Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження та застосування методів комп'ютерного зору для вирішення задач ідентифікації та класифікації твердих відходів
(тема)

Виконав:
студент 2 курсу, групи СШІм-22-3
Бухановський В.О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)
Керівник проф. Рябова Н.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Бухановському Володимиру Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та застосування методів комп'ютерного зору для вирішення задач ідентифікації та класифікації твердих відходів

затверджена наказом університету від 1 квітня 20 24 р. № 260Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 червня 20 24 р.

3. Вихідні дані до роботи Науково-технічні публікації, відкриті джерела в мережі Інтернет, відкриті набори даних, документація мови програмування Python, документація фреймворків для побудови нейронних мереж, документація платформи Raspberry PI

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Огляд методів та алгоритмів на основі глибинних нейронних мереж
3) Програмна реалізація, перенос моделі та оцінка результатів роботи

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.04.2024	виконано
2	Аналіз предметної області та постановка задачі	12.04.2024	виконано
3	Огляд методів та алгоритмів комп’ютерного зору	24.04.2024	виконано
	та аугментації		
4	Обробка даних та аугментація	02.05.2024	виконано
5	Розробка моделі виявлення та класифікації	12.05.2024	виконано
6	Перенесення моделі на мобільний пристрій,	20.05.2024	виконано
	оптимізація роботи		
7	Обробка і оформлення результатів	28.05.2024	виконано
8	Оформлення графічних матеріалів	01.06.2024	виконано
9	Оформлення пояснювальної записки	05.06.2024	виконано
10	Попередній захист	10.06.2024	виконано
11	Захист перед ЕК	13.06.2024	виконано

Дата видачі завдання 1 квітня 20 24 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Рябова Н.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 60 с., 31 рис., 1 дод., 35 джерел

ВІДХОДИ, ГЛИБОКІ НЕЙРОННІ МЕРЕЖІ, ЗАБРУДНЕННЯ, КОМП'ЮТЕРНИЙ ЗІР, МАШИННЕ НАВЧАННЯ, МОБІЛЬНІ ПРИСТРОЇ, НАВКОЛИШНЄ СЕРЕДОВИЩЕ, СМІТТЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об'єкт дослідження – проблема ідентифікації та класифікації твердих відходів.

Предмет дослідження – методи та алгоритми комп'ютерного зору для вирішення проблеми ідентифікації та класифікації твердих відходів.

Мета роботи полягає у проектуванні та імплементації моделі згорткової нейронної мережі, яка ефективно ідентифікує та класифікує тверді відходи для її подальшого широкого застосування на мобільних пристроях.

Методи дослідження – аналітичний (аналіз існуючої проблематики, сучасних технологій та даних), експериментальний (розробка та тестування моделей штучного інтелекту, що засновані на згорткових нейронних мережах, оцінка їх ефективності на мобільних пристроях).

Проведено дослідження теоретичних матеріалів щодо сучасних технологій у сфері комп'ютерного зору, досліджено засадничі підходи щодо роботи згорткових нейронних мереж, проаналізовано існуючі джерела даних, приділено увагу до проблеми обмеженості даних для навчання, запропоновано ефективні методики аугментації даних для покращення становища. Поставлені задачі щодо практичної реалізації, окреслено технології, що мають бути використані для імплементації.

ABSTRACT

Master’s thesis contains: 60 pp., 31 fig., 1 tabl., 1 ann., 35 references.

ARTIFICIAL INTELLIGENCE, COMPUTER VISION, DEEP NEURAL NETWORKS, ENVIRONMENT, GARBAGE, MACHINE LEARNING, MOBILE DEVICES, POLLUTION, SOLID WASTE.

The object of this research is the problem of identification and classification of solid waste.

The subject of the research is computer vision methods and algorithms for solving the solid waste identification and classification problem.

The purpose of the work is to design and implement a convolutional neural network model that effectively identifies and classifies solid waste for its further widespread use on mobile devices.

Research methods are analytical (analysis of existing problems, modern technologies, and data), experimental (development and testing of artificial intelligence models based on convolutional neural networks, evaluate their efficiency on mobile devices).

A study of theoretical materials on modern technologies in the field of computer vision was carried out, basic approaches to the operation of convolutional neural networks were investigated, existing data sources were analyzed, attention was paid to the problem of limited data for training, effective methods of data augmentation were proposed to improve the situation. Tasks regarding practical implementation are set, technologies to be used for implementation are outlined.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БПЛА – безпілотний літальний апарат;

ШІ – штучний інтелект;

ШНМ – штучні нейронні мережі;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

COCO – Common Objects In Context – поширені об’єкти в контексті;

IoT – Internet Of Things – інтернет речей;

ONNX – Open Neural Network Exchange – бібліотека для побудови глибоких ШНМ;

TACO – Trash Annotations In Context – анотації про сміття в контексті;

TPU – Tensor Processing Unit – тензорний процесор;

YOLOvX – You Only Look Once, version X – назва алгоритму для комп’ютерного зору X версії.

ВСТУП

Збереження навколишнього середовища – питання денного порядку в розвинутих країнах світу. До прикладу в 2019 році європейський союз прийняв Європейський кліматичний закон. Цей закон закріплює мету, викладену в Європейській зеленій угоді, щоб економіка та суспільство Європи стали кліматично нейтральними до 2050 року. Закон також встановлює проміжну ціль скорочення чистих викидів парникових газів щонайменше на 55% до 2030 року, порівняно з рівнем 1990 року [1]. Варто звернути увагу на формулювання, що для досягнення мети мають долучитися всі.

Частиною загальної проблеми є побутові відходи. Забруднення побутовими відходами для багатьох людей, нажаль, стає даниною в наш час. І якщо деякі країни активно впроваджують ініціативи задля сортування сміття, його переробки та повторного використання, то в деяких країнах світу – це все ще не є пріоритетом.

Розглянемо приклад Індії. Швидке зростання населення, урбанізація та зміна моделей споживання в Індії призвели до утворення величезної кількості твердих відходів. Відповідно до інформації опублікованої Міністерством житлового будівництва та міських справ у січні 2020 року, 147 613 метричних тон твердих відходів утворюється щодня в 84 475 округах по всій Індії.

Існуючий процес збору, транспортування та утилізації, як частини управління твердими побутовими відходами, зазнає надмірного тиску. Неправильна утилізація твердих відходів становить великий ризик для здоров'я населення.

За даними Цільової групи Комісії з планування, органічні відходи, що утворюються в домогосподарствах, такі як харчові відходи, кухонні відходи, овочеві відходи, фруктові шкірки, папір тощо, становлять близько 52% від загального обсягу відходів.

Ці органічні відходи можуть піддаватися процесу ферментації, створюючи сприятливе середовище для виживання та розвитку мікробних патогенів. Це в свою чергу може стати серйозною загрозою та завдати шкоди здоров'ю. Прямий контакт з такими відходами може спричинити різні типи інфекцій та хронічні захворювання. Також існує кілька шкідливих наслідків твердих відходів. Наприклад, залишене без догляду сміття на узбіччі, яке є звичайним місцем у різних містах Індії, може стати розсадником комарів, тарганів і щурів [2].

Щоб покращити загальну ситуацію, що була описана вище, варто комплексно підійти до проблеми. По-перше, посилити законодавчу базу щодо регулювання процесу взаємодії з відходами. По-друге, покращити інфраструктуру з фокусом на сегрегацію та переробку відходів. По-третє, проводити освітню діяльність щодо донесення важливості правильної утилізації відходів. Останнє, але не менш важливе – збільшити ефективність нагляду за виконанням норм права.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Опис предметної області

Забруднення – це велика проблема сучасного світу. Вона виникає внаслідок введення в навколишнє середовище нових фізичних, хімічних та біологічних факторів, які завдають шкоди природним екосистемам та здоров'ю людини.

Розрізняють декілька категорій забруднення залежно від їх поширення у просторі та походження. Локальні забруднення характерні для конкретних підприємств, населених пунктів або окремих районів, де відбувається видобуток корисних копалин або масове сільське господарство. Регіональні забруднення охоплюють великі території, зазвичай внаслідок промислової діяльності, тоді як глобальні забруднення мають планетарний масштаб і викликані атмосферними викидами.

За типом походження виділяють фізичні, механічні, хімічні та біологічні забруднення. Фізичні забруднення включають зміни теплових, електричних, радіаційних та інших полів у природному середовищі, тоді як механічні – це тверді частки та предмети, які потрапляють в навколишнє середовище. До хімічних речовин відносять різноманітні хімічні речовини, що порушують природні процеси кругообігу, а біологічні – різні організми, які можуть бути шкідливими для природи та здоров'я людини [3].

В контексті цієї роботи особливої уваги заслуговують локальні механічні забруднення неорганічними речовинами. Такий вид забруднення найчастіше можна споглядати в побуті, де переважають тверді відходи, які включають пластикові пляшки, пакети, упаковки та інші об'єкти, виготовлені з неорганічних матеріалів. Розробка системи комп’ютерного зору для виявлення та класифікації таких об'єктів має бути кроком до поліпшення ситуації з кількістю зазначених відходів, та більш оперативного реагування на такого роду забруднення.

У цьому контексті згорткові нейронні мережі (CNN) є потужним інструментом для виявлення та класифікації об'єктів на зображеннях, таких як пластикові відходи та інші. Згорткова модель дозволяє ефективно виділяти деталі та ознаки на зображенні, що робить її ідеальним інструментом для розв'язання проблеми виявлення забруднень в навколишньому середовищі [4].

1.2 Згорткові нейронні мережі

Штучні нейроні мережі (ШНМ) – це математична модель, яка створена подібно до біологічних нейронних мереж, як мозок живої істоти. Такі системи навчаються поставленим задачам, розглядаючи приклади. Вони не потребують спеціального програмування для конкретного застосування. Такі моделі відмінно справляються із задачами розпізнавання натуральних мов, об’єктів на зображеннях. Основою для навчання є частково впорядковані чи невпорядковані данні, такі як тексти, зображення, відео.

Більш технічне визначення цього терміну каже, нейронна мережа – це послідовність нейронів, що поєднані між собою синапсами. Виходить, що нейрон – це одиниця обчислення мережі.

Нейрон отримує інформацію, робить розрахунки, віддає їх результати. З цього можна зробити висновок, що кожен нейрон має вхідні та вихідні дані. Для вхідного нейрона вхідні дані дорівнюють вихідним. В інших випадках вхідні дані нейрону – це зважена сумарна інформація всіх нейронів з попереднього шару, після чого вона нормалізується за допомогою функції активації і потрапляє у вихідні дані.

Основних типів нейронів нараховують три: вхідні, приховані та вихідні. Існують ще інші типи нейронів: нейрон зміщення та контекстний нейрон. Нейрон в свою чергу поділяються на шари в залежності від типів, а також загальної архітектури моделі (наприклад, приховані нейрони можуть бути поділені на декілька шарів). Кількість шарів визначається за багатьма

параметрами: складності задачі, її специфіки, очікуваної ефективності моделі по різним параметрам та інші.

За поєднання нейронів відповідають синапси, вони вказують, який наступний нейрон отримає дані. Завдяки ньому інформація, проходячи від одного нейрону до іншого, змінюється. Синапс, у якого більша вага, передає інформацію, яка буде домінуючою в наступному нейроні.

Сукупність ваг (синапсів) нейронної мережі або матриця ваг – це мозок всієї моделі. Саме завдяки цим вагам, вхідна інформація обробляється та перетворюється на результат [5].

Невід'ємною частиною нейрону є функція активації, яка є способом нормалізації вхідних даних. Велике значення на вході, проходячи функцію активації перетворюється на вихідне значення у потрібному діапазоні. Функцій активації дуже різноманітні та мають свою переваги і недоліки. Найпопулярніші з них: лінійна, сігмоїдальна та гіперболічний тангенс.

Згорткові нейронні мережі (CNN) є потужними інструментом для обробки зображень у глибокому навчанні. Їх основним призначенням є виявлення різних особливостей та шаблонів у зображеннях, що допомагає у різних задачах, таких як класифікація об'єктів, розпізнавання обличчя та багато іншого.

Головним елементом CNN є фільтри, які складаються з ядер – матриць ваг, що навчаються розпізнавати певні особливості на зображеннях. Фільтри застосовуються до різних частин зображення, виконуючи операцію згортки. Вона дозволяє виявляти характеристики, такі як границі, форми та текстури.

Під час роботи з фільтрами важливо враховувати кількість каналів вихідного зображення, оскільки кожен канал вимагає свій власний фільтр для ефективного виявлення особливостей. Додатково, фільтри можуть рухатися з різним кроком (страйдом), що дозволяє контролювати розмірність вихідних сигналів та швидкість обробки. До зображення можуть додаватися падінги, щоб фільтр пройшовся не тільки по середній частині зображення, але і по його боках.

Після застосування згорткових шарів зазвичай використовуються шари, що зменшують розмірності даних. Популярним і простим методом є максимальне об’єднання, що сприяє зменшенню кількості оброблених даних із збереженням загального контексту [6].

Після цього до отриманих даних застосовується функція активації, як і в класичних ШНМ, яка допомагає виявляти нелінійні зв'язки між особливостями. Часто використовується функція ReLu, через свою універсальність та ефективність [7].

Остаточно, для оцінки якості роботи нейронної мережі використовується функція втрат, яка дозволяє визначити, наскільки точно мережа впоралася. Це невід’ємна частина навчання моделі.

Загалом, CNN – це складна, але дуже ефективна модель для обробки зображень, яка знаходить застосування у багатьох областях.

1.3 Технології Computer Vision

Технології Computer Vision (комп’ютерного зору) стали незамінними в багатьох сферах починаючи з охорони здоров’я, безпекових рішень, військових технологій, завершуючи соціальними мережами. Ці технології використовуються для автоматизації, оптимізації та покращення різноманітних процесів, які в минулому не можливо було делегувати комп’ютеру.

Важливим фактором, що сприяє впровадженню комп’ютерного зору, є впровадження глибокого навчання, зокрема архітектури глибокого навчання, які ефективно обробляють візуальні дані.

Далі розглянемо декілька основних архітектур, їх особливості, та порівняємо них.

Згорткові нейронні мережі (CNN – Convolutional Neural Network). Основна більшості методів глибокого навчання для комп’ютерного зору – це згорткові нейронні мережі. Це мережа, яка використовує згорткові шари,

шари об’єднання, і повністю пов’язані шари для аналізу візуальних даних (рисунок 1.1).

Кожен шар у CNN поступово створює комплексне розуміння вхідного зображення. Згорткові шари застосовують набір доступних для навчання фільтрів, які підсвічують різні елементи зображення. Шари об’єднання зменшують просторові розміри, зберігаючи корисну інформацію, що робить мережу більш ефективною та менше схильною до перенавчання. Кінцеві повністю пов’язані шари виконують високорівневі висновки на основі виділених ознак минулими шарами.

Класичні архітектури включають LeNet-5 для розпізнавання цифр, AlexNet ,яка перемогла на конкурсі ImageNet у 2012 році, і VGGNet, яка підтвердила важливість глибини в нейронних мережах.

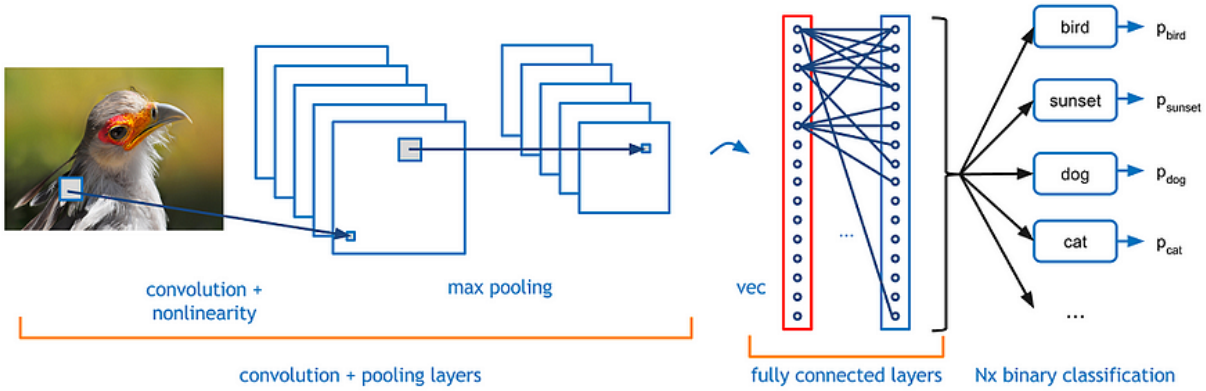


Рисунок 1.1 – Схематичне зображення архітектури CNN [8]

Залишкові мережі (Residual Networks – ResNets). Одною з традиційних проблем глибоких нейронних мереж є зникаючий градієнт, ResNets, представлений Kaiming He та ін., запропонував іноваційне рішення, запровадивши «швидкі» або «пропускні з’єднання» [9]. Ці з’єднання дозволяють градієнтам поширюватися прямо через мережу, покращуючи навчання глибших моделей (рисунок 1.2).

ResNets знайшли широке застосування, від розпізнавання об’єктів до аналізу відео. Варіанти архітектури ResNet, як-от ResNeXt, яка включає згруповані згортки для підвищення потужності та ефективності, були досить успішними.

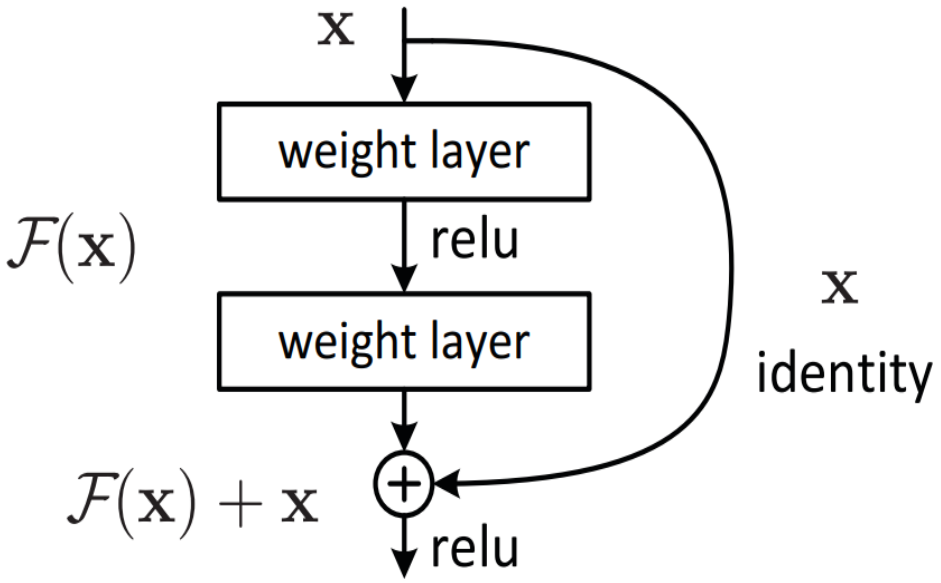


Рисунок 1.2 – Залишкове навчання: будівельний блок

Породжуючи мережі (Inception Networks). Inception мережі, вперше представлені в GoogLeNet [10] для конкурсу ImageNet, пропонують архітектуру «мережа в мережі». Він використовує набір паралельних згорток із різними розмірами ядра на кожному шарі, ефективно дозволяючи моделі вивчати різні представлення просторових ознак одночасно, одночасно зменшити кількість параметрів для навчання завдяки ефективному процесу фільтрації (рисунок 1.3) [11]. Пізніші версії мереж Inception представили такі поняття, як нормалізація пакетів і згладжування міток для більш ефективного та стабільного навчання.

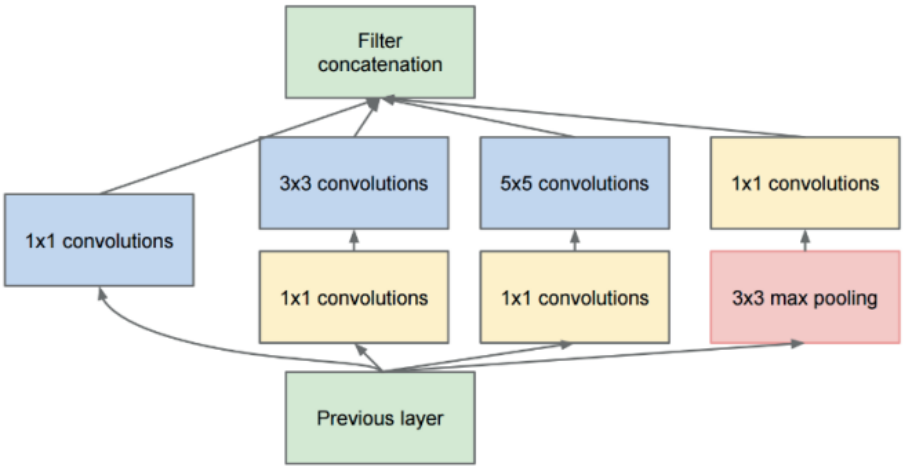


Рисунок 1.3 – Схематичне зображення Inception модуля

Регіональні згорткові нейронні мережі (Region-based Convolutional Neural Networks – R-CNN та нащадки). Хоча CNN успішно класифікують зображення, їм все ж бракує можливості надати контекст про те, де на зображенні розташований об’єкт. R-CNN архітектура заповнила цю прогалину, застосовуючи CNN до різних «регіонів пропозиції» всередині зображення для виявлення та класифікації об’єктів (рисунок 1.4) [12].

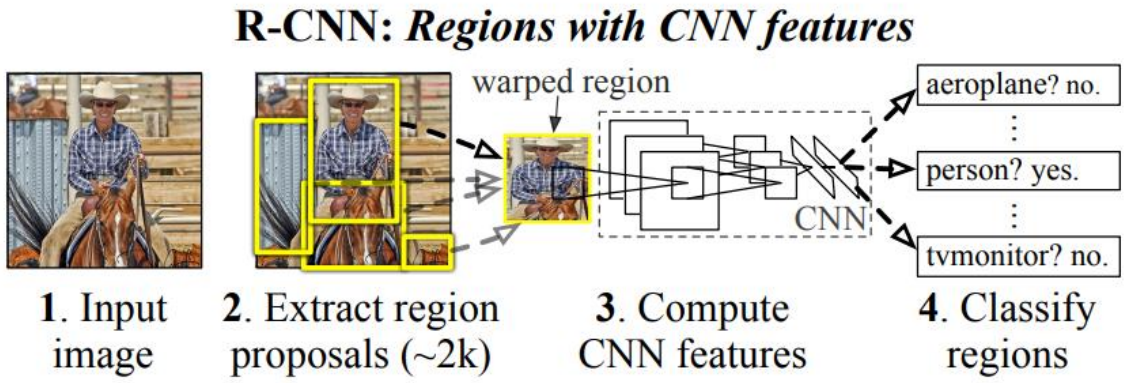


Рисунок 1.4 – Схематичне зображення принципу роботи R-CNN

З часом були розроблені швидші та ефективніші версії, зокрема Fast R-CNN, Faster R-CNN і Mask R-CNN (для сегментації на рівні пікселів).

Генеративні змагальні мережі (Generative Adversarial Networks – GAN). Популярні зараз GAN – це зовсім інша архітектура, яка переважно використовується для генеративних завдань, а не для задач розпізнавання. GAN складається з двох основних компонентів: мережі генератора, яка вчиться створювати дані, схожі на справжній розподіл даних, і мережі дискримінатора, яка вчиться розрізняти реальні та згенеровані дані (рисунок 1.5). Дві мережі тренуються одночасно в грі міні-макс, підштовхуючи одна одну до постійного покращення [13]. Налаштувати таку модель – це виклик, але потенціал у таких моделей неймовірний.

Архітектури GAN, які заслуговують уваги – це DCGAN (Deep Convolutional GAN), CycleGAN для непарного перетворення зображення в зображення та StyleGAN від NVIDIA, відомого тим, що створює неймовірно реалістичні людські обличчя.

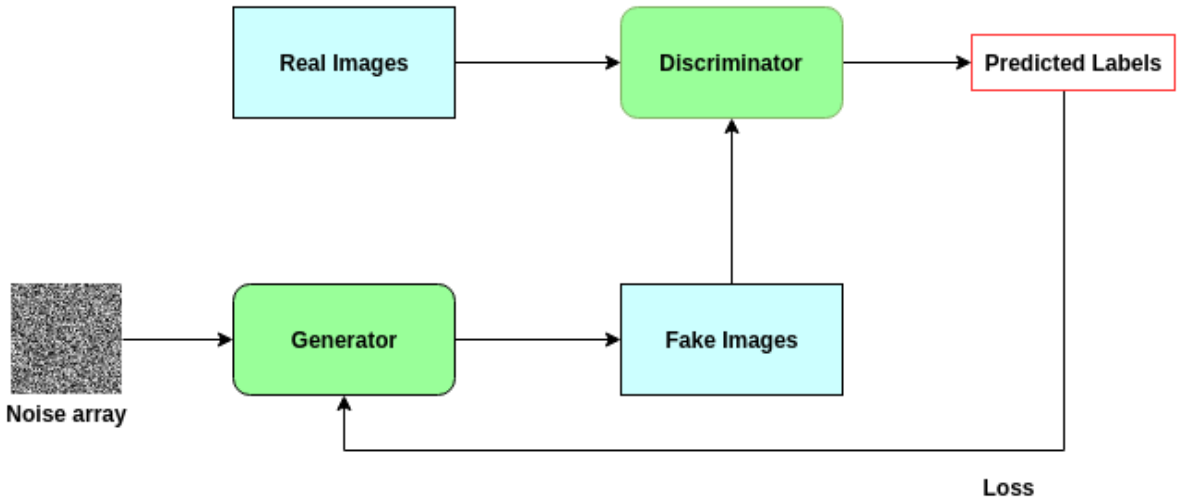


Рисунок 1.5 – схематичне зображення архітектури GAN

Архітектура Трансформер (Transformer) у контексті Computer Vision (рисунок 1.6). Спочатку розроблені для завдань обробки природної мови, трансформаторні моделі знайшли свій шлях до комп'ютерного зору з такими архітектурами, як Vision Transformer (ViT) і DETR (для виявлення об'єктів). Ці моделі, що спираються на механізми самоуважності (self-

attention), показали непогані результати в деяких задачах, іноді перевершуючи традиційні підходи на основі CNN.

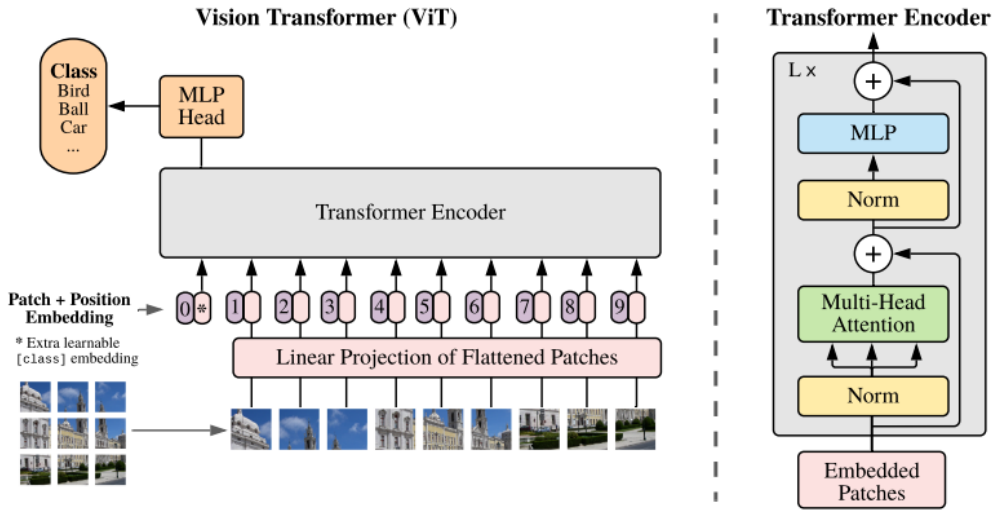


Рисунок 1.6 – Схематичне зображення ViT

Досить незвично бачити використання моделі Transformer для задач Computer Vision. Подивимося більш глибше, як це працює. Зображення розбивається на фрагменти фіксованого розміру, лінійно вбудовується кожен із них, додається вбудовування позиції та передається отримана послідовність векторів у стандартний кодер Transformer. Щоб виконати класифікацію, для цього використовується стандартний підхід додавання додаткового матеріалу для навчання «токен класифікації» до послідовності [14].

Підсумовуючи вибір архітектури має залежати від поставленого завдання. Важливо розуміти, що глибші мережі, такі як ResNet, InceptionNet і моделі на основі трансформерів, зазвичай забезпечують кращу продуктивність, але вони в той же час мають більші витрати з точки зору обчислювальних ресурсів і можуть бути надмірними для більш простих завдань. Тому вибір архітектури – це баланс між складністю завдання,

обчислювальними ресурсами та необхідною точністю. Важливо пам’ятати, що всі моделі мають свої сильні та слабкі сторони, що може дуже впливати на результат конкретної моделі.

Архітектури глибокого навчання значно вплинули на розвиток і прогрес комп’ютерного зору. Сучасні глибокі нейроні мережі у напрямку Computer Vision зазвичай поєднують в собі декілька описаних вище архітектур, їх частин для досягнення найбільшої ефективності.

1.4 Існуючі джерела даних

У відкритому доступі знаходиться велика кількість джерел, які можуть бути корисними для розпізнавання та класифікації твердих відходів, бо цей вид відходів дуже розповсюджений. Розглянемо декілька таких джерел даних. В якості першоджерела списку я використовую проект на гітхабі, автор якого дуже ретельно проаналізував безліч джерел даних доступних онлайн [15].

TrashNet. Це датасет, який містить зображення сміття, розділеного на шість класів: папір, картон, скло, метал, пластик та інші. Цей датасет створено для розпізнавання різних типів сміття та розвитку моделей машинного навчання для автоматичного сортування відходів [16].

TrashCan. Це набір даних з 7 212 зображень, які містять спостереження за сміттям та широкою різноманітністю підводної флори та фауни. Зображення в TrashCan отримано з набору даних J-EDI (JAMSTEC E-Library of Deep-sea Images), куратором якого є Японське агентство морських наук і технологій (JAMSTEC). Цей набір даних містить відео з ROV (remotely operated vehicle – дистанційно керований транспортний засіб), якими керує JAMSTEC з 1982 року, переважно в Японському морі [17].

В розділі 1.7 буде розглянуто існуюче рішення SpotGarbage. Для нього існує окремий набір даних Garbage in Images (GINI). Він складається з 2561 зображенням із невизначеною роздільною здатністю, 1496 зображень

анотовано обмежувальними рамками (один клас – сміття). Для створення набору даних використовувався API пошуку зображень Bing.

Джерело даних, яке буде використовуватися в цій роботі – це TACO (Trash Annotations in Context). Це набір даних з 1500 зображень, які було зібрано в основному на відкритому повітрі, таких як ліси, дороги та пляжі, і анотовано масками сегментації. Є 60 категорій, які належать до 28 супер (верхніх) категорій. Додаткові зображення мають фоновий тег: вода, пісок, сміття, рослинність, приміщення, тротуар [18].

Цей датасет ідеально вписується в концепцію проектованої моделі завдяки різноманітності категорій та умов фіксації твердих відходів.

1.5 Аугментація даних

Аугментація даних (від data augmentation) – це техніка штучного збільшення навчальних даних, яка використовується у машинному навчанні. Цей прийом може використовуватися в різних напрямках машинного навчання, наприклад: комп’ютерне зору, обробка природної мови, розпізнавання мовлення та інше.

Конкретно розглянемо кейс для комп’ютерного зору та роботи із зображеннями. Збільшення кількості даних досягається за рахунок створення нових, в різній мірі змінених версій існуючих даних. У більшості випадків це відбувається за допомогою різних трансформацій, таких як обертання, обрізання, масштабування, відображення, додавання шуму та інших модифікацій до вхідних даних. Отримані змінені дані можна використовувати для покращення ефективності та стійкості моделей машинного навчання, а також закривати потребу моделей у великій кількості даних, маючи при цьому обмежений датасет.

Важливість цієї техніки важко переоцінити. Звісно, в деяких предметних областях, наприклад NLP, безмірна кількість даних, бо люди по всьому світу генерують кожен день велику кількість контенту. Але якщо

мова йде про більш вузько спеціалізовані напрямки, то відсутність достатньої кількості даних – велика проблема.

Як було вже зазначено, головна ціль аугментації даних – це збільшення кількості даних. Це в свою чергу може поліпшити ефективність моделі. У багатьох випадках моделі вимагають великої кількості даних для вивчення складних шаблонів та взаємозв'язків, що призводить до покращення їх розуміння та результативності. Більш того аугментація даних може покращити здатність моделей до узагальнення. Додавання варіацій у данні, дозволяє моделі розпізнавати важливі ознаки, які залишаються незмінними під час цих змін. Це допомагає у запобіганні перенавчанню, коли модель запам'ятовує тренувальні дані та недостатньо узагальнюється на нові, невидимі дані [19].

Якщо підсумовувати, то в більшості випадків аугментація даних допомагає підвищити ефективність моделей, коли доступні обмежені обсяги даних. У багатьох випадках збір великої кількості тренувальних даних є нереальним. А за рахунок введення різноманітних прикладів у тренувальні дані, аугментація даних поліпшує точність моделей. Це особливо важливо в області комп'ютерного зору, де можуть відзначатися варіації в освітленні, положенні та орієнтації об'єктів, які модель має розпізнавати.

Існує набір методів, який використовується для збільшення даних. Приклад їх використання можна побачити на рисунку 1.7. Загалом ці методи можна розділити на дві великі групи: зміна форми зображення, коригування зображення. Розглянемо декілька прикладів з кожної з груп.

Перша група методів включає в себе наступні:

– ротація – метод, що передбачає поворот зображення на певний кут, це допомагає розпізнавати об’єкти з різних кутів;

– обрізання – метод, який передбачає випадковий вибір частини зображення та використання лише цієї частини для навчання, це допомагає моделі навчитися розпізнавати важливі особливості об’єкта, який розпізнається, навіть якщо він не в центрі зображення;

– масштабування – це метод, який передбачає зміну розміру зображення до меншого або більшого розміру, це допомагає моделі навчитися розпізнавати об’єкти в різних масштабах;

– перегортання – це метод, який передбачає горизонтальне або вертикальне перегортання зображення, це допомагає моделі навчитися розпізнавати об’єкти, орієнтовані в різних напрямках;

– переклад – це метод, який передбачає зміщення зображення на певну відстань по горизонталі або вертикалі, це допомагає моделі навчитися розпізнавати об’єкти, розташовані в різних частинах зображення;

– зсув – це метод, який передбачає розтягування зображення вздовж однієї з його осей і стиснення вздовж іншої осі, це допомагає моделі навчитися розпізнавати спотворені об’єкти.

Друга група методів включає в себе наступні:

– шум – це метод, який передбачає додавання різноманітних «дефектів» на зображення, що допомагає моделі фокусуватися не на конкретних пікселях та їх послідовностях, а узагальнювати шаблони;

– різкість – це метод, який передбачає збільшення ступінь виразності кордону між двома ділянками зображення які отримали різні експозиції, це допомагає урізноманітнити шаблони одного і того ж типу об’єктів, їх обрисів;

– розмиття – це метод, який протилежний до різкості, він передбачає згладжування переходу між об’єктами, це допомагає моделі навчитися розпізнавати об’єкти, які не знаходять у фокусі;

– яскравість – це метод, який передбачає зміну кількості білого кожного пікселя зображення, це допомагає моделі враховувати той факт, що зображення зроблені при різних умовах світла;

– контраст – це метод, який передбачає зміну різниці між білим та чорним кольором, це допомагає моделі враховувати той факт, що зображення зроблені при різних умовах світла;

– насиченість – це метод, який передбачає зміну в меншу чи більшу сторону інтенсивність кольорів на зображенні, це допомагає моделі не будувати шаблони на конкретних кольорах.

а) б)

Рисунок 1.7 – Приклад простої аугментації: а) вихідне зображення, б) змінене зображення

Важливо вказати і на недоліки цієї техніки. Якщо дані недостатньо змінені, то модель отримує на вхід повторюванні дані, що приводить до перенавчання моделі. Це означає, що модель запам’ятає конкретні кейси замість вивчення закономірностей та зв’язків. Ще одним обмеженням є те, що обсяг варіацій все ж обмежений. Ця техніка може вводити варіації лише в межах діапазону вихідних даних. Одного лише розширення даних може бути недостатньо для покриття повного діапазону мінливості, з якою очікується зіткнення моделі в реальному світі. До обмежень можна додати той факт, що аугментація може бути затратною. І звісно ж надмірна аугментація може призвести до спотворення даних, що негативно вплине на продуктивність та точність моделі.

1.6 Існуючі рішення

За останній час було прикладено значних зусиль у напрямку розробки різноманітних моделей для ідентифікації та класифікації сміття. Багато з цих досліджень фокусуються на різних аспектах навколишнього середовища: від домашніх умов до вуличних сценаріїв і дикої природи. Наприклад, одне з досліджень звертає увагу на розпізнаванні сміття в домашніх умовах, здатному впізнавати різноманітні класи сміття зблизька [20]. А інше дослідження, наприклад, зосереджується на використанні технічних засобів, таких як безпілотні літальні апарати (БПЛА), для виявлення забруднень.

Велика кількість систем моніторингу та управління сміттям також стала можливою завдяки розвитку Internet of Things (IoT). Ці системи використовують невеликі пристрої з обмеженою обчислювальною потужністю, які можуть забезпечувати зв'язок між собою та навколишнім середовищем. Наприклад, одна з таких систем використовує ультразвукові датчики для визначення рівня сміття у сміттєвих контейнерах та спілкується з диспетчерською за допомогою GSM [21].

Однією з дійсно надихаючих систем є SpotGarbage – мобільний додаток, розроблений у 2016 році для ідентифікації сміття за допомогою глибокого навчання [22].

Як висновок, питання ідентифікації сміття залишається актуальним. Існують досить багато досліджень та прототипів систем у цій області. Доцільним є продовжувати розвивати застосування комп'ютерного зору на таких пристроях, як смартфони та IoT, для різнобічного застосування спроектованої моделі.

1.7 Постановка задачі

Взявши до уваги все вище описане, можна ствердно зазначити, що обрана предметна область є важливою для вивчення. Задача ідентифікації

сміття може стати першим кроком у процесі покращення добробут жителів міст, які страждають від забруднень побутовими відходами, а також позитивно вплинути на загальний стан навколишнього середовища.

Мета цієї роботи сфокусуватися на дослідженні найсучасніших технологій у напрямку Computer Vision, вивченні методики аугментації даних та прикласти зусилля для всебічного покращення роботи розпізнавання сміття. Загалом можна виділити наступні задачі:

– дослідження сучасних технологій глибоких нейронних мереж у сфері Computer Vision;

– дослідження методики аугментації;

– розширення кількості класів розпізнаваних об’єктів та їх систематизація;

– збільшення кількості даних за допомогою методики аугментації;

– дослідження сімейства моделей YOLO, їх порівняння та обрання кращої опції;

– використання моделі на мобільному пристрої.

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Фреймворки та бібліотеки

Поставлені задачі можна розділити на декілька великих груп. Для кожної з них були вибрані інструменти для ефективного вирішення спеціалізованих задач.

Перша група задач – робота з даними. Обрані дані представлені за допомогою двох форматів, а саме JSON (COCO формат) для опису зображень, та самі зображення у JPG форматі.

Для цього етапу була обрана мова програмування Python. Загалом вона має велику кількість бібліотек для роботи з зображеннями та текстовими файлами.

Для попередньої обробки даних, їх аналізу корисними є бібліотеки numpy, pandas для обчислювань, matplotlib для візуалізації даних, pycocotools для роботи з форматом COCO і звісно pillow чи imageio для відображення картинок. Окремою задачею стоїть аугментація даних. Для її вирішення можна використати бібліотеку imgaug та вбудовані можливості моделей сімейства YOLO. Після всіх вище перерахованих операцій, дані треба переформатувати у потрібну для навчання моделі структуру. В цьому допоможуть базові бібліотеки роботи з операційної системою та файлами *os* та shutil. Після цього дані потрібно розділити на навчальну, валідаційну та тестову вибірки. Для цього найкраще підходить бібліотека splitfolders.

Друга група задач – навчання моделі. На зараз компанія Ultralytics, яка займається розробкою 5 і 8 версій моделей YOLO пропонує зручний CLI (Command Line Interface – інтерфейс командного рядку) для роботи з їх моделями, що дуже спрощує взаємодію з моделлю, допомагає сфокусуватися на аспектах навчання замість вирішення технічних проблем.

Остання група задач – це перенесення моделі на мобільний пристрій. Для цього я планую використати Raspberry PI в якості мобільного пристрою,

додатково до цього знадобиться модуль камери, а з програмного забезпечення бібліотека libcamera.

2.2 Попередня обробка даних

TACO датасет складається з 1500 зображень та 4784 анотацій. TACO містить зображення високої роздільної здатності, як показано на рисунку 2.1, що зняті переважно мобільними телефонами.

Ці анотації розділені на категорії, яких 60 звичайних та 28 супер-категорій, включаючи спеціальну категорію: немаркований сміття для об’єктів, які або неоднозначні, або не охоплені іншими категоріями [18]. Як можна побачити на рисунку 2.2 а) для більш, ніж половини звичайних категорій не набирається і сотні анотацій на категорію, що досить мало даних для навчання моделі. Якщо подивитися на другий графік, то видно, що з супер-категоріями ситуація краще, це не дивно, бо вони виконують функцію агрегації звичайних категорій. В подальшому ми будемо працювати більше з супер-категоріями. Потенційно варто також зменшити і їх кількість.

На рисунку 2.3 можна побачити перші 10 категорій за кількістю анотацій та їх розміри. Видно, що більшість об’єктів класу цигарки (3 за кількістю клас) має розмір менший за 64 на 64 пікселі, що може бути проблемою для більшості моделей, в тому числі і для YOLO.

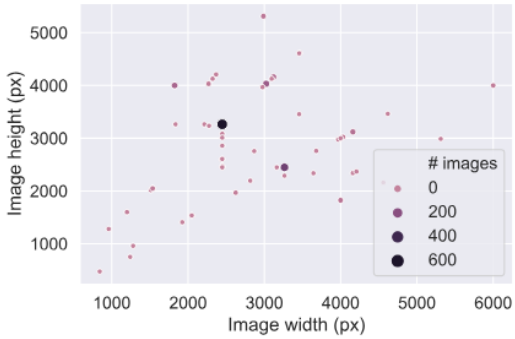


Рисунок 2.1 – Графік кількості зображень за їх розміром

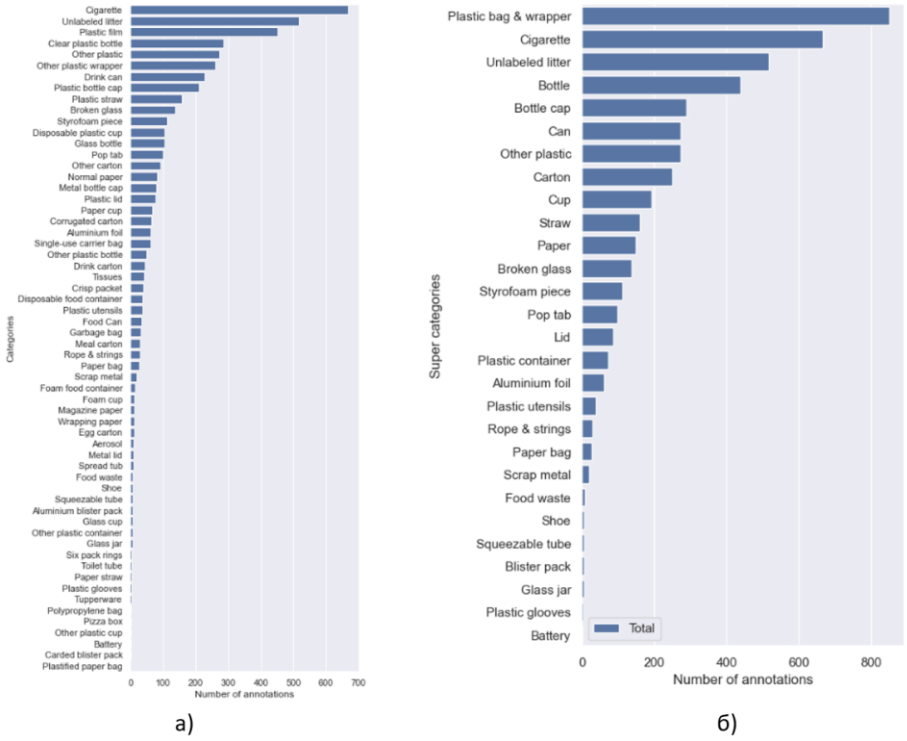


Рисунок 2.2 – Графік кількості анотацій: а) по категоріям, б) по супер-категоріям

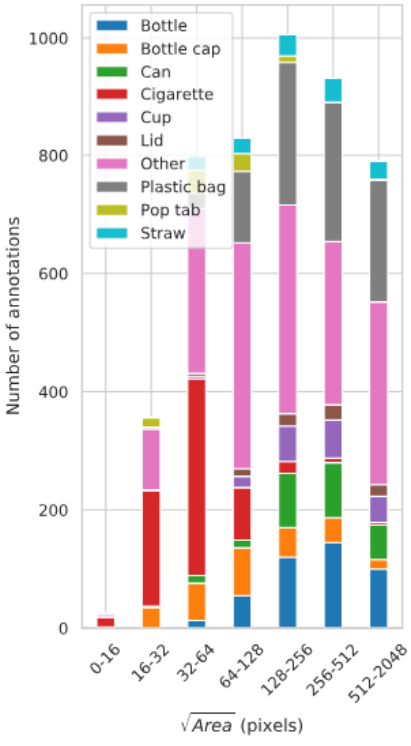


Рисунок 2.3 – Графік кількості топ 10 категорій за площею на зображеннях

2.3 Аугментація даних

Під час проектування системи, а зокрема кроку попередньої обробки даних важливо врахувати особливості даних. Розглянемо обраний датасет. В ньому використовуються Bounding Boxes для маркування об’єктів для подальшого розпізнавання. Це, наприклад, накладає деякі обмеження на методи аугментації, які можна застосувати до зображень. Існує проблема обертання зображення на 45 градусів [23]. Коли це відбувається із зображенням із обраним методом маркування, то ці прямокутники фізично не можуть повернутися за зображенням і починають містити некорисну інформацію про об’єкт. Варто звернути увагу на те, що в сімействі YOLO також використовується технологія мозаїки, коли на вхід до моделі потрапляє не одне зображення, а декілька скомбінованих в одне. На це теж треба звертати увагу під час обрання методів аугментації зображень.

Нажаль, через те, що на зображені зазвичай знаходиться більше одного об’єкту та різних категорій, неможливо за допомогою аугментації вирівняти кількість анотацій для кожної категорій, тому навіть після застосування такої техніки, буде зберігатися нерівномірність даних.

Звертаючи увагу на всі вище перелічені особливості датасету, варто звернути увагу на наступні методи аугментації. Більшість засобів з групи корегування зображення, такі як додавання шуму, зміну контрасту, яскравості, насиченості, різкості, розмиття та інші. Головне не використовувати занадто екстремальні значення, бо це може призвести до того, що зображення перестане нести корисну інформацію, буде занадто темним, чи світлим, занадто розмитим чи різким, з великою кількістю шуму і зовсім буде відрізнятися від того, що може отримати користувач під час використання камери мобільного пристрою. Також можна використати метод перегортання з групи зміни форми зображень, деякі методи з цієї групи реалізовані в алгоритмі YOLO.

Методику аугментації під назвою Mosaic було вперше представлено в YOLOv4. ЇЇ ідея дуже проста. Беруться 4 зображення та об’єднуються в одне. Mosaic робить це шляхом зміни розміру кожного з чотирьох зображень, зшивання їх разом, а потім випадкового вирізання з’єднаних зображень, щоб отримати остаточне зображення на вхід до моделі.

Загалом описаний алгоритм складається з наступних кроків (рисунок 2.4):

– змініть розмір зображень, тут також може бути застосовані інші методи обробки зображення, як от поворот, масштабування, обрізання, зсув, при цьому зміни відбуваються і з обмежувальними рамками;

– об’єднайте всі зображення в одне зображення;

– розмістіть обмежувальні рамки в правильних областях нового зображення.

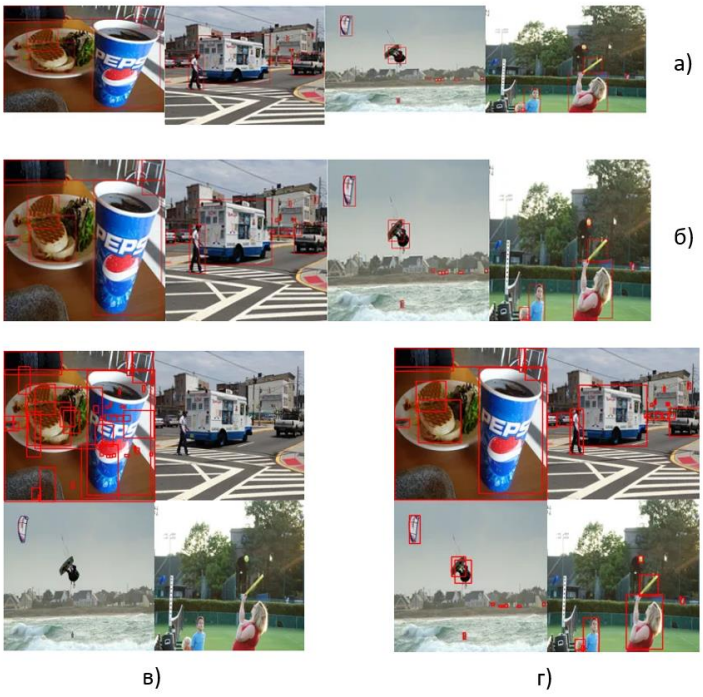


Рисунок 2.4 – Застосування методу Mosaic: а) вихідні зображення, б) змінені зображення, в) об’єднанні зображення, г) зображення з правильними обмежувальними рамками

2.4 Сімейство моделей YOLO

Сімейство моделей комп'ютерного зору YOLO – це нейронні мережі, яка визначають положення обмежувальних прямокутників (Bounding Boxes) і ймовірності класифікації для зображення за один прохід. Простими словами, це метод розпізнавання об'єктів на фотографіях у реальному часі.

Модель стала широко відомою після того, як її розробники Джозеф Редмон та інші, представили нову архітектуру на конференції CVPR Computer Vision and Pattern Recognition Conference в 2016 році [24] і навіть отримали за неї нагороду OpenCV People Choice Awards.

YOLO була першою мережею виявлення об'єктів, яка об'єднала завдання малювання обмежувальної рамки (Bounding Boxes) та ідентифікації міток класу в одну наскрізну диференційовану мережу.

Методи розпізнавання, засновані на глибокому навчанні, можна розділити на дві групи: перша включає двоетапні алгоритми виявлення, які роблять багатоетапне передбачення (наприклад, RCNN, Fast-RCNN Fast-RCNN та інші), друга група детекторів називаються одноступеневими детекторами (наприклад, SSD, EfficientDet і звісно YOLO).

Хоча YOLO не єдина одноетапна модель виявлення, вона, як правило, ефективніша за інші з точки зору швидкості та точності. Зрештою, якщо розглядати проблему виявлення як одноетапний регресійний підхід для визначення обмежувальної рамки, моделі YOLO часто дуже швидкі та дуже маленькі, що робить їх швидшими для вивчення та легшими для розгортання, особливо на пристроях з обмеженими обчислювальними ресурсами.

Основна суть моделі не міняється з її першої версії. Розглянемо всі покращення, які внесли наступні версії цієї моделі (рисунок 2.5).

YOLOv1 була першою офіційною моделлю YOLO. Вона використовувала одну згорткову нейронну мережу (CNN) для виявлення об'єктів на зображенні та була відносно швидкою порівняно з іншими

моделями розпізнавання об’єктів. Однак вона не була такою точною, як деякі двоступеневі моделі того часу.

Рисунок 2.5 – Часова шкала версій моделей сімейства YOLO [27]

YOLOv2 була випущена у 2016 році та внесла кілька покращень порівняно з першою версією. Вона використовувала блоки прив’язки для підвищення точності виявлення, було додано шар Upsample, який покращив роздільну здатність вихідної карти функцій.

YOLOv3 була представлена у 2018 році з метою підвищити точність та швидкість алгоритму. Основним удосконаленням YOLOv3 порівняно з попередніми версіями було використання архітектури Darknet-53, варіанту архітектури ResNet, спеціально розробленого для виявлення об’єктів [25].

Покращення YOLOv3 на цьому не завершуються. У YOLOv3 також були покращена робота з блоками прив’язки (anchor boxes), що дозволило працювати з різними масштабами та співвідношеннями сторін, краще відповідати розміру та формі виявлених об’єктів. Вперше використано FPN – Feature Pyramid Network, функції втрат GHM – Gradient Harmonizing

Mechanism R. Всі ці зміни позитивно вплинули на точність і стабільність YOLOv3.

YOLOv4 була представлена у 2020 році з низкою вдосконалень, що включали нову магістральну мережу, покращення процесу навчання та збільшення потужності моделі. YOLOv4 також було розширено новим методом нормалізації Cross Mini-Batch Normalization, який призначений для підвищення стабільності процесу навчання.

YOLOv5 була представлена так само у 2020 році. Ця модель спирається на успіх попередніх версій і була випущена Ultralytics, як проект із відкритим кодом. YOLOv5 використовувала архітектуру EfficientDet, засновану на мережі EfficientNet [26], а також кілька нових функцій і вдосконалень для досягнення покращеної продуктивності виявлення об’єктів. У 2020 році YOLOv5 став найсучаснішим у світі репозиторієм для виявлення об’єктів завдяки своїй гнучкій структурі.

У YOLOv6 зосередилися на тому, щоб зробити систему більш ефективною та зменшити обсяг пам’яті. Нова модель використовувала архітектуру CNN під назвою SPP-Net (Spatial Pyramid Pooling Network). Ця архітектура розроблена для роботи з об’єктами різних розмірів і пропорцій, що робить її ідеальною для завдань виявлення об’єктів.

Модель YOLOv7 була представлений у 2022 році. Одним із ключових удосконалень є використання нової архітектури CNN ResNeXt. Також ця модель використовує нову стратегію навчання, яка передбачає навчання моделі на зображеннях у кількох масштабах з подальшим об’єднанням прогнозів. Це допомагає моделі ефективніше обробляти предмети різних розмірів і форм. Більш того YOLOv7 містить нову техніку під назвою Focal loss, яка розроблена для вирішення проблеми дисбалансу класів, яка часто виникає в задачах виявлення об’єктів. Функція Focal Loss надає більшої ваги важким прикладам і зменшує вплив простих прикладів.

Протягом останнього часу Ultralytics наполегливо працювали над дослідженням і створенням найновішої версії YOLO, YOLOv8. YOLOv8

було запущено 10 січня 2023 року. Зараз на їх сайті вже існують матеріали щодо YOLOv9. Команда Ultralytics ще провела порівняння YOLOv8 із набором даних COCO та досягла гарних результатів порівняно з попередніми версіями YOLO для всіх п’яти розмірів моделей (рисунок 2.6).

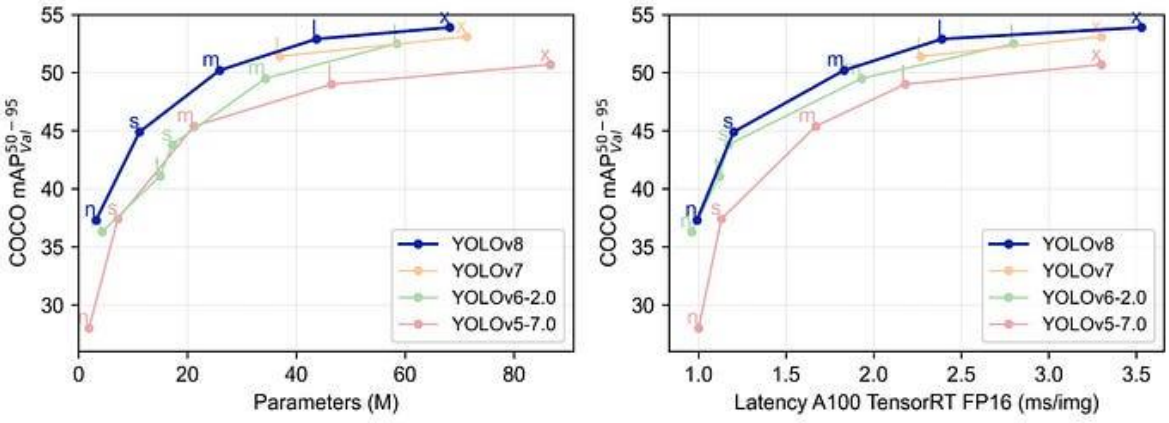


Рисунок 2.6 – Графік порівняння моделей YOLO [28]

2.5 Архітектура YOLOv8

У YOLOv8 ще немає опублікованої статті, тому все ж бракує прямого уявлення про методологію дослідження під час його створення. З огляду на це було проаналізовано репозиторій і доступну інформацію про модель.

На рисунку 2.8 можна побачити деталізовану архітектуру мережі YOLOv8.

Архітектура YOLOv8 має структуру, що складається з трьох основних частин: хребет, шия та голова:

– хребет – це згортковий шар, який об'єднує ознаки зображення в різних масштабах;

– шия – це кілька з'єднаних шарів, які обробляють дані за змістом та передають їх для прогнозування в шар голови;

– голова – шар, який отримує ознаки від шийних шарів та виконує локалізацію та класифікацію об'єктів.

Важливі елементи алгоритму YOLOv8.

Функції активації. Використовуються дві основні функції активації: Сігмоїд та SiLU (Sigmoid Linear Unit – сигмоподібна лінійна одиниця). Сігмоїд функція активації використовується у фінальному шарі для класифікації. Функція SiLU застосовується у прихованих шарах мережі.

SiLU – це поєднання ReLU та Сігмоїду (рисунок 2.7). Вона має наступне рівняння (формула 2.1) [29]:

$$\text{silu}(x) = x * \sigma(x), \tag{2.1}$$

де $\sigma(x)$, – це функція сигмоїду від x.

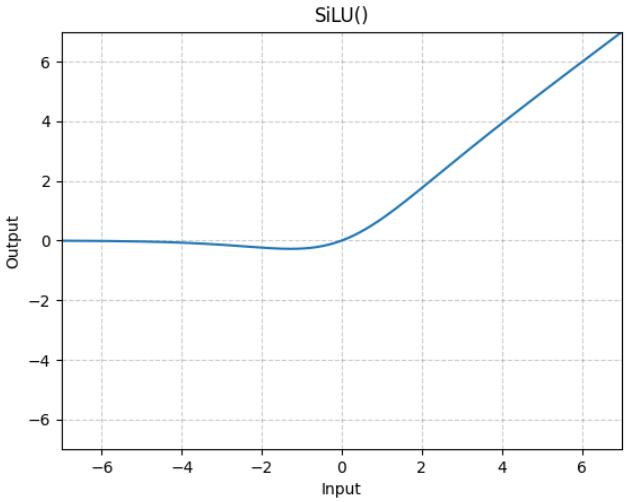


Рисунок 2.7 – SiLU візуалізація

Функція оптимізації. Тут використовується алгоритм ADAM, який себе гарно зарекомендував своєю ефективністю, гнучкістю (через параметри, що настроюються автоматично) та універсальністю.

Функції втрат. Використовується бінарна перехресна ентропія (Binary Cross Entropy – BCE) для класифікації, а також Complete IoU (Intersection Over Union) та Distribution Focal Loss (DFL) для обмежувальних рамок.

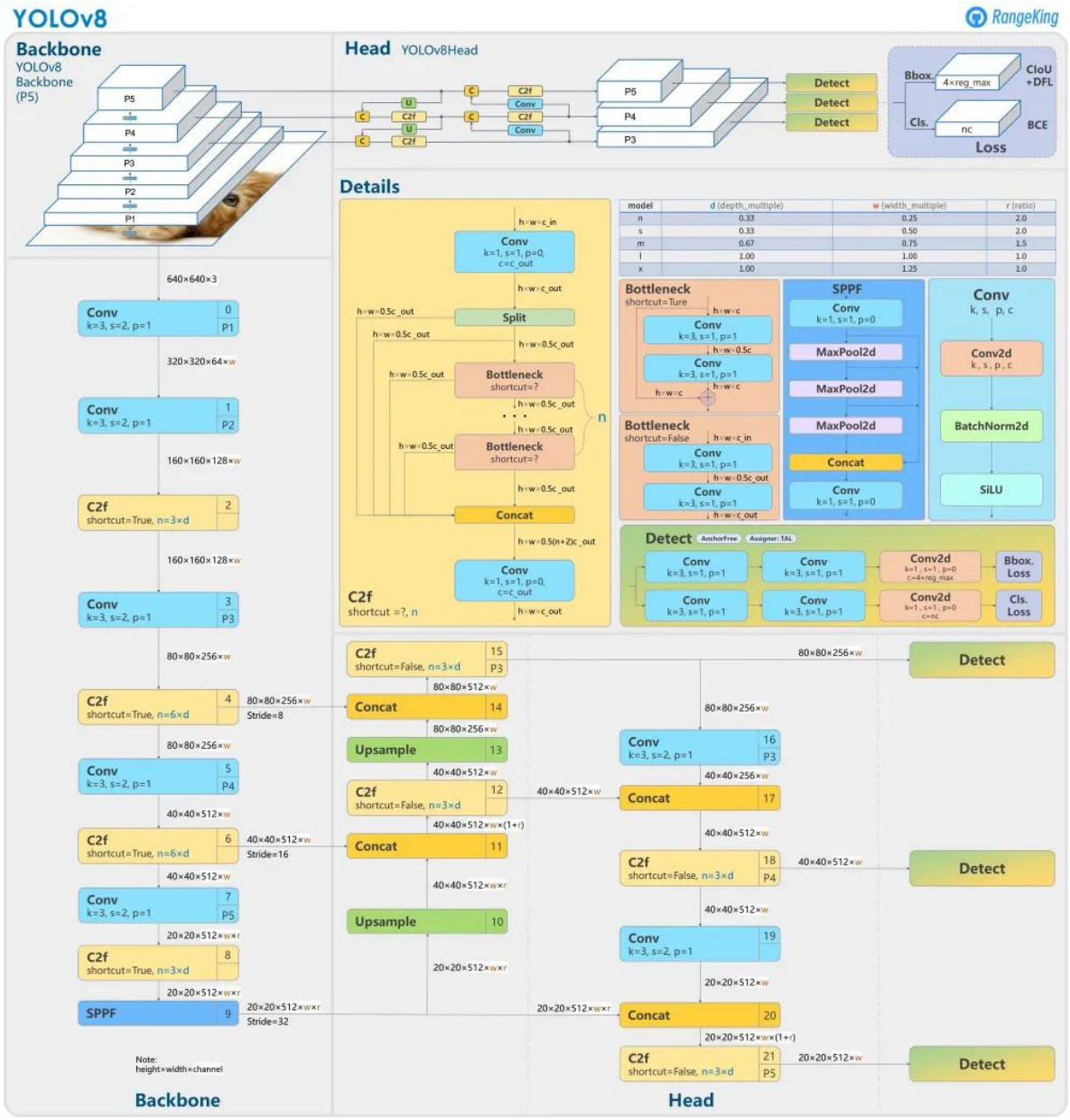


Рисунок 2.8 – Детальна візуалізація архітектури мережі

Розглянемо покращення, які з’явилися у YOLOv8 версії алгоритмів в порівнянні з YOLOv5.

Виявлення без прив’язки. Це означає, що модель безпосередньо прогнозує центр об’єкта замість зміщення від відомого вузла прив’язки.

Блоки прив’язки були складною частиною попередніх моделей YOLO. Блоки прив’язки – це попередньо визначений набір блоків із певною

висотою та шириною, які використовуються для виявлення класів об’єктів із бажаним масштабом і співвідношенням сторін. Вони вибираються на основі розміру об’єктів у навчальному наборі даних і розміщуються на зображенні під час виявлення.

Мережа виводить ймовірність і атрибути, як фон, IoU та зміщення для кожного блоку, які використовуються для коригування прив’язок. Для різних розмірів об’єктів можна визначити кілька блоків прив’язки, які служать фіксованими початковими точками для припущень рамки межі.

Виявлення без прив’язки зменшує кількість блоків, які було розпізнано, що прискорює немаксимальне придушення (NMS – Non-Maximum Suppression), складний етап постобробки, який просіює кандидатів, щоб не дублювати результати. На рисунку 2.9 можна побачити різницю підходів до процесу розпізнавання об’єктів у YOLOv5 та YOLOv8.

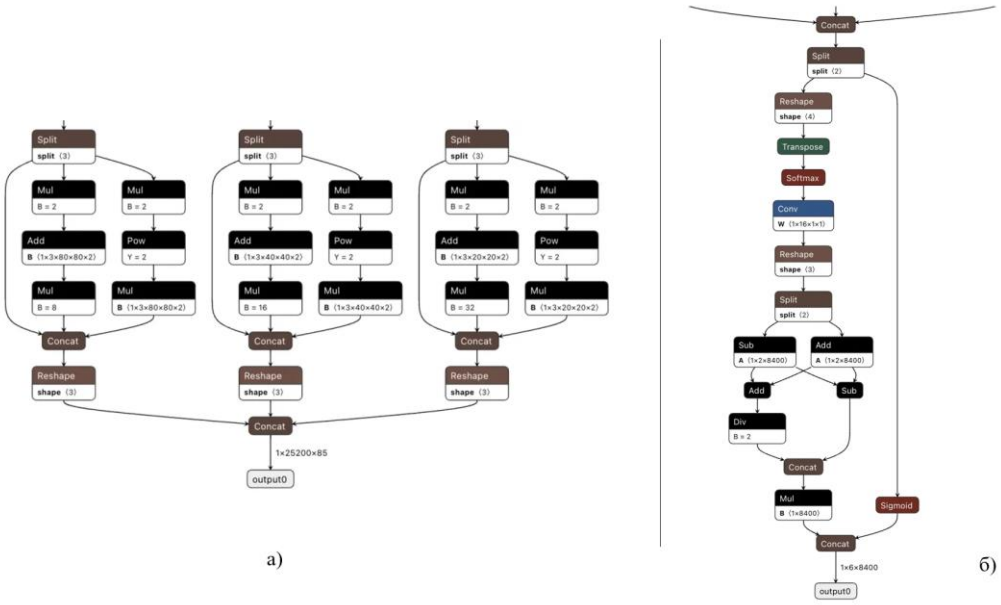


Рисунок 2.9 – Візуалізація «голови» розпізнавання: а) YOLOv5, б) YOLOv8

Також при порівнянні YOLOv5 та YOLOv8 видно, що відбулися зміни в архітектурі хребта. Перші дві згортки з розміру 6×6 змінилися на 3×3. Так само змінився і основний будівельний блок, який мав назву C3, а тепер

називається C2f (рисунок 2.10). Модуль показано на рисунку 2.X нижче, де «f» – кількість функцій, «e» – швидкість розширення, а CBS – це блок, що складається з Conv (згортки), BatchNorm (нормалізації) і SiLU (функція активації) [30].

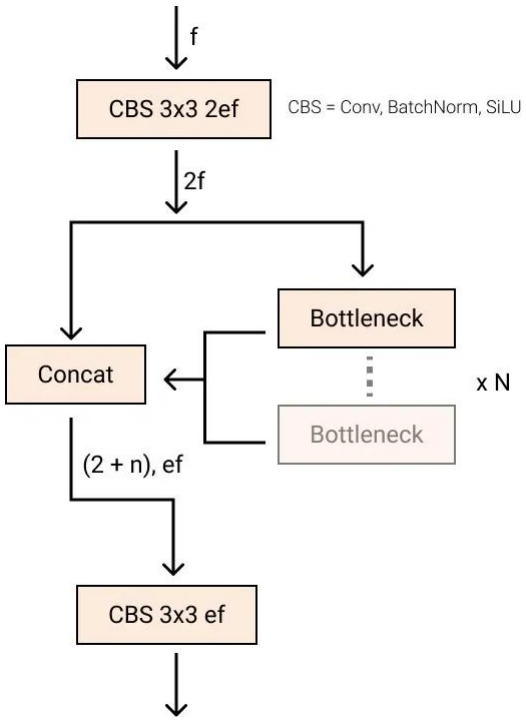


Рисунок 2.10 – Новий YOLOv8 C2f модуль

Головна відмінність полягає в тому, що у C2f усі виходи з Bottleneck (назва для двох згорток 3×3 із залишковими з'єднаннями) об'єднані. Тоді як у C3 використовувався лише вихід останнього Bottleneck. Це значить, що на виході зберігається більше корисної інформації.

Варто зазначити, що Bottleneck таке ж, як і в YOLOv5, але розмір ядра першої згортки змінено з 1×1 на 3×3. З цієї інформації ми бачимо, що YOLOv8 починає повертатися до блоку ResNet,

У «шиї» деталі з'єднуються безпосередньо без приведення до однакових розмірів каналу. Це зменшує кількість параметрів і загальний розмір тензорів. Особливості виділяються на кожному масштабі окремо, спрощуючи цим цей етап.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Обробка даних

Видалення категорій. Як вже зазначалося вище, в обраному датасеті досить велика кількість категорій, які поєднані в супер-категорії. Загалом з 28 супер-категорій залишилося 12 достатньо великих категорій (більше 200 анотацій), список яких можна побачити на рисунку 3.1.

```
# Classes
names:
  0: aluminium foil
  1: bottle
  2: bottle cap
  3: can
  4: cup
  5: lid
  6: plastic bag & wrapper
  7: plastic container
  8: pop tab
  9: styrofoam piece
  10: unlabeled litter
  11: cigarettes
```

Рисунок 3.1 – Список категорій після їх зменшення

Наступним кроком є аугментація даних. Для цього етапу була використана бібліотека imgaug. Ця бібліотека Python допоможе з аргументацією зображень для проектів машинного навчання. Вона перетворює набір вхідних зображень у новий, набагато більший набір дещо змінених зображень. Вона має широкі можливості роботи із зображеннями, їх видозміною, та накладанням змін з досить тонкими налаштуваннями, при цьому гарно справляється зі зображеннями, що мають обмежувальні рамки [31].

Через те, що в YOLOv8 використовується технологія Mosaic (рисунок 3.4), я не видозмінював форму зображень, а зупинився тільки на змінах коригування. На рисунку 3.2 можна побачити код функції, що застосовувався до зображень. Зміни включають коригування контрасту, додавання шуму, додавання різних видів розмиття, роботу з різними каналами кольору. Загалом вийшло збільшити датасет в двічі. Також ця функція включає просту імплементацію повторної спроби застосувати фільтри через те, що цей крок не завжди успішно завершувався.

```
def augment_image(image_data, img_ann):
    image = imageio.v2.imread(base_img_path + image_data['file_name'])
    image = ia.imresize_single_image(image, (image_data['height'], image_data['width']))

    img_bbs = []
    for a in img_ann:
        box = a['bbox']
        img_bbs.append(BoundingBox(x1=box[0], y1=box[1],
            x2=box[0]+box[2], y2=box[1]+box[3], label=a['category_id']))
    bbs = BoundingBoxesOnImage(img_bbs, shape=image.shape)

    seq = iaa.Sequential([
        iaa.GammaContrast((0.5,1.8)),
        iaa.LinearContrast((0.5, 1.5)),
        iaa.AdditiveGaussianNoise(loc=0, scale=(0.0, 0.05*255), per_channel=0.5),
        iaa.Multiply((0.8, 1.3), per_channel=0.2),
        iaa.Sometimes(
            0.5,
            iaa.OneOf([
                iaa.GaussianBlur((0, 2.5)),
                iaa.AverageBlur(k=(2, 6)),
                iaa.MedianBlur(k=(1, 7)),
            ])
        ),
    ], random_order=True)

    tries = 0
    max_tries = 10
    while tries < max_tries:
        try:
            image_aug, bbs_aug = seq(image=image,bounding_boxes=bbs)
            bbs_result = []
            for b_a in bbs_aug.bounding_boxes:
                bbs_result.append(b_a.get_cords())
            return {
                "raw_image": image_aug,
                "raw_bbs": bbs_aug,
                "bbs": bbs_result
            }
        except Exception as e:
            tries = tries + 1
```

Рисунок 3.2 – Лістинг коду аугментації зображення

Загалом робота із даними цього датасету полягає в одночасному опрацюванні зображень та json файлу з анотаціями, в якому зберігається важлива метадані. На рисунку 3.3 можна побачити, як описана вище функція аугментації застосовується, а також оновлюються дані в файлі з анотаціями.

```
anns = data['annotations']
images = data['images']

max_img_id = max(images, key=lambda ev: ev['id'])['id']
ann_len = len(anns)

batch_number = 16
counter = 0
new_ann_id = 4783

for i in range(max_img_id):
    if(counter == 100):
        counter = 0
        batch_number += 1

    if not os.path.exists(base_img_path + f"batch_{batch_number}"):
        os.makedirs(base_img_path + f"batch_{batch_number}")

    if batch_number > 30:
        break
    image = images[i]
    image_id = image['id']
    new_image_id = max_img_id + i + 1
    image_anns = []
    for a in range(0, ann_len):
        if(anns[a]['image_id'] == image_id):
            new_ann_id = new_ann_id + 1
            image_ann = anns[a]
            image_ann['id'] = new_ann_id
            image_anns.append(image_ann)
    new_image = {
        "id": new_image_id,
        "width": image['width'],
        "height": image['height'],
        "file_name": f"batch_{batch_number}/{counter:06d}.jpg",
    }
    data['images'].append(new_image)

    result = augment_image(image, image_anns)


    for bb in image_anns:
        new_ann_id = new_ann_id + 1
        new_ann = {
            "id": new_ann_id,
            "image_id": new_image_id,
            "category_id": bb['category_id'],
            "bbox": bb['bbox']
        }
        data['annotations'].append(new_ann)
    imageio.imwrite(base_img_path + new_image['file_name'], result['raw_image'])
    print(counter, batch_number, new_image_id, image_anns)
    counter += 1
file_path = "output.json"
with open(file_path, 'w') as json_file:
    json.dump(data, json_file, indent=2)
```

Рисунок 3.3 – Лістинг коду роботи з зображеннями та анотаціями під час аугментації даних

Рисунок 3.4 – Приклад аугментації вбудованими інструментами YOLO

Останнім кроком роботи з даними є приведення їх до структури, якої очікує на вхід модель. На рисунку 3.6.а можна побачити вихідну структуру датасету та очікувану моделлю на рисунку 3.6.б. Для реалізації цієї трансформації було використано вбудовані можливості мови програмування Python та os, shutil, numpy, tqdm та split-folders бібліотеки. Лістинг коду трансформації представлено на рисунку 3.5.

```python
for index, img_id in tqdm.tqdm(enumerate(img_ids), desc='change .json file to .txt file'):
    img_info = data_source.loadImgs(img_id)[0]
    save_name = img_info['file_name'].replace('/', '_')
    file_name = save_name.split('.')[0]

    height = img_info['height']
    width = img_info['width']

    save_path = save_base_path + file_name + '.txt'
    is_exist = False
    with open(save_path, mode='w') as fp:
        annotation_id = data_source.getAnnIds(img_id)
        boxes = np.zeros((0, 5))
        if len(annotation_id) == 0:
            fp.write('')
            continue

        annotations = data_source.loadAnns(annotation_id)
        lines = ''

        for annotation in annotations:
            label = coco_labels_inverse[annotation['category_id']]
            if label in label_transfer.keys():
                is_exist = True
                box = annotation['bbox']
                if box[2] < 1 or box[3] < 1:
                    continue

                box[0] = round((box[0] + box[2] / 2) / width, 6)
                box[1] = round((box[1] + box[3] / 2) / height, 6)
                box[2] = round(box[2] / width, 6)
                box[3] = round(box[3] / height, 6)
                label = label_transfer[label]
                if label not in class_num.keys():
                    class_num[label] = 0
                class_num[label] += 1
                lines = lines + str(label)

                for i in box:
                    lines += ' ' + str(i)
                lines += '\n'
        fp.writelines(lines)
    if is_exist:
        shutil.copy('C:\\study\\23-24\\diploma\\TACO\\data\\train\\{}'.format(img_info['file_name']),
                    os.path.join(save_image_path, save_name))
    else:
        os.remove(save_path)
```

Рисунок 3.5 – Лістинг коду трансформації датасету

```
TACO
├── annotations
│   └── annotations.json
└── data
    ├── batch_1
    │   ├── 000000.JPG
    │   ├── ...
    │   └── 000100.JPG
    ├── ...
    └── batch_15
```

а)

```
TACO
├── test
│   ├── images
│   │   ├── batch_1_000010.JPG
│   │   ├── ...
│   │   └── batch_30_000080.JPG
│   └── labels
│       ├── batch_1_000010.txt
│       ├── ...
│       └── batch_30_000080.txt
├── train
└── val
```

б)

Рисунок 3.6 – Структура даних: а) вихідна, б) очікувана моделлю на вхід

3.2 Навчання моделі

Навчання CNN моделі досить кропіткий та затратний процес. Загалом він потребує великих обсягів даних та обчислювальних ресурсів. На моменти роботи над проектом в якості обчислювального засобу могла бути використана тільки робоча станція. Загалом робочий комп'ютер має 11th Gen Intel Core(TM) i7-1185G7 3.00GHz процесор з вбудованим GPU та 32 гб оперативної пам'яті. Загалом дана конфігурація може гарно підійти тільки для навчання невеликих моделей з невеликими датасетами. І все одно це буде досить довго. Але враховуючи той факт, що модель має запускатися на мобільній платформі Raspberry, яка в свою чергу теж не дуже потужна, то нам і не потрібна дуже велика модель. Ми можемо вдовольнитися тими обчислювальними ресурсами, які є в наявності.

Приклад команди запуску навчання моделі представлений на рисунку 3.7.

```
а) yolo train
б)     data=trash_detection.yaml model=C:/study/23-24/diploma/runs/detect/320_n_500/weights/last.pt name=320_n_600
в)     epochs=100 batch=8 imgsz=320  optimizer=Adam
г)     degrees=180 scale=0.5 shear=90 fliplr=0.5
```

Рисунок 3.7 – Приклад запуску навчання моделі з гіперпараметрами: а) основна команда, б) джерела даних та назва запуску, в) кількість епох, кількість зображень у пакеті, розмір зображень та тип оптимізатора, г) параметри аугментації даних

Для навчання було обрано модель маленького розміру (S – Small), як найбільш оптимальну для поставлених задач, враховуючий той факт, що запускатися розпізнавання буде на мобільному пристрої. Більш того, наявні ресурси і не дозволять ефективно навчати модель більшого розміру. На рисунку 3.7 можна побачити технічний вигляд моделі. Звісно, він дуже

схожий на те, що було представлено схематично на рисунку 2.8 з поправкою на розмір моделі.

На рисунку 3.8 видно, що модель складається з 23 елементів, загалом має 225 шарів, 11140244 параметрів, 11140228 градієнтів.

```
                 from  n    params  module                                       arguments
  0                -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
  1                -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                -1  1     29056  ultralytics.nn.modules.block.C2f             [64, 64, 1, True]
  3                -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  4                -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  5                -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  6                -1  2    788480  ultralytics.nn.modules.block.C2f             [256, 256, 2, True]
  7                -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                -1  1   1838080  ultralytics.nn.modules.block.C2f             [512, 512, 1, True]
  9                -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11            [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                -1  1    591360  ultralytics.nn.modules.block.C2f             [768, 256, 1]
 13                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14            [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 16                -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 17           [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 19                -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 20            [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                -1  1   1969152  ultralytics.nn.modules.block.C2f             [768, 512, 1]
 22      [15, 18, 21]  1   2120692  ultralytics.nn.modules.head.Detect           [12, [128, 256, 512]]
Model summary: 225 layers, 11140244 parameters, 11140228 gradients, 28.7 GFLOPs
```

Рисунок 3.8 – Схематичне зображення моделі YOLOv8s

1.3 Порт моделі на мобільний пристрій

В якості мобільного пристрою було розглянуто декілька варіантів: Android пристрій, iPhone та Raspberry PI. Загалом всі перелічені пристрої варті уваги, бо мають потрібні параметри, такі як достатня обчислювальна потужність та модуль камери.

Вибір був зроблений на користь Raspberry PI, бо це ідеальна мобільна платформа для експериментів з модульним підходом, що робить її дуже гнучкою. Плюсом також є простота роботи з моделлю, бо в порівнянні з Android та iOS має операційну систему Linux, яка без проблем запускає Python код, а тим більше без проблем працює з C++. Це в свою чергу дозволяє запускати модель у більш оптимізованому форматі ONNX. А якщо, наприклад, говорити про iOS, то на цій операційній системі потрібно

конвертувати файл навченої моделі в спеціальний формат, що підтримується Apple [32].

Загалом для перенесення моделі нам потрібно наступне:

– мобільний пристрій Raspberry PI 4;
– камера PI camera;
– 64-бітна Raspberry Pi операційна система та libcamera бібліотека.

Також корисними будуть така периферія, як монітор, блок живлення, картка пам’яті, клавіатура та миша без яких розробка на Raspberry PI або утруднена, або неможлива.

Налаштування Raspberry PI для роботи з YOLOv8. В порівнянні з YOLOv5 все стало ще більш простіше через CLI. Для початку роботи треба відтворити наступні кроки.

Для початку треба оновити Raspberry PI (рисунок 3.9.а).

Далі треба встановити потрібні пакети (рисунок 3.9.б).

Після цього необхідно перезавантажити систему. Далі треба запустити та налаштувати TCP-потік з камерою (рисунок 3.9.в).

Для перевірки того, що все працює, запустити код (рисунок 3.10) [33]. На цьому лістингу можна побачити, що з моделями YOLO можна працювати як через CLI, так і за допомогою мови програмування Python, що може бути зручно, наприклад, під час розпізнавання об’єктів з одночасним виводом результатів роботи модель в окремому вікні.

а)
```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get autoremove -y
```

б)
```
pip3 install opencv-contrib-python
pip3 install ultralytics
```

в)
```
libcamera-vid -n -t 0 --width 1280 --height 960 --framerate 1 --inline --listen -o tcp://127.0.0.1:8888
```

Рисунок 3.9 – Лістинг коду: а) оновлення Raspberry PI, б) інсталяція OpenCV та Ultralytics пакетів, в) запуск камери

```python
from ultralytics import YOLO

model = YOLO('yolov8n.pt')
results = model('tcp://127.0.0.1:8888', stream=True)

while True:
    for result in results:
        boxes = result.boxes
        probs = result.probs
```

Рисунок 3.10 – Лістинг коду для перевірки роботи моделі

Важливо зазначити, що мобільна платформа Raspberry PI 4 не дуже потужна, якщо порівнювати її з сучасними комп'ютерами, які мають дискретні GPU, а тим більше TPU. По цій причині треба максимально оптимізувати роботу моделі на цій платформі. Для цього можна використати ONNX. ONNX – це відкритий формат, створений для представлення моделей машинного навчання. Він визначає загальний набір операторів – будівельних блоків моделей машинного та глибокого навчання, і загальний формат файлу, який дозволяє використовувати моделі з різними фреймворками, інструментами, середовищами виконання та компіляторами [34].

ONNX формат має кращу оптимізацію для запуску моделей на CPU, ніж вихідна модель у форматі PyTorch. На рисунку 3.9 можна побачити приклад лістингу коду для запуску моделі у форматі ONNX. Зручно, що під час запуску коду потрібні бібліотеки будуть автоматично встановлені.

Для наочності порівняємо швидкість роботи моделей, які представлені в однаковій конфігурації (таблиця 3.1).

```python
from ultralytics import YOLO
import cv2

model = YOLO('yolov8s.pt')
model.export(format='onnx')
onnx_model = YOLO('yolov8s.onnx')

results = onnx_model('tcp://127.0.0.1:8888', stream=True)

for result in results:
    annotated_frame = result.plot()
    cv2.imshow("YOLOv8 Inference", annotated_frame)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

cv2.destroyAllWindows()
```

Рисунок 3.9 – Лістинг коду запуску моделі у форматі ONNX

Таблиця 3.1 – Порівняння часу розпізнавання об’єктів моделлю, що представлена в різних форматах та розмірах

Формат	Час виконання	
	Розмір Nano	Розмір Small
PyTorch	1523 мс	3400 мс
ONNX	914 мс	1036 мс

Спираючись на емпіричні дані, що занесені в таблицю 3.1, можна зробити висновок, що виконання розпізнавання об’єктів моделі у форматі ONNX відбувається на ~40% швидше для Nano розміру моделі, та на ~70% швидше для Small розміру, ніж у звичайному PyTorch форматі такого ж розміру. Якщо порівняти час виконання різних розмірів моделей у форматі ONNX, то отримаємо, що різниця у швидкості виконання ~10%, що спонукає до використання більшої моделі при невеликій втраті в ефективності.

Ще однією потенційною можливістю збільшити ефективність моделі є покращення системи охолодження платформи Raspberry PI 4. За замовчуванням її немає. Можливі рішення включають додавання радіаторної решітки чи додавання системи на базі вентиляторів. Цей крок дозволяє

працювати процесору на більшій частоті та не зменшувати її задля регулювання температури.

1.4 Результати роботи моделі

За результатами проведеної роботи було отримано модель згорткової нейронної мережі. Навчання моделі проводилося локально з використанням наявних ресурсів, та з використанням сервісу хмарних обчислень Azure Machine Learning.

Загалом було запущено 300 епох, що є бажаним мінімумом для моделей сімейства YOLO [35], ще зайняло близько 7 діб навчання. Результати навчання (останні 50 епох), інші метрики можна побачити на рисунку 3.10. Також цікавим може бути графік PR-curve по категоріям на рисунку 3.11. За графіком можна побачити, що з деякими категоріями об’єктів модель справляється значно краще, аніж з іншими. На це безпосередньо впливає кількість об’єктів за категорією у вихідній вибірці. Той самий результат можна побачити на рисунку 3.12, де перераховано параметри mAP50 та mAP95 для кожної з категорій окремо.

Навіть не зважаючи на покращення моделі YOLOv8 щодо роботи з малими об’єктами, використанням середнього розміру зображення (640 × 640 пікселів) та порівняльно великій кількості об’єктів (топ 3 в датасеті), з категорією «цигарки», модель справляється погано. Загалом це можна звести до того, що малі предмети несуть менше корисної інформації щодо їх форми та інших параметрів. Це дуже ускладнює роботу моделі щодо їх розпізнавання та класифікації. Більш того негативно впливає на загальні метрики моделі (рисунок 3.13).

В якості мобільного пристрою була обрана платформа Raspberry PI завдяки її компактності, універсальності і зручності використання. Враховуючі можливості системи, було обрано розмір моделі S (Small), це другий знизу розмір моделі після N (Nano). Більш того для оптимізації

роботи моделі було використано формат ONNX, що значно прискорив виконання задач на CPU.

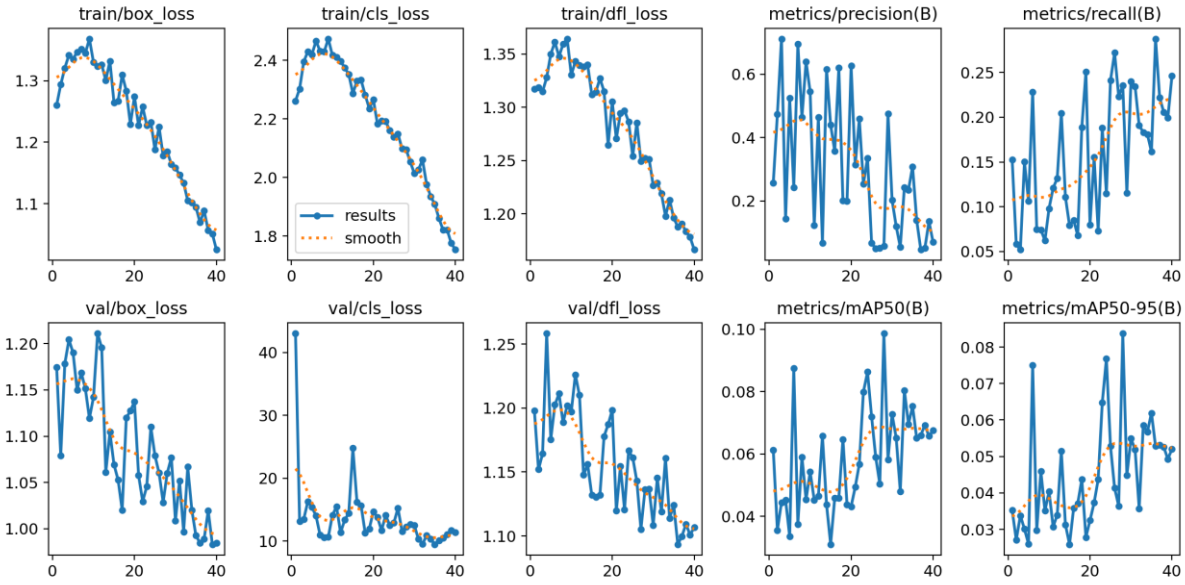


Рисунок 3.10 – Графіки результатів навчання останніх 50 епох

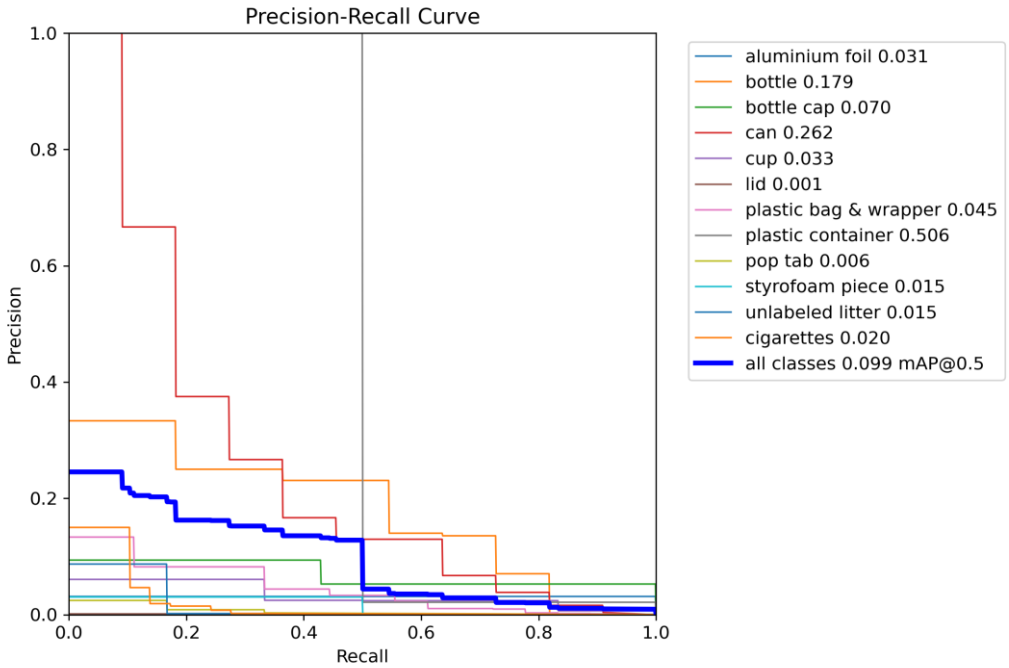


Рисунок 3.11 – Графіки PR-curve

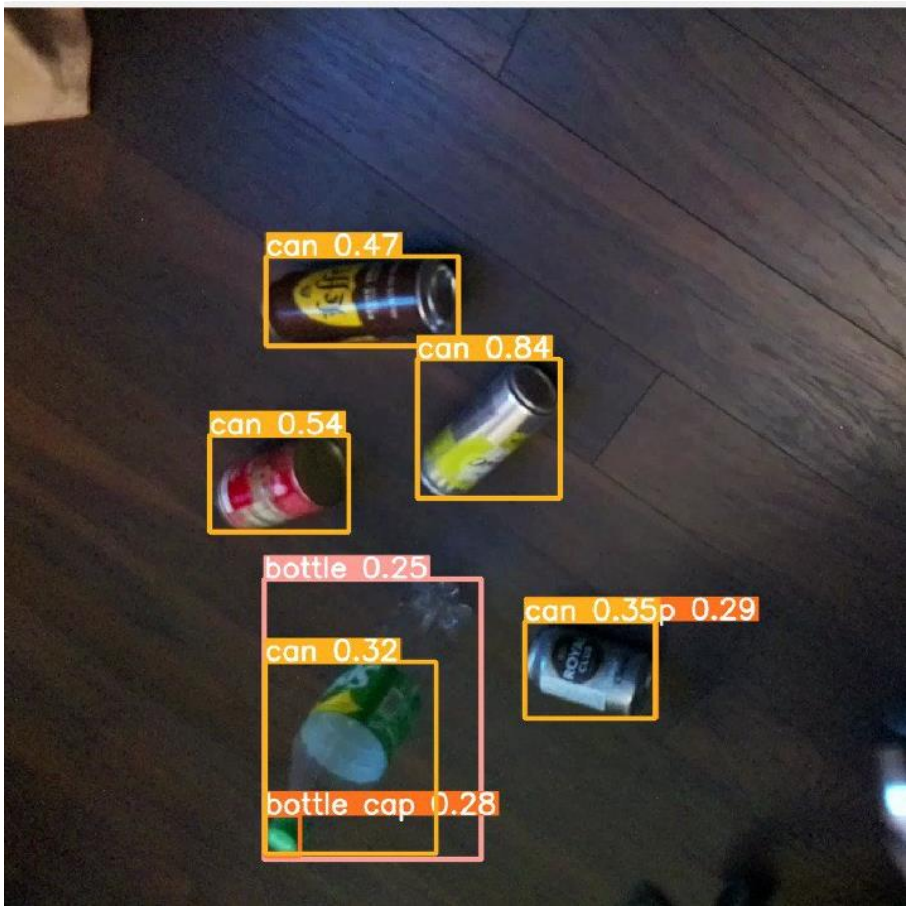


Рисунок 3.12 – Результати роботи моделі на мобільному пристрої

Class	mAP50	mAP50-95
all	0.0987	0.0837
aluminium foil	0.0311	0.028
bottle	0.179	0.13
bottle cap	0.0698	0.033
can	0.262	0.224
cup	0.0334	0.0231
lid	0.000659	0.000132
plastic bag & wrapper	0.0452	0.0349
plastic container	0.506	0.5
pop tab	0.00646	0.00258
styrofoam piece	0.0151	0.0106
unlabeled litter	0.0149	0.00969
cigarettes	0.0199	0.00793

Рисунок 3.13 – Метрики mAP50, mAP50-95 по категоріям

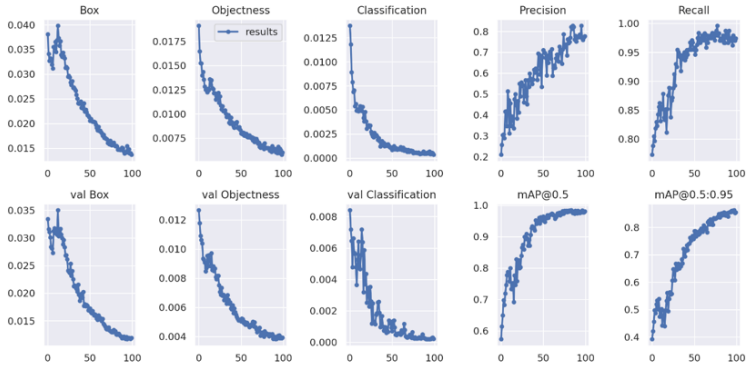
Метрика mAP50 (mean Average Precision) при IoT, що дорівнює 0,5. Ця метрика показує наскільки ефективно модель розпізнає об’єкти у простих випадках.

Метрика mAP50-95 (mean Average Precision) при IoT, що дорівнює 0,5–0,95. Ця метрика показує наскільки ефективно модель розпізнає об’єкти у складних випадках.

Для порівняння на рисунку 3.14 приведено більш успішні результати навчання моделі YOLOv5 з двома найбільшими класами (пластикова пляшка, бляшанка) та ігноруванням інших.

На рисунку 3.12 можна побачити, як модель працює на мобільному пристрої. На підлозі лежить декілька бляшанок та пляшка. Модель успішно розпізнала бляшанки, трохи помилилася з пляшкою, правильно визначила кришечку від пляшки.

а)



б)

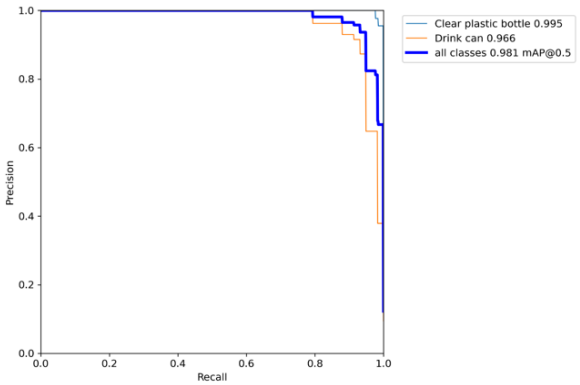


Рисунок 3.14 – Результати роботи моделі YOLOv5: а) динаміка навчання моделі, б) PR-curve результатів роботи

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було проведено аналіз обраної предметної області: проблематики забрудненості навколишнього середовища, сучасних технологій штучного інтелекту в області комп’ютерного зору та глибоких нейронних мереж, які стають в пригоді для вирішення вище зазначеної проблеми.

Перш за все були прикладені зусилля для аналізу актуальності предметної області, звернено увагу на важливість застосування технологій для вирішення поставлених задач.

Завдяки швидкому розвитку технологій, ми можемо говорити про часткову оптимізацію процесу утилізація твердих відходів. Сучасні технології можуть стати корисними в процесі ідентифікації несанкціонованих звалищ, чи, наприклад, в оцінці чистоти вулиць міста.

Напрямок комп’ютерного зору невпинно розвивається, ця область дуже розгалужена, вона пропонує велику кількість інструментів для вирішення ще більшого обсягу задач. Перевіреним інструментом в цій області є глибокі згорткові нейронні мережі. Різноманітність їх архітектур була досліджена в першому блоці роботи. Визначальними факторами обрання тієї чи іншої моделі були ефективність та можливість роботи на мобільних пристроях.

Окрім того було приділено увагу теорії щодо методики аугментації даних. Аугментація даних – важливий інструмент для збільшення кількості даних у випадку, коли їх кількість обмежена. Загалом було проаналізовано сталі кращі підходи, існуючі інструменти.

Як було зазначено вище, кількість даних для навчання обмежена. Цей висновок зроблений після того, як було проаналізовано існуючі джерела даних. Загалом існує достатня кількість датасетів щодо відходів, але всі вони різної направленості починаючи від «академічних», в яких на зображеннях предмети на білому фоні, закінчуючи світлинами з дна морів та океанів. В

результаті аналізу ми прийшли до висновку використати той самий датасет TACO, з яким вже мали справу. Він складається із зображень, що зроблені на мобільні телефони, що дуже гарно співвідноситься з обраною нами концепцією використання мобільних пристроїв. Це означає, що датасет складається зі схожих даних, які ми плануємо отримувати на вхід моделі. Ми збільшили кількість зображень за допомогою методики аугментації. З іншого боку через нерівномірність даних за категоріями, деякі з цих категорій було проігноровані (залишилося 11 з 28 суперкатегорій).

Для вирішення поставленої задачі було обрано сімейство моделей YOLO, які протягом декількох років підтвердили свою ефективність та швидкість роботи. У другому блоці роботи було позначено основні особливості YOLOv8 у порівнянні з YOLOv5.

В третьому блоці роботи було описано прикладні кроки, які були зроблено. Це все, що пов’язано з аргументацією. Частково її було проведено за допомогою бібліотек Python, іншу частину було покладено на вбудовані в YOLOv8 алгоритми аугментації, які були налаштовані за допомогою параметрів під час запуску навчання моделі. З допомогою цих дій вдалося вдвічі збільшити кількість зображень.

В якості мобільного пристрою було обрано мобільну платформу Raspberry PI 4 через простоту її використання, компактність, універсальність та модульність. Ще великим плюсом цієї платформи є операційна система Linux, з якою відносно просто працювати в порівнянні з Android чи IOS. А її потужності вистачає для розрахунків офлайн.

Після того, як мобільний пристрій було обрано, також було обрано і оптимальний розмір моделі YOLOv8S (Small). Ці розрахунки базувалися на швидкості виконання розпізнавання об’єктів на зображенні. Для оптимізації цього процесу і можливості обрати найбільший можливий розмір, модель було конвертовано у формат ONNX, який більш оптимально працює на системах, що фокусуються на CPU.

В результаті роботи було отримано модель зготкової нейронної мережі YOLOv8S, яку можна запустити на мобільному пристрої Raspberry PI 4. Навчання відбувалося на TACO датасеті, який було розширено за допомогою методики аугментації даних.

Загалом оцінити результати навчання моделі можна як задовільні, враховуючі той факт, що вона ефективно класифікує найбільші категорії з датасету, а це пляшки та бляшанки.

Особливості моделі. Через нерівномірність даних в обраному датасеті, ефективність розпізнавання та класифікації об’єктів різна за категоріями. Деякі об’єкти модель класифікує відмінно, деякі гірше. Окрім неконсистентності даних, на результат вплинув той факт, що третя за кількістю анотацій категорія «цигарки» має описані дуже малі предмети, що негативно впливає на метрики.

Узагальнюючи модель показала свою ефективність на мобільному пристрою з деякими обмеженнями в продуктивності. Метрики вказують на ефективність моделі розпізнавати декілька з перелічених категорій об’єктів. Результати її роботи можна назвати реалістичними. Загалом існують перспективи розвитку як з боку обладнання (наприклад, додавання системи активного охолодження), так і з боку програмного забезпечення, покрашення навчальних даних, більш тонке налаштування моделі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. European climate law. Climate Action. URL: https://climate.ec.europa.eu/eu-action/european-climate-law_en (дата звернення: 27.05.2024).

2. Kenekar A. Negative effects of improper solid waste disposal on human health. Organica Biotech. URL: https://organicabiotech.com/negative-effects-of-improper-solid-waste-disposal-on-human-health/ (дата звернення: 27.05.2024).

3. Забруднення. Класифікація забруднень довкілля – WikiLegalAid. Платформа правових консультацій – WikiLegalAid. URL: https://wiki.legalaid.gov.ua/index.php/Забруднення._Класифікація_забруднень_довкілля (дата звернення: 27.05.2024).

4. Бухановський В. О. Дослідження та застосування методів computer vision для розпізнавання та класифікації забруднень. *Радіоелектроніка та молодь у XXI столітті. Т. 6 : Конференція «Інформаційні інтелектуальні системи»* : матеріали 28-го Міжнар. молодіж. форуму, м. Харків, 16–18 квіт. 2024 р. Харків, 2024. С. 61–63. URL: https://openarchive.nure.ua/entities/publication/1d16e319-fde0-4d2d-ae64-f3ade2a135e1 (дата звернення: 27.05.2024).

5. Fumo J. A gentle introduction to neural networks series – part 1. *Medium*. URL: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc (дата звернення: 27.05.2024).

6. Introduction to convolutional neural networks | rubix code. *Rubix Code*. URL: https://rubikscode.net/2018/02/26/introduction-to-convolutional-neural-networks/ (дата звернення: 27.05.2024).

7. Xu B., Wang N. Empirical evaluation of rectified activations in convolutional network. *arXiv.org*. URL: https://arxiv.org/abs/1505.00853 (дата звернення: 27.05.2024).

8. Convolutional neural network. *Medium*. URL: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529 (дата звернення: 27.05.2024).

9. He K. Deep residual learning for image recognition. *arXiv.org*. URL: https://arxiv.org/abs/1512.03385 (дата звернення: 27.05.2024).

10. Russakovsky O., Deng J., Su H. ImageNet large scale visual recognition challenge. *arXiv.org*. URL: https://arxiv.org/abs/1409.0575 (дата звернення: 27.05.2024).

11. Bezdan T., Bacanin N. Convolutional neural network layers and architectures.
URL: https://www.researchgate.net/publication/333242381_Convolutional_Neural_Network_Layers_and_Architectures (дата звернення: 27.05.2024).

12. Rich feature hierarchies for accurate object detection and semantic segmentation / Ross Girshick та ін. *arXiv.org*. URL: https://arxiv.org/abs/1311.2524 (дата звернення: 27.05.2024).

13. Generative adversarial networks / I. J. Goodfellow та ін. *arXiv.org*. URL: https://arxiv.org/abs/1406.2661 (дата звернення: 27.05.2024).

14. An image is worth 16x16 words: transformers for image recognition at scale / A. Dosovitskiy та ін. *arXiv.org*. URL: https://arxiv.org/abs/2010.11929 (дата звернення: 27.05.2024).

15. Trash detection – review of useful resources. *GitHub*. URL: https://github.com/majsylw/litter-detection-review (дата звернення: 27.05.2024).

16. Yang M., Thung G. Classification of trash for recyclability status. *CS229: Machine Learning*. URL: https://cs229.stanford.edu/proj2016/report/ThungYang-ClassificationOfTrashForRecyclabilityStatus-report.pdf (дата звернення: 28.05.2024).

17. Hong J., Fulton M., Sattar J. TrashCan: a semantically-segmented dataset towards visual detection of marine debris. *arXiv.org*. URL: https://arxiv.org/abs/2007.08097 (дата звернення: 28.05.2024).

18. Proença P. F., Simões P. TACO: trash annotations in context for litter detection. *arXiv.org*. URL: https://arxiv.org/abs/2003.06975 (дата звернення: 28.05.2024).

19. Rosebrock A. Deep learning for computer vision with python : Practitioner Bundle. PYIMAGESEARCH, 2017. 208 p.

20. A garbage detection and classification method based on visual scene understanding in the home environment / Y. Wu та ін. *ResearchGate*. URL: https://www.researchgate.net/publication/356569155_A_Garbage_Detection_and_Classification_Method_Based_on_Visual_Scene_Understanding_in_the_Home_Environment (дата звернення: 28.05.2024).

21. Saji R. M. A survey on smart garbage management in cities using iot. *International journal of engineering and computer science*. 2016. URL: https://doi.org/10.18535/ijecs/v5i11.04 (дата звернення: 28.05.2024).

22. SpotGarbage / G. Mittal та ін. *UbiComp '16: the 2016 ACM international joint conference on pervasive and ubiquitous computing*, м. Heidelberg Germany. New York, NY, USA, 2016. URL: https://doi.org/10.1145/2971648.2971731 (дата звернення: 28.05.2024).

23. Examples: Bounding Boxes. *imgaug 0.4.0 documentation*. URL: https://imgaug.readthedocs.io/en/latest/source/examples_bounding_boxes.html (дата звернення: 28.05.2024).

24. You only look once: unified, real-time object detection / J. Redmon та ін. *Computer vision and pattern recognition (CVPR)* : Proceedings of the IEEE Conference, м. Piscataway, New Jersey, 2016. С. 779–788.

25. Redmon J., Farhadi A. YOLOv3: an incremental improvement. *arXiv.org*. URL: https://arxiv.org/abs/1804.02767v1 (дата звернення: 28.05.2024).

26. Tan M., Yu A. EfficientDet: towards scalable and efficient object detection. *Google Research - Explore Our Latest Research in Science and AI*. URL: https://research.google/blog/efficientdet-towards-scalable-and-efficient-object-detection/#:~:text=EfficientDet%20uses%20EfficientNet%20as%20the,-the-art%20detection%20models. (дата звернення: 28.05.2024).

27. Encord. YOLOv8 for object detection explained [practical example]. *Medium*. URL: https://medium.com/cord-tech/yolov8-for-object-detection-explained-practical-example-23920f77f66a (дата звернення: 28.05.2024).

28. NEW – YOLOv8 in PyTorch > ONNX > OpenVINO > CoreML > TFLite. *GitHub*. URL: https://github.com/ultralytics/ultralytics (дата звернення: 28.05.2024).

29. SiLU – PyTorch 2.3 documentation. *PyTorch*. URL: https://pytorch.org/docs/stable/generated/torch.nn.SiLU.html (дата звернення: 28.05.2024).

30. Solawetz J. What is yolov8? The ultimate guide. 2024. *Roboflow Blog*. URL: https://blog.roboflow.com/whats-new-in-yolov8/ (дата звернення: 28.05.2024).

31. Image augmentation for machine learning experiments. *GitHub*. URL: https://github.com/aleju/imgaug (дата звернення: 28.05.2024).

32. Core ML, apple developer documentation. *Apple Developer Documentation*. URL: https://developer.apple.com/documentation/coreml (дата звернення: 28.05.2024).

33. Ultralytics. Raspberry pi. *Ultralytics YOLO Docs*. URL: https://docs.ultralytics.com/guides/raspberry-pi/ (дата звернення: 28.05.2024).

34. ONNX. *ONNX*. URL: https://onnx.ai/ (дата звернення: 28.05.2024).

35. Tips for best training results. *GitHub*. URL: https://github.com/ultralytics/yolov5/wiki/Tips-for-Best-Training-Results (дата звернення: 28.05.2024).

ДОДАТОК А

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	60 с.
	Інші документи	
2.	Презентаційні матеріали	22 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Бухановський В.О.			Дослідження та застосування методів комп'ютерного зору для вирішення задач ідентифікації та класифікації твердих відходів	Шифр групи	Код напр./спец.
Перевірив		Рябова Н. В.				СШІм-22-3	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					