

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Навчально-науковий інститут прикладного системного аналізу

Кафедра штучного інтелекту

До захисту допущено:

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«___» 20__ р.

**Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи і методи штучного
інтелекту»
спеціальності 122 «Комп'ютерні науки»
на тему: «Застосування глибокого навчання для виявлення військових
об'єктів»**

Виконав:

студент IV курсу, групи КІ-03
Капусткін Артем Олександрович

Керівник:

к.т.н., доц. Тимошенко Юрій Олександрович

Консультант з економічного розділу:

доц. Мажара Гліб Анатолійович

Консультант з нормоконтролю:

фахівець першої категорії кафедри ШІ,
Кравець П.В.

Рецензент:

декан ФПМ, д.т.н., професор
Дичка Іван Андрійович

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

Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри

_____ Ірина Джигирей

«31» січня 2024 р.

ЗАВДАННЯ
на дипломну роботу студента
Капусткіну Артему Олександровичу

1. Тема роботи «Застосування глибокого навчання для виявлення військових об'єктів», керівник роботи к.т.н., доц. Тимошенко Ю.О., затверджені наказом по університету від «31» травня 2024 р. №2240-С
2. Термін подання студентом роботи «10» червня 2024 року.
3. Вихідні дані до роботи: власноруч зібраний набір зображень військової техніки.
4. Зміст роботи: Аналіз предметної області та її актуальності. Дослідження та обробка даних, вибір доцільних методів глибокого навчання та архітектур нейронних мереж для роботи, процес навчання, оцінка отриманих результатів.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): зображення вхідного набору даних, ілюстративні матеріали до розділів, графіки метрик оцінок навчання, презентація до виступу.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	доц. Мажара Г. А.		

7. Дата видачі завдання «05» лютого 2024 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за темою	15.04.2024	Виконано
2	Підготовка першого розділу	22.04.2024	Виконано
3	Підготовка другого розділу	28.04.2024	Виконано
4	Розробка програмного продукту	05.05.2024	Виконано
5	Підготовка третього розділу	12.05.2024	Виконано
6	Підготовка четвертого розділу	19.05.2024	Виконано
7	Проведення функціонально-вартісного аналізу та оформлення п'ятого розділу	26.05.2024	Виконано
8	Оформлення розділів відповідно до нормоконтролю	01.06.2024	Виконано
9	Оформлення дипломної роботи	07.06.2024	Виконано
10	Підготовка презентації доповіді	10.06.2024	Виконано

Студент

Артем КАПУСТКІН

Керівник

Юрій ТИМОШЕНКО

РЕФЕРАТ

Дипломна робота: 100 с., 47 рис., 6 табл., 33 посилання, 1 додаток.

ГЛИБОКЕ НАВЧАННЯ, КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ОБ'ЄКТІВ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, АНАЛІЗ ЗОБРАЖЕНЬ, АНОТАЦІЯ ДАНИХ.

Об'єкт дослідження – військове застосування глибокого навчання.

Предмет дослідження – нейронна мережа та її застосування для задачі розпізнавання об'єктів.

Мета роботи – отримати навчену нейронну мережу, здатну розпізнавати військові об'єкти.

Методи дослідження – аналіз літератури, дослідження існуючих методів та алгоритмів глибокого навчання, збір зображень військових об'єктів та їх анотація, тренування моделей глибокого навчання та проведенням експериментів для оцінки їх ефективності.

Результати дослідження демонструють високу точність виявлення класу броньованої техніки і помітно нижчу для класу легкої техніки через меншу кількість зображень цього класу. В цілому, результати є прийнятними, враховуючи нерівномірність даних. Новизна роботи полягає у створенні унікального набору даних з актуальними зображеннями військових об'єктів, зібраними зокрема під час сучасних військових конфліктів, таких як російсько-українська війна. Відображення реальних бойових умов робить набір даних цінним для подальших досліджень і розробки систем автоматичного розпізнавання.

ABSTRACT

Master's thesis: 100 p., 47 figures, 6 tables, 33 references, 1 appendix.

DEEP LEARNING, COMPUTER VISION, OBJECT RECOGNITION,
CONVOLUTIONAL NEURAL NETWORK, IMAGE ANALYSIS, DATA
ANNOTATION.

The object of the study is the military application of deep learning.

The subject of research is a neural network and its application to the task of object recognition.

The purpose of the work is to obtain a trained neural network capable of recognizing military objects.

Research methods include literature analysis, research of existing deep learning methods and algorithms, collection of images of military objects and their annotation, training of deep learning models, and experiments to evaluate their effectiveness.

The results of the study demonstrate high accuracy of detecting the class of armored vehicles and significantly lower for the class of light vehicles due to the smaller number of images of this class. Overall, the results are acceptable given the unevenness of the data. The novelty of the work is the creation of a unique dataset with actual images of military objects collected in particular during modern military conflicts, such as the Russo-Ukrainian war. The reflection of real combat conditions makes the dataset valuable for further research and development of automatic recognition systems.

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1 ТЕХНОЛОГІЯ КОМП'ЮТЕРНОГО ЗОРУ ТА ЇЇ АКТУАЛЬНІСТЬ ДЛЯ ВІЙСЬКОВОГО ЗАСТОСУВАННЯ	10
1.1 Проблематика військового розвідування	12
1.2 Використання дронів та БПЛА у військових цілях	15
1.3 Перспективи впровадження інтелектуальних систем	18
Висновки до розділу 1	19
РОЗДІЛ 2 ГЛИБОКЕ НАВЧАННЯ ТА ЙОГО ЗАСТОСУВАННЯ В КОМП'ЮТЕРНОМУ ЗОРІ	20
2.1 Методи комп'ютерного зору	27
2.2 Підходи до розпізнавання об'єктів	36
Висновки до розділу 2	45
РОЗДІЛ 3 НАВЧАННЯ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ	46
3.1 Середовище розробки	46
3.2 Тренувальний набір даних	47
3.3 Процес навчання моделі	49
Висновки до розділу 3	57
РОЗДІЛ 4 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТУ	59
4.1 Розпізнавання об'єктів на фото	59
4.2 Розпізнавання об'єктів на відео	64
4.3 Перспектива розгортання моделі на дроні	66
Висновки до розділу 4	67
РОЗДІЛ 5 ФУНКЦІОНАЛЬНО-ВАРТИСНИЙ АНАЛІЗ	68
5.1 Постановка задачі проектування	68
5.2 Обґрунтування функцій програмного продукту	69
5.3 Обґрунтування системи параметрів програмного продукту	72
5.4 Аналіз експертного оцінювання параметрів	76

5.5 Аналіз рівня якості варіантів реалізації функцій	80
5.6 Економічний аналіз варіантів розробки ПП	81
5.7 Вибір кращого варіанту ПП техніко-економічного рівня	86
Висновки до розділу 5	87
ВИСНОВКИ	88
ПЕРЕЛІК ПОСИЛАНЬ	90
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ	94

ВСТУП

В останні роки спостерігається тенденція збільшення використання штучного інтелекту, особливо в напрямі комп'ютерного зору в різноманітних сферах. Від охорони здоров'я до транспорту – технології комп'ютерного зору, керовані штучним інтелектом, змінюють процеси в галузях та взаємодію людей із технологіями.

На сьогоднішній день, багато аспектів автоматичного розпізнавання військових об'єктів уже досліджено та частково впроваджено. Зокрема, розроблено алгоритми для розпізнавання та класифікації об'єктів на супутникових знімках і відео з дронів. Однак, більшість таких розробок ведуться секретно військовими організаціями різних країн світу, що створює прогалини у загальнодоступних знаннях та обмежує доступ до передових технологій для широкого кола дослідників. Існує також проблема недостатньої кількості відкритих та актуальних наборів даних, що є критично важливими для тренування та тестування моделей глибокого навчання.

На глобальному рівні, дослідники активно працюють над покращенням точності та ефективності моделей глибокого навчання для розпізнавання військових об'єктів. Наприклад, на платформах, таких як Roboflow, доступні відкриті моделі, навчені на зображеннях із різноманітних джерел, включаючи відеоігри, де представлена сучасна та історична військова техніка. Ці відкриті моделі демонструють прогрес у публічних дослідженнях і сприяють спільному розвитку технологій.

Актуальність даної роботи обумовлена необхідністю створення більш точних та адаптивних систем розпізнавання військових об'єктів в умовах реальних бойових дій. Збір даних під час сучасних військових конфліктів, таких як Російсько-українська війна, дозволяє зібрати унікальні набори даних, які відображають реальні умови на полі бою. Це надає можливість для покращення існуючих моделей і розробки нових алгоритмів, здатних ефективно працювати в

умовах сучасних конфліктів.

Метою роботи є отримати нейронну мережу, здатну розпізнавати військові об'єкти з використанням актуальних даних. Галузь застосування результатів охоплює військову розвідку, автоматизоване управління бойовими системами та аналітичні платформи для оцінки військової ситуації.

Ця робота тісно пов'язана з попередніми дослідженнями у сфері комп'ютерного зору та глибокого навчання. Використання методів аугментації даних, що застосовуються в інших галузях, таких як охорона здоров'я та транспорт, може сприяти покращенню точності моделей для військових застосувань. Дослідження базуються на відкритих моделях та методиках, що дозволяє інтегрувати нові досягнення та розширювати існуючі знання.

Таким чином, дана робота не тільки сприяє розвитку технологій автоматичного розпізнавання військових об'єктів, але й заповнює прогалини у знаннях, створюючи основи для майбутніх досліджень та інновацій.

РОЗДІЛ 1 ТЕХНОЛОГІЯ КОМП'ЮТЕРНОГО ЗОРУ ТА ЇЇ АКТУАЛЬНІСТЬ ДЛЯ ВІЙСЬКОВОГО ЗАСТОСУВАННЯ

Задача комп'ютерного бачення полягає у навчанні комп'ютерів отримувати значущу інформацію з цифрових зображень, відео та інших візуальних даних, а також надавати рекомендації або вживати заходів, коли вони бачать дефекти або проблеми. У цьому такі системи перевершують людські здібності, оскільки здатні аналізувати тисячі продуктів або процесів за хвилину. Комп'ютерний зір працює майже так само, як людський, за винятком того, що людина навчається розпізнавати об'єкти та їх положення впродовж життя, тоді як комп'ютерний зір вимагає машини виконувати ці функції за набагато менший час за допомогою камер, даних і алгоритмів, а не сітківки, зорових нервів і зорової кори [4].

Метою розпізнавання об'єктів (англ. object detection) [1] є ідентифікація та локалізація об'єктів на цифрових зображеннях або відеокадрах (рис. 1.1).

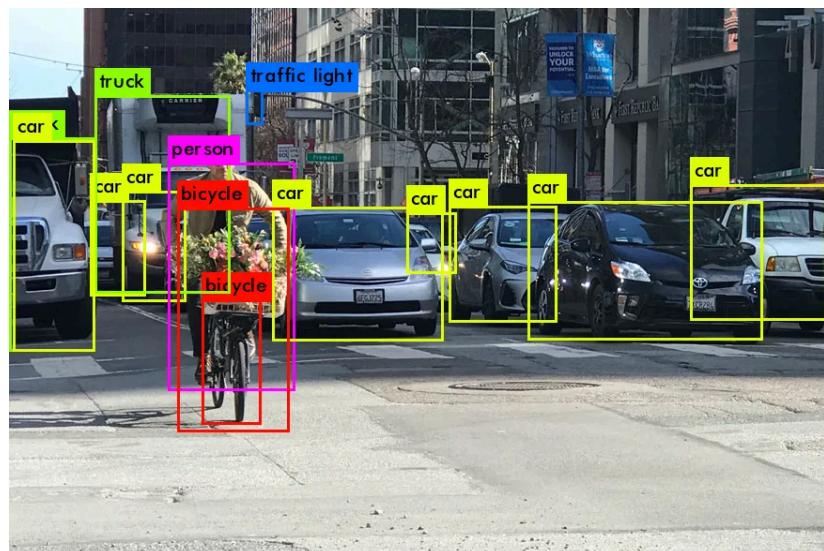


Рисунок 1.1 – Результат розпізнавання об'єктів в міській місцевості¹

¹ Джерело зображення:

https://miro.medium.com/v2/resize:fit:828/format:webp/0*XLOCUJnALL6XKto9.png

Системи виявлення об'єктів на основі штучного інтелекту використовують складні алгоритми, зокрема методи глибокого навчання, для автоматизації цього процесу. Зі стрімким розвитком мереж глибокого навчання для задач виявлення, продуктивність детекторів об'єктів значно покращилася.

Системи автопілотів є важливим кроком у розвитку автомобільної промисловості. Вони дозволяють автомобілям автоматично керувати рухом без пряմого втручання водія. Ця технологія існує вже десятиліттями, але останні роки принесли значний прогрес у цьому напрямі. Системи автопілотів відкривають нові можливості для безпеки на дорозі, зменшення аварій та комфорту для водіїв. Наприклад, Tesla [33] вже впроваджує систему автопілоту в свої автомобілі, яка здатна автоматично керувати автомобілем, уникати перешкод та дотримуватися правил дорожнього руху.

Аеророзвідка та спостереження трансформуються, дрони та БПЛА, оснащені системами комп'ютерного бачення, автономно ідентифікують об'єкти на землі, надаючи розвідувальну інформацію для різноманітних додатків, включаючи охорону кордонів, реагування на стихійні лиха та моніторинг навколошнього середовища, адже за допомогою комп'ютерного зору можна ефективно сканувати територію. Останнім часом прослідовується зростання ролі технологій у військових діях, зокрема використання безпілотних літальних апаратів та дронів, відкрило нові можливості та викиди для стратегічного та тактичного планування військових операцій.

Використання технологій комп'ютерного зору у вище перелічених сферах є підґрунтям для її застосування у військових цілях, зокрема для виявлення військових об'єктів.

1.1 Проблематика військового розвідування

Традиційні методи військової розвідки століттями відігравали ключову роль у збиранні розвідувальних даних і формуванні стратегій ведення бойових дій. Неможливо переоцінити важливість інформації про маневри противника, місцезнаходження укриття особового складу або артилерійської батареї, умови місцевості, адже той хто нею володів мав стратегічну перевагу на полі бою. Однак традиційні методи не позбавлені недоліків, це спонукало шукати нові більш хитрих, ефективних і дієвих рішень у сфері розвідки.

Найрозповсюдженішим у всі часи було використання невеликих наземних розвідувальних груп, переважно до 20 військовослужбовців. Зазвичай така розвідка мала обмежений радіус дії через фізичні обмеження особового складу. Кількісне розширення розвідників в такій групі збільшує ризики виявлення, що загрожує викриттям і ставить під загрозу цілі завдання. Крім того така розвідка вимагає значних витрат часу і ресурсів, що затримує отримання своєчасних розвідувальних даних, необхідних для прийняття рішень.

З винаходом фотографії з'явилається можливість застосовувати її для розвідки [21]. Ще на початку Першої світової війни для спостереженням за противником почали використовувати аерофотозйомку. Камери закріплялись на повітряних кулях та аеростатах на великий висоті, з якої було видно ворожі позиції. Згодом камери почали закріплювати й на літаки, що дозволило за більш короткий час отримувати різноманітні знімки більшої території, хоча й вона була обмежена кількістю палива. Аналіз цих знімків був важливим елементом планування та картографії. Звісно навіть такий передовий вид розвідки мав свої недоліки. Під час несприятливої погоди такі заходи малоефективні, також в протидію розвідувальним літакам виступали винищувачі-перехоплювачі та зенітні установки, що накладало обмеження на їх використання.

Поява супутникових технологій [21] значно розширила сферу застосування військової розвідки, забезпечивши глобальне покриття і високу

роздільну здатність зображень (див. рис. 1.2). Проте, вона стикається з власними проблемами. Зазвичай такі знімки мають недостатню роздільність для детального аналізу наземних цілей або дій. Супутники обмежені своїми заздалегідь визначеними орбітальними траєкторіями, що обмежує їхню здатність фіксувати динамічні події, які відбуваються на землі, в реальному часі. Вразливі до глушіння методами радіоелектронної боротьби, яка виводить з ладу супутників лінії зв'язку або погіршує можливості отримання зображень.

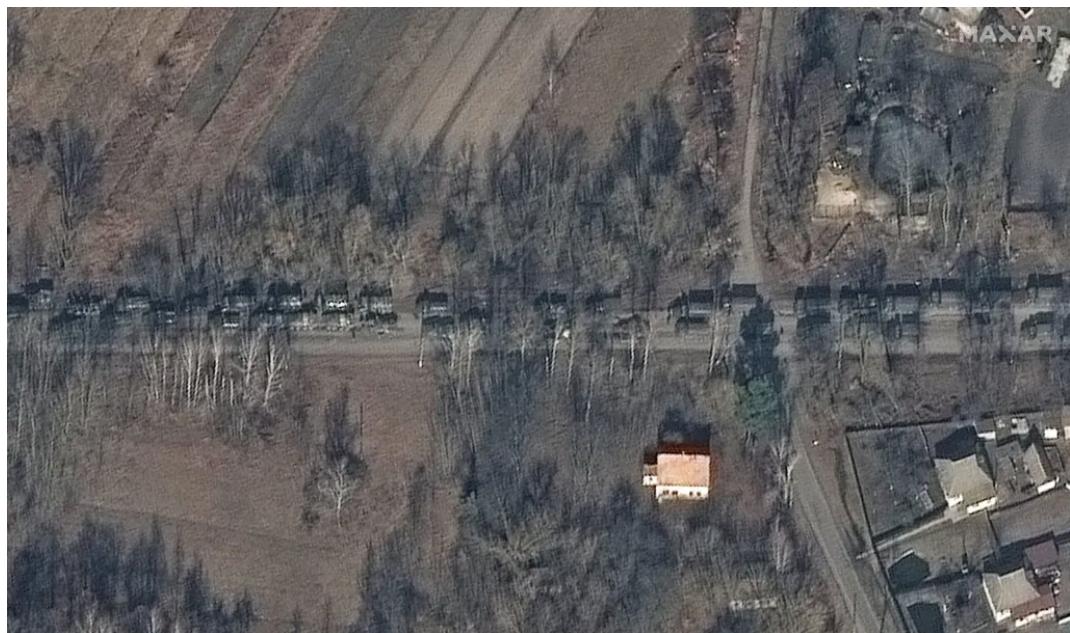


Рисунок 1.2 – Зображення військової колони з супутника Maxar²

Інші види розвідки такі як проникнення силами спеціальних операцій в тил противника, не позбавлені ризиків і викликів. Таким завданнями притаманні ризики, в тому числі виявлення ворожими силами, захоплення в полон або участь у ворожих зіткненнях. Підрозділи спеціальних операцій обмежені чисельністю особового складу і ресурсів, що накладає обмеження на масштаби їх застосування.

² Джерело зображення:

<https://i0.wp.com/spacenews.com/wp-content/uploads/2022/02/FModdsiXEAIZgAD-scaled.jpeg?resize=1024%2C606&ssl=1>

Традиційні методи військової розвідки слугували основою збору розвідувальної інформації протягом багатьох поколінь, надаючи цінну інформацію про можливості та наміри противника. Проте, ці методи не позбавлені недоліків і стикаються з проблемами, пов'язаними з досяжністю, вразливістю і обмеженістю ресурсів. Російсько-українська війна продемонструвала світу мінливість характеру війни і важливість технологічної переваги на ворогом (див. рис. 1.3). Беручи це до уваги збройні сили провідних країн адаптуються до сучасного формату війни, інтеграція передових технологій, таких як БПЛА та дронів, пропонує перспективні рішення для подолання цих викликів і посилення розвідувальних спроможностей на сучасному полі бою.

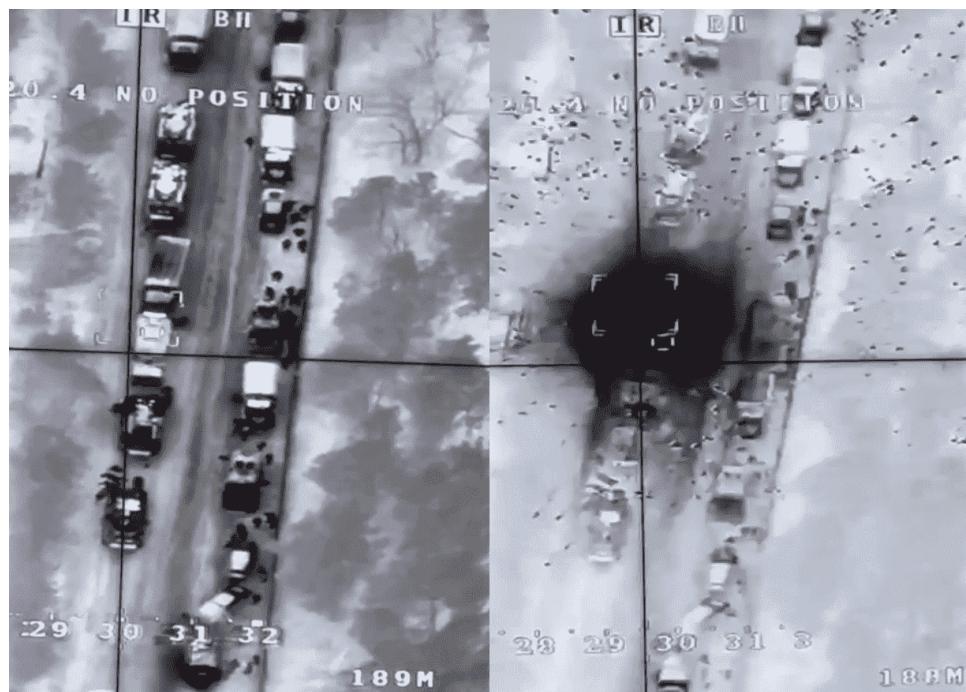


Рисунок 1.3 – Знімок знищеної військової колони з турецького БПЛА Bayraktar TB2³

³ Джерело зображення: <https://www.spectator.co.uk/wp-content/uploads/2022/03/8875.png?resize=1010,720>

1.2 Використання дронів та БПЛА у військових цілях

Безпілотні літальні апарати (БПЛА) або дрони стали частиною нового витка технологічної революції військової справи. Запропонувавши можливості, які покращують спостереження, ведення бою, логістику тощо. Їх застосування продовжує зростати завдяки технологічному прогресу і стратегічним перевагам, які вони надають.

БПЛА мають вирішальне значення для збору розвідданих і спостереження за полем бою. Оснащені камерами з високою роздільною здатністю і сучасними датчиками, такі безпілотники, як MQ-9 Reaper (див. рис. 1.4), надають дані про пересування і позиції противника в режимі реального часу. Вони також відіграють життєво важливу роль в операціях з патрулювання кордонів, відстежуючи величезні території на предмет незаконної діяльності та вторгнень. Додатково можуть бути оснащені високоточними боєприпасами, що дозволяє військовим наносити точкові удари з мінімальними супутніми втратами. Ці безпілотники забезпечують безпосередню повітряну підтримку наземним військам, знищуючи сили противника і водночас зменшуючи ризик для пілотів-людей. Іншим завданням безпілотників є радіоелектронна боротьба, коли вони застосовуються для виведення з ладу РЛС і ведення радіоелектронної розвідки [22].

На сьогодні деякі види БПЛА (наприклад К-МАХ) вже здатні доставляти вантажі у віддалені або ворожі райони. Це дозволяє забезпечити війська необхідними боєприпасами, медикаментами та іншими предметами першої необхідності, не наражаючи конвої постачання на ворожі атаки. Окрім логістики БПЛА допомагають у пошуково-рятувальних місіях, визначаючи місцезнаходження збитих пілотів або військ, що зазнали лиха. Дрони, оснащені тепловізійними та іншими датчиками, можуть швидко охоплювати великі території, що значно підвищує шанси на успішну рятувальну операцію.



Рисунок 1.4 – Військовий БПЛА MQ-9 Reaper, оснащений камерою для збору інформації та підвісним арсеналом для завдання ударів по ворожим цілям⁴

БПЛА слугують гарними мішенями для тренувань зенітних підрозділів та випробувань нових систем озброєння. Їх використання в навчальних сценаріях допомагає підвищити ефективність стратегій протиповітряної оборони і гарантує, що війська добре підготовлені до реальних бойових ситуацій.

Російсько-українська війна відзначалась широким застосуванням ударних безпілотників, що перетворило її на полігон для тестування та вдосконалення цієї технології. Одними з таких є FPV-дрони [23] (з англ. FPV - з видом від першої особи), оснащені камерами і вибуховим пристроєм, які зарекомендували себе серед військових як ефективний і недорогий інструмент для знищення ворожих цілей без наражання особового складу на прямі бойові дії. Пряма трансляція відеосигналу дозволяє вносити корективи в реальному часі, і завдавати точні удаres під час дистанційного пілотування. Такі дрони доволі

⁴ Джерело зображення:

<https://static01.nyt.com/images/2024/02/19/multimedia/19dc-houthisdrone-pjbv/19dc-houthisdrone-pjbv-jumbo.jpg?quality=75&auto=webp>

прості в виробництві і часто збираються на базі цивільних дронів, через те відносно недорогі у порівнянні з іншими видами зброї та тим більш з технікою, яку вони призначені вражати. Не рідкісні випадки, коли такий дрон собівартістю декілька тисяч доларів вражає багатомільйонну ворожу техніку. Через таку доступність, військові можуть широко застосовувати їх. Проте FPV-дрони можуть бути використані не тільки для ударів по наземній техніці чи особовому складу ворога, а я для розвідки та знаходження ворожих позицій чи об'єктів. Така можливість подвійного застосування робить їх універсальними засобами на полі бою.

Серйно з боку Російської Федерації у поточній війни зарекомендували себе спеціальні ударні безпілотники Shahed-136 і Lancet. Shahed-136 (відомий на російському озброєнні як "Герань-2") призначений для атак на великі відстані і може нести значне вибухове навантаження. У конфлікті він використовувався переважно для атак на об'єкти української інфраструктури, такі як електростанції та мережі зв'язку, спричиняючи значні руйнування та пошкодження. Ланцет – російський баражуючий боєприпас, що має більш досконалу систему наведення, це дозволяє йому завдавати точних ударів по важливих цілях. Ланцет використовується для ураження українських військових об'єктів, в тому числі артилерійських позицій, бронетехніки і систем протиповітряної оборони. Його точність і універсальність роблять його грізною зброєю на полі бою [24].

БПЛА особливо добре підходять для ведення асиметричної війни, коли нетрадиційна тактика використовується для протидії більш технологічно розвиненому супротивнику, наприклад використання для моніторингу поля бою в режимі реального часу, що дозволяє командирам приймати обґрунтовані рішення на полі бою, корегувати артилерійського вогню, проводити удари по критично важливих цілях. Як російські, так і українські сили застосовують ці безпілотники з різною метою, що має значний вплив на перебіг конфлікту.

1.3 Перспективи впровадження інтелектуальних систем

Майбутнє БПЛА та дронів у військових операціях очікує на значний прогрес, зумовлений постійними дослідженнями і технологічними інноваціями. Досягнення в галузі штучного інтелекту і машинного навчання, дозволять дронам виконувати завдання по типу автоматизованої розвідки. Оснащені високоточними камерами та сенсорами дрони й зараз збирають величезні обсяги візуальної та інфрачервоної інформації, ці дані можуть бути швидко оброблені для визначення ворожих об'єктів та оцінки загроз, що дозволить значно скоротити час від отримання інформації до прийняття рішень. Збільшуючи кількість таких дронів, можна отримати цілі рої дронів, які будуть синхронізовано збирати інформацію на великих ділянках фронту, тим самим гарантуючи більше покриття території і точність розвідувальних даних.

Це також крок до автоматизації бойових місій, до прикладу виявлення, наведення та знищення цілі з мінімальним втручаннями людини. Важливим аспектом є підвищення безпеки особового складу. Використання автономних БПЛА дозволяє зменшити кількість солдатів, які безпосередньо беруть участь у розвідувальних та бойових операціях. Це значно знижує ризик втрат серед військових і дозволяє зосередити людські ресурси на стратегічному плануванні та управлінні [25].

Незважаючи на всі переваги, впровадження інтелектуальних систем в безпілотники викликає також низку етичних та правових питань. Автономні бойові системи можуть приймати рішення про знищенння цілей без участі людини, що піднімає питання щодо відповідальності за такі дії та можливих помилок. Крім того, існує ризик неправильного використання цих технологій, що може привести до загибелі мирного населення або неправомірного застосування сили.

Висновки до розділу 1

У першому розділі дипломної роботи розглянута технологія комп’ютерного зору, приклади її застосування у різних сферах, розглянуті можливості її застосування у військових цілях. Військова розвідка протягом своєї історії змінювалась завдяки новим технологіям, поява на полі бою дронів не стала винятком. Сучасні війська широко використовують дрони, як для збору інформації, так і для бойових чи рятувальних задач. Впровадження інтелектуальних систем для тих чи інших задач вже стала наступним кроком у розвитку застосувань дронів для військових задач. Тому доволі актуальним є задача дослідження застосування комп’ютерного зору для виявлення військових об’єктів.

РОЗДІЛ 2 ГЛИБОКЕ НАВЧАННЯ ТА ЙОГО ЗАСТОСУВАННЯ В КОМП'ЮТЕРНОМУ ЗОРИ

Глибоке навчання знайшло застосування у таких областях як комп'ютерний зір (англ. Computer Vision), обробку природної мови (англ. Natural Language Processing, NLP) та розпізнавання мови (англ. Speech Recognition). Його сила в здатності моделювати та навчатися на великих обсягах даних, витягуючи складні закономірності за допомогою багаторівневих архітектур, відомих як нейронні мережі. В основі глибокого навчання лежать нейронні мережі [14], які натхненні структурою і функціями людського мозку. Нейронна мережа складається з взаємопов'язаних шарів вузлів (нейронів), де кожен зв'язок має відповідну вагу. Нейрон (рис. 2.1) є базовою обчислювальною одиницею, основна функція якого полягає в обробці вхідних сигналів, зважуванні їх за допомогою вагових коефіцієнтів, застосуванні зсуву (bias) та передаванні обчисленого значення через активаційну функцію, що визначає вихідний сигнал.

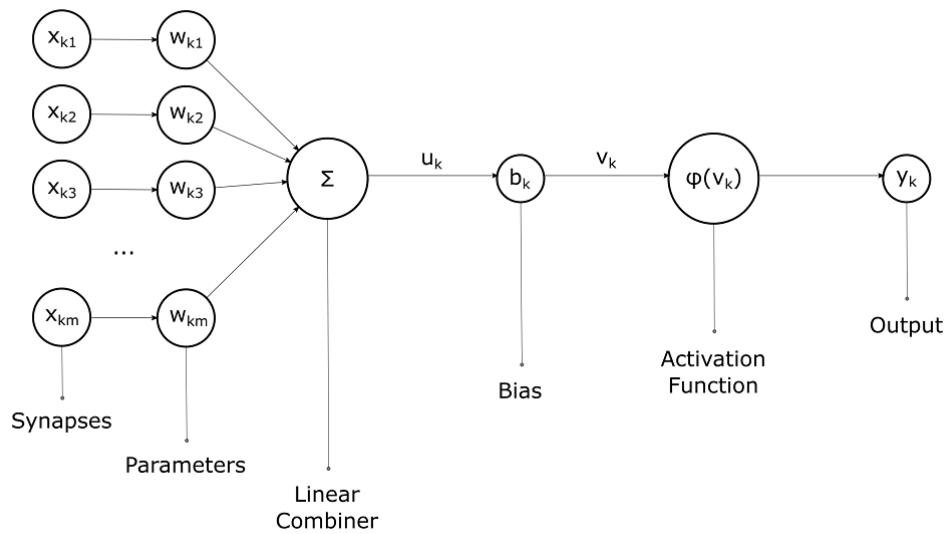


Рисунок 2.1 – Схематичне зображення нейрону в нейронній мережі⁵

⁵ Джерело зображення: https://miro.medium.com/v2/resize:fit:1000/1*vbRXwvbk4ep_kK3dgfAWzQ.png

Нейрон у штучній нейронній мережі (ШНМ) складається з кількох ключових компонентів:

- синапси (англ. synapses) – вхідні сигнали, позначені як $x_{k1}, x_{k2}, x_{k3}, \dots, x_{km}$, що подаються на нейрон;
- параметри (англ. parameters) – вагові коефіцієнти $w_{k1}, w_{k2}, w_{k3}, \dots, w_{km}$, що визначають важливість кожного вхідного сигналу;
- лінійний комбайнер (англ. linear combiner), у якому вхідні сигнали і їх ваги комбінуються лінійно, створюючи зважену суму, позначену як u_k , вона обчислюється за формулою 2.1;

$$u_k = \sum_{i=1}^m x_{ki} w_{ki} \quad (2.1)$$

- зсув (англ. bias) b_k додається до зваженої суми, що допомагає змістити функцію активації, після чого отримуємо значення v_k , що дорівнює рівнянню 2.2;

$$v_k = u_k + b_k \quad (2.2)$$

- значення v_k проходить через функцію активації (англ. activation function) $\phi(v_k)$, яка визначає вихід нейрону y_k і служить для введення нелінійності у модель.

Існує три основні типи шарів: вхідний шар отримує початкові дані, приховані шари (англ. hidden layers) виконують різні обчислення для виявлення закономірностей та особливостей у даних, вихідний шар виробляє остаточний прогноз або класифікацію. Кожен нейрон обробляє свої входи, застосовує зважену суму, пропускає її через функцію активації та надсилає вихід на наступний рівень [14].

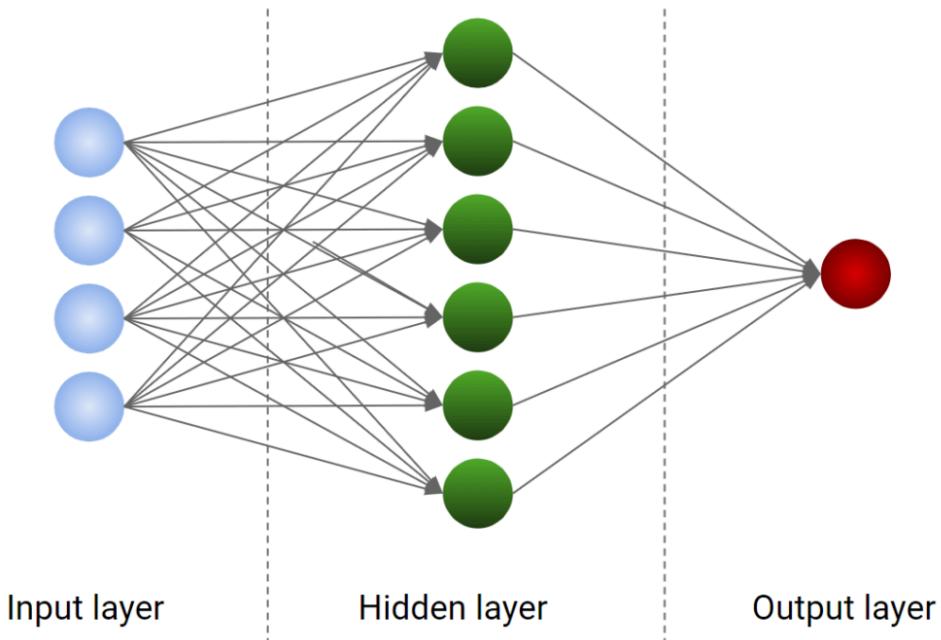


Рисунок 2.2 – Ілюстративне зображення структури шарів нейронної мережі⁶

Функції активації вносять нелінійність у мережу, дозволяючи їй вивчати складні закономірності. Частіше всього зустрічаються наступні функції активації:

- сигмоїд виводить значення від 0 до 1, корисні для бінарної класифікації;
- тангенс виводить значення від -1 до 1, що дозволяє відцентрувати дані;
- ReLU (випрямлена лінійна одиниця) [15] виводить безпосередньо вхідні дані, якщо вони додатні; в іншому випадку виводить нуль.

Зворотне розповсюдження (англ. back propagation) [16] використовується для навчання нейронних мереж, а саме для мінімізації функції втрат. Він включає прямий прохід, коли вхідні дані пропускаються через мережу для генерації прогнозів, і зворотний прохід, коли помилки мережі обчислюються і передаються назад для оновлення вагових коефіцієнтів., яка вимірює різницю між фактичними та прогнозованими значеннями (рис. 2.3).

⁶ Джерело зображення:

<https://www.yourdatateacher.com/wp-content/uploads/2021/04/neural-network-1024x665.png>

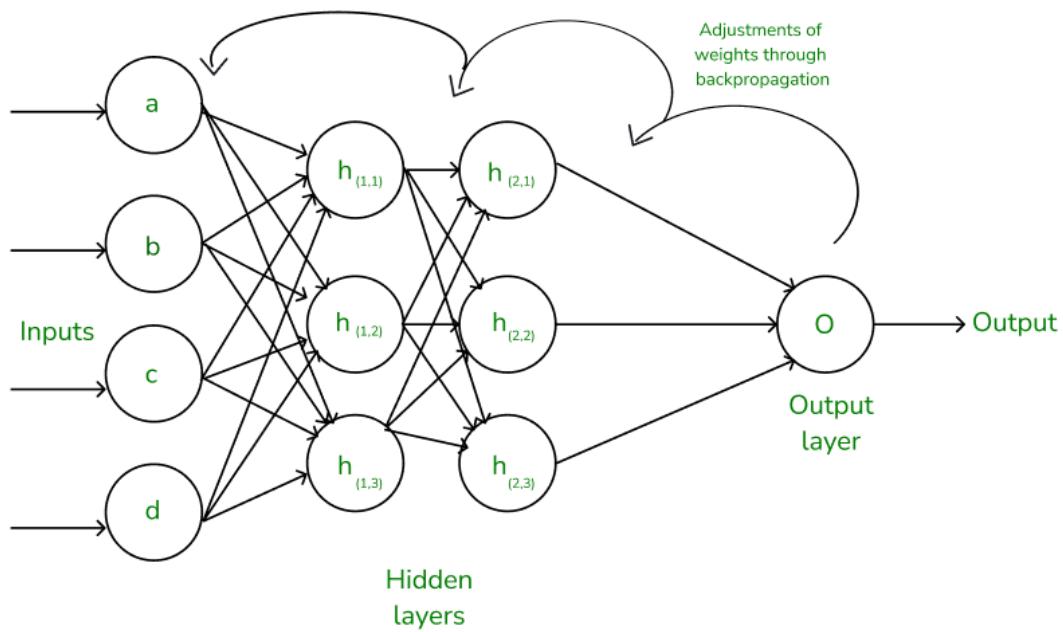


Рисунок 2.3 – Ілюстрація алгоритму зворотного поширення⁷

Функція втрат [32], також відома як функція вартості (англ. loss/cost function) кількісно визначає, наскільки добре прогнози моделі відповідають фактичним даним, керуючи процесом оптимізації для покращення продуктивності моделі. Мета функції втрат - надати єдине значення, яке містить в собі продуктивність моделі на заданому наборі даних. Під час навчання метою є мінімізація цієї функції втрат, тим самим підвищуючи точність і надійність моделі. Функція втрат надає сигнал, який алгоритм оптимізації (наприклад, градієнтний спуск) використовує для налаштування параметрів моделі. Мінімізуючи втрати, параметри моделі ітеративно оновлюються, щоб краще відповідати даним. Дозволяє порівнювати різні моделі або конфігурації, оцінюючи, яка з них дає найнижчі втрати на перевірочному наборі даних.

Різні завдання і моделі вимагають різних типів функцій втрат. До прикладу для задачі розпізнавання зображень часто використовують комбінації функцій втрат.

⁷ Джерело зображення: <https://media.geeksforgeeks.org/wp-content/uploads/20240217152156/Frame-13.png>

- Localization loss (також Smooth L1 loss) [11] використовується для регресії обмежувальних коробок (рамки навколо об'єкту в просторі), яка зазвичай передбачає прогнозування чотирьох значень для кожної обмеженої рамки: координат x та y центру рамки, а також ширини та висоти рамки. Smooth L1 loss менш чутлива до викидів, ніж MSE loss та поєднує в собі найкращі аспекти L1 і L2 loss.

$$\text{Localization Loss} = \sum_{i=1}^n \text{smooth}_{L1}(x_i - y_i) \quad (2.3)$$

- Confidence Loss [11] використовується для класифікації визначених об'єктів, вимірює впевненість у тому наскільки об'єкт належить певному класу.

$$\text{Confidence Loss} = \sum_{i=1}^n \left[1_i^{obj} (-\log(p_i)) + 1_i^{noobj} (-\log(1 - p_i)) \right] \quad (2.4)$$

де 1_i^{obj} – функція індикатор, за умови що в опорної рамці i є об'єкт рівна одиниці, в іншому випадку нуль; 1_i^{noobj} – функція індикатор, за умови що в опорній рамці i немає об'єкта дорівнює одиниці, в іншому випадку нуль; p_i – прогнозована ймовірність наявності об'єкта в обмежувальній рамці i .

Градієнтний спуск (англ. Gradient Descent) [31] є ітеративним алгоритмом оптимізації, який використовується для мінімізації згаданої вище функції втрат. Для цього йому потрібні дві точки даних - напрямок і швидкість навчання (англ. learning rate). Ці фактори визначають обчислення часткових похідних для наступних ітерацій, що дозволяє поступово досягти локального і глобального

мінімуму (тобто точки збіжності). Швидкість навчання часто називають розміром кроку або альфа α . Зазвичай це невелика величина, яка оцінюється і оновлюється на основі поведінки функції витрат. Висока швидкість навчання призводить до більших кроків, проте існує ризик перевищення мінімуму (див. рис. 2.4). І навпаки, низька швидкість навчання має невеликі розміри кроків.Хоча вона має перевагу в більшій точності, кількість ітерацій ставить під загрозу загальну ефективність, оскільки для досягнення мінімуму потрібно більше часу і обчислень.

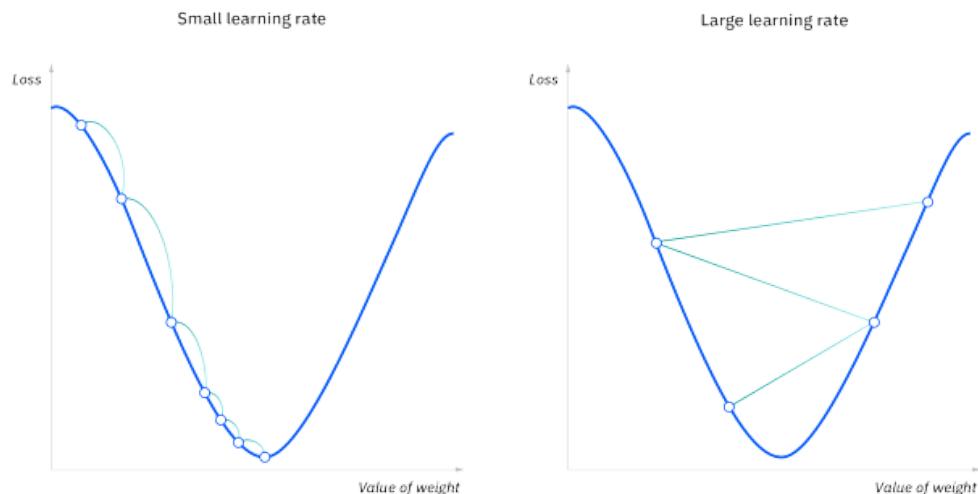


Рисунок 2.4 – Порівняння різних значень швидкостей навчання⁸

Ініціалізація початкового припущення щодо параметрів (ваг та зміщень), їх значення можуть бути встановлені випадковим чином або за допомогою евристики. Обчислюється градієнт функції втрат для кожного параметра. Цей градієнт є вектором часткових похідних, що вказує напрямок і швидкість найкрутішого підйому функції втрат. Оновлення параметрів у напрямку,

⁸ Джерело зображення:

https://www.ibm.com/content/dam/connectedassets-adobe-cms/worldwide-content/cdp/cf/ul/g/c2/0f/ICLH_Diagram_Batch_03_21-AI-ML-GradientDescent.component.simple-narrative-m.ts=1712347664958.png/content/adobe-cms/us/en/topics/gradient-descent/jcr:content/root/table_of_contents/body/content_section_styled/content-section-body/simple_narrative_1771421240/image

протилежному градієнту (оскільки ми хочемо мінімізувати втрати, а не максимізувати їх). Правило оновлення має наступний вигляд:

$$\theta = \theta - \alpha \nabla_{\theta} J(\theta) \quad (2.5)$$

де θ – відображає параметри, α – швидкість навчання (контролює розмір кроку), $\nabla_{\theta} J(\theta)$ – градієнт функції втрат $J(\theta)$ відносно параметра θ .

Процес обчислення градієнта повторюється, відповідно оновлюються параметри до збіжності або конвергенції, коли зміна функції втрат не стане нижчою за певний поріг. Розрізняють наступні види градієнтного спуску.

1. Пакетний градієнтний спуск (англ. Batch Gradient Descent) використовує весь набір даних для обчислення градієнта функції втрат, збігається до глобального мінімуму для опуклих функцій, стабільно оновлюється, але для великих наборів даних може виявлятись обчислювально дорогим та мати повільну збіжність.
2. Стохастичний градієнтний спуск (англ. Stochastic Gradient Descent, SGD) використовує один навчальний приклад за раз для обчислення градієнта, має швидше оновлення та може уникати локальних мінімумів через зашумлені оновлення. Проте має більшу дисперсію функції втрат та менш стабільну збіжність.
3. Міні-пакетний градієнтний спуск (англ. Mini-Batch Gradient Descent) використовує невелику випадкову підмножину набору даних (міні-пакет) для обчислення градієнта. Потребує вибору розміру міні-пакету, зате поєднує в собі переваги пакетного та стохастичного градієнтного спуску, а саме ефективні обчислення та плавну збіжність.

Згорткові нейронні мережі (англ. Convolutional Neural Network, CNN) [11] призначені для обробки структурованих сіткових даних, таких як зображення.

Вони складаються зі згорткових шарів, які застосовують фільтри для виявлення локальних шаблонів, таких як краї та текстури, а потім об'єднують шари, які зменшують розмірність даних, зберігаючи важливі ознаки. Успішно показали себе в задачах розпізнавання і класифікації зображень.

Рекурентні нейронні мережі (англ. Recurrent Neural Network, RNN) призначені для послідовних даних, таких як часові ряди або текст, де порядок точок даних є важливим. Вони мають зв'язки, які формують спрямовані цикли, що дозволяє інформації зберігатися на різних часових етапах. Це робить їх придатними для таких завдань, як моделювання мови та розпізнавання мовлення. Однак такі нейронні мережі схильні до затухаючих градієнтів, які обмежують їхню здатність вивчати довгострокові залежності.

Генеративні змагальні мережі (англ. Generative Neural Network, GAN) [17] складаються з двох нейронних мереж, генератора і дискримінатора, які навчаються одночасно. Генератор створює фальшиві дані, а дискримінатор оцінює їхню справжність. Мета полягає в тому, щоб генератор створював дані, які неможливо відрізити від справжніх, що має застосування в синтезі зображень, доповненні даних тощо.

Навчання з передаванням (англ. transfer learning) [18] використовує попередньо навчені моделі на великих наборах даних для вирішення споріднених завдань з обмеженими даними. Шляхом точного налаштування попередньо навченої моделі на новому, меншому наборі даних, навчання з перенесенням знижує обчислювальні витрати і підвищує продуктивність, особливо в областях з обмеженими даними.

2.1 Методи комп’ютерного зору

Обробка зображень є важливим етапом комп’ютерного зору. Попередня нормалізація яскравості та контрастності вхідних даних робить їх більш

однорідними, що дозволяє моделі краще розпізнавати різні ознаки зображення. Розрізняють наступні методи, обидва представлені в бібліотеці OpenCV [2].

1. Вирівнювання гістограми (англ. histogram equalization) підвищує контрастність зображення шляхом перерозподілу значень інтенсивності світла. Процес коригує гістограму зображення, щоб розподілити найчастіші значення інтенсивності. Ця техніка особливо ефективна для покращення контрастності зображень з різними умовами освітлення, роблячи деталі більш помітними.
2. Адаптивне вирівнювання гістограми (англ. Adaptive Histogram Equalization, AHE) розширює концепцію вирівнювання гістограми, застосовуючи її до локалізованих областей, а не до всього зображення. Цей метод ефективніше підвищує контрастність у місцях, де умови освітлення значно відрізняються по всьому зображеню.

Зміна розмірів і форматування зображень має вирішальне значення для стандартизації розмірів зображень перед їх введенням до моделі. Нижче наведені деякі ключові методи, обидва представлені в бібліотеці OpenCV [2].

1. Білінійна інтерполяція [3] розглядає найближчі чотири пікселі до вхідних координат (квадрат 2×2), і обчислює значення вихідного пікселя на основі середньозваженого значення цих пікселів. На виході зображення більш згладжене, ніж початкове.
2. Бікубічна інтерполяція [3] розширяє білінійну інтерполяцію, враховуючи 16 найближчих пікселів (квадрат 4×4) для обчислення вихідного значення пікселя. На виході отримуємо трохи більш різке зображення, ніж початкове.

Фільтрація використовується для маніпулювання або покращення зображень шляхом зміни значень їхніх пікселів. Його застосовують для різних цілей, таких як зменшення шуму, підвищення різкості та виділення деталей [5]. Існує кілька поширених типів фільтрів, що використовуються для фільтрації зображень.

1. Гауссівський фільтр [6] використовується для розмиття зображень і зменшення шуму. Він застосовує функцію Гауса для зважування значень пікселів, створюючи згладжене зображення, ступінь розмиття якого залежить від стандартного відхилення σ . Застосовуючи гаусове ядро (рис. 2.5), фільтр надає центральним пікселям більшої ваги, ніж навколошнім областям, ефективно зменшуєчи шум і зберігаючи структуру зображення. Гаусові фільтри гарно видаляють випадкові, малопомітні патерни шуму на зображеннях, що робить їх життєво важливими у багатьох програмах обробки зображень (рис. 2.6).

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Рисунок 2.5 – Квадратні опорні гаусові ваги, де стандартне відхилення $\sigma=3$ і нормалізаційною константою $n=273^9$



Рисунок 2.6 – Застосування гаусівського фільтрування з різним розміром ядра до зображення¹⁰

⁹ Джерело зображення: https://miro.medium.com/v2/resize:fit:1100/format:webp/0*QJ2rEGwi4F2nEb8j

¹⁰ Джерело зображення: https://miro.medium.com/v2/resize:fit:828/format:webp/0*DMJRbrJdNdknIVSD

2. Медіанний фільтр [7] замінює значення кожного пікселя медіаною його самого та восьми навколошніх пікселів (рис.2.7), зменшуючи різницю між усіма пікселями та згладжуячи зображення. Він особливо ефективний для видалення шуму "сіль-перець" (представляє себе як рідкісні білі та чорні пікселі) зі збереженням країв, але для усунення гаусівського шуму така фільтрація є менш ефективною. Результат медіанної фільтрації зображено на рис. 2.8.

$(i-1, j-1)$	$(i-1, j)$	$(i-1, j+1)$
$(i, j-1)$	(i, j)	$(i, j+1)$
$(i+1, j-1)$	$(i+1, j)$	$(i+1, j+1)$

Рисунок 2.7 – Згортка розмірністю 9×9 ¹¹

Source Image:



Result after median filter:



Рисунок 2.8 – Результат застосування медіанного фільтру до вихідного зображення¹²

¹¹ Джерело зображення:

https://miro.medium.com/v2/resize:fit:1100/format:webp/1*iGp8sUqeazOdnJDXdGoJlg.png

¹² Джерело зображення:

https://miro.medium.com/v2/resize:fit:828/format:webp/1*AA-_nFP4eeZoUgSZcg4S_Q.png

Виявлення країв використовується для ідентифікації та локалізації різких розривів на зображенні, які зазвичай відповідають межам об'єктів. Це важливий крок в аналізі зображень і додатках комп'ютерного зору, таких як виявлення і розпізнавання об'єктів.

1. Канні-крайовий детектор (англ. Canny edge detector) передбачає зменшення шуму за допомогою гаусового фільтра, знаходження градієнтів інтенсивності, не максимальне придушення для розрідження країв та відстеження країв за допомогою гістерезису.
2. Фільтрація Собела [6] дозволяє знаходити край, його суть полягає в здатності обчислювати градієнт функції інтенсивності в кожній дискретній точці зображення (пікселі). Під час цього ядра виконують множення між значеннями пікселів і відповідними коефіцієнтами ядра. Отримані результати потім підсумовуються, отримуючи значення від фільтрованого пікселя. Після завершення згорток фільтр Собела обчислює величину градієнта, використовуючи теорему Піфагора. Обчислення цієї величини дає уявлення про швидкість зміни інтенсивності та наявність країв на зображенні. Цей процес охоплює дві окремі згортки (рис. 2.9): одну з горизонтальним ядром, яку часто називають G_x , і другу з вертикальним ядром, відому як G_y . Горизонтальне ядро Собела найкраще ідентифікує вертикальні краї, тоді як вертикальне ядро Собела вправно виділяє горизонтальні краї. Більше того, орієнтація цих країв визначається проти годинникової стрілки відносно напрямку максимального контрасту, що простягається від чорного до білого. Результат фільтрації Собела зображенено на рисунку 2.10.

$Gx = \begin{array}{ c c c } \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$	$Gy = \begin{array}{ c c c } \hline -1 & -2 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$
---	--

Рисунок 2.9 – Горизонтальне та вертикальне ядро (згортки) Собеля¹³



Рисунок 2.10 – Результат застосування фільтрації Собеля¹⁴

3. Лапласіанський фільтр [8] для виявлення країв використовує похідну другого порядку (див. рівняння 2.6) для виявлення областей, де інтенсивність швидко змінюється, таким чином виділяючи краї. Техніка згладжування Лапласа в основному використовується для виявлення країв зображення. Вона виділяє розриви на рівні сірого, базуючись на другій просторовій деривації зображення.

¹³ Джерело зображення: https://miro.medium.com/v2/resize:fit:828/format:webp/0*1sasFNgx_MQ5hyWd

¹⁴ Джерело зображення: https://miro.medium.com/v2/resize:fit:450/format:webp/0*tcZ2CckcVY_rInCc

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (2.6)$$

Лапласівський детектор країв використовує лише одне ядро. Для виявлення країв зображення це ядро визначає похідні 2-го порядку від рівнів інтенсивності зображення, використовуючи лише один прохід. Ми можемо використовувати "Kernel 2" для виявлення країв з діагоналями. Це дасть кращу апроксимацію (див. рис. 2.11 та 2.12). Крім того, метод Лапласа дає швидші обчислення, ніж інші.

0	-1	0
-1	4	-1
0	-1	0

Kernel 1

-1	-1	-1
-1	8	-1
-1	-1	-1

Kernel 2

Рисунок 2.11 – Ядра, що можуть бути використані для лапласівського фільтру¹⁵



Kernel 1

Kernel 2

Рисунок 2.12 – Застосування лапласівського фільтрування з різними ядрами¹⁶

¹⁵ Джерело зображення:
https://miro.medium.com/v2/resize:fit:640/format:webp/1*H5cP-t6IEDFUeFAn3YX-Fw.png

¹⁶ Джерело зображення:
https://miro.medium.com/v2/resize:fit:828/format:webp/1*o_was4oP7irQ06dNecz1KA.png

Морфологічні операції використовуються для обробки бінарних зображень та вилучення компонентів зображення, які є корисними для представлення та опису форми. Ці операції спираються на структуру зображення і застосовуються для очищення бінарних зображень, видалення шуму та відокремлення пов'язаних компонентів. Розглянемо дві основні морфологічні операції.

1. Операція ерозії (англ. erosion) [9] видаляє пікселі на межах об'єктів.

В результаті об'єкти зменшуються, а дрібні, не пов'язані між собою точки шуму усуваються (рис. 2.13). Ерозія використовує структуруючий елемент для дослідження форм, що містяться у вхідному зображення.

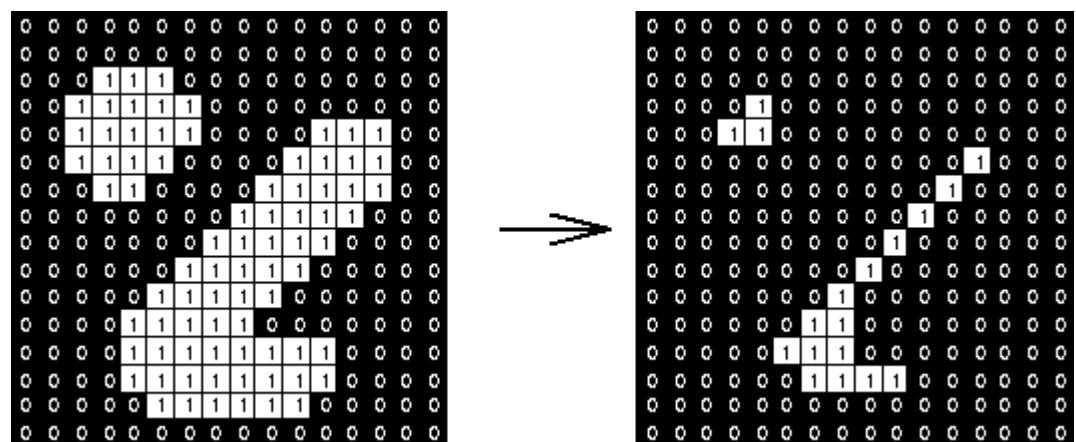


Рисунок 2.13 – Ефект еrozії, застосований з використанням структурного елемента розміром 3×3 пікселі¹⁷

2. Операція розширення (англ. dilation) [9] додає пікселі до меж об'єктів на зображенні, ефективно збільшуючи форми (рис. 2.14). Часто використовується для заповнення невеликих отворів і проміжків всередині об'єктів.

¹⁷ Джерело зображення: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/figs/erodebin.gif>

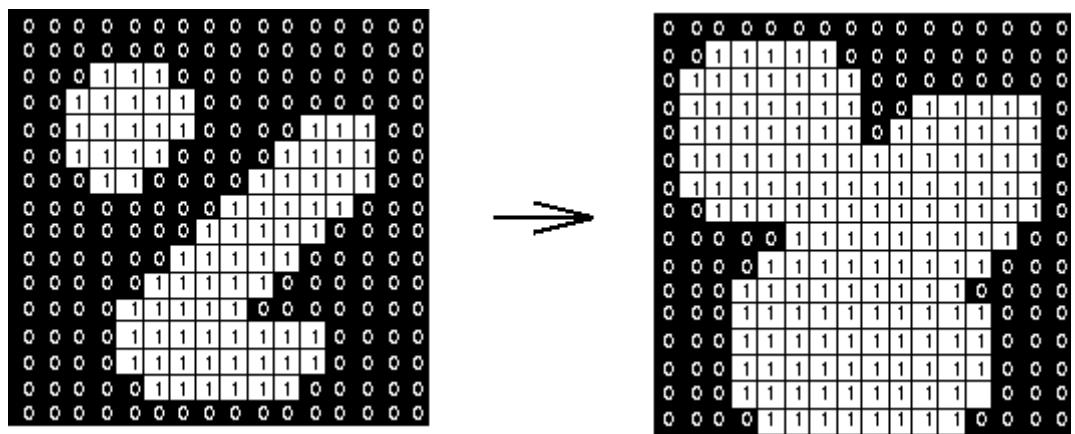


Рисунок 2.14 – Ефект ерозії, застосований з використанням структурного елемента розміром 3×3 пікселі¹⁸

Фільтрація, виявлення країв і морфологічні операції дозволяють покращувати та аналізувати цифрові зображення. Розуміючи і застосовуючи ці методи, можна витягувати значущу інформацію і виконувати широкий спектр завдань у комп'ютерному зорі.

Сегментація базується на певних критеріях, таких як колір, інтенсивність або текстура, що дозволяє проводити більш змістовний аналіз зображення. Існують різні підходи до сегментації, розглянемо найпоширеніші.

1. Порогова сегментація (англ. thresholding) [10] передбачає перетворення зображення у відтінках сірого у двійкове зображення, де пікселям присвоюється одне з двох значень (зазвичай чорне та біле) на основі порогового значення. Основна ідея полягає у виборі порогового значення та віднесення всіх пікселів зі значенням інтенсивності вище цього порогу до одного класу, а тих, що нижче - до іншого. Існує два підходи для вибору порогового значення: глобальне порогове значення - обирається для всього зображення, добре працює, коли фон і передній план мають різні рівні інтенсивності; адаптивне порогове значення - для різних областей зображення розраховуються різні порогові значення, корисно, коли

¹⁸ Джерело зображення: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/figs/diltbin.gif>

умови освітлення змінюються на різних ділянках зображення.

2. Методи сегментації на основі областей (англ. region-based segmentation) [5] працюють шляхом групування сусідніх пікселів зі схожими властивостями в одну область. Двома найпоширенішими методами є розростання областей (англ. region growing) та алгоритм вододілу (англ. watershed algorithm). Зростання областей починається з набору початкових точок і збільшує області шляхом додавання сусідніх пікселів зі схожими властивостями (наприклад, інтенсивністю або кольором). Зростання зупиняється, коли більше не залишається пікселів, що відповідають критеріям. Алгоритм вододілу розглядає зображення як поверхню, де значення пікселів представляють висоту. Він знаходить лінії, які формують межі водозбірних басейнів. Цей метод особливо ефективний для сегментації об'єктів, які дотикаються або перекриваються.
3. Сегментація зображень із застосуванням згорткових нейронних мереж (CNN) [11], що вивчають складні шаблони та особливості з великих наборів даних.

2.2 Підходи до розпізнавання об'єктів

Розпізнавання об'єктів ідентифікує та класифікує виявлені сегменти за заздалегідь визначеними категоріями. Одним з методів, який дозволяє розпізнавати об'єкти є ковзаюче вікно (англ. sliding window) фіксованого розміру, що систематично переміщується по зображення, скануючи об'єкти в різних місцях. Оскільки вимагає обробки декількох вікон у різних масштабах і положеннях, щоб покрити все зображення, то може бути доволі обчислювально дорогим і повільним.

У порівнянні з методом ковзаючого вікна, сучасні підходи для задачі

розділення об'єктів, такі як згорткові нейронні мережі (CNN) та трансформери, забезпечують високу точність і ефективність.

CNN [11] є основою багатьох систем розпізнавання об'єктів, основною ідеєю є використання згорткових шарів, які автоматично виявляють важливі особливості зображень, такі як краї, текстири та форми. Такі мережі мають шарову структуру (рис. 2.15), окрім згорткових шарів, ще розрізняють наступні: шари об'єднання (англ. pooling layers) зменшують розмірність даних, зберігаючи важливу інформацію; повнозв'язні шари (англ. fully connected layers) завершують мережу та виконують класифікацію на основі виділених особливостей. Знаходження мережею оптимальних фільтрів відбувається за допомогою зворотного розповсюдження та градієнтного спуску.

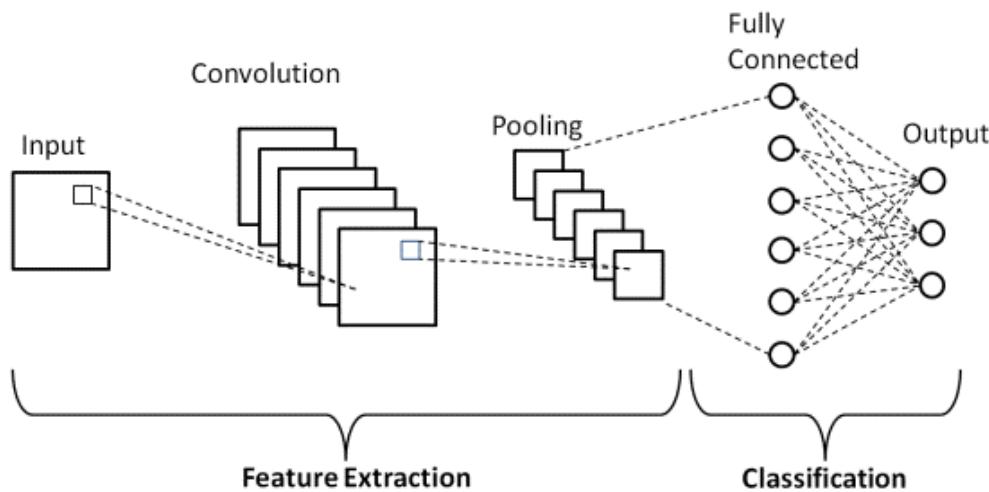


Рисунок 2.15 – Ілюстрація архітектури згорткової нейронної мережі¹⁹

Ці шари працюють разом, щоб навчитися складним представленням даних, що дозволяє моделі розпізнавати різноманітні об'єкти на зображеннях.

Згорткові шари працюють наступним чином[30]. Представимо вхідне зображення у вигляді кубоїда (рис. 2.16), що має довжину, та ширину (розмір зображення) і висоту (канал RGB, з червоним, зеленим і синім каналами).

¹⁹ Джерело зображення: <https://d14b9ctw0m6fid.cloudfront.net/ugblog/wp-content/uploads/2020/12/1-4.png>

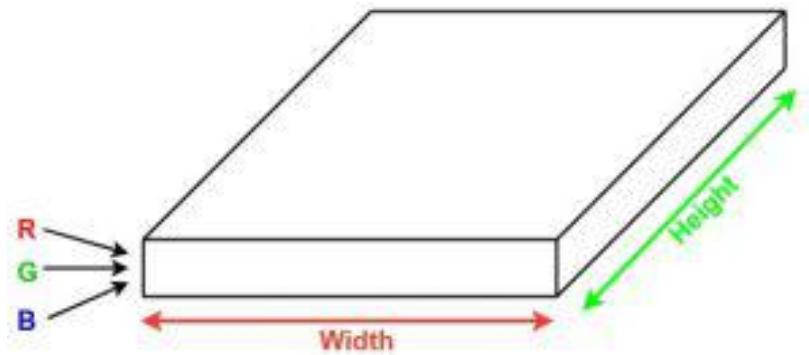


Рисунок 2.16 – Вхідне зображення як кубоїда²⁰

Візьмемо невеликий фрагмент цього зображення і запустимо на ньому невелику нейронну мережу, яка називається фільтром або ядром, з k виходами і представимо їх вертикально. Просунувши цю нейронну мережу по всьому зображення, на виході маємо інше зображення з іншими ширинou, висотою і глибиною (рис. 2.17).

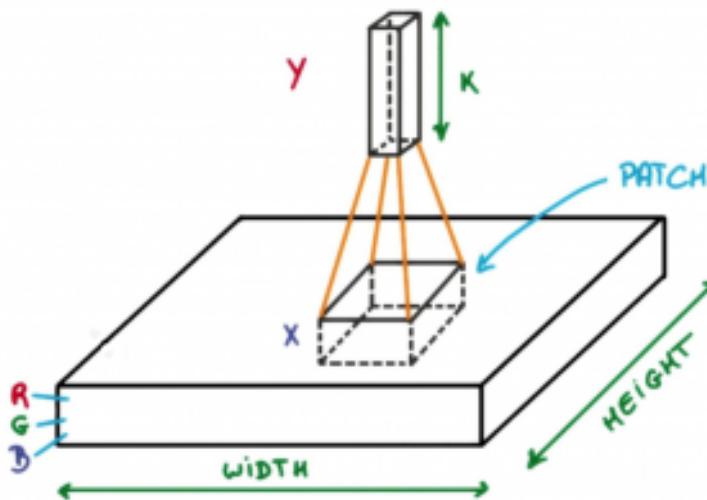


Рисунок 2.17 – Ілюстрація зображення з новою шириною, довжиною та глибиною після проходження по вихідному зображеню фільтром²¹

²⁰ Джерело зображення:

<https://media.geeksforgeeks.org/wp-content/uploads/20231218174321/cnn-2-300x133.jpg>

²¹ Джерело зображення:

<https://media.geeksforgeeks.org/wp-content/uploads/20231218174335/Screenshot-from-2017-08-15-13-55-59-300x217.png>

Замість каналів RGB тепер маємо більше каналів, але меншої ширини і висоти. Ця операція називається згортка. Завдяки цьому маленькому патчу ми маємо менше вагових коефіцієнтів. Згорткові шари складаються з набору фільтрів (або ядер), що навчаються, мають невелику ширину і висоту і таку ж глибину, як і вхідний об'єм (3, якщо вхідним шаром є вхідне зображення). Під час прямого проходу ми переміщуємо кожен фільтр по всьому вхідному об'єму крок за кроком, де кожен крок називається кроком, і обчислюємо точковий добуток між вагами ядра і патча від вхідного об'єму. Пересуваючи фільтри, ми отримаємо двовимірний вихід для кожного фільтра і складемо їх разом, в результаті чого отримуємо вихідний об'єм з глибиною, що дорівнює кількості фільтрів. Мережа вивчить всі фільтри.

1. Faster R-CNN (Region-based Convolutional Neural Networks) [28] У ній вхідне зображення пропускається через серію згорткових шарів для створення згорткової карти ознак (англ. convolutional feature map). Ця карта слугує основою як для генерації пропозицій областей, так і для класифікації об'єктів (рис. 2.18).

Мережа пропозицій областей (англ. Region Proposition Network, RPN) – це окрема згорткова мережа, яка одночасно прогнозує межі об'єктів в кожній позиції на карті об'єктів. Вона генерує пропозиції щодо області, ковзаючи невеликою мережею по згорнутій карті об'єктів. Ця невелика мережа повністю підключена до вікна розмірністю 3×3 на карті об'єктів. Кожне ковзне вікно зіставляється з об'єктом нижчого розміру, який потім подається на два споріднені повністю з'єднані шар регресії та шар класифікації.

Вихідні дані RPN, які включають набір пропозицій областей, подаються на рівень об'єднання RoI (Region of Interest), який витягує карту об'єктів фіксованого розміру для кожної пропозиції області зі спільної карти об'єктів. Карти об'єктів фіксованого розміру потім використовуються для класифікації та регресії граничного поля.

Додатково повнозв'язні шари обробляють карти об'єктів фіксованого розміру з шару об'єднання RoI. Мережа закінчується двома спорідненими вихідними шарами: один з них виробляє ймовірності softmax для класифікації, а інший уточнює координати граничної області.

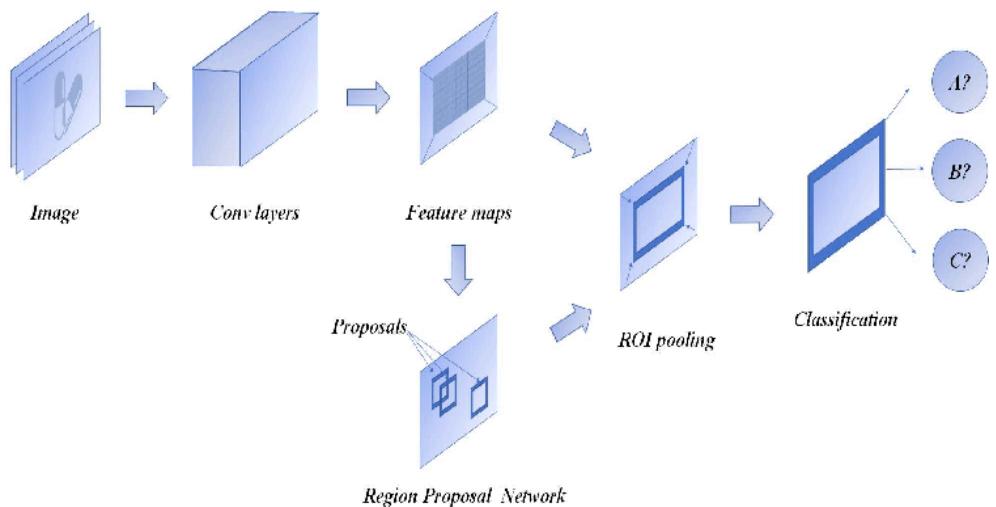


Рисунок 2.18 – Ілюстрація архітектури мережі Faster R-CNN²²

На відміну від своїх попередників, Faster R-CNN інтегрує генерацію пропозицій областей у саму мережу виявлення, усуваючи необхідність окремого етапу генерації пропозицій. Це об'єднання значно підвищує швидкість роботи моделі. Основною інновацією Faster R-CNN є Region Proposal Network (RPN), яка генерує високоякісні пропозиції областей за допомогою полегшеної мережі, що використовується спільно з мережею виявлення об'єктів. RPN використовує набір опорних блоків, званих якорями, у кожній позиції ковзного вікна. Ці якорі призначені для охоплення різних масштабів і співвідношень сторін, покращуючи виявлення об'єктів

²² Джерело зображення:

<https://www.researchgate.net/profile/Liang-Bi-Chen/publication/331980242/figure/fig9/AS:743367642202118@1554244065952/Flowchart-of-the-Faster-R-CNN-module.png>

різних розмірів і форм. Вся мережа (включаючи RPN і мережу виявлення) навчається наскрізним способом за допомогою зворотного розповсюдження, що забезпечує більш згуртовану та оптимізовану модель. Використовуючи ієрархічні карти ознак, утворені згортковими шарами, Faster R-CNN ефективно виявляє об'єкти різних масштабів [13].

2. YOLO (You Only Look Once) [27] використовує архітектуру єдиної нейронної мережі, яка прогнозує обмежувальні рамки та ймовірності класів безпосередньо з повних зображень за одну оцінку. Вхідне зображення розбивається на сітки комірок. Кожна комірка відповідає за прогнозування граничних областей та ймовірностей класів для об'єктів, розташованих у ній. YOLO прогнозує граничні області шляхом регресії відожної комірки сітки (рис. 2.19). Кожна гранична область представлена набором координат, а також оцінкою довіри до наявності об'єкта та ймовірністю класу. YOLO використовує згорткові шари для виділення ознак, що дозволяє йому захоплювати як локальну, так і глобальну інформацію з вхідного зображення.

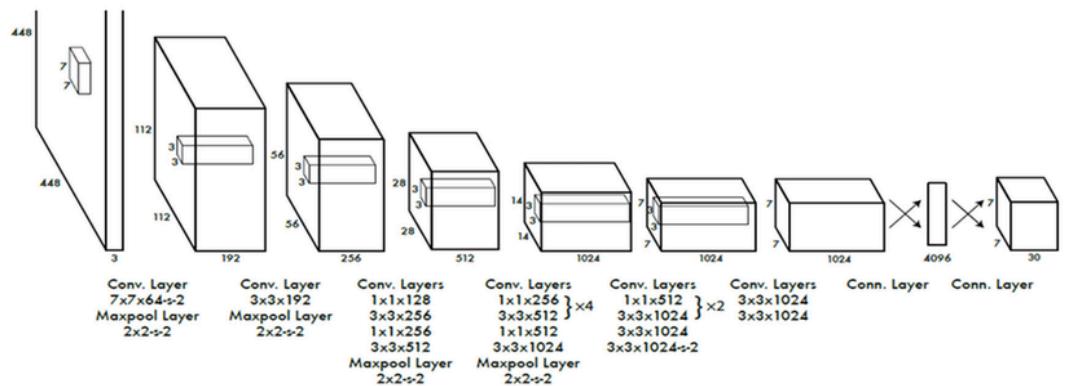


Рисунок 2.19 – Ілюстрація архітектури мережі YOLO²³

²³ Джерело зображення:

<https://www.researchgate.net/publication/329038564/figure/fig2/AS:694681084112900@1542636285619/YOLO-architecture-YOLO-architecture-is-inspired-by-GooLeNet-model-for-image.ppm>

Для усунення повторних виявлень YOLO застосовує не максимальне придушення для уточнення остаточного набору прогнозованих обмежувальних рамок.

YOLO відомий своїми можливостями виявлення в реальному часі, здатний обробляти зображення на вражаючих швидкостях без втрати точності. Ефективна конструкція YOLO забезпечує швидке і точне виявлення об'єктів на пристроях з обмеженими ресурсами, що робить його придатним для розгортання в різних практичних сценаріях.

Попередньо розроблені для обробки природної мови (NLP), моделі трансформери почали використовуватися і в комп'ютерному баченні. Вони пропонують потужну альтернативу традиційним CNN. Ці моделі використовують механізми самоуваги для зважування впливу різних вхідних елементів при створенні вихідних даних, що дозволяє їм фіксувати довгострокові залежності більш ефективно. Одними з представників моделей трансформерів, які активно застосовуються для задач комп'ютерного бачення є ViT та Swin Transformer.

1. ViT (Vision Transformer) [20] – замість згорток, зображення ділиться на невеликі батчі, які потім подаються у трансформер. Кожен батч, зазвичай розміром 16x16 пікселів, перетворюється на лінійний вектор і подається на вхід трансформера як послідовність токенів. Механізм самоуваги допомагає охоплювати глобальні залежності між усіма частинами зображення. Архітектура ViT (рис.2.20) складається з послідовності шарів, кожен з яких містить механізм багатоголової самоуваги (англ. multi-head self-attention) і позиційні вбудування (англ. position embeddings), які додаються до кожного патчу для збереження інформації про позицію. Крім того, ViT включає шари нормалізації (англ. layer normalization) та повнозв'язні шари для подальшої обробки даних. ViT показує конкурентоспроможні результати на багатьох задачах розпізнавання.

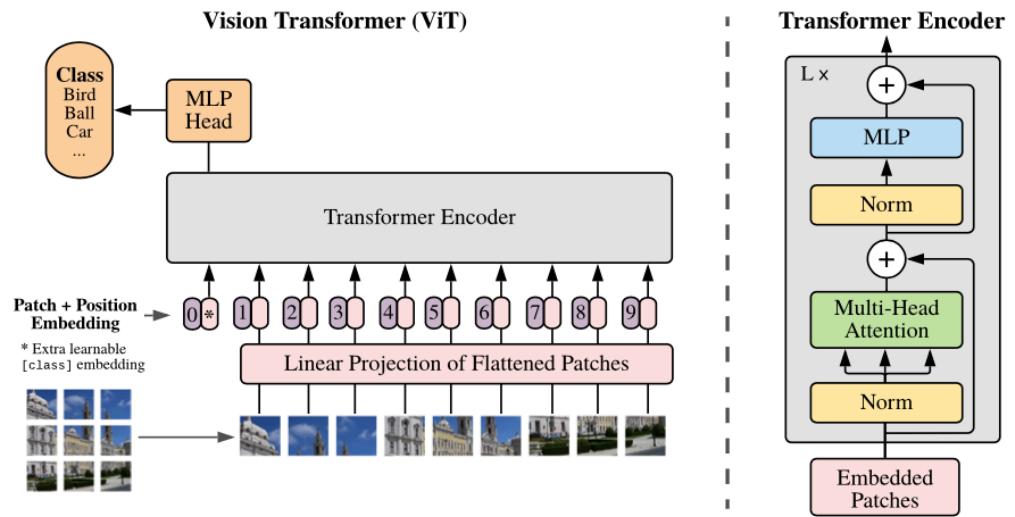


Рисунок 2.20 – Ілюстрація Архітектури трансформеру ViT²⁴

2. Swin Transformer – ієрархічна архітектура трансформера з ковзними вікнами, що дозволяє ефективно обробляти великі зображення. Вона розбиває зображення на патчі і обробляє їх у ковзних вікнах, забезпечуючи локальну обробку та знижуючи складність. Механізм дозволяє захоплювати глобальні залежності між частинами зображення. Архітектура включає самоувагу та багатошарові перцептрони (рис. 2.21) для обробки патчів на кожному рівні, поєднуючи переваги трансформерів і CNN. Модель показує високу ефективність у класифікації, сегментації та детекції об'єктів [20].

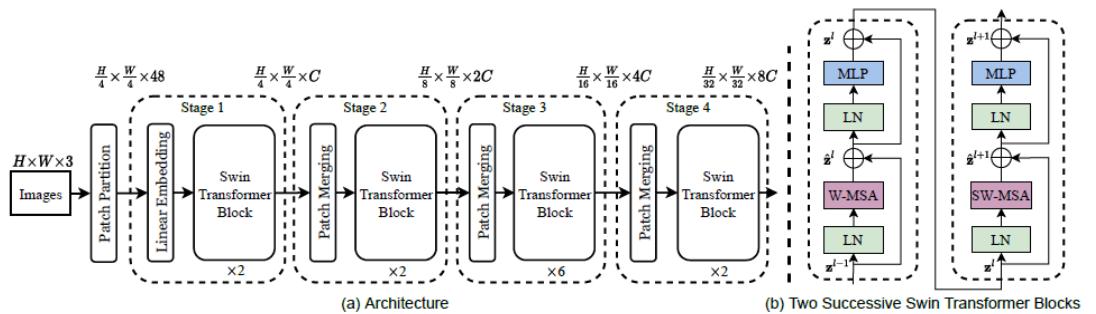


Рисунок 2.21 – Ілюстрація архітектури Swin Transformer²⁵

²⁴ Джерело зображення: <https://viso.ai/wp-content/uploads/2021/09/vision-transformer-vit.png>

²⁵ Джерело зображення: https://miro.medium.com/v2/resize:fit:1400/1*x50n-zyiU6TMHamEPM2HeA.png

Нейронна мережа YOLOv8 була обрана для вирішення задачі розпізнавання військових об'єктів через її ряд переваг. Висока швидкість обробки зображень в режимі реального часу, що є критично важливим для задач розпізнавання об'єктів на відео. На відміну від інших нейронних мереж, обробка зображення відбувається за один прохід, що суттєво знижує вимоги до обчислювальних ресурсів. Орієнтуючись на баланс між швидкістю та точністю, вибір YOLOv8 (найбільш сучасної версії архітектури YOLO) є доволі підходящим варіантом для задачі розпізнавання військових об'єктів, знятих дронами.

Важливо зазначити, що YOLOv8 постачається з великою кількістю зручних для розробника функцій, від простого у використанні CLI, який робить навчання моделі більш інтуїтивно зрозумілим, до добре структурованого пакета Python. Існує велика спільнота навколо YOLO, багато експертів з комп'ютерного зору знають про цю архітектуру, існує багато практичних прикладів і рішень різноманітних проблем, все це значно полегшує роботу з мережею.

До прикладу візьмемо оцінку архітектур виявлення об'єктів проводилася на внутрішньому наборі даних, що включає декілька класів таких як транспортні засоби та будівельні об'єкти зі спільною кількістю понад 7 тис. зразків зображень і понад 12 тис. обмежувальних рамок [26]. Найкращу точність та швидкістю показала YOLOv8 серед чотирьох оцінених архітектур, досягнувши показника mAP50 0,62 на тестовому наборі. Інші мережі досягли відповідних показників mAP50 на тестовому наборі даних: YOLOv5 – 0.58, EfficientDet – 0.47 і, Faster R-CNN – 0.41.

Лакмусовим папірцем для визначення точності вважається датасет Microsoft COCO (Common Objects in Context). На рис. 2.22 наведено дослідження точності різних версій архітектури YOLO авторами цієї нейронної мережі. Вони стверджують, що вона перевершує попередні випуски YOLO (а саме версії v7, v6 і v5) на вищезгаданому наборі даних.

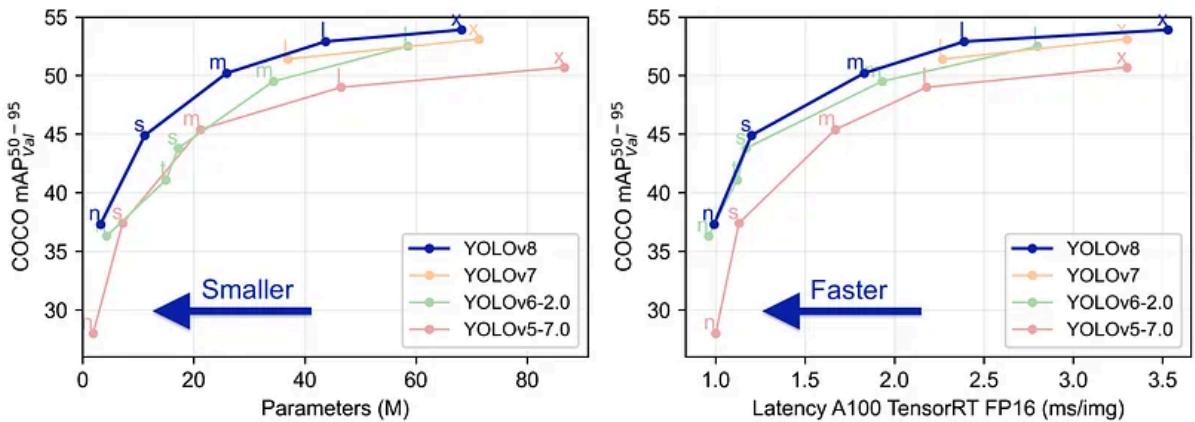


Рисунок 2.22 – Порівняння точності різних версій YOLO на датасеті COCO²⁶

Висновки до розділу 2

У другому розділі дипломної роботи розглянуті поняття машинного та глибокого навчання, які потрібні були для того, щоб розібратись в тому як саме працюють ці технології. Детально було розглянуто структуру нейронної мережі, нейрону алгоритм зворотного розповсюдження, функцію втрат та навчання з передаванням. Щодо комп’ютерного зору було досліджено основні методи та алгоритми обробки зображень, виділення ознак, сегментація зображень і насамкінець розпізнавання об’єктів. Розглянуто підходи до вирішення задачі комп’ютерного зору, а саме детально оглянути згорткові нейронні мережі як основний підхід та оглядово можливість застосування моделей трансформерів для цієї задачі. Подальший вибір впав на згорткову нейронну мережу архітектури YOLOv8 через її одні з найкращих показників для задач розпізнавання об’єктів, зручністю в роботі та широкою спільнотою.

²⁶ Джерело зображення:

https://miro.medium.com/v2/resize:fit:828/format:webp/1*6Sn93qyovio0qn3Y3q7Vgw.png

РОЗДІЛ З НАВЧАННЯ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

Починаючи процес навчання згорткової нейронної мережі для розпізнавання військових об'єктів, необхідно ретельно розглянути середовище розробки та доступний тренувальний набір даних. У сучасних умовах, коли штучний інтелект швидко розвивається, вибір правильних інструментів для розробки є важливим завданням, майже на рівні зі знаходженням потрібного набору даних. У цьому розділі детально розглянуто кожен аспект цього процесу, починаючи з вибору інструментів та завершуючи налаштуванням моделі та оцінкою результатів.

3.1 Середовище розробки

Для локальної розробки та тренування моделі важлива наявність графічного процесору (англ. Graphics Processing Unit, GPU) через його здатність виконувати паралельну обробку, що значно прискорює обчислення великих матриць і складних операцій, пов'язаних з навчанням CNN. Тому має значно більшу швидкість у порівнянні з центральним процесором (англ. Central Processing Unit, CPU). Платформа Google Colab здається кращим варіантом для навчання нейронної мережі, адже надає безкоштовний доступ до потужних обчислювальних ресурсів, включаючи GPU і TPU (Tensor Processing Unit). Інтеграція з Google Drive спрощує зберігання та управління даними, особливо великими навчальними наборами даних для тренування моделей. Google Colab також має вбудовану підтримку багатьох популярних бібліотек, таких як TensorFlow, PyTorch, Keras, що дозволяє легко розробляти та тестувати моделі.

3.2 Тренувальний набір даних

Однією з основних проблем, з якими стикаються дослідники в галузі глибокого навчання, є відсутність готових датасетів, особливо для специфічних задач, мій випадок не став виключенням. Набори даних, які я знаходив на просторі мережі, на ресурсах по глибокому навчанню (наприклад Roboflow), не відповідали моїм очікуванням, адже зачасту були наповнені зображеннями військової техніки з близьких ракурсів або матеріалами з відео ігрових симуляцій. Такі дані не відповідають вимогам, тому було прийнято рішення збирати власний датасет.

Для створення необхідного тренувального набору даних доводиться самостійно переглядати сотні фотографій та відео, задачу ускладнюють те, що підходять зображення військових об'єктів лише з певних ракурсів, зняті трошки або повністю згори, та масштаби об'єктів чи техніки. Відеоматеріали додатково мають бути розбитими на відеокадри, все це доволі часозатратний процес. Попередньо усі зображення мають бути приведені в один формат, в моєму випадку до розширень .jpg, .jpeg або .png. Дані зібрани виключно з відкритих інтернет ресурсів.

Для розмітки об'єктів на зображеннях обрано сервіс анотації даних для комп'ютерного бачення CVAT.ai [19] через його зручний інтерфейс та можливості гнучкої роботи з великими наборами даних. Не дивлячись на вищеперелічені переваги, все ж процес розмітки залишається трудомістким. Розібравшись з інтерфейсом та потрібним мені процесом, загалом розмітка всього набору даних зайняла близько тижня роботи. На зображеннях було виділено наступні два загальні класи військових об'єктів: броньована техніка (англ. armor vehicle) та легка техніка (англ. light vehicle). Розглядання більшої кількості класів, або розбиття поточних на більш конкретні збільшило об'єм роботи зі знаходженням відповідних зображень та їх розміткою, тому було прийнято рішення зупинитись саме на цих двох. Важливе зауваження, що

кількість зображень класу легкої техніки в декілька разів менше, ніж броньованої.

Одним із важливих кроків стало застосування технік препроцесингу, таких як приведення зображень до однієї розмірності. Це забезпечує послідовну та ефективну обробку, а також зменшує обчислювальне навантаження та використання пам'яті. Збереження співвідношення сторін може запобігти спотворенню зображень, зберігаючи риси, необхідні для ефективного навчання. Була обрана розмірність 416×416 зі збереженням співвідношення сторін оскільки вона забезпечує хороший баланс між роздільною здатністю і обчислювальною ефективністю. Ця роздільна здатність є достатньо високою, щоб зафіксувати основні деталі та ознаки, необхідні для розпізнавання військових об'єктів, і водночас керованою з точки зору обчислювальних ресурсів, необхідних для навчання та висновків. Процес ускладнюється і тим, що рамки об'єктів, які ми отримали на етапі розмітки зображень, також мають бути зменшенні в розмірності пропорційно до зображення на якому вони знаходяться (рис. 3.1). В кінці кінців було отримано наступний набір даних, що містить: 912 навчальних і 166 валідаційних зображень.



Рисунок 3.1 – Зображення від масштабованої обмежувальної рамки об'єкту

3.3 Процес навчання моделі

Обрана на попередніх етапах модель YOLOv8 має готовий функціонал, реалізований у бібліотеці ultralytics [29]. Завдяки ній процес навчання моделі можна точно налаштувати за допомогою різних гіперпараметрів, які є важливими для оптимізації продуктивності. Якщо гіперпараметри не встановлені заздалегідь, саме так трапилось у моєму випадку, вони можуть бути автоматично оптимізовані та встановлені.

Конфігурацію моделі змінено з початкових 80 класів до 2 класів відповідно до моєї задачі. Модель має 225 шарів із загальною кількістю параметрів 3011238, з яких 3011222 є градієнтами, що навчаються. Обчислювальна складність моделі вимірюється на рівні 8.2 GFLOP, що свідчить про баланс між продуктивністю та ефективністю. Таке налаштування гарантує, що модель добре оснащена для точного і ефективного розпізнавання різних типів військових об'єктів.

Під час навчання виконуються автоматичні перевірки змішаної точності (англ. Automatic Mixed Precision або AMP) для підвищення обчислювальної ефективності. Для покращення узагальнення моделі застосовано методи доповнення даних, такі як Blur, MedianBlur, ToGray та CLAHE. Налаштування оптимізатора, включаючи швидкість навчання та імпульс, визначаються автоматично, а в якості оптимізатора використовується AdamW (модифікація традиційного оптимізатору Adam шляхом включення спадання ваги безпосередньо в правило оновлення, що призводить до кращого узагальнення та ефективнішої збіжності). Він гарно підходить для глибокого навчання моделей, оскільки допомагає зменшити надмірне пристосування, караючи великі ваги, покращуючи таким чином продуктивність і стабільність моделі. Навчання було проведено з наступними кількостями епох – 100 та 300. Результати реєструються та візуалізуються за допомогою TensorBoard (рис. 3.2 та 3.3), що полегшує комплексний моніторинг та аналіз продуктивності моделі в часі.

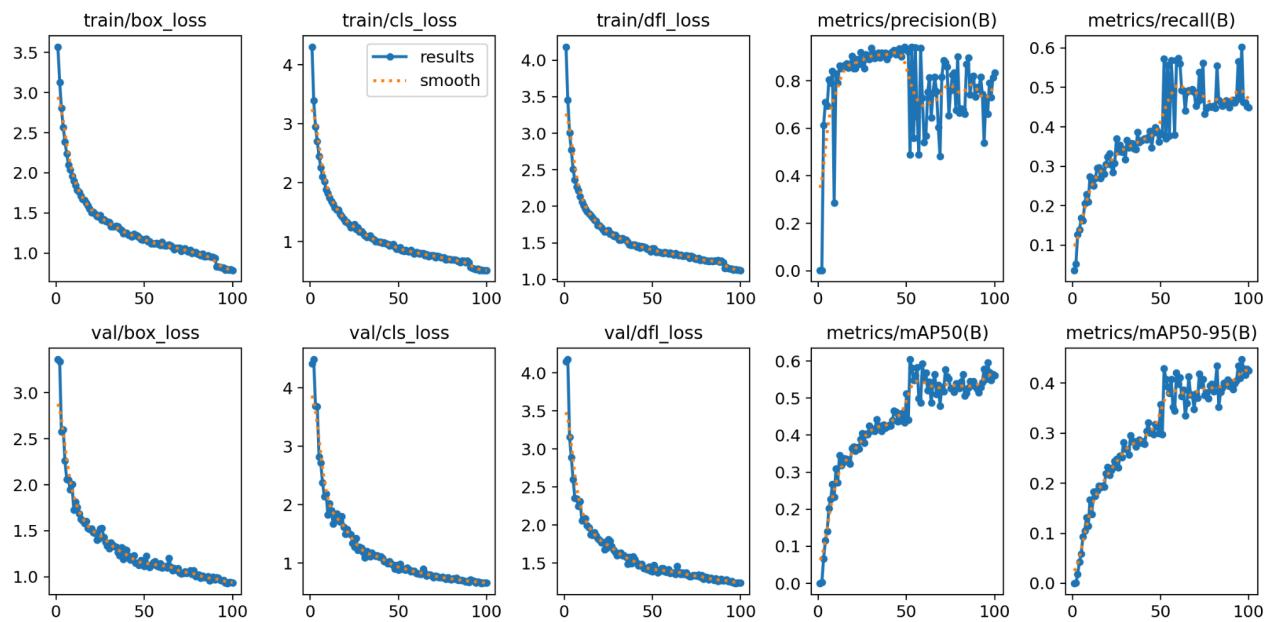


Рисунок 3.2 – Графічне зображення метрик тренування протягом 100 епох

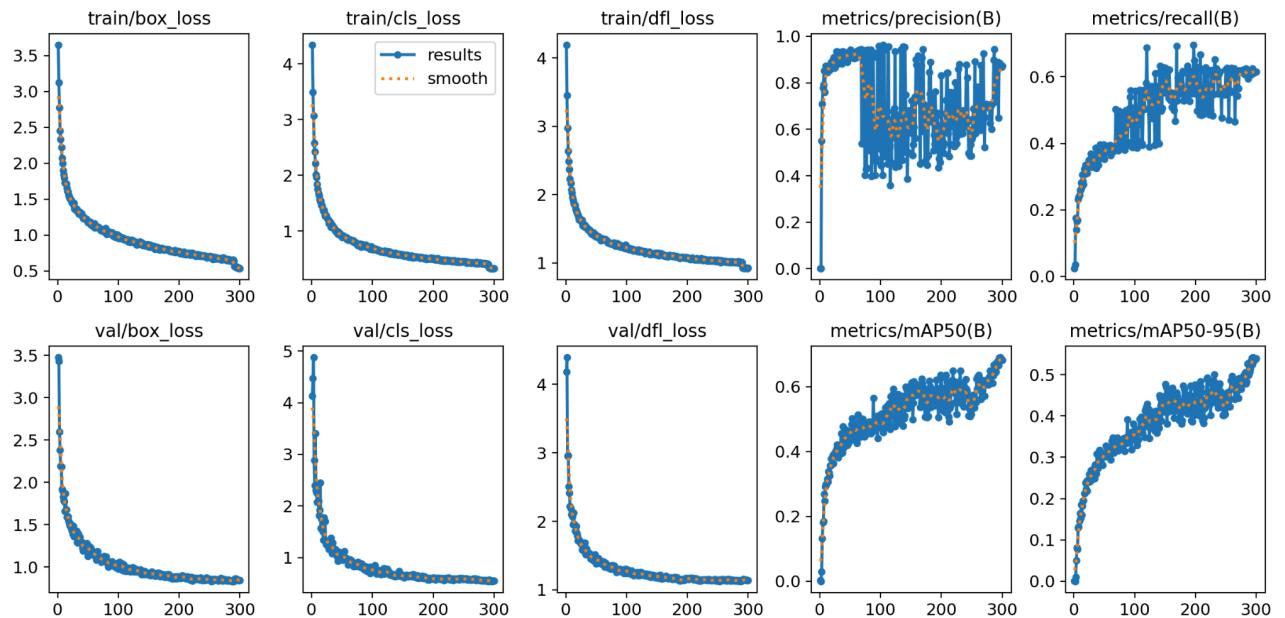


Рисунок 3.3 – Графічне зображення метрик тренування протягом 300 епох

На графіках представлено як змінювались значення втрат при навченні та валідації з різною кількістю епох, а саме такі метрики втрат як box loss, classification loss, distributional focal loss (DFL).

1. Втрати обмежувальної рамки (англ. box loss). Протягом 100 епох втрати в навчальному наборі зменшуються з приблизно 3.5 до приблизно 1.0, тоді як втрати в валідаційному зменшуються з 3.0 до трохи вище 1.0. Таке стійке зниження вказує на те, що модель ефективно навчається більш точно передбачати обмежувальні рамки в міру навчання. Розглядаючи навчання на 300 епохах, бачимо як втрати на навчальному та валідаційному наборі зменшуються до 0.5, що свідчить про подальше покращення прогнозів обмежувальних рамок. Узгодженість між втратами при навчанні та валідації свідчить про високі можливості узагальнення.
2. Втрати при класифікації (англ. classification loss). Протягом 100 епох втрата класифікації при навчанні зменшилася з 4.0 до приблизно 1.0, втрата валідаційної класифікації має аналогічну тенденцію. Таке зменшення означає, що модель стає більш вправною в класифікації об'єктів в межах прогнозованих обмежувальних рамок. При більш тривалому навчанні на 300 епохах втрата класифікації ще більше зменшується як для навчання так і для валідації, приблизно до 0.5, демонструючи підвищену точність класифікації об'єктів. Це постійне поліпшення підкреслює здатність моделі навчатися і зберігати важливі класифікаційні ознаки.
3. Розподілені фокальні втрати (англ. Distribution Focal Loss, DFL). На 100 епохах як під час навчання, так і під час валідації втрати DFL зменшуються з близько 4.0 до 1.0. Ця втрата вимірює зосередженість моделі на складних прикладах, і її зменшення свідчить про те, що модель ефективно навчається визначати пріоритети складних прикладів під час навчання. Під час навчання на 300 епохах втрата DFL ще більше зменшується, приблизно до 0.5, що свідчить про те, що модель стає все більш вправною у фокусуванні на складних прикладах, тим самим підвищуючи свою надійність і точність.

Також на графіках зображені показники оцінювання такі як влучність (англ. precision), повнота (англ. recall), mAP50 – вимірює середню середню точність при пороговому значенні 0.5 перетину над об'єднанням (англ. Intersection over Union або IoU), mAP50-95 (вимірює середню середню точність за пороговими значеннями IoU в діапазоні від 0.5 до 0.95). Останні два показники визначаються шляхом обчислення точності та запам'ятовування передбачуваних обмежувальних прямокутників порівняно з фактичними даними. mAP50-95 забезпечує більш комплексну оцінку, враховуючи ширший діапазон порогових значень IoU, фіксуючи як високі, так і низькі перекриття між прогнозованими та істинними обмежувальними прямокутниками. Розглянемо як змінювались вище перелічені показники оцінювання обмежувальної рамки для навчання з різною кількістю епох.

1. Влучність на 100 епохах починається з коливань, але загалом зростає до 0.8, що говорить про здатність моделі доволі точно ідентифікувати позитивні випадки серед прогнозованих обмежувальних рамок. Під час тренування на 300 епохах точність стабілізується на рівні 0.9, що свідчить про високу довіру до прогнозів моделі. Зменшення коливань свідчить про покращення узгодженості та надійності виявлення об'єктів.
2. Повнота на 100 епохах постійно зростає від 0.1 до 0.5, що свідчить про покращення здатності моделі ідентифікувати всі релевантні об'єкти на зображеннях. Повнота стабілізується на позначці 0.6 під час навчання на 300 епохах, що свідчить про подальше покращення здатності моделі виявляти істинно позитивні стани. Поступове збільшення показника свідчить про зростання ефективності моделі у розпізнаванні всіх станів об'єктів.
3. mAP50 на 100 епохах покращився з 0.0 до приблизно 0.5, що демонструє значне покращення загальної ефективності виявлення при пороговому значенні 50% перетину над об'єктом (IoU). Тоді як на 300 епохах, показник збільшується приблизно до 0.6, що свідчить

про подальше покращення точності виявлення. Постійне зростання цього показника свідчить про покращення здатності моделі прогнозувати обмежувальні рамки, що точно відповідають дійсності.

4. mAP50-95 на 100 епохах починається від 0.0 і зростає приблизно до 0.4, що відображає зростаючу точність моделі для різних порогових значень ВНП. Тоді як на 300 епохах mAP50-95 покращується приблизно до 0.5, тобто модель зберігає високу точність навіть при більш суворих порогових значеннях IoU. Покращення цієї метрики говорить про надійність та універсальність моделі в різних сценаріях виявлення.

Метрики навчання та валідації демонструють, що модель YOLOv8 демонструє значне покращення продуктивності як за 100, так і за 300 епох. Послідовне зменшення втрат і постійне зростання показників точності, повноти і mAP вказують на те, що модель ефективно навчається і узагальнює навчальні дані. Розширення навчання до 300 епох призводить до покращення цих значень. Ці результати свідчать про те, що тривале навчання дозволяє моделі вдосконалювати свої прогнози і покращувати здатність точно виявляти і класифікувати військові об'єкти. Таким чином, модель YOLOv8, навчена на моєму наборі даних з військовими об'єктами, демонструє багатообіцяючі результати, а тривале навчання дає суттєве покращення точності та надійності виявлення.

Розглянемо криві F1-Confidence та Precision-Recall та їх графіки.

1. Крива F1-Confidence відображає оцінку F1 для різних рівнів довірчого інтервалу. Показник F1 є середнім гармонійним влучності та повноти, що забезпечує єдину метрику, яка дає узагальнене збалансоване значення, розраховується за формулою 3.1

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (3.1)$$

Показник F1 для класу “armor-vehicle” становлять близько 0.8 та 0.9 на 100 та 300 епохах навчання в широкому діапазоні довірчих інтервалів, що вказує на високу та стабільну продуктивність. Тоді як для класу “light-vehicle” максимум лежить близько 0.4 та 0.6 на навчаннях з відповідною кількістю епох з помітними коливаннями, що свідчить про непостійну ефективність виявлення. Комбінована оцінка F1 для обох класів досягає значення 0.61 при довірчій ймовірності 0.098 для 100 епох (рис. 3.4) та 0.69 при довірчій ймовірності 0.283 для 300 епох (рис. 3.5), на що значною мірою впливає висока ефективність класу “armor-vehicle”.

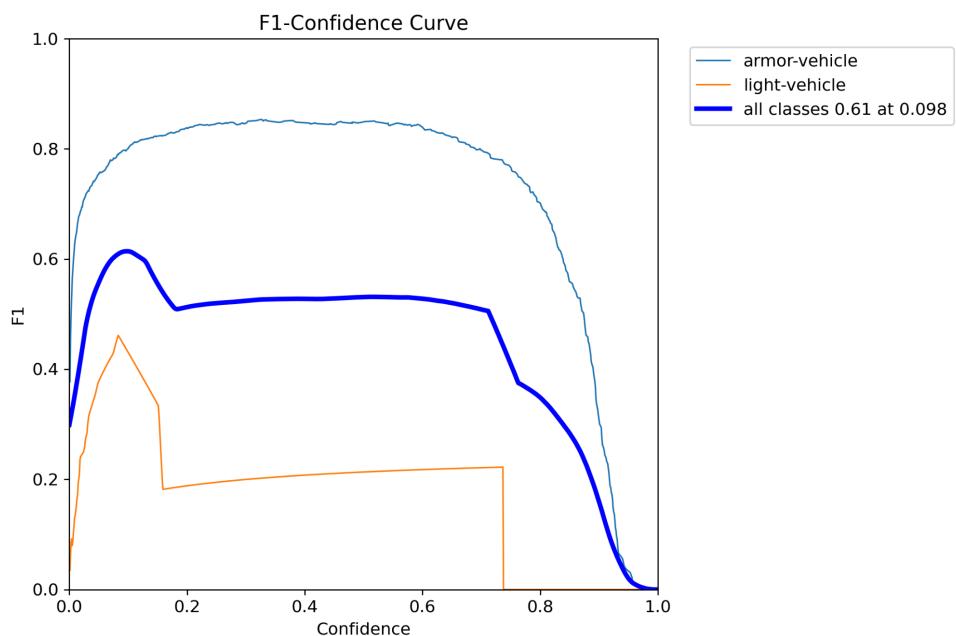


Рисунок 3.4 – Графічне зображення зміни показника F1 відносно впевненості протягом 100 епох

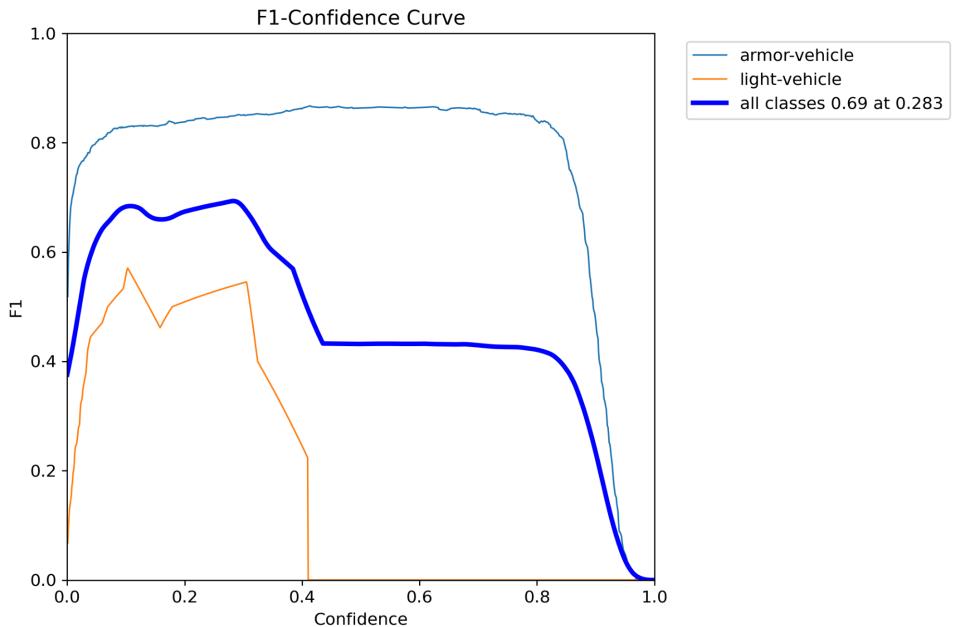


Рисунок 3.5 – Графічне зображення зміни показника F1 відносно впевненості протягом 300 епох

2. Крива Precision-Recall показує залежність влучності від повноти для різних довірчих інтервалів. Як правило, збільшення повноти призводить до зменшення влучності, і навпаки. Площа під кривою є єдиною метрикою, що підсумовує ефективність моделі: більша площа вказує на кращу ефективність.

Влучність вимірює частку істинно позитивних прогнозів від усіх позитивних прогнозів, тоді як повнота вимірює частку істинно позитивних прогнозів від усіх фактичних позитивних прогнозів, зроблених моделлю, обчислюються за формулами 3.2 та 3.3 відповідно.

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (3.2)$$

$$recall = \frac{true\ positives}{true\ positives + false\ negative} \quad (3.3)$$

Для класу “armor-vehicle” крива підтримує високі значення близько 0.9, із середньою точністю (англ. Average Precision або AP) 0.897 та 0.912 для навчання на 100 та 300 епохах відповідно, що відображає мінімальну кількість хибних спрацювань і негативних результатів.

Для класу “light-vehicle” крива показує різке падіння значень із середньою точністю 0.294 та 0.464 для навчання на 100 та 300 епохах відповідно, що свідчить про низьку точність виявлення. Середня точність (mAP@0.5) 0.595 та 0.688 (рис. 3.6 та 3.7), що свідчить про помірну загальну ефективність.

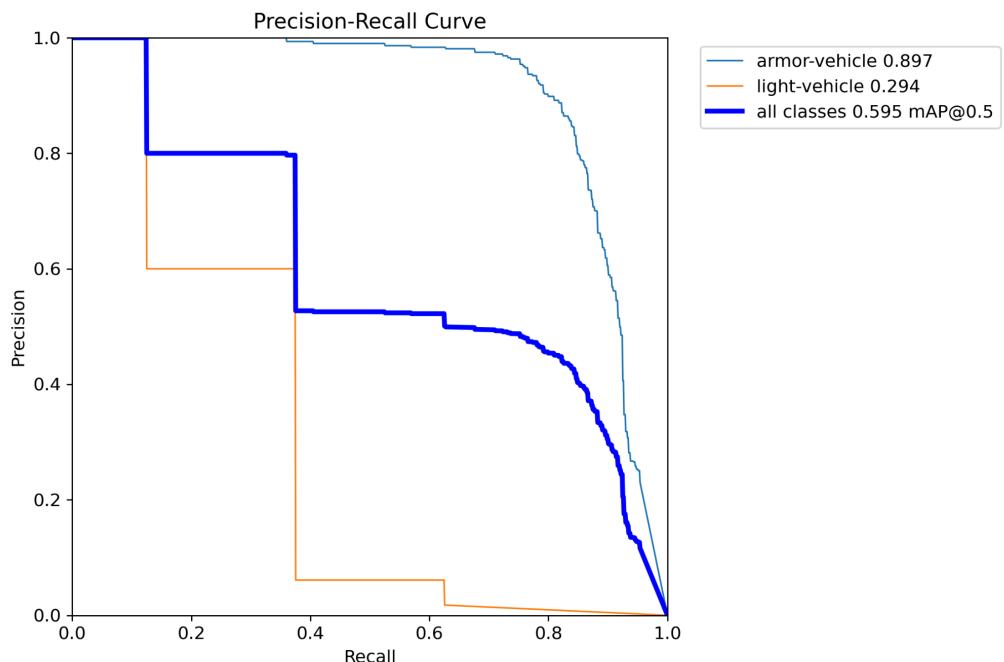


Рисунок 3.6 – Графічне зображення Precision-Recall кривої протягом 100 епох

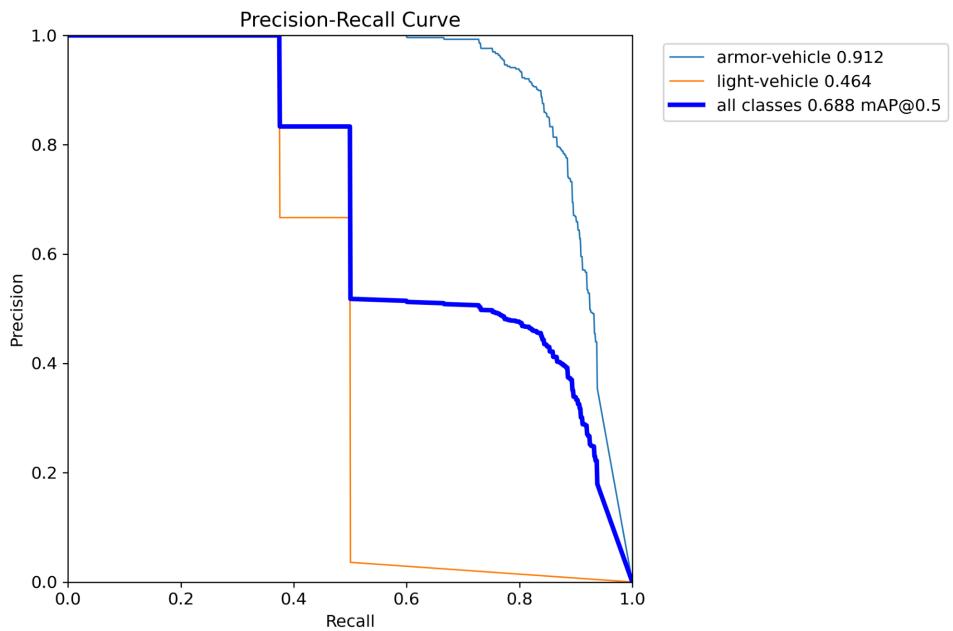


Рисунок 3.7 – Графічне зображення Precision-Recall кривої протягом 300 епох

Графіки вказують на явний дисбаланс класів, показники для різних класів мають значну розбіжність, причому високі показники одного класу перекривають і тим самим маскують слабші результати іншого, це вказує на необхідність проведення оцінок для кожного класу. Стабільно вищі показники для класу “armor-vehicle” контрастують з нестабільністю для класу “light-vehicle”, що вказує на необхідність збільшення кількості зображень для другого класу. Так як збільшення кількості епох навчання принесло корелює з покращенням результатів моделі, то в наступному розділі з результатами експерименту фігуруватиме саме модель, навчена на 300 епохах.

Висновки до розділу 3

У третьому розділі дипломної роботи обґрунтований описані процеси збору та розмітки тренувального і валідаційного наборів даних, проблеми

пов'язані з цими процесами. Детально пояснені графіки результатів навчання нейронної мережі YOLOv8 на 100 та 300 епохах, зі збільшенням кількості епох прослідковується покращення ефективності розпізнавання об'єктів обох класів. Розпізнавання класу “armor-vehicle” має значно вищі показники, ніж класу “light-vehicle”, що є наслідком меншої кількості зображень другого класу в тренувальному та валідаціоному наборах даних.

РОЗДІЛ 4 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТУ

Після успішного навчання моделі виявлення об'єктів наступним кроком є дослідження її практичного застосування та оцінка її ефективності в реальних умовах. Далі у цьому розділі буде представлено результати експериментів із застосуванням навченої моделі для виявлення військових об'єктів, а також можливості її розгортання на дронах для сканування території з метою виявлення військових об'єктів.

4.1 Розпізнавання об'єктів на фото

Під час свого навчання модель YOLOv8 оцінювала свою ефективність завдяки валідаційному набору, що містив зображення з різними військовими об'єктами та їх різною кількістю. Модель успішно ідентифікувала та позначила ці об'єкти класу “armor-vehicle” з високою точністю, трохи гірше вона розрізняла об'єкти класу “light-vehicle”. Нижче наведено приклад відпрацювання моделі на зображеннях з валідаційного набору, що демонструє виявлені військові об'єкти з числом поряд, що являє собою ймовірність того, що об'єкт обраного класу знаходиться в обмежувальній рамці.



Рисунок 4.1 – Результати розпізнавання об'єктів на валідаційному наборі даних (перші 16 зображень)

Аналіз результатів розпізнавання об'єктів на валідаційному наборі даних навченої моделі YOLOv8 надає важливі висновки щодо її ефективності. Особливу увагу слід звернути на значення впевненості моделі у визначені техніки, яка перебуває в обмежувальних рамках (див. рис. 4.1 та 4.2). За результатами, можна зробити висновок, що модель успішно навчилась визначати клас "armor-vehicle" з високою впевненістю, яка коливається від 0.7 до 1.

Однак для класу "light-vehicle" ситуація виявилась менш задовільною, що можна пояснити меншою кількістю зображень цього класу в тренувальному датасеті. Впевненість моделі на одному зі знімків навіть не досягає рівня 0.5, що свідчить про нестабільність розпізнавання об'єктів даного класу. Такі

результати вказують на необхідність подальшої роботи з покращенням роботи моделі для класів з меншим обсягом навчальних даних.



Рисунок 4.2 – Результати розпізнавання об’єктів на валідаційному наборі даних (другі 16 зображень)

Далі була спроба застосувати модель до зображень, які не входили до навчальних або валідаційних наборів даних (див. рис. 4.3 – 4.7). Цей крок був вирішальним для оцінки здатності моделі до узагальнення та її продуктивності

на невидимих даних. Результати з визначенням класу “armor-vehicle” наведені нижче відзначаються доволі гарною точністю. Чого не можна сказати відносно розпізнавання іншого класу “light-vehicle”, з розпізнаванням якого є проблеми, хоча все ж деякі об’єкти помічаються коректно цим класом (рис. 4.7).



Рисунок 4.3 – Результат розпізнавання об’єкту класу “armor-vehicle”²⁷

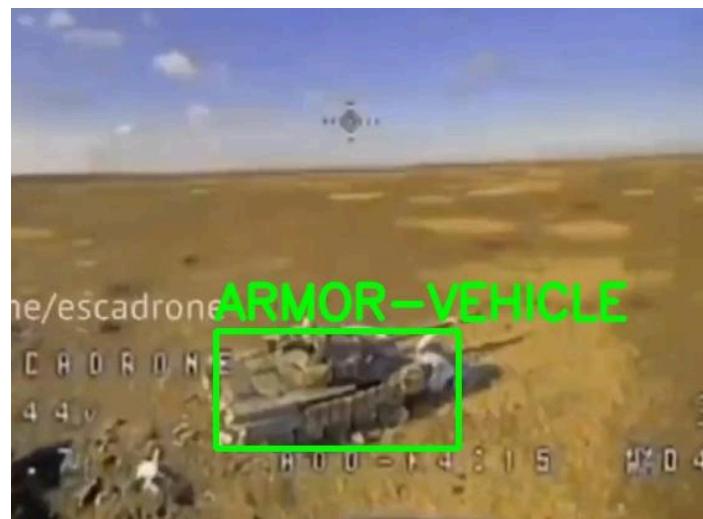


Рисунок 4.4 – Результат розпізнавання об’єкту класу “armor-vehicle”²⁸

²⁷ Джерело вихідного зображення:

https://media.cnn.com/api/v1/images/stellar/prod/231212211052-ukraine-drone-footage-vpx.jpg?c=16x9&q=w_1280,c_fill

²⁸ Джерело вихідного зображення:

https://media.licdn.com/dms/image/sync/D4E27AQF23fVPPu5Xnw/article-share-shrink_800/0/1711825645324?e=2147483647&v=beta&t=KaNNgeCdMoR9IQJSWtx3_eCnPY0N-wmL3f-_6Pgc8ys



Рисунок 4.5 – Результат розпізнавання об’єкту класу “armor-vehicle” в міській місцевості²⁹

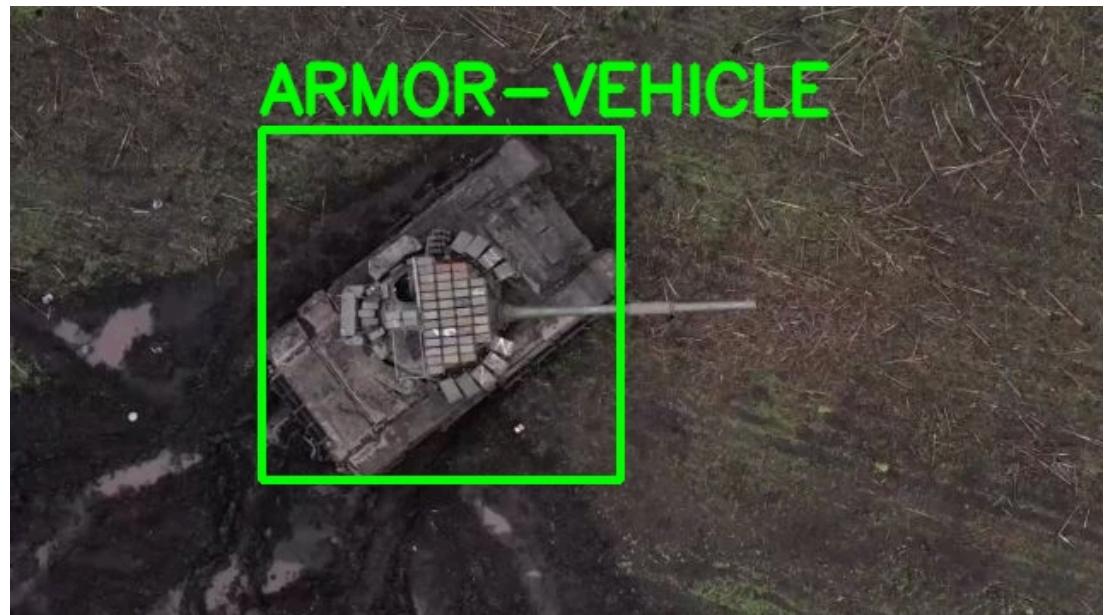


Рисунок 4.6 – Результат розпізнавання об’єкту класу “armor-vehicle” знятого згори за допомогою дрона³⁰

²⁹ Джерело вихідного зображення:

https://e3.365dm.com/22/03/1600x900/skynews-ukraine-russia_5701408.png?20220310145805

³⁰ Джерело вихідного зображення:

<https://bulgarianmilitary.com/wp-content/uploads/2023/05/Russia-is-already-putting-ERAs-on-the-tanks-anti-drone-grids.jpg>



Рисунок 4.7 – Результат розпізнавання об’єктів класів “armor-vehicle” та “light-vehicle” знятих згори за допомогою дрона³¹

4.2 Розпізнавання об’єктів на відео

Після успішного застосування натренованої моделі до статичних зображень, було вирішено спробувати використати її для обробки відео. Це дозволило нам оцінити роботу моделі в динамічних і безперервних сценаріях. Нижче наведено кілька кадрів з обробленого відео, що демонструють можливості моделі.

³¹ Джерело вихідного зображення:

https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcR_iZd9M5fx2iDBx9DIXs2-xj0iaorh7NmA9LS689EpuheCcfO



Рисунок 4.8 – Результат розпізнавання об’єкту класу “armor-vehicle” на відео

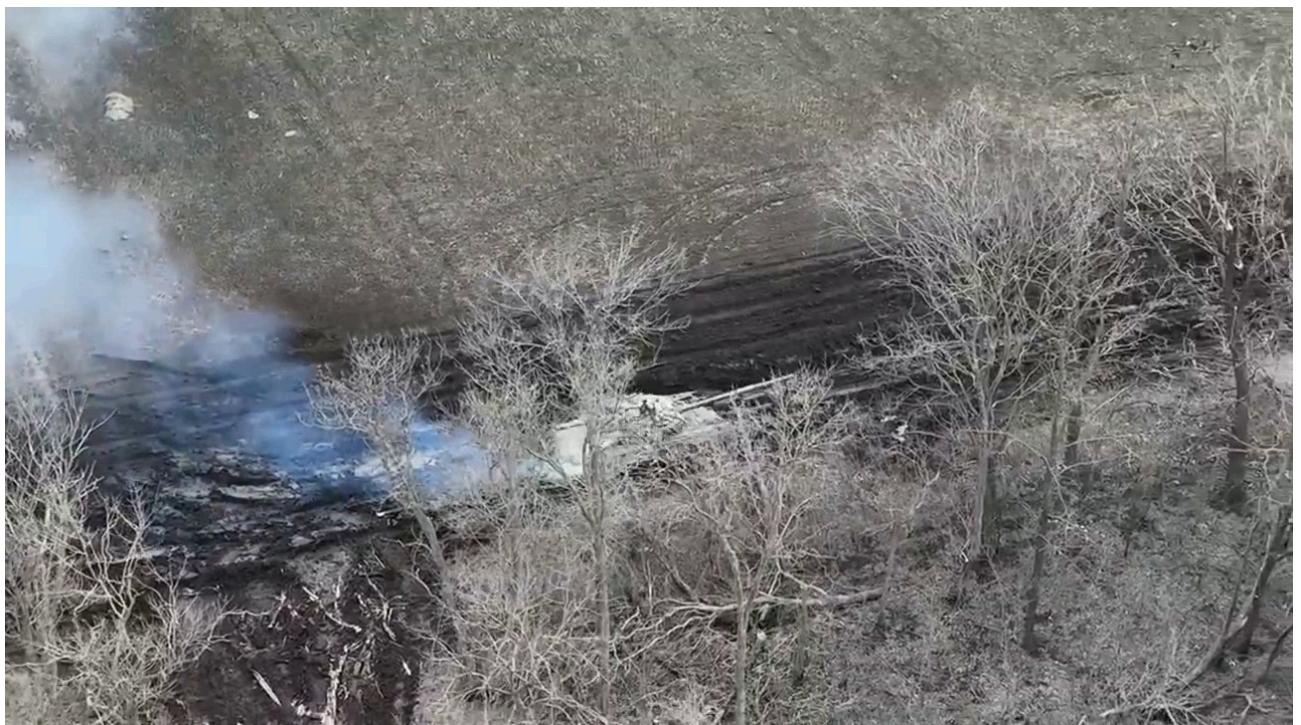


Рисунок 4.9 – Результат розпізнавання об’єкту класу “armor-vehicle” при наявності перешкоди на відео

Результати були успішними, майже протягом всього відео, модель успішно ідентифікували об'єкт, правильно визначаючи його обмежувальну рамку. Однак присутній короткий момент, коли силует танка був затулений деревом, що призвело до тимчасової втрати виявлення. Незважаючи на це, модель швидко відстежила об'єкт, в даному випадку танк, як тільки він знову з'явився з-за перешкоди.

4.3 Перспектива розгортання моделі на дроні

Незважаючи на наявність навченої моделі, здатної виявляти об'єкти на фотографіях і відео, що вже є досягненням, справжній потенціал цієї технології полягає у її військовому застосуванні. Просте виявлення на статичних зображеннях і відео були лише початковими кроками в тестуванні та перевірці моделі. Щоб повністю використати цю можливість у військовому контексті, розглянемо більш складні реалізації.

Одним з підходів було б розгортання моделі на безпілотниках, оснащених бортовими обчислювальними ресурсами, що дозволяє обробляти відеопотік безпосередньо в реальному часі. Це б зробило процес розвідки більш автономним, запрограмований на сканування території дрон міг би сам помічати об'єкти, і звітувати про це умовному серверу. Також у такому варіанті відкривається можливість автономних бойових дронів, які б при помічанні цілі автоматично атакували б її. Проте обчислення відеофрагментів у реальному часі доволі затратний процес з оглядом на обмежені обчислювальні ресурси дрону, який має бути невеликим.

Іншим підходом до реалізації є дрон, здатний передавати відеопотік на наземний сервер, як це вже і влаштовано, коли пілот дрону має відео у себе на пристрої і т.п. На серверній стороні можна виконати більш інтенсивну обчислювальну обробку, а головне що обмеження потужностей в такому

варіанті відсутнє. Такий підхід дозволив би ефективно сканувати територію, можливо навіть цілий зграї дронів, що робили б це синхронізовано, і миттєво виявляти військові об'єкти, надаючи цінну розвідувальну інформацію.

Висновки до розділу 4

У четвертому розділі дипломної роботи були продемонстровані результати розпізнавання об'єктів на зображеннях та відео. У цілому результати можна вважати успішними, хоча б через результати розпізнавання класу “armor-vehicle”, яке має велику кількість успішні практичних результатів. Також розглянута можливість розгортання навченої моделі у парі з дронами для автоматизації процесів збору інформації та бойових завдань.

РОЗДІЛ 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ

У даному розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на розпізнаванні військових об'єктів. У дослідженні розглянуто різні підходи до реалізації для виборі найбільш оптимальної, враховуючи економічні аспекти та сумісність з майбутнім продуктом. Для цього застосувався апарат функціонально-вартісного аналізу (ФВА).

ФВА передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою зниження витрат за рахунок ефективніших варіантів виробництва та оптимального співвідношення між вартістю для споживача та витратами на виробництво. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм ФВА включає в себе визначення послідовності етапів розробки продукту, розрахунок річних витрат та кількості робочих годин, ідентифікацію джерел витрат та кінцевий розрахунок вартості програмного продукту.

5.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту.

Технічні вимоги до програмного продукту:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- швидкість обробки даних;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

5.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні: F_1 – вибір мови програмування, F_2 – вибір методу реалізації алгоритмів, F_3 – вибір середовища розробки. Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1 :

- a) C++;
- б) Python.

Функція F_2 :

- а) Застосування готових бібліотек;
- б) Написання алгоритмів власноруч.

Функція F_3 :

- а) Google Colab;
- б) PyCharm IDE.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 5.1).

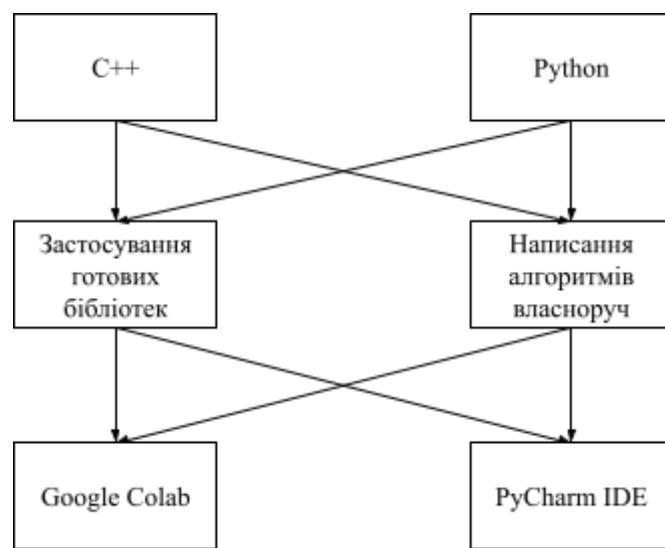


Рисунок 5.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в табл. 5.1.

Таблиця 5.1 – Позитивно-негативна матриця

<i>Функції</i>	<i>Варіанти реалізації</i>	<i>Переваги</i>	<i>Недоліки</i>
F_1	A	Компільована мова програмування, тому має вищу швидкість виконання коду.	Складніша у використанні.
	Б	Мова проста у освоєнні та використанні, має багато бібліотек для машинного навчання та велику спільноту.	Інтерпретована мова програмування, тому має меншу нижчу швидкість виконання коду.

Закінчення табл. 5.1

<i>Функції</i>	<i>Варіанти реалізації</i>	<i>Переваги</i>	<i>Недоліки</i>
F_2	A	Готові бібліотеки розроблені спеціалістами є надійним рішенням, яке має велику спільноту. Зазвичай є швидшими за самописні реалізації.	Обмежена гнучкість.
	Б	Можливість гнучкого написання алгоритму під задану задачу.	Потребує значних часових ресурсів, високого рівня компетенції. Ризик помилки.
F_3	A	Можливість залучення до навчання хмарних обчислювальних потужностей, інтеграція з хмарним сховищем для зберігання датасету. Зручність використання Jupyter Notebook завдяки його інтерактивності та зручним відображенням зображень/графіків.	Складність керування великими проектами, відлагодження коду.
	Б	Наявність інтеграції з системою контролю версій Git та базами даних. Можливість відлагодження коду.	Потребує обчислювальних ресурсів, менш інтерактивна.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1 :

Перевагу даємо простоті та доступності мови Python. Для спрощення роботи по написанню коду варіант А має бути відкинутий.

Функція F_2 :

Існуючі рішення для задач машинного навчання є більш надійними, адже вони написані та відтестовані спеціалістами в області і активно підтримуються. Через це варіант Б має бути відкинутий.

Функція F_3 :

Вибір середовища розробки не є таким важливим. Та скоріше за все будуть використані обидві платформи через їх переваги.

Таким чином, будемо розглядати такий варіант реалізації ПП:

$$F_1 \delta - F_2 a - F_3 a,$$

$$F_1 \delta - F_2 a - F_3 \delta.$$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- $X1$ – швидкодія мови програмування;
- $X2$ – об’єм пам’яті, необхідний для обчислень та збереження даних;
- $X3$ – час навчання нейронної мережі;
- $X4$ – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 5.2.

Таблиця 5.2 – Основні параметри програмного продукту

Назва параметру	Умовні позначення	Одиниці виміру	Значення параметру		
			Гірші	Середні	Кращі
Швидкодія мови програмування	$X1$	оп/мс	4	20	36
Необхідний об’єм пам’яті	$X2$	Гб	8	3	1
Час навчання нейронної мережі	$X3$	год	5	2.5	1
Потенційний об’єм програмного коду	$X4$	Рядки коду	2000	1200	700

За даними таблиці 5.2 будуються графічні характеристики параметрів – рис. 5.2 – рис. 5.5.



Рисунок 5.2 – X1, швидкодія мови програмування

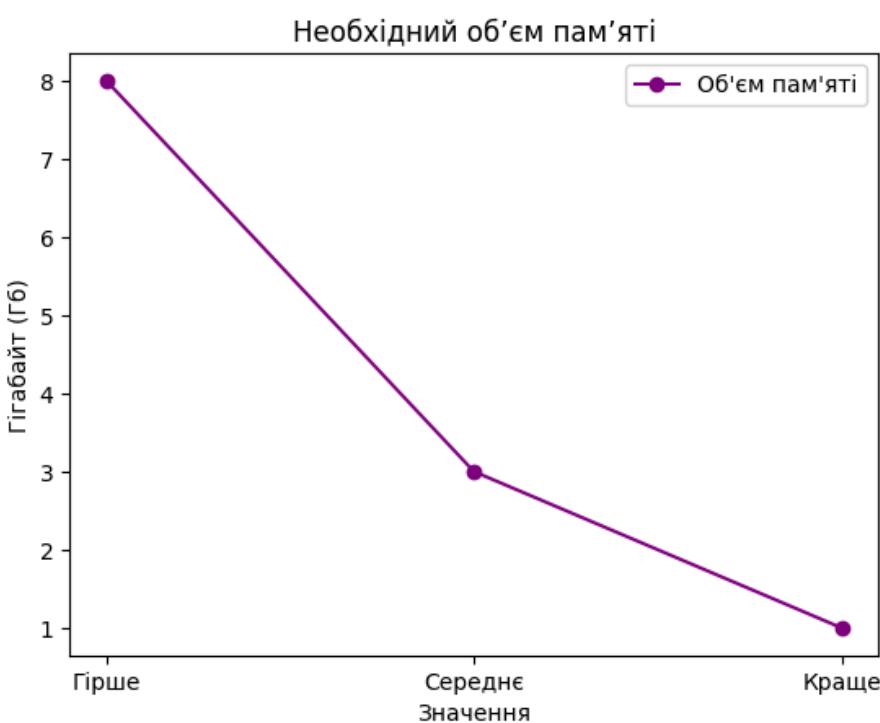


Рисунок 5.3 – X2, необхідний об'єм пам'яті

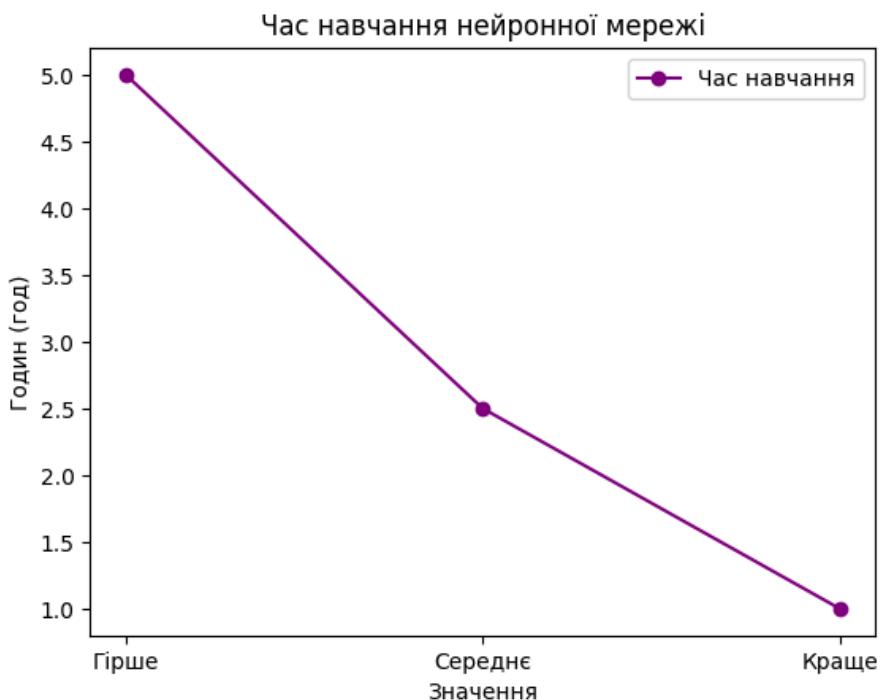


Рисунок 5.4 – X3, час навчання нейронної мережі

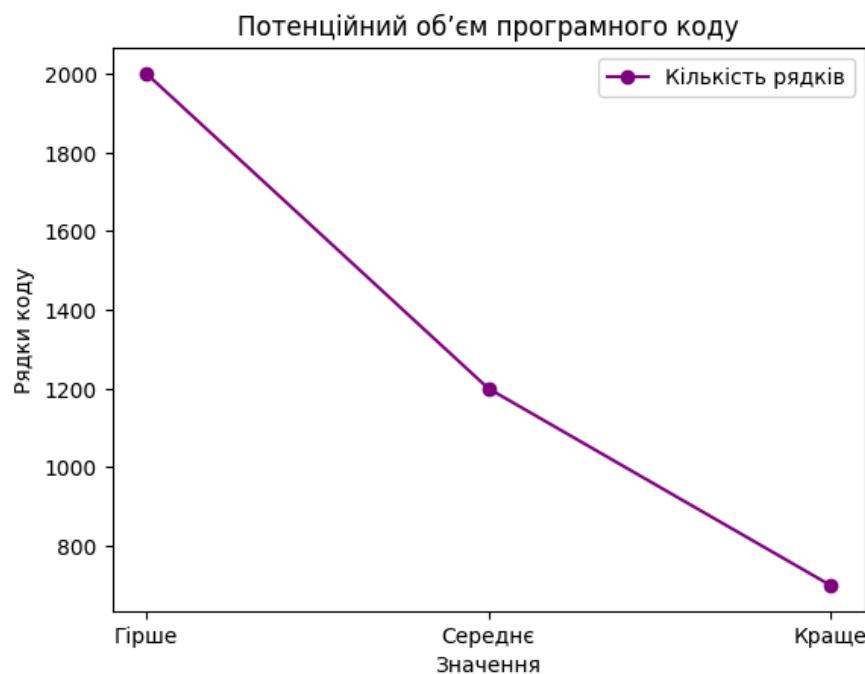


Рисунок 5.5 – X4, потенційний об'єм програмного коду

5.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати розпізнавання об'єктів.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значущості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 5.3.

Таблиця 5.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці вимірю	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X_1	Швидкодія мови програмування	оп/мс	1	2	2	1	1	1	2	10	-7.5	56.25
X_2	Необхідний об'єм пам'яті	Гб	3	4	3	3	4	3	4	24	6.5	42.25

Закінчення табл. 5.3

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X_3	Час навчання нейронної мережі	год	2	1	1	2	2	2	1	11	-6.5	42.25
X_4	Потенційний об'єм програмного коду	Рядки коду	4	3	4	4	3	4	3	25	7.5	56.25
	Разом		10	10	10	10	10	10	10	70	0	197

Для перевірки ступеня достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70 \quad (5.1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17.5 \quad (5.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \quad (5.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 197 \quad (5.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 197}{7^2(4^3-4)} = 0,754 > W_k = 0.67 \quad (5.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.67. Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати заносимо у таблицю 5.4.

Таблиця 5.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	<	<	<	<	0.5
X1 і X3	<	>	>	<	<	<	>	<	0.5
X1 і X4	<	<	<	<	<	<	<	<	0.5
X2 і X3	>	>	>	>	>	>	>	>	1.5
X2 і X4	<	>	<	<	>	<	<	<	0.5
X3 і X4	<	<	<	<	<	<	<	<	0.5

Числове значення, що визначає ступінь переваги i -го параметра над j -тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 & \text{при } X_i > X_j \\ 1.0 & \text{при } X_i = X_j \\ 0.5 & \text{при } X_i < X_j \end{cases} \quad (5.6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \{a_{ij}\}$.

Для кожного параметра зробимо розрахунок вагомості K_{bi} за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^N b_i}, \quad (5.7)$$

$$b_i = \sum_{j=1}^N a_{ij}. \quad (5.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^N b_i}, \quad (5.9)$$

$$b_i = \sum_{j=1}^N a_{ij} b_j. \quad (5.10)$$

Як видно з таблиці 5.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	b_i	K_{bi}	b_i^1	K_{bi}^1	b_i^2	K_{bi}^2
X1	1	0.5	0.5	0.5	2.5	0.16	9.25	0.16	34.125	0.16
X2	1.5	1	1.5	0.5	4.5	0.28	16.25	0.28	59.125	0.28
X3	1.5	0.5	1	0.5	3.5	0.22	12.25	0.21	41.875	0.2
X4	1.5	1.5	1.5	1	5.5	0.34	21.25	0.35	77.875	0.36
Всього:					16	1	59	1	213	1

5.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів $X2$ (необхідний об'єм пам'яті), $X3$ (час навчання нейронної мережі) та $X4$ (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра $X1$ (швидкодія мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 5.6):

$$K_K(j) = \sum_{i=1}^n K_{bi,j} B_{ij} \quad (5.11)$$

де n – кількість параметрів, K_{bi} – коефіцієнт вагомості i -го параметра, B_i – оцінка i -го параметра в балах.

Таблиця 5.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіанти реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	Б	$X1$	27	9	0.18	1.62
F2	А	$X2$	4.5	5	0.27	1.35
F3	А	$X3$	1.75	7	0.34	2.38
	Б	$X4$	800	8	0.11	0.88

За даними з таблиці 5.6 за формулою:

$$K_K = K_{\text{Ty}}[F_{1k}] + K_{\text{Ty}}[F_{2k}] + \dots + K_{\text{Ty}}[F_{zk}] \quad (5.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1.62 + 1.35 + 2.38 = 5.35; K_{K2} = 1.62 + 1.35 + 0.88 = 3.85$$

Як видно з розрахунків, кращим виявився варіант 1, для нього коефіцієнт технічного рівня має більше значення.

5.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості. Всі варіанти включають в себе два окремих завдання: розробка проекту програмного продукту, розробка програмної оболонки. Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3. Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних. Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як:

$$T_0 = T_p \cdot K_p \cdot K_{CK} \cdot K_M \cdot K_{CT} \cdot K_{CT.M} \quad (5.13)$$

де T_p – трудомісткість розробки ПП, K_p – поправочний коефіцієнт, K_{CK} – коефіцієнт на складність вхідної інформації, K_M – коефіцієнт рівня мови

програмування, K_{CT} – коефіцієнт використання стандартних модулів і прикладних програм, $K_{CT.M}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_p = 59$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_{\Pi} = 1.8$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{CK} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{CT} = 0.9$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 32 \cdot 1.8 \cdot 0.9 = 51.84 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, ступінь новизни Б), тобто $T_p = 29$ людино-днів, $K_{\Pi} = 0.9$, $K_{CK} = 1$, $K_{CT} = 0.8$:

$$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (51.84 + 19.44 + 5.35 + 19.44) \cdot 8 = 768.56 \text{ людино-годин,}$$

$$T_{II} = (51.84 + 19.44 + 3.85 + 19.44) \cdot 8 = 756.56 \text{ людино-годин.}$$

Найбільшу високу трудомісткість має варіант I.

У розробці беруть участь один програміст з окладом 20000 грн. та один аналітик в області даних з окладом 18000. Визначаємо середню зарплату за годину за формулою:

$$C_{\text{ч}} = \frac{M}{T_m \cdot t} \text{ грн}, \quad (5.14)$$

де M – місячний оклад працівників, T_m – кількість робочих днів тиждень, t – кількість робочих годин в день.

$$C_{\text{ч}} = \frac{20000+18000}{2 \cdot 21 \cdot 8} = 113.09 \text{ грн.} \quad (5.15)$$

Тоді, розраховуємо заробітну плату за формулою:

$$C_{\text{зп}} = C_{\text{ч}} \cdot T_i \cdot K_D, \quad (5.16)$$

де $C_{\text{ч}}$ – величина погодинної оплати праці програміста, T_i – трудомісткість відповідного завдання, K_D – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$\text{I. } C_{\text{зп}} = 113.09 \cdot 768.56 \cdot 1.2 = 104299.74 \text{ грн,}$$

$$\text{II. } C_{\text{зп}} = 113.09 \cdot 756.56 \cdot 1.2 = 102671.24 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становить 22%:

$$\text{I. } C_{\text{від}} = C_{\text{зп}} \cdot 0.22 = 104299.74 \cdot 0.22 = 22945.94 \text{ грн,}$$

$$\text{II. } C_{\text{від}} = C_{\text{зп}} \cdot 0.22 = 102671.24 \cdot 0.22 = 22587.67 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години (C_M).

Так як одна ЕОМ обслуговує одного програміста з окладом 20000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_{\Gamma} = 12 \cdot M \cdot K_3 = 12 \cdot 20000 \cdot 0.2 = 48000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{3П} = C_{\Gamma} \cdot (1 + K_3) = 48000 \cdot (1 + 0.2) = 57600 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{ВД} = C_{3П} \cdot 0.22 = 57600 \cdot 0.22 = 12672 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 24000 грн.

$$C_A = K_{TM} \cdot K_A \cdot \Pi_{PP} = 1.4 \cdot 0.25 \cdot 24000 = 8400 \text{ грн.},$$

де K_{TM} – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; K_A – річна норма амортизації; Π_{PP} – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{TM} \cdot \Pi_{PP} \cdot K_P = 1.4 \cdot 24000 \cdot 0.08 = 2688 \text{ грн.},$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{E\Phi} = (\Delta_K - \Delta_B - \Delta_C - \Delta_P) \cdot t_3 \cdot K_B = (365 - 104 - 12 - 16) \cdot 8 \cdot 0.35 = \\ = 627.2 \text{ години,}$$

де Δ_K – календарна кількість днів у році, Δ_B , Δ_C – відповідно кількість вихідних та свяtkovих днів, Δ_P – кількість днів планових ремонтів устаткування, t – кількість робочих годин в день, K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{EL} = T_{E\Phi} \cdot N_C \cdot K_3 \cdot \Pi_{EH} = 627.2 \cdot 0.2 \cdot 0.3 \cdot 5.23 = 210.93 \text{ грн,}$$

де N_C – середньо-споживча потужність приладу, K_3 – коефіцієнтом зайнятості приладу, Π_{EH} – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_H = \Pi_{PR} \cdot 0.67 = 24000 \cdot 0.67 = 16080 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{EKC} = C_{зп} + C_{від} + C_A + C_P + C_{EL} + C_H. \quad (5.17)$$

$$C_{EKC} = 57600 + 12672 + 8400 + 2688 + 210.93 + 16080 = 97650.93 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{M-G} = C_{EKC} / T_{E\Phi} = 97650.93 / 627.2 = 155.69 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_M = C_{M-\Gamma} \cdot T. \quad (5.18)$$

$$\text{I. } C_M = 155.69 \cdot 768.56 = 119657.1 \text{ грн,}$$

$$\text{II. } C_M = 155.69 \cdot 756.56 = 117788.82 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_H = C_{ЗП} \cdot 0.67. \quad (5.19)$$

$$\text{I. } C_H = 104299.74 \cdot 0.67 = 69880.82 \text{ грн,}$$

$$\text{II. } C_H = 102671.24 \cdot 0.67 = 68789.73 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{\text{пп}} = C_{ЗП} + C_{\text{від}} + C_M + C_H \quad (5.20)$$

$$\text{I. } C_{\text{пп}} = 104299.74 + 22945.94 + 119657.1 + 69880.82 = 316783.6 \text{ грн,}$$

$$\text{II. } C_{\text{пп}} = 102671.24 + 22587.67 + 117788.82 + 68789.73 = 311837.46 \text{ грн.}$$

5.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{TEP}j} = K_{kj}/C_{\Phi j} \quad (5.21)$$

$$K_{\text{TEP}1} = 5.35 / 316783.6 = 1.68885 \cdot 10^{-5},$$

$$K_{\text{TEP}2} = 3.85 / 311837.46 = 1.23462 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{TEP1} = 1.68885 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розробляється, можна зробити висновок, що з альтернатив, що залишились після первого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $K_{TEP} = 1.68885 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- вибір мови програмування – Python;
- вибір методу реалізації алгоритмів – застосування готових бібліотек;
- вибір середовища розробки – Google Colab.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

Висновки до розділу 5

У п'ятому розділі було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту. В результаті виконання функціонально-вартісного аналізу програмного комплексу що розробляється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують. На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

Дипломна робота на тему "Застосування глибокого навчання для виявлення військових об'єктів" охоплює сучасні підходи до використання технологій комп'ютерного зору та ставить за мету застосування глибокого навчання для вирішення актуальних військових завдань, наприклад розвідки та бойових операцій. Оглянувши підходи та технології, вибір впав на використання сучасної архітектури згорткової нейронної мережі – YOLOv8. Завдяки навчанню з передаваннями даної моделі, стало можливо дослідити застосування технологій зору для виявлення військових об'єктів. Серед двох класів військової техніки – броньованої та легкої, які були задані моделі, перший вирізняється високою успішністю розпізнавання порівнюючи з другим. Насамперед такі результати є наслідком дисбалансу кількості зображень класів у навчальному наборі даних. Проте явно прослідкована залежність: чим більше зображень класу – тим кращі результати розпізнавання. Збільшення часу навчання за рахунок більшої кількості епох також позитивно впливає на процес навчання, результати навчання на 300 епохах є кращими, ніж на 100.

Навчену модель застосовано для розпізнавання на реальних знімках та відеоматеріалах з військовою технікою, зокрема танками. Результатом є правильно визначена обмежувальна рамка об'єкту, та у випадку з відео – майже неперервне прослідковування об'єкту протягом всього запису. Загалом, результати дослідження підтверджують доцільність та ефективність застосування технологій глибокого навчання для виявлення військових об'єктів. Використання таких систем може значно підвищити точність та швидкість аналізу даних у військових операціях, забезпечуючи перевагу в інформаційному забезпеченні та оперативному прийнятті рішень. Світові тенденції показують зростаюче використання штучного інтелекту у військових технологіях, зокрема для автономних розвідувальних та бойових систем. Впровадження цих систем у

військову практику може значно підвищити ефективність оборонних операцій, зменшуючи ризики для людських життів та підвищуючи точність ударів.

Значущість роботи полягає у потенціалі для покращення інформаційного забезпечення військових операцій та створення нових технологічних рішень для національної безпеки. Подальший розвиток цих технологій та їх впровадження може мати значний науковий, економічний та соціальний вплив, сприяючи зміцненню обороноздатності країни та покращенню управління військовими ресурсами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Real-time Object Detection Using Deep Learning / K. Vaishnavi et al. Journal of Advances in Mathematics and Computer Science. 2023. Vol. 38, no. 8. P. 24–32. URL: <https://doi.org/10.9734/jamcs/2023/v38i81787> (дата звернення 20.05.2024).
2. OpenCV: OpenCV modules. URL: <https://docs.opencv.org/4.x/> (дата звернення 20.05.2024).
3. Khaledyan, Donya & Amirany, Abdolah & Jafari, Kian & Moaiyeri, Mohammad & Zargari, Abolfazl & Mashhadi, Najmeh. (2020). Low-Cost Implementation of Bilinear and Bicubic Image Interpolation for Real-Time Image Super-Resolution. 1-5. 10.1109/GHTC46280.2020.9342625.
4. What is Computer Vision? | IBM. IBM in Deutschland, Österreich und der Schweiz. URL: <https://www.ibm.com/topics/computer-vision> (дата звернення 20.05.2024).
5. Szeliski R. Computer Vision: Algorithms and Applications. Springer, 2010.
6. Vedoveli H. Image Filtering Techniques in Image Processing – Part 1. URL: <https://medium.com/@henriquevedoveli/image-filtering-techniques-in-image-processing-part-1-d03362fc73b7> (дата звернення 20.05.2024).
7. Lin Z. Digital Image Processing in C (Chapter 1): Mean and Median Filter. URL: <https://medium.com/@wilson.linzhe/digital-image-processing-in-c-chapter-1-mean-and-median-filter-b4c4d0775e14> (дата звернення 20.05.2024).
8. Dilhani S. Digital Image Processing Filters. URL: <https://medium.com/@shashikadilhani97/digital-image-processing-filters-832ec6d18a73> (дата звернення 20.05.2024).

9. Vedoveli H. Image Filtering Techniques in Image Processing – Part 2. URL: <https://medium.com/@henriquevedoveli/image-filtering-techniques-in-image-processing-part-2-ffd4b5531a1> (дата звернення 20.05.2024).
10. Woods R. E., Gonzalez R. C. Digital Image Processing. Pearson, 2017. 1192 p.
11. Bengio Y., Courville A., Goodfellow I. Deep Learning. MIT Press, 2016. 800 p.
12. Lavanya G., Pande S. D. Enhancing Real-time Object Detection with YOLO Algorithm. EAI Endorsed Transactions on Internet of Things. 2023. Vol. 10. URL: <https://doi.org/10.4108/eetiot.4541> (дата звернення 20.05.2024).
13. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks / S. Ren et al. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017. Vol. 39, no. 6. P. 1137–1149. URL: <https://doi.org/10.1109/tpami.2016.2577031> (дата звернення 20.05.2024).
14. Nielsen M. Neural networks and deep learning. URL: <http://neuralnetworksanddeeplearning.com/chap1.html> (дата звернення 20.05.2024).
15. Chollet F. Deep Learning with Python, Second Edition. Manning Publications Co. LLC, 2021.
16. 3Blue1Brown. What is backpropagation really doing? | Chapter 3, Deep learning, 2017. YouTube. URL: <https://www.youtube.com/watch?v=Ilg3gGewQ5U> (дата звернення 20.05.2024).
17. Generative adversarial networks / I. Goodfellow et al. Communications of the ACM. 2020. Vol. 63, no. 11. P. 139–144. URL: <https://doi.org/10.1145/3422622> (дата звернення 20.05.2024).
18. Fagbuyiro D. Guide To Transfer Learning in Deep Learning. URL: <https://medium.com/@davidfagb/guide-to-transfer-learning-in-deep-learning-1f685db1fc94> (дата звернення 20.05.2024).
19. Documentation. CVAT. URL: https://docs_cvat_ai_docs/ (дата звернення 22.05.2024).

20. Global Context Vision Transformers / A. Hatamizadeh et al. 2023. URL: <https://openreview.net/forum?id=75k3jJqAnm> (дата звернення 20.05.2024).
21. Military Reconnaissance. Homepage | National Air and Space Museum. URL: <https://airandspace.si.edu/explore/stories/military-reconnaissance> (дата звернення 20.05.2024).
22. General Atomics MQ-9 Predator B. Designation-Systems.Net. URL: <https://www.designation-systems.net/dusrm/app2/q-9.html> (дата звернення 20.05.2024).
23. Join Army Recognition Web TV membership with exclusive videos series training military equipment. Defense News security global military army equipment industry. URL: https://armyrecognition.com/focus-analysis-conflicts/army/conflicts-in-the-world/russia-ukraine-war-2022/russia-intensifies-use-of-armed-fpv-first-person-view-drones-to-strike-ukrainian-positions#google_vignette (дата звернення 20.05.2024).
24. Lancet and Shahed: Production, Weaknesses and Connections in the Russian Defense Industry. OSINT-агенція Molfar. URL: <https://molfar.com/en/blog/rosiyany-vyroblyayut-shahedy-ta-lancety-v-trc-dean-on-golovnogo-konstruktora> (дата звернення 20.05.2024).
25. The Impact of Drones on Future of Military Warfare. INTI MEDIA. URL: <https://media.inti.asia/read/the-impact-of-drones-on-future-of-military-warfare> (дата звернення 20.05.2024).
26. Glue R. YOLOv8, EfficientDet, Faster R-CNN or YOLOv5 for remote sensing. Medium. URL: <https://medium.com/@rustemgal/yolov8-efficientdet-faster-r-cnn-or-yolov5-for-remote-sensing-12487c40ef68> (дата звернення 22.05.2024).
27. You Only Look Once: Unified, Real-Time Object Detection. arXiv.org. URL: <https://arxiv.org/abs/1506.02640> (дата звернення 22.05.2024).
28. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks / S. Ren et al. IEEE Transactions on Pattern Analysis and Machine

- Intelligence. 2017. Vol. 39, no. 6. P. 1137–1149. URL:
<https://doi.org/10.1109/tpami.2016.2577031> (дата звернення 22.05.2024).
29. GitHub - ultralytics/ultralytics. GitHub. URL:
<https://github.com/ultralytics/ultralytics> (дата звернення 08.06.2024).
30. Introduction to Convolution Neural Network - GeeksforGeeks. GeeksforGeeks.
URL:
<https://www.geeksforgeeks.org/introduction-convolution-neural-network/> (дата звернення 08.06.2024).
31. What is Gradient Descent? | IBM. IBM - United States. URL:
<https://www.ibm.com/topics/gradient-descent> (дата звернення 08.06.2024).
32. Alake R. Loss Functions in Machine Learning Explained. Learn Data Science and AI Online | DataCamp. URL:
<https://www.datacamp.com/tutorial/loss-function-in-machine-learning> (дата звернення 09.06.2024).
33. Autopilot and Full Self-Driving Capability. Tesla. URL:
<https://www.tesla.com/support/autopilot> (дата звернення 09.06.2024).

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```

# -*- coding: utf-8 -*-
"""/diploma.ipynb

Automatically generated by Colab.

Original file is located at
https://colab.research.google.com/drive/1Xa5FXb6u4c6r-D9nqhaIsexyoEIW7lWo
"""

!pip install ultralytics

from google.colab import drive

drive.mount('/content/gdrive')

ROOT_DIR = '/content/gdrive/My Drive'

def resize_image(image, target_size):
    height, width = image.shape[:2]
    new_width, new_height = target_size
    scale = min(new_width / width, new_height / height)
    resized_image = cv2.resize(image, (0, 0), fx=scale, fy=scale)
    return resized_image

def scale_bounding_boxes(bboxes, original_size, target_size):
    original_height, original_width = original_size
    target_height, target_width = target_size

    scaled_bboxes = []
    for bbox in bboxes:
        class_id, x_center, y_center, width, height = bbox

        # Convert coordinates from relative to absolute
        x_min = (x_center - width / 2) * original_width
        y_min = (y_center - height / 2) * original_height
        x_max = (x_center + width / 2) * original_width
        y_max = (y_center + height / 2) * original_height

        x_min_scaled = x_min * (target_width / original_width)
        y_min_scaled = y_min * (target_height / original_height)
        x_max_scaled = x_max * (target_width / original_width)
        y_max_scaled = y_max * (target_height / original_height)

        x_center_scaled = (x_min_scaled + x_max_scaled) / (2 * target_width)
        y_center_scaled = (y_min_scaled + y_max_scaled) / (2 * target_height)
        width_scaled = (x_max_scaled - x_min_scaled) / target_width
        height_scaled = (y_max_scaled - y_min_scaled) / target_height

        scaled_bboxes.append([class_id, x_center_scaled, y_center_scaled, width_scaled, height_scaled])

    return scaled_bboxes

import os
import cv2

input_images_dir = os.path.join(ROOT_DIR, "data/images/")
input_labels_dir = os.path.join(ROOT_DIR, "data/labels/")
output_images_dir = os.path.join(ROOT_DIR, "scaled_data/images")

```

```

output_labels_dir = os.path.join(ROOT_DIR, "scaled_data/labels")

for dir_path in [output_images_dir, output_labels_dir]:
    if not os.path.exists(dir_path):
        os.makedirs(dir_path)

for filename in os.listdir(input_images_dir):
    if filename.endswith(".jpg") or filename.endswith(".png"):
        image_path = os.path.join(input_images_dir, filename)
        image = cv2.imread(image_path)
        original_size = image.shape[:2]

        target_size = (416, 416)
        resized_image = resize_image(image, target_size)

        labels_file = os.path.join(input_labels_dir, filename[:-4] + ".txt")
        with open(labels_file, 'r') as f:
            labels = [list(map(float, line.strip().split())) for line in f]

        for label in labels:
            label[0] = int(label[0])

        scaled_labels = scale_bounding_boxes(labels, original_size, target_size)

        output_image_path = os.path.join(output_images_dir, filename)
        cv2.imwrite(output_image_path, resized_image)

        output_labels_file = os.path.join(output_labels_dir, filename[:-4] + ".txt")
        with open(output_labels_file, 'w') as f:
            for bbox in scaled_labels:
                f.write(' '.join(map(str, bbox)) + '\n')

import numpy as np
from google.colab.patches import cv2_imshow

def visualize_boxes(image, labels_file):
    image_with_boxes = np.copy(image)

    with open(labels_file, 'r') as f:
        lines = f.readlines()
        for line in lines:
            print(line)
            parts = line.strip().split()
            class_id = int(parts[0])
            x_center = float(parts[1])
            y_center = float(parts[2])
            width = float(parts[3])
            height = float(parts[4])

            image_height, image_width = image.shape[:2]
            x_center_abs = int(x_center * image_width)
            y_center_abs = int(y_center * image_height)
            box_width = int(width * image_width)
            box_height = int(height * image_height)

            x1 = int(x_center_abs - box_width / 2)
            y1 = int(y_center_abs - box_height / 2)
            x2 = int(x_center_abs + box_width / 2)
            y2 = int(y_center_abs + box_height / 2)

            color = (0, 255, 0)
            thickness = 2

```

```

cv2.rectangle(image_with_boxes, (x1, y1), (x2, y2), color, thickness)

return image_with_boxes

image_path = os.path.join(ROOT_DIR, "scaled_data/images/993.png")
labels_file = os.path.join(ROOT_DIR, "scaled_data/labels/993.txt")
image = cv2.imread(image_path)
cv2_imshow(image)

image_with_boxes = visualize_boxes(image, labels_file)

cv2_imshow(image_with_boxes)

import os
import random
import shutil

input_images_dir = os.path.join(ROOT_DIR, "scaled_data/images")
input_labels_dir = os.path.join(ROOT_DIR, "scaled_data/labels")
output_images_train_dir = os.path.join(ROOT_DIR, "datasetV1/images/train")
output_images_val_dir = os.path.join(ROOT_DIR, "datasetV1/images/val")
output_labels_train_dir = os.path.join(ROOT_DIR, "datasetV1/labels/train")
output_labels_val_dir = os.path.join(ROOT_DIR, "datasetV1/labels/val")

for dir_path in [output_images_train_dir, output_images_val_dir, output_labels_train_dir, output_labels_val_dir]:
    if not os.path.exists(dir_path):
        os.makedirs(dir_path)

validation_percentage = 15

for filename in os.listdir(input_images_dir):
    if filename.endswith(".jpg") or filename.endswith(".jpeg") or filename.endswith(".png"): # Assuming images are in
        # Load image
        image_path = os.path.join(input_images_dir, filename)
        labels_path = os.path.join(input_labels_dir, filename[:-4] + ".txt")

        is_validation = random.random() < validation_percentage / 100

        output_images_dir = output_images_val_dir if is_validation else output_images_train_dir
        output_labels_dir = output_labels_val_dir if is_validation else output_labels_train_dir

        shutil.copy(image_path, os.path.join(output_images_dir, filename))
        shutil.copy(labels_path, os.path.join(output_labels_dir, filename[:-4] + ".txt"))

import os
from ultralytics import YOLO

# Load a model
model = YOLO("yolov8n.yaml")

# Use the model
results = model.train(data=os.path.join(ROOT_DIR, "google_colab_config.yaml"), epochs=300) # train the model

!zip -r /content/filenew.zip /content/runs

import locale
locale.getpreferredencoding = lambda: "UTF-8"

import matplotlib.pyplot as plt

```

```

def plot(values, label, title, xlabel, ylabel):
    labels = ['Гірше', 'Середнє', 'Краще']

    plt.plot(labels, values, marker='o', color='purple', label=label)

    plt.legend()

    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)

    plt.show()

plot([4, 20, 36], "Швидкодія", "Швидкодія мови програмування", "Значення", "Операцій за мілісекунду (оп/мс)")
plot([8, 3, 1], "Об'єм пам'яті", "Необхідний об'єм пам'яті", "Значення", "Гігабайт (ГБ)")
plot([5, 2.5, 1], "Час навчання", "Час навчання нейронної мережі", "Значення", "Годин (год)")
plot([2000, 1200, 700], "Кількість рядків", "Потенційний об'єм програмного коду", "Значення", "Рядки коду")

from ultralytics import YOLO

import cv2

model_path = 'best.pt'

image_path = 'data/images/train/41.jpg'

img = cv2.imread(image_path)
H, W, _ = img.shape

model = YOLO(model_path)

results = model(img)

for result in results:
    for box in result.boxes:
        print(box)
        #x1, y1, x2, y2 = detection['bbox']
        #class_label = detection['class_name']

        #cv2.rectangle(img, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2)
        #cv2.putText(img, class_label, (int(x1), int(y1) - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

        #cv2.imwrite('./output_with_detections.png', img)

import os
from ultralytics import YOLO
import cv2

VIDEOS_DIR = os.path.join('.', 'videos')

video_path = os.path.join(VIDEOS_DIR, 'tank2.mp4')
video_path_out = '{}_out.mp4'.format(video_path)

cap = cv2.VideoCapture(video_path)
ret, frame = cap.read()
H, W, _ = frame.shape
out = cv2.VideoWriter(video_path_out, cv2.VideoWriter_fourcc(*'MP4V'), int(cap.get(cv2.CAP_PROP_FPS)), (W, H))

model_path = "best2.0.pt"

model = YOLO(model_path)

```

```

threshold = 0.5

while ret:
    results = model(frame)[0]

    for result in results.bboxes.data.tolist():
        x1, y1, x2, y2, score, class_id = result

        if score > threshold:
            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 4)
            cv2.putText(frame, results.names[int(class_id)].upper(), (int(x1), int(y1 - 10)),
                       cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3, cv2.LINE_AA)

    out.write(frame)

cv2.imshow('Processed Video', frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

ret, frame = cap.read()

cap.release()
out.release()
cv2.destroyAllWindows()

import os
from ultralytics import YOLO
import cv2

IMAGES_DIR = os.path.join('.', 'images')
OUTPUT_DIR = os.path.join('.', 'processed_images')

if not os.path.exists(OUTPUT_DIR):
    os.makedirs(OUTPUT_DIR)

model_path = "best2.0.pt"

model = YOLO(model_path)

threshold = 0.5

for image_name in os.listdir(IMAGES_DIR):
    if image_name.lower().endswith('.png', '.jpg', '.jpeg'):
        image_path = os.path.join(IMAGES_DIR, image_name)
        output_path = os.path.join(OUTPUT_DIR, image_name)

        image = cv2.imread(image_path)
        if image is None:
            print(f"Error: Could not read image {image_name}")
            continue

        results = model(image)[0]

        for result in results.bboxes.data.tolist():
            x1, y1, x2, y2, score, class_id = result

            if score > threshold:
                cv2.rectangle(image, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 4)
                cv2.putText(image, results.names[int(class_id)].upper(), (int(x1), int(y1 - 10)),
                           cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3, cv2.LINE_AA)

cv2.imwrite(output_path, image)

```

```

cv2.imshow('Processed Image', image)
cv2.waitKey(1000)

cv2.destroyAllWindows()

import os
from ultralytics import YOLO
import cv2

VIDEOS_DIR = os.path.join('.', 'videos')

video_path = os.path.join(VIDEOS_DIR, 'tank2.mp4')
video_path_out = '{}_out.mp4'.format(video_path)

cap = cv2.VideoCapture(video_path)
ret, frame = cap.read()
H, W, _ = frame.shape
out = cv2.VideoWriter(video_path_out, cv2.VideoWriter_fourcc(*'MP4V'), int(cap.get(cv2.CAP_PROP_FPS)), (W, H))

model_path = "best2.0.pt"

model = YOLO(model_path)

threshold = 0.5

while ret:
    results = model(frame)[0]

    for result in results.boxes.data.tolist():
        x1, y1, x2, y2, score, class_id = result

        if score > threshold:
            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 4)
            cv2.putText(frame, results.names[int(class_id)].upper(), (int(x1), int(y1 - 10)),
                       cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3, cv2.LINE_AA)

    out.write(frame)

    cv2.imshow('Processed Video', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    ret, frame = cap.read()

cap.release()
out.release()
cv2.destroyAllWindows()

import os

txt_directory = "C:/Users/akapu/Pictures/unprocessed_v2/labels"
img_directory = "C:/Users/akapu/Pictures/unprocessed_v2/images"

print(os.listdir(txt_directory))

txt_files = [f for f in os.listdir(txt_directory) if f.endswith('.txt')]

txt_numbers = [os.path.splitext(f)[0] for f in txt_files]
print("Number of txt files: ", len(txt_files))

img_files = [f for f in os.listdir(img_directory) if f.endswith('.jpg', '.jpeg', '.png')]


```

```
for img_file in img_files:  
    img_number = os.path.splitext(img_file)[0]  
    if img_number not in txt_numbers:  
        print(img_number)  
        os.remove(os.path.join(img_directory, img_file))
```