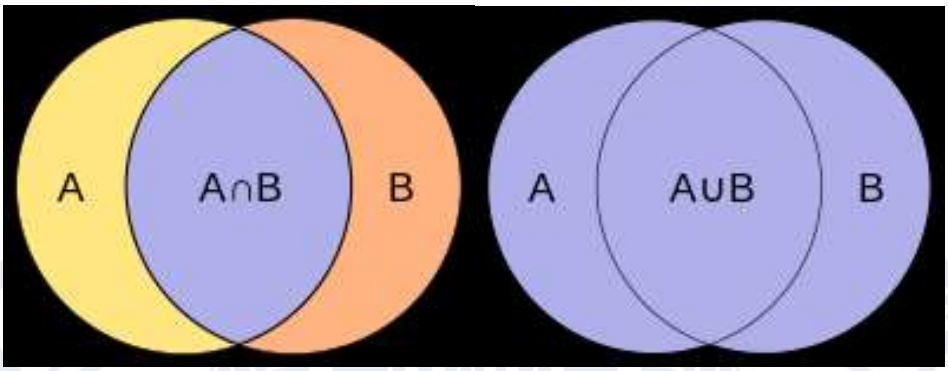
Рисунок 13. Результат виконання коду

Якщо казати про розпізнавання собаки на зображенні, то деякі якірні рамки краще окреслюють собаку, ніж інші. Але самі по собі якірні рамки не зберігають ніякої інформації. Щоб зрозуміти, наскільки точно якірні рамки позначають об’єкт, потрібно їх порівняти з рамками обмеження, що були надані для зображення в тренувальному датасеті. Це можна дізнатись через визначення коефіцієнту Жаккара [12].

Коефіцієнт Жаккара - це міра подібності двох множин, яка визначається у кількості спільних частин, поділених на міру об’єднання множин. Наприклад, для множин A і B для обчислення коефіцієнту застосовується формула:

$$J(\mathcal{A},\mathcal{B}) = \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}.$$

У випадку з алгоритмами розпізнавання об’єктів множинами будуть вважатись рамки обмеження та якірні рамки, а елементами множин будуть пікселі.



Рисунки 13 і 14. Візуалізація спільної частини та об’єднання множин

Але навіть розуміння схожості якірних рамок та рамок обмежень з датасету недостатньо, адже необхідно також виявити, який клас мають позначати якірні рамки і чи відносяться вони до певного класу взагалі.

Припустімо, що для одного зображення існують якірні рамки, які складають множину:

$$A_1, A_2, \ldots, A_{n_a}$$

А також рамки обмеження, що позначають об’єкти в датасеті, зі знанням про те, які класи вони позначають, в множині:

$$B_1, B_2, \ldots, B_{n_b}$$

Де □□ > □□. Також визначемо матрицю X, яка відповідає множині:

$$\mathbf{X} \in \mathbb{R}^{n_a \times n_b},$$

І в якій елемент □□□є точкою пересічення якірних рамок, згенерованих алгоритмом та рамок обмежень з тренувального датасету. Алгоритм присвоєння певних якірних рамок до певних рамок обмежень буде виглядати наступним чином:

1. Спочатку потрібно знайти найбільший елемент в матриці X. Позначимо його місцезнаходження в матриці через □*I*для рядка та □*I*для стовбчика. Рамки обмеження з тренувального датасету

□□1визначаються як відповідні до якірних рамок □□1. Після їх призначення очистити всі інші елементи в рядку □1та стовбці □1.

2. Тепер потрібно знайти найбільший елемент в матриці X серед тих, що залишились. Для цього елемента його місцезнаходження позначається як □2 для рядка та □2 для стовбця. Рамки обмеження з тренувального датасету □□2визначаються як відповідні до якірних рамок □□2. Після їх призначення очистити всі інші елементи в рядку □2та стовбці □2.
3. Повторювати ці кроки, поки не будуть оброблені елементи в усіх стовбцях та рядках матриці X.

Рисунок 15. Візуалізація алгоритму присвоєння якірних рамок до рамок обмежень.

На цьому етапі вже можна присвоювати класи для кожного екземпляру якірних рамок. Можна припустити, що для кожних якірних рамок має бути присвоєний той клас, який був присвоєний їх рамкам обмежень. Але проблема в тому, що по місцезнаходженню на зображенні якірні рамки та рамки обмежень можуть співпадати не повністю. Наприклад, для якірних

рамок A присвоєно клас рамок обмежень B, але координати їх пікселів не співпадають повністю. Отож спочатку необхідно визначити, наскільки великий зсув між A і B та визначити область, для якої можна присуджувати клас. Якщо для A і B координати центрів будуть (□□, □□) та (□□, □□), їхні значення ширини □□ та □□, а їхні значення висоти $h_□$ та $h_□$ відповідно, тоді можливо вирахувати зсув для A за допомогою формули:

$$\left(\frac{\frac{x_b - x_a}{w_a} - \mu_x}{\sigma_x}, \frac{\frac{y_b - y_a}{h_a} - \mu_y}{\sigma_y}, \frac{\log \frac{w_b}{w_a} - \mu_w}{\sigma_w}, \frac{\log \frac{h_b}{h_a} - \mu_h}{\sigma_h} \right)$$

Де для констант задані наступні значення:

$$\mu_x = \mu_y = \mu_w = \mu_h = 0, \sigma_x = \sigma_y = 0.1,$$

$$\sigma_w = \sigma_h = 0.2.$$

Деякі якірні рамки можуть бути не прив’язані до рамок обмежень. Тоді вони називаються негативними якірними рамками. Якщо ж якірні рамки прив’язані до рамок обмежень, тоді вони називаються позитивними якірними рамками.

Для демонстрації прикладу нижче наведено функцію на Python, яка на вхід приймає якірні рамки та класи. Функція визначає зсуви для якірних рамок і присвоює їм класи:

```
#@save
def multibox_target(anchors, labels):
    """Label anchor boxes using ground-truth bounding boxes."""
    batch_size, anchors = labels.shape[0], anchors.squeeze(0)
    batch_offset, batch_mask, batch_class_labels = [], [], []
    device, num_anchors = anchors.device, anchors.shape[0]
    for i in range(batch_size):
    label = labels[i, :, :]
    anchors_bbox_map = assign_anchor_to_bbox(
    label[:, 1:], anchors, device)
    bbox_mask = ((anchors_bbox_map >= 0).float().unsqueeze(-1)).repeat(
    1, 4)
```

```
        # Initialize class labels and assigned bounding box coordinates with
        # zeros
        class_labels = torch.zeros(num_anchors, dtype=torch.long,
                                   device=device)
        assigned_bb = torch.zeros((num_anchors, 4), dtype=torch.float32,
                                  device=device)
        # Label classes of anchor boxes using their assigned ground-truth
        # bounding boxes. If an anchor box is not assigned any, we label its
        # class as background (the value remains zero)
        indices_true = torch.nonzero(anchors_bbox_map >= 0)
        bb_idx = anchors_bbox_map[indices_true]
        class_labels[indices_true] = label[bb_idx, 0].long() + 1
        assigned_bb[indices_true] = label[bb_idx, 1:]
        # Offset transformation
        offset = offset_boxes(anchors, assigned_bb) * bbox_mask
        batch_offset.append(offset.reshape(-1))
        batch_mask.append(bbox_mask.reshape(-1))
        batch_class_labels.append(class_labels)
        bbox_offset = torch.stack(batch_offset)
        bbox_mask = torch.stack(batch_mask)
        class_labels = torch.stack(batch_class_labels)
        return (bbox_offset, bbox_mask, class_labels)
```

Далі необхідно викликати цей метод, передавши список якірних рамок та класи:

```
ground_truth = torch.tensor([[0, 0.1, 0.08, 0.52, 0.92],
                             [1, 0.55, 0.2, 0.9, 0.88]])
anchors = torch.tensor([[0, 0.1, 0.2, 0.3], [0.15, 0.2, 0.4, 0.4],
                        [0.63, 0.05, 0.88, 0.98], [0.66, 0.45, 0.8, 0.8],
                        [0.57, 0.3, 0.92, 0.9]])
```

```
labels = multibox_target(anchors.unsqueeze(dim=0),
                         ground_truth.unsqueeze(dim=0))
fig = plt.imshow(img)
show_bboxes(fig.axes, ground_truth[:, 1:] * bbox_scale, ['dog', 'cat'], 'k')
show_bboxes(fig.axes, anchors * bbox_scale, ['0', '1', '2', '3', '4']);
```

Результат виконання коду показано на зображенні 16.

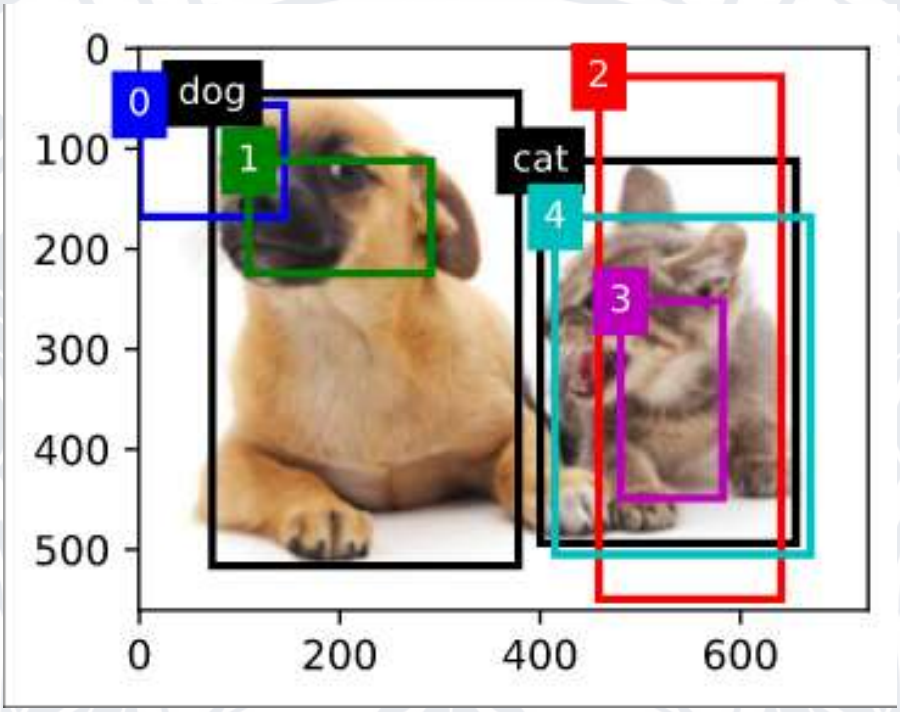


Рисунок 16. Результат виконання коду

4.2.3 Передбачення рамок обмежень

Процесс, що відбувався до цього, був співставлення якірних рамок та рамок обмежень з тренувального датасету, тобто фактично тренуванням моделі. Тепер, щоб передбачити клас об’єкту на зображенні, для зображення потрібно згенерувати якірні рамки і по них визначити рамки обмежень та їх клас.

Потенційно одному об’єкту, що може передбачити модель, можуть відповідати декілька якірних рамок, які мають спільні пікселі. Щоб зменшити кількість якірних рамок та зробити так, щоб вони не

перетинались, можна використати метод, який називається *non-maximum suppression* (скорочено NMS).

Принцип роботи NMS описано в наступному прикладі. Для передбачених моделлю рамок обмежень B алгоритм детекції рахує вірогідність приналежності рамок обмежень до певного класу. Клас, для якого визначено найвищу вірогідність, присуджується до рамок обмежень B. Всі інші рамки обмежень, для яких поки не присуджено класи, формують список L, де відсортовані по вірогідності приналежності до класу, що був присуджений для рамок обмежень B. Після чого список L обробляється по наступному алгоритму:

1. Зі списка беруться рамки обмеження $□_1$, які є першими в списку. Зі списку L видаляються рамки обмежень, коефіцієнт Жаккара яких в парі з рамками обмежень $□_1$ більше за задану константу ϵ. Таким чином, на цьому етапі уже частина зайвих рамок обмежень видалена зі списку.
2. Береться другий елемент зі списку $□_2$. Так само рі списку L видаляються рамки обмежень, коефіцієнт Жаккара яких в парі з рамками обмежень $□_2$ більше за задану константу ϵ.
3. Повторити дані кроки для усіх елементів списку.

4.3 Вибір типу нейронної мережі

На момент написання роботи існує декілька найбільш популярних алгоритмів машиного навчання, які використовуються при розпізнаванні об’єктів на зображеннях. Усі вони є варіантами так званих згорткових нейронних мереж [13].

У минулих підрозділах були визначенні поняття рамок обмежень та якірних рамок. В наступних підрозділах ці терміни будуть активно застосовуватись при описі моделей машинного навчання для розпізнавання об’єктів на зображеннях.

4.3.1 Region-based Convolutional Neural Network

Region-based Convolutional Neural Network (R-CNN) - варіант згорткової нейронної мережі, розроблений спеціально для розпізнавання візуальних об’єктів [14]. Основою R-CNN є селектор регіонів, що використовує так званий “вибірковий пошук”, алгоритм, що знаходить на зображеннях регіони пікселів, які можуть представляти об’єкти, також відомі як регіони інтересів (RoI, скорочено від англ. Regions of Interest). Для кожного зображення селектор регіонів визначає близько 2000 регіонів. Обмежене число регіонів дозволяє пришвидшити процес розпізнавання об’єктів.

Для розпізнавання об’єктів на зображеннях R-CNN працює за наступним алгоритмом:

1. На вхід подається зображення. З зображення витягуються близько 2000 запропонованих регіонів (в оригінальному документі, що описує R-CNN, називаються region proposals), цей термін позначає якірні рамки в контексті R-CNN. Обмеження числа запропонованих регіонів до такого числа дозволяє R-CNN працювати швидше за своїх попередників.
2. З кожного запропонованого регіона витягується *ознака* (англійською *feature*).

   Ознака - в розпізнаванні об’єктів та машинному навчанні це характеристика або окрема властивість спостережуваного явища, яку можливо виміряти.

   Ознаки витягуються у вигляді 4096-вимірних векторів з фрагменту розміром 276 на 276 пікселів в форматі RGB та обробляються п’ятьма згортковими шарами нейронної мережі.
3. Відбувається присвоєння класу для запропонованих регіонів. Щоб розрізнити об’єкт від заднього фону, оптимальний коефіцієнт Жаккара має бути 0,3.

Недоліки:

- Тренування моделі потребує багато часу;
- Не може бути використаним в режимі реального часу (наприклад, розпізнавання об’єктів з відео з прямої трансляції);

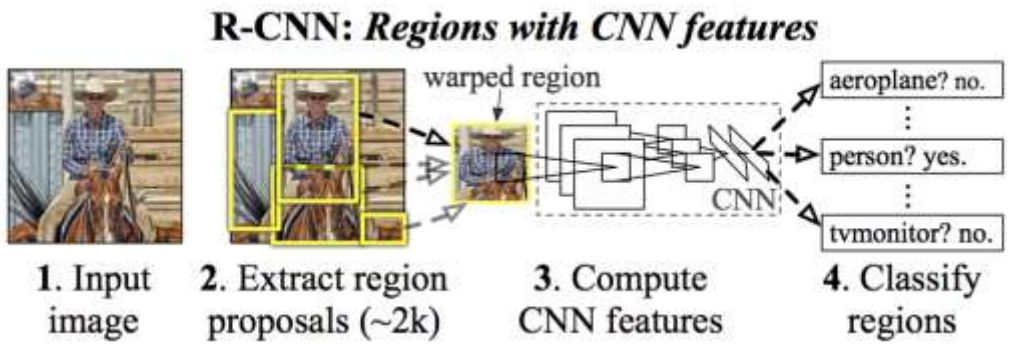


Рисунок 17. Приклад роботи R-CNN

4.3.2 Fast R-CNN

Fast R-CNN - згорткова нейронна мережа, заснована на R-CNN, призначена для більш швидкого процесу розпізнавання об’єктів [15]. Fast R-CNN працює швидше за R-CNN за рахунок того, що замість вибіркових регіонів, витягнутих з зображення, на вхід подається саме зображення, в процесі генеруючи карту ознак, де ознака - це векторне представлення об’єкту. З карти ознак ідентифікуються RoI і подаються на fully connected layer.

Основне слабке місце в продуктивності R-CNN полягає в тому, що механізм генерації якірних рамок (вони ж запропоновані регіони), для кожного об’єкту йде окремо, відповідно на кожний з них витрачається більше обчислювальної потужності, якщо об’єкти пересікаються. І якраз одним з найголовніших досконалень Fast R-CNN є те, що обчислення для запропонованих регіонів відбувається одразу для всього зображення.

Fast R-CNN працює за наступним алгоритмом:

1. На відміну від R-CNN, в Fast R-CNN на вхід подається повне зображення, а не фрагменти у вигляді запропонованих регіонів.
2. Селективний пошук генерує *n* запропонованих регіонів різного розміру. Дані запропоновані регіони позначають регіони інтересів на виході нейронної мережі. Для кожного регіону інтересів витягуються їх ознаки того ж розміру, що і регіони для того, щоб їх можна було легко об'єднати. Для цього Fast R-CNN використовує так званий *шар об'єднання регіонів інтересів* або ж RoI (від англійського region *of interest pooling layer*). Вихід CNN та запропоновані регіони подаються на вхід на шар RoI.
3. За допомогою під'єднаного шару розмір об'єднаних ознак змінюється на *n*d*, де *d* - це константа, значення якої залежить від реалізації Fast R-CNN.
4. Відбувається передбачення класів та рамок обмежень для кожного запропонованого регіону. Під час передбачення вихід під'єднаного шару трансформується до розміру *n*q*, де *q* це кількість класів.

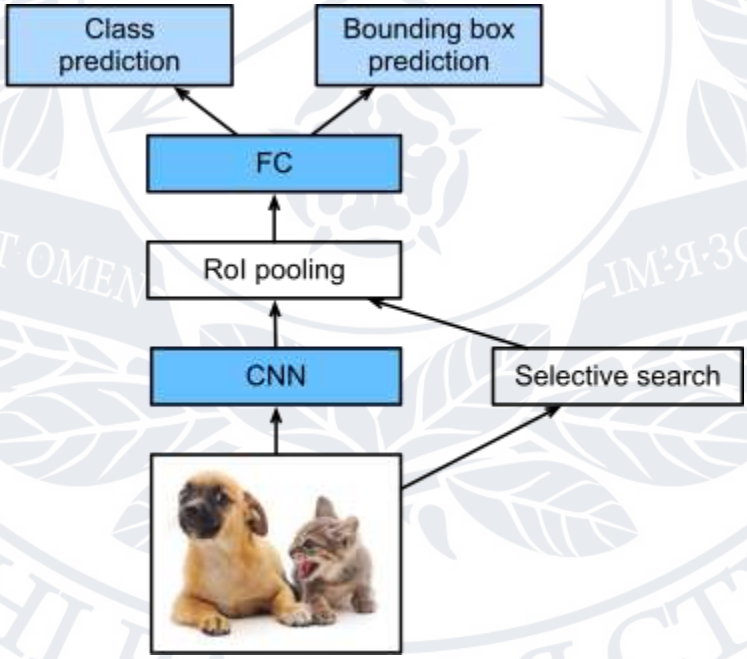


Рисунок 18. Візуалізація Fast R-CNN

На зображенні 19 показано візуалізацію шару об'єднання регіонів інтересів.

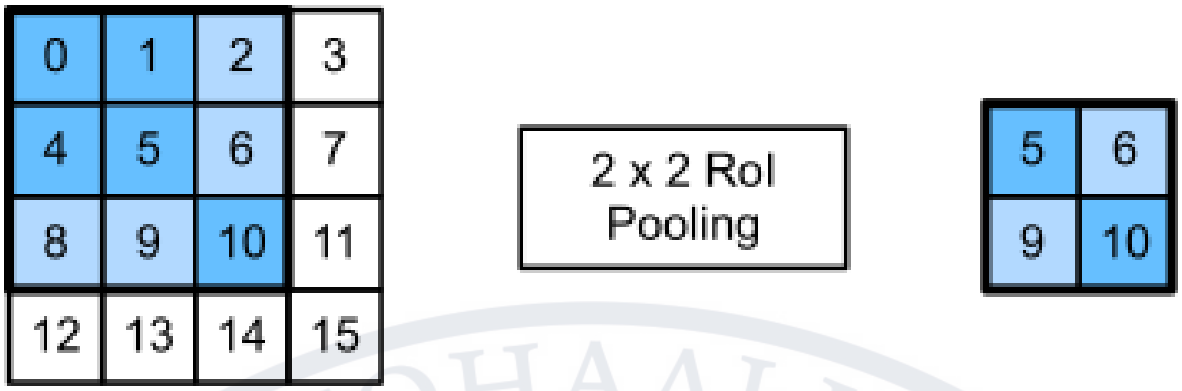


Рисунок 19. Візуалізація шару об’єднання регіонів інтересів

Недоліки:

- Незважаючи на те, що Fast R-CNN швидше за R-CNN, ця модель є недостатньо швидкою для обробки зображень в режимі реального часу.

4.3.3 Faster R-CNN

Faster R-CNN - згорткова нейронна мережа, заснована на Fast R-CNN та працює швидше за останню. Мережа працює ідентично до Fast R-CNN, але в даному варіанті замість селектора регіонів використовується окрема мережа для пропозицій регіонів (RPN, скорочено від англ. Region proposal network) [16].

Основною проблемою в Fast R-CNN є те, що необхідно згенерувати велику кількість запропонованих регіонів під час селективного пошуку, що негативно впливає на продуктивність моделі. Щоб вирішити цю проблему, в Faster R-CNN замість селективного пошуку використовується мережа запропонованих регіонів. Це єдина відмінність від Fast R-CNN.

Алгоритм роботи Faster R-CNN наступний:

1. Згортковий шар розміром 3x3 трансформує вихід CNN в новий вихід з *c* каналами. Таким чином, кожне значення з кожного виміру ознак

витягнутих з CNN, перетворюється в нову ознаку з розмірністю *c*, де *c* - це кількість каналів.

2. Для центру кожної ознаки генеруються декілька якірних рамок різних розмірів.
3. Використовуючи вектор розмірності c для кожних якірних рамок відбувається бінарна класифікація - чи позначають якірні рамки об’єкт або ж фон.
4. Якщо в якірних рамках визначається об’єкт, тоді всі інші передбачені результати в межах якірних рамок видаляються.

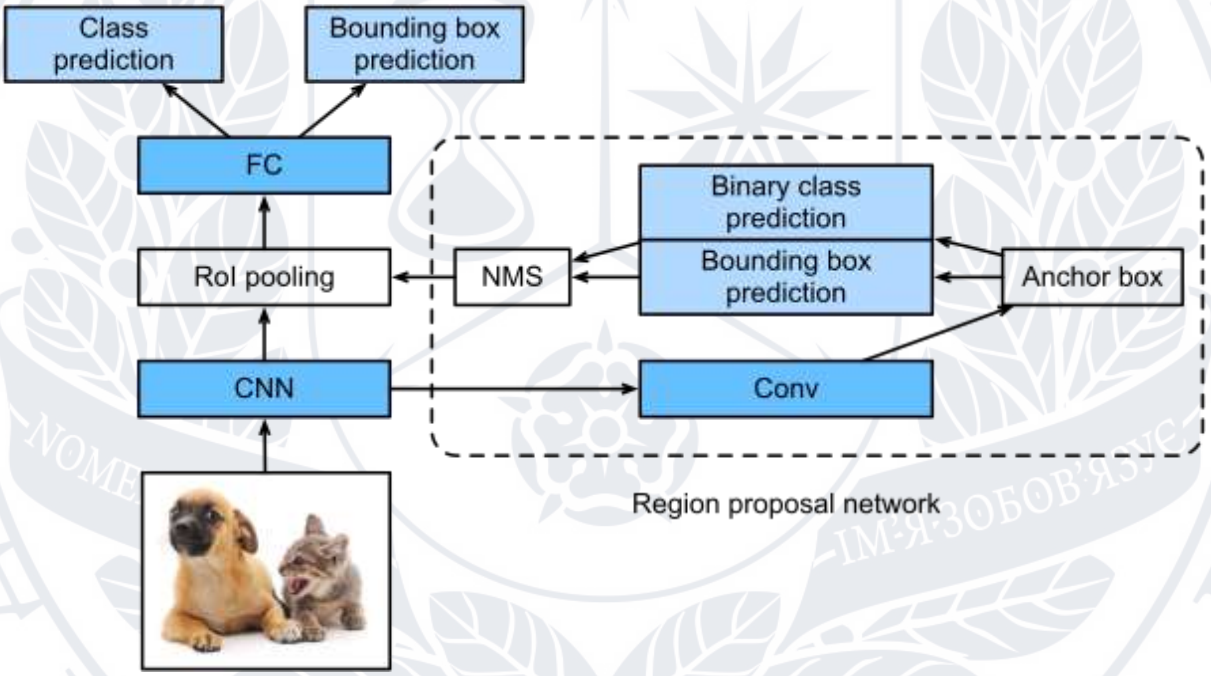


Рисунок 20. Візуалізація роботи Faster R-CNN

Недоліки:

- Незважаючи на те, що Faster R-CNN швидше за Fast R-CNN, ця модель є недостатньо швидкою для обробки зображень в режимі реального часу.

4.3.4 YOLO

YOLO (від англ. You Only Look Once) - набір нейронних мереж, створених для обробки зображень в режимі реального часу. YOLO працює швидше за Faster R-CNN за рахунок того, що процеси сегментації та класифікації об’єднані в одну мережу [17]. Для цього алгоритм використовує наступні техніки:

- Залишкові блоки (residual blocks)
- Регресія рамок обмежень (bounding box regression)
- Intersection over Union (IOU)

Спочатку на зображення накладається сітка розміром SxS, де S - це константа, що залежить від розміру зображення. Приклад сітки наведено на рисунку 21.

Рисунок 21. Приклад накладання сітки на зображення

Далі необхідно визначити рамки обмежень. Кожні рамки можуть вимірюватись наступними параметрами:

- Ширина
- Висота
- Клас
- Координати центру фрагменту

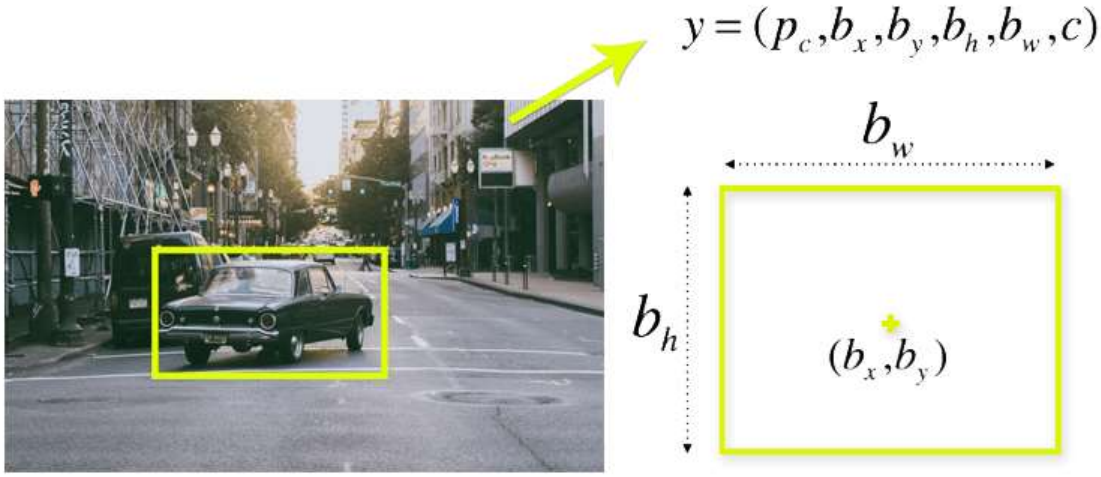


Рисунок 22. Приклад рамок обмежень

Особливість роботи YOLO в тому, що алгоритм комбінує генерацію рамок обмежень та класифікацію одночасно, що дозволяє проводити детекцію об’єктів в режимі реального часу.

Висновок - для вирішення задачі обрано набір нейронних мереж YOLO, оскільки вони дозволяють обробляти зображення в режимі реального часу, що необхідно за умов, в яких модель може використовуватися.

4.4 Навчання моделі

4.4.1 Налагодження середовища для розробки

В якості платформи для навчання моделі було обрано Google Colab - онлайн-сервіс, який дозволяє виконувати код, написаний на Python на серверах Google, розрахованих для роботи з даними та моделями машинного навчання.

Інтерфейс Google Colab показано на рисунку 23.

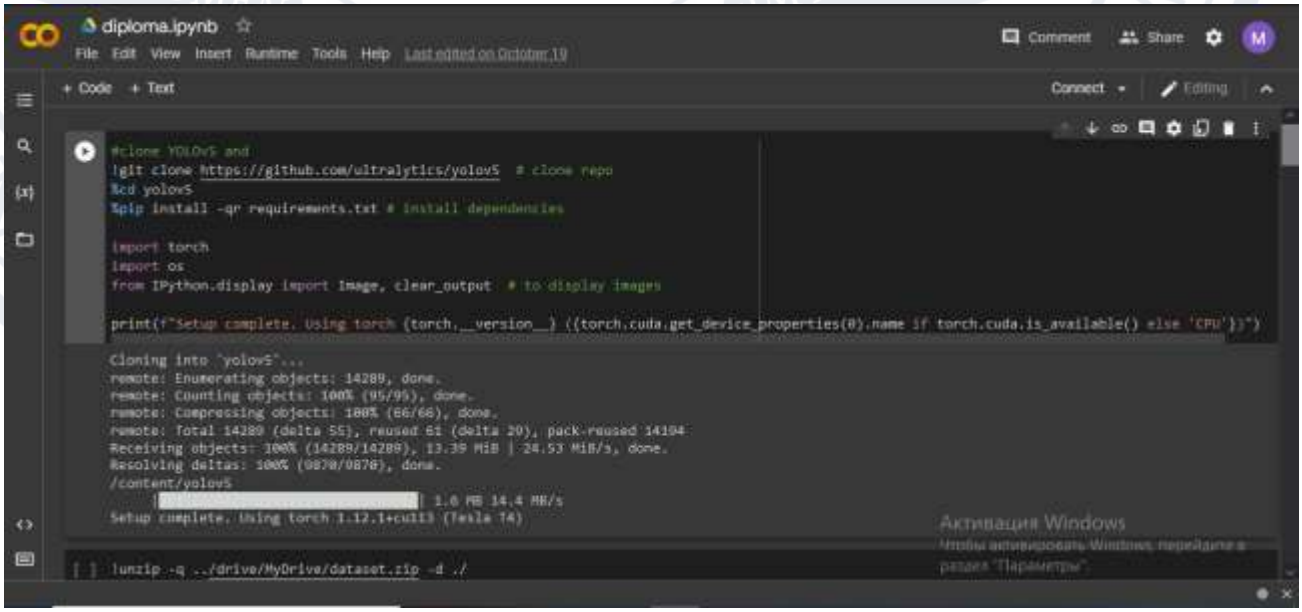


Рисунок 23. Інтерфейс Google Colab

В якості реалізації моделі нейронної мережі YOLO було обрано бібліотеку YOLOv5 - бібліотеку з відкритим вихідним кодом, написану на Python.

Щоб тренувати модель в Google Colab за допомогою бібліотеки YOLOv5, спочатку створимо новий документ Google Drive, натиснувши на кнопку New, обравши пункт More та підпункт Google Colaboratory, як показано на рисунку 24.

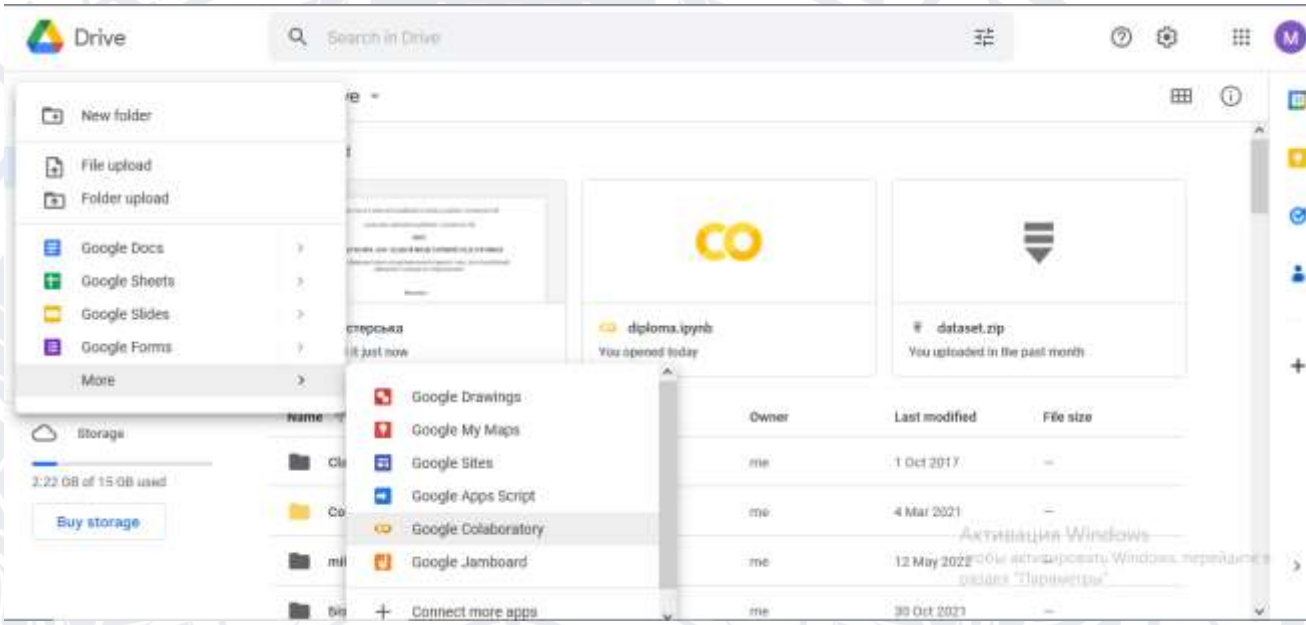


Рисунок 24. Створення нового документу Google Colab

Далі необхідно створити поле для введення коду, використавши в інтерфейсі кнопку Code, як показано на рисунку 25.

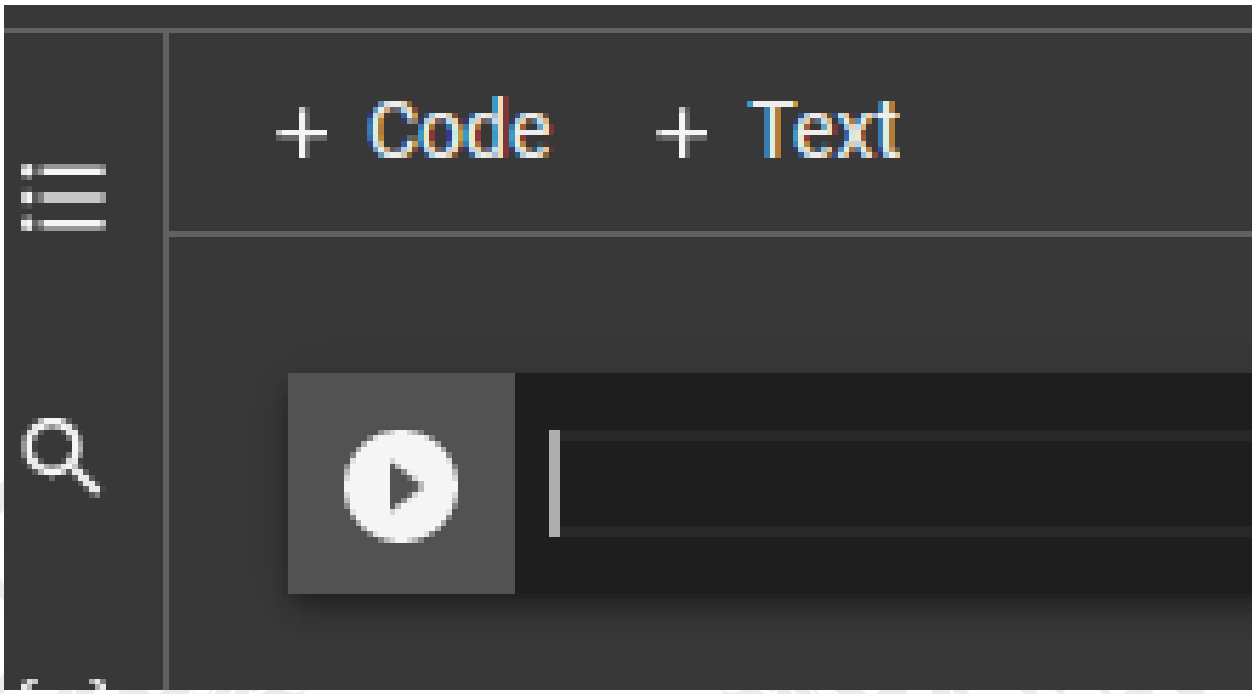


Рисунок 25. Кнопка Code в інтерфейсі Google Colab

Далі в новоутворене поле потрібно ввести код для завантаження та встановлення бібліотеки YOLOv5:

```
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies
```

Далі потрібно створити ще одне поле для введення коду і ввести код, що показує, яка версія бібліотеки PyTorch та який графічний процесор (GPU) будуть використовуватись:

```
import torch
import os
from IPython.display import Image, clear_output  # to display images

print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")
```

Оскільки YOLOv5 написана з використанням бібліотеки PyTorch, немає потреби її окремо завантажувати. Після виконання коду було виведено текст: «Setup complete. Using torch 1.12.1+cu113 (Tesla T4)». Це означає, що для навчання моделі буде використовуватись бібліотека PyTorch версії 1.12.1 на базі платформи для паралельних комп'ютерних обчислень CUDA версії 11.3, а в якості платформи для виконання коду буде

використовуватись сервер з встановленим графічним процесором NVIDIA Tesla T4.

Далі необхідно створити файл в форматі YAML, в якому будуть описані сутності, які має класифікувати на зображеннях модель машинного навчання. В даному випадку це наступні різновиди військової техніки:

- IMP-2 (рисунок 26)
- МіГ-31 (рисунок 27)
- Т-72 (рисунок 28)
- Торнадо-Г (рисунок 29)
- Мі-8 (рисунок 30)

Рисунок 26. IMP-2

Рисунок 27. МіГ-31

Рисунок 28. Т-72

Рисунок 29. Торнадо-Г

Рисунок 30. Mi-8

Кінцеве наповнення файлу буде мати наступний вигляд:

path: ./dataset/

train: train

val: val

names:

0: IMR-2

1: MiG-31

2: T-72

3: Tornado-G

4: Mi-8

Поле path позначає шлях до директорії, де зберігаються файли датасету. Поля train та val позначають директорії датасету, де зберігаються дані для для тренування та валідації моделі відповідно. В полі names перераховуються назви об’єктів, що мають розпізнаватись.

Після того, як всі необхідні файли підготовлені, можна почати тренувати модель. Для цього необхідно запустити команду: python train.py --batch 16 --epochs 300 --data coco128.yaml --weights yolov5s.pt.

СПИСОК ЛІТЕРАТУРИ

1. Researchers train AI on ‘synthetic data’ to uncover Syrian war crimes [Електроний ресурс]. - Режим доступу до ресурсу - https://www.ft.com/content/8399873e-0dda-4c87-ba59-0e2678166fba.

2. Agency - Forensic Architecture [Електроний ресурс]. - Режим доступу до ресурсу - https://forensic-architecture.org/about/agency.

3. Гіґґінс Еліот. Ми - Bellingcat. Онлайн-розслідування міжнародних злочинів та інформаційна війна з Росією / пер. з англ. Орина Ємельянова. - К. : Наш Формат, 2022. - 240 с.

4. What Hundreds of Photos of Weapons Reveal About Russia’s Brutal War Strategy [Електроний ресурс]. - Режим доступу до ресурсу - https://www.nytimes.com/interactive/2022/06/19/world/europe/ukraine-munitions-war-crimes.html.

5. People + AI Guidebook. User Needs + Defining Success [Електроний ресурс]. - Режим доступу до ресурсу - https://pair.withgoogle.com/chapter/user-needs/.

6. People + AI Guidebook. What’s new when working with AI [Електроний ресурс]. - Режим доступу до ресурсу - https://pair.withgoogle.com/chapter/user-needs/#whats-new.

7. People + AI Guidebook. Find the intersection of user needs & AI strengths [Електроний ресурс]. - Режим доступу до ресурсу - https://pair.withgoogle.com/chapter/user-needs/#section1.

8. People + AI Guidebook. Assess automation vs. augmentation [Електроний ресурс]. - Режим доступу до ресурсу - https://pair.withgoogle.com/chapter/user-needs/#section2.

9. People + AI Guidebook. Design & evaluate the reward function [Електроний ресурс]. - Режим доступу до ресурсу - https://pair.withgoogle.com/chapter/user-needs/#section3.

10. Object Detection and Bounding Boxes [Електроний ресурс]. - Режим доступу до ресурсу - https://d2l.ai/chapter_computer-vision/bounding-box.html.

11. Anchor Boxes [Електроний ресурс]. - Режим доступу до ресурсу - https://d2l.ai/chapter_computer-vision/anchor.html.

12. Statistics How To. Statistics for the rest of us! Jaccard Index / Similarity Coefficient [Електроний ресурс]. - Режим доступу до ресурсу - https://www.statisticshowto.com/jaccard-index/.

13. Region-based CNNs (R-CNNs) [Електроний ресурс]. - Режим доступу до ресурсу - https://d2l.ai/chapter_computer-vision/rcnn.html.

14. Girshick, R., Donahue, J., Darrell, T., & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition (2014).

15. Girshick, R. Fast R-CNN. Proceedings of the IEEE international conference on computer vision (2015).

16. Ren, S., He, K., Girshick, R., & Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. Advances in neural information processing systems (2015).

17. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. 2016 (University of Washington, Allen Institute for AI, Facebook AI Research).

18. Russian aggression in Ukraine: Overview, Evidence and Map [Електроний ресурс]. - Режим доступу до ресурсу - https://informnapalm.org/en/russian-military-intervention-ukraine-overview-evidence-map/

19. The Battle of Ilovaisk. Mapping Russian Military Presence in Eastern Ukraine. August–September 2014 [Електроний ресурс]. - Режим доступу до ресурсу - https://ilovaisk.forensic-architecture.org/.

20. New evidence emerges of Russian role in Ukraine conflict [Електроний ресурс]. - Режим доступу до ресурсу -

https://amp.theguardian.com/world/2019/aug/18/new-video-evidence-of-russian-tanks-in-ukraine-european-court-human-rights.

21. DeepAI. Jaccard Index [Електроний ресурс]. - Режим доступу до ресурсу - https://deepai.org/machine-learning-glossary-and-terms/jaccard-index.

22. A Simple Explanation of the Jaccard Similarity Index [Електроний ресурс]. - Режим доступу до ресурсу - https://www.statology.org/jaccard-similarity/.

23. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way [Електроний ресурс]. - Режим доступу до ресурсу - https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

24. Introduction to YOLO Algorithm for Object Detection [Електроний ресурс]. - Режим доступу до ресурсу - https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/.

Марчук Михайло Борисович
Прізвище, ім’я по батькові

Факультет інформаційних і прикладних технологій
Факультет

122 Комп’ютерні науки
Шифр і назва спеціальності

Комп’ютерні технології обробки даних (Data Science)
Освітня програма

ДЕКЛАРАЦІЯ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (магістерська) робота на тему «Використання алгоритмів комп’ютерного зору для розпізнавання військової техніки на зображеннях» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувалась іншими особами, а також дані та інформація не отримувалась у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

_____

_____

(дата) (підпис здобувача освіти)