Міністерство освіти і науки України
Державний університет «Одеська політехніка»

Навчально-науковий інститут штучного інтелекту та роботехніки
Кафедра комп’ютерних систем

Кошутіна Дар’я Валеріївна,
студентка групи УК-162

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Дослідження методів підвищення ефективності системи виявлення об'єктів на платформі Raspberry Pi

Спеціальність:
123 - Комп'ютерна інженерія

Спеціалізація, освітня програма:
Спеціалізовані комп’ютерні системи

Керівник:
Стрельцов Олег Васильович,
доцент

Одеса – 2021

Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Навчально-науковий інститут штучного інтелекту та роботехніки
Кафедра комп’ютерних систем

Рівень вищої освіти: магістерський
Спеціальність: 123 - Комп'ютерна інженерія
Спеціалізація, освітня програма: Спеціалізовані комп’ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Ситніков В.С.
_____2021р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кошутіної Дар’ї Валеріївни, УК-162

1. Тема роботи: Дослідження методів підвищення ефективності системи виявлення об'єктів на платформі Raspberry Pi

Керівник роботи: Стрельцов Олег Васильович, доцент

- затверджені наказом ректора від 01.10.2021 р. № 346-в

2. Зміст роботи: Аналітичний огляд, дослідження методів виявлення об’єктів, прототипи системи визначення об’єктів, постанова задачі, висновки на основі першого розділу, дослідження методів виявлення об’єктів, методи виявлення об’єктів на зображені ,програмне забезпечення, висновки до другого розділу, 3 перевірка ефективності запропонованих рішень, перевірка ефективності системи, тестування, висновки до третього розділу, висновки

3. Перелік ілюстративного матеріалу: Блок-схема алгоритму роботи методу виявлення об’єктів, Структура методу SSD, графік залежності точності спрацьовування методів,

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2,3 - Нормоконтроль	Нестерюк О.Г.		

5. Дата видачі завдання: 01 жовтня 2021р.

- ***КАЛЕНДАРНИЙ ПЛАН***

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Пошук наукових публікацій, що містять відомості про методи виявлення об’єктів на зображенні та відеопотоці	3 тиждень	виконано
2	Аналіз існуючих методів виявлення об’єктів на зображенні та відеопотоці	5 тидень	виконано
3	Розробка алгоритму	8 тиждень	виконано
4	Формування критеріїв доцільності вибору використання запропонованого алгоритму виявлення об’єктів на зображенні	12 тиждень	виконано
5	Визначення ефективності	14 тиждень	виконано
6	Оформлення пояснювально записки	15-17 тиждень	виконано
7	Захист	17 тиждень	виконано

Здобувач вищої освіти Кошутіна Д.В.

Керівник роботи Стрельцов О.В.

ЗМІСТ ст.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AP – Average Precision

CNN – Convolutional neural network

CPU – Central processing unit

CUDA – Compute Unified Device Architecture

GPU – Graphics processing unit

mAP – mean Average Precision

SSD – Single-Shot MultiBox Detector

YOLO − You Only Look Once

ЗНМ – Згорткові нейронні мережі

ВСТУП

Сучасне життя стає все більш автоматизованим, прискорюючи темпи економіки та життя суспільства загалом. Однак деякі області досі повністю не опрацьовані, не досягнуті бажані результати, здатні вплинути на життя людства. Одним із таких напрямків є розробка системи розпізнавання образів.

Розвиток Інтернету та процеси глобалізації сприяли тому, що з’явилося дуже багато інформації, опрацювати яку самотужки людина фізично не в змозі.

Штучний інтелект, як наукова галузь активно розвивається з 50-х років XX століття. У 1952 році Артур Самуель створив першу шашкову гру для IBM і трохи пізніше додав до цієї програми здатність до самонавчання. Іншими словами – комп'ютер навчили грати в шашки. Таким чином, Артура Самуеля можна назвати піонером в області штучного інтелекту. Вчені розробляли алгоритми, створювали багато дослідницьких проектів. В 1959 році їм вдалося створити першу нейронну мережу [1].

Нейронна мережа – це послідовність нейронів, з'єднаних між собою синапсами. Структура нейронної мережі прийшла у світ програмування прямо з біології. Завдяки такій структурі машина отримує можливість аналізувати і навіть запам'ятовувати різну інформацію. Нейронні мережі також здатні не тільки аналізувати вхідну інформацію, а й відтворювати її зі своєї пам'яті. Іншими словами, нейронна мережа це машинна інтерпретація мозку людини, в якому знаходяться мільйони нейронів які передають інформацію у вигляді електричних імпульсів.

Нейронні мережі знайшли застосування у:

- аналізі та класифікації даних за заданими параметрами;
- формуванні аналітичних прогнозів, керуючись вхідною інформацією;
- порівнянні та розпізнаванні ідентичних даних.

Останній пункт, наприклад, використовується в системах безпеки аеропортів. Виконується це шляхом фіксації обличчя людей, та порівняння їх із базою злочинців. Ще один приклад — функція Google по пошуку схожого зображення. Достатньо завантажити фото і система знайде усі схожі картинки.

Останнім часом все частіше і частіше говорять про так звані нейронні мережі, мовляв незабаром вони будуть активно застосовуватися і в роботехніці, і в машинобудуванні, і в багатьох інших сферах людської діяльності, ну а алгоритми пошукових систем, того ж Google вже потихеньку починають на них працювати.

Такі системи мають велику популярність у космічних розробках та в органах з управління безпекою життєдіяльності населення, зокрема, у місцях масового скупчення людей, з метою запобігання тероризму, а також розпізнавання осіб, які вчинили злочини (аеропорти, вокзали, банки, супермаркети та торгові центри, культурно-розважальні та спортивні об'єкти). Також важливо контролювати порядок на вулицях міста.

В даний час однією з проблем розвитку інтелектуальних систем розпізнавання осіб є відсутність доступного технічного оснащення. Камери, що передають дуже якісне зображення, необхідне обробки системи, мають найвищу вартість. Відповідно, дуже мала частина суб'єктів світової економіки може собі дозволити такі витрати. Часто вони є недоцільними. До того ж сама система розпізнавання осіб досить витратною.

На основі машинного навчання дещо пізніше з'явилося поняття комп'ютерного зору. Комп'ютерний зір – це технологія, за допомогою якої машини можуть знаходити, відстежувати, класифікувати та ідентифікувати об'єкти, витягуючи дані і аналізуючи отриману інформацію суто з зображень. Комп'ютерний зір застосовується для розпізнавання об'єктів, відеоаналітики, опису змісту зображень і відео, розпізнавання жестів і рукописного введення, а також для інтелектуальної обробки всього що можна побачити людським оком.

Актуальність. Дане дослідження є актуальним в сучасному світі, тому що все більше використовують розумні девайси, роботи та роботехнічні системи для покращення життя. Тому система визначення об’єктів є важливою частиною роботизації та автоматизації. Відеоспостереження на дорогах, система технічного зору в машин Tesla чи роботів-помічників, FaceID на телефонах – це все варіанти виявлення об’єктів.

Мета дослідження – підвищити ефективність системи виявлення об'єктів на платформі Raspberry Pi.

Задачі дослідження:

- провести аналіз існуючих методів виявлення об’єктів;
- удосконалити методи підвищення ефективності;
- удосконалити метод виявлення об’єктів;
- моделювання ефективності запропонованого рішення.

Об’єкт дослідження – процес виявлення об’єктів на зображенні.

Предмет дослідження – методи розпізнавання об’єктів в реальному часі.

Наукова новизна.

Існує багато методів та підходів, які намагаються вирішити задачу виявлення об’єктів на зображені. Серед, них є ряд алгоритмів, які аналізують зображення за допомогою методів опорних векторів, якими можна здійснювати класифікацію об'єкта, чи завдяки семантичного аналізу, шляхом чергування згорткових та субдискретизуючих шарів, а також наявність операції згортки, в процесі якої на матрицю згортки поелементно множиться кожен фрагмент зображення, підсумовується та записується у відповідну позицію у вихідному зображенні. Натомість, більшість з запропонованих підходів працюють повільно і не можуть похвалитись стабільною роботою у реальному часі, а також вони погіршують якість вхідного зображення.

Запропонований алгоритм вирішує задачу виявлення об’єктів у реальному часі та класифікація його.

Розроблений алгоритм розбиває отримане зображення на шари і паралельно аналізує кожний шар на наявність об’єкту.

Публікації. Підготовлена стаття «Дослідження методів підвищення ефективності системи виявлення об'єктів на платформі Raspberry Pi» для публікації в науковому видані.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз методів виявлення об’єктів

Виявлення об'єктів - це комп'ютерна технологія, пов'язана з комп'ютерним зором та обробкою зображень, що займається виявленням екземплярів семантичних об'єктів певного класу (таких як люди, будівлі чи автомобілі) у цифрових зображеннях та відео. [2] Виявлення об'єктів знаходить застосування в багатьох областях комп'ютерного зору, включаючи пошук зображень та відеоспостереження.

Обробка зображень чи аналіз зображень, в основному зосереджені на роботі з двомірними зображеннями, тобто як перетворити одне зображення в інше. Наприклад, попіксельного операції збільшення контрастності, операції по виділенню країв, усунення шумів або геометричні перетворення, такі як Афінний перетворення. Дані операції припускають, що обробка / аналіз зображення діють незалежно від змісту самих зображень.[3]

Комп'ютерне зір зосереджується на обробці тривимірних сцен, спроектованих на одне або декілька зображень. Наприклад, відновленням структури або іншої інформації про тривимірній сцені за одним або кількома зображень. Комп'ютерне зір часто залежить від більш чи менш складних припущень щодо того, що представлено на зображеннях.[4]

Методи виявлення об'єктів, як правило, засновані або на машинному або на глибинному навчанні.

Машинне навчання є одним з видів штучного інтелекту, де комп'ютери «вчаться самі», будучи ще не запрограмованими на дії, які вони вміють, після свого «навчання». Замість розробки алгоритму, силами машинного навчання можна дозволити комп'ютерам самим розробляти і вдосконалювати будь-які

алгоритми, проводити пошук відомих конструкцій і шаблонів у великих обсягах даних.

Так про машинне навчання говорять уже в IT-сфері – розпізнавання зву- 9 ків і зображень; рекламні компанії – передбачення втрати клієнтів, маркетингові дослідження; медична діагностика – аналізуючи історії хвороб пацієнтів, можна виявляти непомітні для людини зв'язку і встановлювати невідомі раніше симптоми небезпечних захворювань; технічна діагностика та біоінформатика. Прогнозування покупок користувачів цілком покладається на нейронні мережі та дерева рішень з використанням даних з інтернет-магазину [5].

Глибинне навчання – це галузь машинного навчання, що ґрунтується на наборі алгоритмів, які намагаються моделювати високорівневі абстракції в даних, застосовуючи глибинний граф із декількома обробними шарами, що побудовано з кількох лінійних або нелінійних перетворень.[6]

Для методів, заснованих на машинному навчанні, спочатку потрібно визначити особливості об'єкта, перш ніж класифікувати його. Далі, за допомогою технік схожих з методом опорних векторів, вже можна здійснювати класифікацію об'єкта. Приклади методів із використанням машинного навчання:

− алгоритм Віоли-Джонса, заснований на ознаках Хаара;

− масштабно-інваріантна трансформація ознак;

− гістограма спрямованих градієнтів.

Алгоритм Віоли-Джонса включає чотири стадії:

− виділення ознак Хаара на зображенні;

− створення інтегрального зображення (для кожного пікселя записується сума пікселів вище та лівіше за нього);

− навчання з використанням алгоритму Adaptive Boost (алгоритм машинного навчання для покращення алгоритмів класифікації);

− каскадна класифікація.

Ознаки, які алгоритм шукає, включають суму пікселів всередині певної прямокутної області. Але на відміну від ознак Хаара алгоритм використовує

більше однієї області, тим самим роблячи ознаки комплекснішими. Переважно цей алгоритм використовується для виявлення осіб на зображеннях та відео.

Алгоритм масштабно-інваріантної трансформації ознак ґрунтується на виділенні ключових ознак (точок) об'єкта з набору контрольних зображень та подальшому зберіганні даних ознак у базі даних. Розпізнавання об'єкта на новому зображенні відбувається за рахунок порівняння ознак на новому зображенні з ознаками, що зберігаються у базі даних. Збіг ознак ґрунтується на евклідовій відстані між векторами ознак. Даний метод може використовуватися для трекінгу, розпізнавання жестів, зшивання зображень та тривимірного моделювання.

Алгоритм гістограми спрямованих градієнтів ґрунтується на підрахунку випадків певного напрямку градієнтів на виділеній ділянці зображення. Цей алгоритм схожий з алгоритмом масштабно-інваріантної трансформації ознак, але обчислюється на щільній сітці рівномірно розподілених клітин і підвищення точності використовує нормалізацію локального контрасту, що перекриває. Алгоритм заснований на можливості опису зовнішнього вигляду та форми об'єкта за допомогою розподілу градієнтів інтенсивності. Перевагою даного алгоритму є стійкість до геометричних та фотометричних змін зображення. Завдяки цьому він добре підходить для виявлення людей на зображеннях.

Для методів, заснованих на глибинному навчанні, властиве використання згорткових нейронних мереж, які дозволяють виявляти об'єкт без використання списку специфічних особливостей даного об'єкта.

Згорткові нейронні мережі (ЗНМ) у машинному навчанні — це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень. [7]

ЗНМ використовують різновид багатошарових перцептронів, розроблений так, щоб вимагати використання мінімального обсягу попередньої обробки.

Згорткові мережі взяли за основу біологічний процес, а саме схему з'єднання нейронів зорової кори тварин. Окремі нейрони кору реагують на

стимули лише в обмеженій області зорового поля, відомої як рецептивне поле. Рецептивні поля різних нейронів частино перекриваються таким чином, що вони покривають усі зорове поле.

Приклади методів із використанням глибокого навчання:

− Region Proposals (з використанням різних регіональних згорткових нейронних мереж: R-CNN, Fast R-CNN, Faster R-CNN, cascade R-CNN);

− Single-Shot MultiBox Detector;

− You Only Look Once;

− RetinaNet;

− Single-Shot Refinement Neural Network for Object Detection (RefineDet);

− Deformable convolutional networks.

Дані методи базуються на згорткових нейронних мережах, основною ідеєю яких є чергування згорткових та субдискретизуючих шарів, а також наявність операції згортки, в процесі якої на матрицю згортки поелементно множиться кожен фрагмент зображення, підсумовується та записується у відповідну позицію у вихідному зображенні.

У методі Region Proposals використовуються регіональні згорткові нейронні мережі - різновид згорткових нейронних мереж, основною метою яких є виділення на зображенні областей інтересу, і подальше визначення та класифікація об'єктів у кожній з областей.

Метод You Only Look Once (YOLO) менш точний, ніж регіональні згорткові нейронні мережі, але при цьому значно швидше, що дозволяє виявляти об'єкти в реальному часі. Суть даного методу полягає в початковому поділі зображення на сітку комірок. Кожна комірка відповідає за розташування області об'єкта на зображенні, якщо центр цієї області знаходиться в межах комірки. Для кожної області визначаються координати x і y, ширина і висота області, а також коефіцієнт впевненості, який показує можливість наявності в даній області будь-якого об'єкта. Крім того, кожна клітина визначає клас об'єкта в ділянці, що відноситься до цієї клітини [8]. Основна ідея даного підходу є визначення

об'єкту, як проблему регресії до просторово розділених обмежувальних блоків і пов'язаних з ними ймовірностей класів. Розроблена нейронна мережа передбачає обмежувальні поля і ймовірності класу безпосередньо з повних зображень в одній оцінці, що показано на рисунку 1.1. Оскільки весь конвеєр виявлення є єдиною мережею, він може бути оптимізований наскрізь безпосередньо на продуктивності виявлення. У порівнянні з найсучаснішими системами виявлення YOLO робить більше помилок локалізації. Нарешті, YOLO дізнається дуже загальні уявлення про об'єкти. Він перевершує інші методи виявлення, включаючи DPM і R-CNN, при узагальненні від природних зображень до інших доменів, наприклад, до ілюстрації.

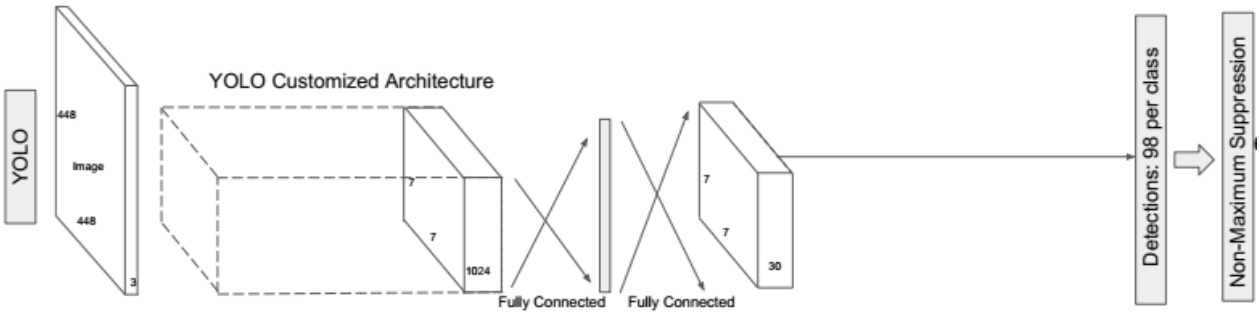

Рисунок 1.1 – Структура методу YOLO [9]

Метод RetinaNet також є одноетапним методом виявлення об'єкта. Він включає піраміду ознак Feature Pyramid Network і функцію помилки для навчання нейронних мереж Focal Loss.

Feature Pyramid Network складається з трьох частин: висхідного шляху, низхідного шляху та бічних з'єднань. Висхідний шлях схожий на піраміду і виглядає як послідовність шарів згорткової нейронної мережі з розмірністю, що зменшується. Таким чином, нижні шари мають більшу роздільну здатність, але менше семантичне значення; верхні ж шари – навпаки. Ця частина вразлива до

шуму на зображенні, оскільки через зашумленість може втратитися інформація про об'єкт.

У низхідному шляху розмірності шарів відповідають розмірності шарів у висхідному шляху, але з-за руху вниз шарами відбувається збільшення карти ознак за допомогою методу найближчого сусіда.

Бічні з'єднання потрібні для того, щоб прибрати загасання сигналів у процесі руху шарами, поєднуючи семантично важливу інформацію, отриману до кінця першої піраміди і більш детальну інформацію, отриману в ній раніше.

Focal Loss, зазвичай, використовую на вирішення проблеми дисбалансу класів з урахуванням одноступінчастих моделей виявлення об'єктів, тобто. Ця функція необхідна для виправлення проблеми перекриття об'єкта з меншою областю якоря об'єктом з більшою областю якоря. Через цю проблему спосіб може виявити невеликий об'єкт великому тлі. Focal Loss на кожному рівні піраміди залишає лише кілька якірних областей як шуканий об'єкт. Усі інші області вважатимуться об'єктами задньому фоні [10].

Метод RetinaNet добре зарекомендував себе у виявленні об'єктів на зображеннях із супутників та з повітря.

Метод Single-Shot MultiBox Detector представлений двома компонентами: нейронною мережею для класифікації зображень та згорткового шару для виявлення та класифікації об'єктів на зображенні. Структура методу показана на рисунку 1.2. Цей метод, як і YOLO, ділить зображення на сітку, але також оперує поняттям якірної області. Для кожній клітинці сітки можна призначити кілька якірних областей. Кожна з них зумовлена і відповідає за розмір та форму об'єкта всередині комірки [11]. Перевагою даного методу, як і методу YOLO, є швидкість виконання та можливість виявлення об'єкта за один прогін зображення, на відміну від регіональних нейронних згорткових мереж, яким необхідно пройти два етапи (виділення можливого регіону з об'єктом і власне виявлення об'єкта в даному регіоні зображення).

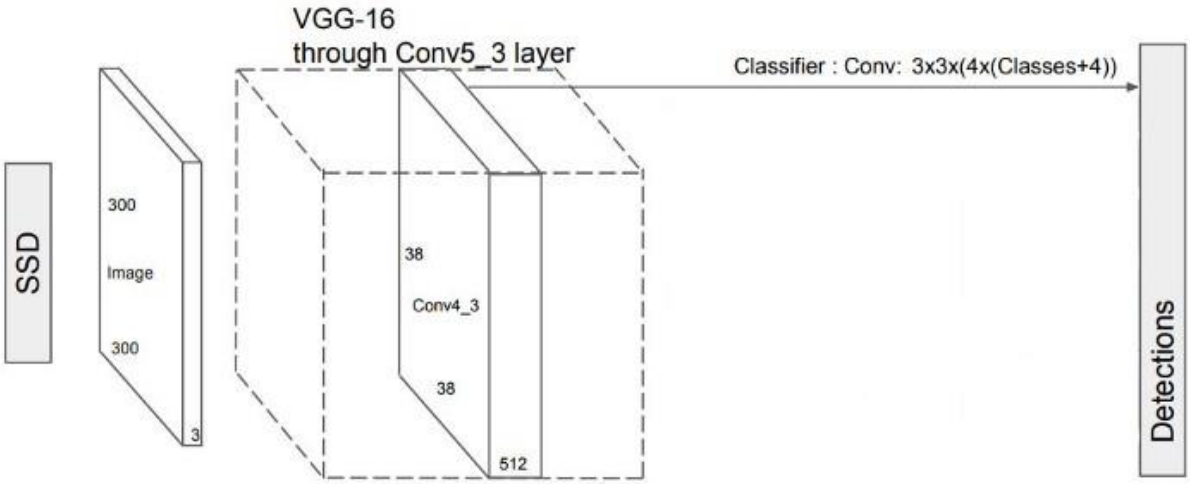

Рисунок 1.2 – Структура методу SSD [9]

SSD не використовує мережу пропозицій делегованого регіону. Натомість він зводиться до дуже простого методу. Він обчислює як місце розташування, і оцінки класу, використовуючи невеликі фільтри згортки. Після вилучення карток характеристик SSD застосовує фільтри згортки 3x3 для кожної комірки, щоб зробити прогнози, це показано на рисунку 1.3. (Ці фільтри обчислюють результати так само, як і звичайні фільтри CNN.) Кожен фільтр виводить 25 каналів: 21 бал для кожного класу плюс одне граничне поле.

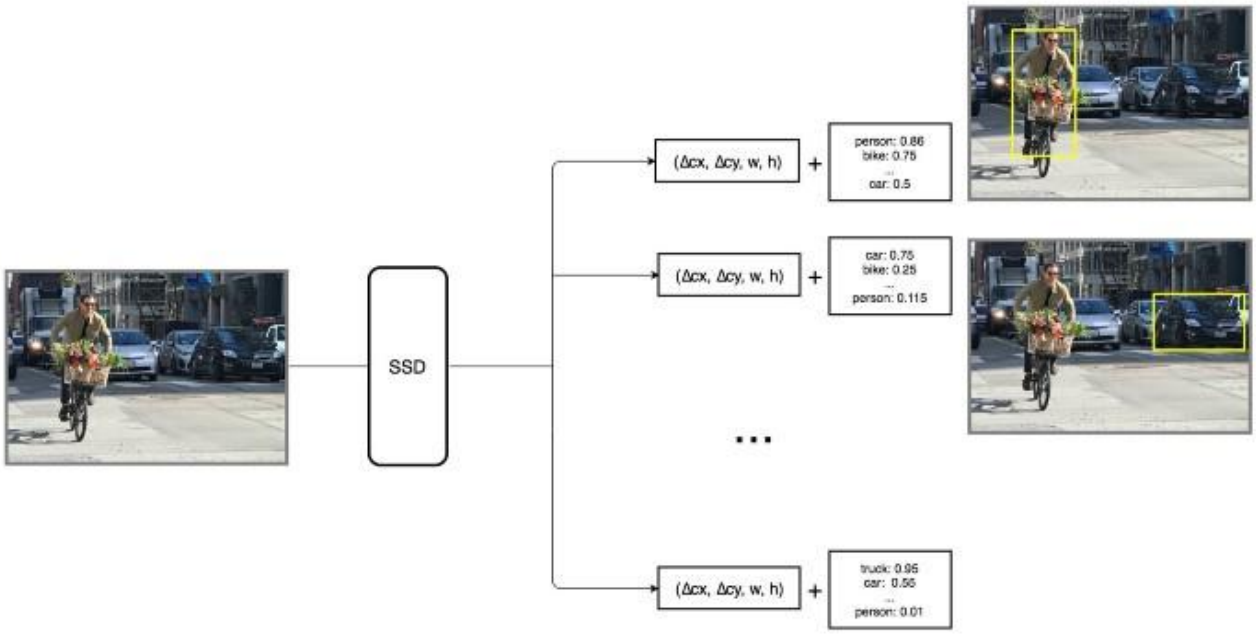

Рисунок 1.3 – Прогнози вхідного зображення [12]

На сьогоднішній день так і не розроблено оптимальний алгоритм розпізнавання, існує кілька провідних бібліотек, яким вдається розпізнавати образи на картинці, шукати ідентичні особи в мережі. Однак мета пошуку місцезнаходження об'єкта або його ідентифікації не досягнуто.

Провідні світові компанії області інформаційних технологій, такі як Google, Microsoft, Facebook, Apple, Intel створили відділи розробки бібліотек розпізнавання образів. Поки що результати їх роботи обмежуються простими додатками з розпізнаванням порід тварин, людей, проте їх очікування у найближчій перспективі є досить високими. У вересні 2017 р. стало відомо, що вчені Індії та Великобританії навчили нейронну мережу розпізнавати злочинців із предметами маскування [13].

Компанія FaceBook розробила алгоритм під назвою DeepFace, яка дозволить візуально аналізувати, порівнювати та ідентифікувати людські обличчя з неймовірно високою точністю (До 97,25%). DeepFace буде використовувати техніку 3D-моделювання для сканування об'єкта, але сам

алгоритм будується на основі процесу «фронталізації», тобто зміни кута зображення таким чином, щоб обличчя людини дивилося прямо вперед. Потім отримані дані перетворюються на числове значення і обробляються для наступного порівняння. На даний момент, DeepFace проходить етапи тестування, для якого FaceBook вже ідентифікували близько 4 млн фотографій своїх користувачів [14].

Компанія Google була розроблена відкрита програмна бібліотека для машинного навчання під назвою "TensorFlow". Вона дозволяє вирішувати завдання побудови та тренування нейронної мережі з метою автоматичного знаходження та класифікації образів, досягаючи якості людського сприйняття. У той час як еталонна реалізація працює на одиничних пристроях, TensorFlow може працювати на багатьох паралель них процесорах, як CPU, так і GPU, спираючись на архітектуру CUDA підтримки обчислень загального призначення на графічних процесорах). Унікальність бібліотеки полягає в наступних характеристиках:

– основна бібліотека підходить для широкого сімейства технік машинного навчання, а не тільки для глибинного навчання;

– лінійна алгебра та інші нутрощі добре видно зовні;

– на додаток до основної функціональності машинного навчання, TensorFlow також включає власну систему логування, власний інтерактивний візуалізатор логів і навіть потужну архітектуру з доставки даних;

– модель виконання TensorFlow відрізняється від scikit-learn мови Python та від більшості інструментів у R.

Обчислення TensorFlow виражаються як графи потоків даних із збереженням стану (stateful). Бібліотека алгоритмів від Google інструктує нейронні мережі сприймати інформацію та розмірковувати подібно до людини, так що нові додатки спочатку мають такі «людські» якості. Сама назва TensorFlow походить від назви операцій, які ці нейромережі здійснюють над багатовимірними масивами даних. Ці багатовимірні масиви називаються

«тензорами», як однойменні математичні об'єкти, що лінійно перетворюють елементи одного лінійного простору елементи іншого. Завдання TensorFlow – вивчати нейромережі виявляти та розпізнавати патерни та кореляції в масивах даних [15].

OpenCV – бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Реалізована на C/C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов. Може вільно використовуватись в академічних та комерційних цілях – поширюється в умовах ліцензії BSD [16]. OpenCV надає різні класифікатори, які можна використовувати для розпізнавання осіб, очей, автомобілів та багатьох інших об'єктів. Ці класифікатори, однак, досить прості, вони не навчені з використанням технологій машинного навчання, тому, при розпізнаванні осіб точність складе приблизно 80 % [17].

1.2 Прототипи систем визначення об’єктів

Одним із прикладів роботи системи є мобільний додаток Google Lens. Google Lens (Google Об'єктив) – це один із безкоштовних сервісів компанії Google LCC, доступний з червня 2018 року. В основі програми лежить технологія розпізнавання зображень. Для отримання зображення на обротку використовується камера смартфону. Основний функціонал додатку це:

- Сканування тексту. Це може бути книга, роздрукований документ чи рукописні записи. Потрапивши в об'єктив камери текст можна виділити та скопіювати, а потім вставити в електронний лист, текстовий документ, нотатку або надіслати повідомлення через будь-який месенджер;
- Збереження рукописних заміток. З одним із останніх оновлень у Google Об'єктиві з'явилася функція розпізнавання рукописного тексту. Наприклад, якщо ви пишете нотатки на аркушах

паперу або в блокноті, ви можете швидко переносити їх до електронного формату;

- Прослуховування тесту;
- Переклад тексту;
- Ідентифікація рослин та тварин;
- Сканування QR- та штрихкодів;
- Пошук по зображенню чи тексту на зображенні.

Програма візуального пошуку Google Lens знаходить інформацію за запитом з урахуванням місцезнаходження користувача, що підвищує точність результату та дозволяє вказати відомості аж до географічних координат. Наприклад, якщо сфотографувати вивіску магазину (або просто навести на неї камеру, адже тепер Google додала функцію в пошукову програму), Google Об'єктив докладно розповість про напрямок його діяльності, графік роботи, місцезнаходження та інші деталі, а також поділиться відгуками відвідувачів про це місце. Система штучного інтелекту вдосконалюється і що більше користувачі знімають об'єктів (зокрема тих самих), тим якісніше стають результати видачі.

Ще один додаток на мобільні пристрої – Camfind. Воно аналогічне до попереднього додатку. Однією з можливостей даного додатку є пошук товарів в інтернеті за їхніми фотографіями та штриховими кодами. Його основний принцип роботи це розпізнавання об’єкту на зображені, а для цього спочатку необхідно зробити фото на камеру смартфону. Додаток має зв’язок с пошуковою системою Google, де у кінці розпізнавання воно буде шукати знайдений об’єкт. Крім того, є і більш стандартні методи ідентифікації: сканування штрих- та qr-кодів, розпізнавання голосу.

Розпізнавання образів - перспективний напрямок у рекламі та маркетингу. Нейросети дозволяються за лічені години дізнатися про речі, для пошуку яких в інших випадках потрібна велика команда професіоналів і тижня, а то й місяці досліджень. Наприклад, сервіс YouScan, система моніторингу соціальних медіа, відстежує згадку про бренди в соцмережах. Причому робить це у тексті постів, а

й у фотографіях, і навіть допомагає зробити певні висновки про продукт. За допомогою розпізнавання образів на фото знайшли цікаву закономірність, пошук якої нікому б і не спало на думку: серед тварин коти частіше зустрічаються з технікою Apple, а собаки — з брендом Adidas. Ця незвичайна інформація може стати в нагоді для таргетування реклами.

Розпізнавання образів на камерах міського відеоспостереження - це, мабуть, найневідворотніша перспектива використання машинного зору. Окрім камер «Безпечного міста», в систему інтегровані поліцейські камери та апаратура, що з'явилася у знакових туристичних місцях за програмою «Відкрите місто Одеса».

Для реалізації системи відео спостереження був обраний китайський виробникапрограмно-апаратного комплексу HikVision, що зарекомендував себе у Києві. Він не тільки не поступається лідеру ринку Milestone, а й значно дешевшим. Зараз система здатна одночасно обробляти інформацію із 2025 відеокамер.

Здібності камер та програмного забезпечення вражають. Одна група приблизно з 50 пристроїв дозволяє фіксувати прохід через контрольну точку транспортного засобу, що знаходиться в розшуку або під арештом. Система порівнює номерний знак із базою даних номерів, наданою правоохоронцями, і у разі збігу моментально сигналізує оператору. Йому залишається лише доповісти про подію суб'єкту розшуку (поліції, СБУ тощо) і спостерігати за переміщенням об'єкта.

Розпізнавання образів вже стало справжнім проривом у медицині — у багатьох випадках комп'ютери помічають речі, які пропускають навіть найдосвідченіші лікарі. Вони виступають своєрідними помічниками, чия «технічна» думка підтверджує гіпотезу лікаря або дає привід для більш глибоких досліджень.

Створена Google нейромережа Inception, яка аналізує мікроскопічне дослідження біопсії лімфатичних вузлів у пошуку ракових клітин у молочних

залозах. Для людини це дуже довгий і трудомісткий процес, під час якого легко помилитися або пропустити щось важливе, оскільки в деяких випадках розмір зображення становить 100 000 х 100 000 пікселів. Нейросітка Inception забезпечує чутливість близько 92% проти 72% у лікаря. Нейросітка не прогавить всі підозрілі ділянки знімків, хоча і допускаються помилкові спрацьовування, які пізніше відфільтрує лікар.

Дрони із розпізнаванням зображень використовуються використовується як у розважальних, так і в наукових цілях. Наприклад, норвезька компанія eSmart Systems розробила інтелектуальні рішення для енергомереж. В рамках одного з їхніх проектів – Connected Drone – дрони використовуються для пошуку несправностей на лініях електропередач. Навчені розпізнавання елементів енергомереж, вони перевіряють цілісність дротів, ізоляторів та інших частин ЛЕП. Це особливо важливо для швидкої локалізації несправності, коли від лінії залежить електропостачання міста чи підприємства. Враховуючи, що часто ЛЕП побудовані у важкодоступних місцях, послати бригаду дронів на пошук несправності десь у тайзі чи горах набагато ефективніше, ніж послати бригаду людей.

Розпізнавання об'єктів в автомобілях – це необхідна частина систем безпеки ADAS (Advanced driver-assistance systems). ADAS можуть бути реалізовані як складними засобами, на зразок радара та інфрачервоних датчиків, так і за допомогою монокулярної камери. Однієї відеокамери цілком достатньо для того, щоб автомобіль у реальному часі зміг розпізнати пішоходів, знаки та світлофори. Однак таке розпізнавання «на льоту» — дуже ресурсомістке завдання, для виконання якого потрібний спеціалізований процесор. Toshiba протягом кількох років розвиває серію таких процесорів. Вони будують тривимірну модель на основі зображення з однієї камери, що рухається, і тим самим помічають невідомі перешкоди на дорозі. Адже якщо нейромережа навчена розпізнавати тільки людей, розмітку і знаки, то покришка або шматок огорожі, що лежить на асфальті, не будуть розпізнані і розцінені, як небезпека.

1.3 Постанова задачі

Проблема виявлення об’єктів є дуже актуальною у наш час. Існує безліч різноманітних задач які вирішуються шляхом обробки та аналізу зображення, про що свідчать багато опрацьованих досліджень у цій сфері, а також багато реалізованих систем. У рамках даної роботи, було обрано задачу дослідження методу підвищення ефективності виявлення об’єктів.

Виявлення об’єктів є однією з найпопулярніших систем для забезпечення безпеки, як окремих девайсів, так і великих систем. Основним джерелом реалізації системи є правоохоронні служби, які використовують їх для забезпечення порядку та безпеки в людських містах, а також компанії по розробці програмного забезпечення для виявлення облич, що слугують мірою безпеки девайсів. Тому й і не дивно, що дана проблема э дуже актуальною у наш час.

Отож, необхідно створити алгоритмічне забезпечення, яке буде приймати на вхід потокове зображення та знаходити об’єкти, які будуть класифікуватися відповідно до бібліотеки об’єктів. Для цього слід вирішити задачу пошуку об’єктів на зображені, що фіксується з відео потоку, для реалізації програми на платформі Raspberry Pi.

Результатом даного дослідження буде система, в основі якої лежить розроблений алгоритм. Дана система здатна фіксувати зображення в відео потоці та у реальному часі класифікувати об’єкти на зображені.

Метою ж даної роботи є підвищення ефективності методу виявлення об'єктів на зображенні у реальному часі та реалізація даної системи на платформі Raspberry Pi.

Основні завдання, які мають бути виконані для досягнення цієї мети:

а) розглянути теоретичні аспекти задачі аналізу зображення:

1) проаналізувати сучасні методи виявлення об’єктів на зображені;

2) оцінити доцільність використання даних методів.

б) розв’язати задачу аналізу зображення в реальному часі і знаходження об’єктів на ньому:

1) розробити формальну постановку задачі;

2) ознайомитися з сучасними підходами до розв’язку даної задачі;

3) оцінити можливі переваги та недоліки запропонованих методів розв’язку даної задачі;

4) на основі детального аналізу основних алгоритмів створити власний алгоритм, що враховує особливості та обмеження задачі аналізу відео потоку в реальному часі і знаходження рекламних оголошень відповідно до відео контенту;

в) програмно реалізувати розроблений алгоритм у вигляді системи.

Висновки до першого розділу

Виявлення об'єктів – це технологія, яка відноситься до галузі комп'ютерного зору та цифрової обробки зображень. Її завдання – виявлення на цифровому зображенні чи відео об'єктів певного виду (живі істоти, машини, будівлі). [17]

Для кожного виду об'єктів існує набір специфічних особливостей за допомогою яких можна класифікувати об'єкт. Наприклад, для ідентифікації особи такими особливостями будуть очі, губи, ніс, колір шкіри, відстань між очима. Ці ж специфічні особливості застосовуються і виявлення об'єктів. Або ще один приклад з розвитком технологій розумних міст потрібна наявність швидких та ефективних систем для розпізнавання об'єктів, щоб мінімізувати вимоги до апаратного забезпечення даних технологій, а також підвищити точність результатів їхньої роботи. Області застосування алгоритмів виявлення об'єктів

на зображенні різноманітні: медична допомога, роздрібна торгівля, охоронні системи, ідентифікація особи, віртуальні помічники та багато іншого [18].

Виявлення об'єктів на зображенні є актуальним завданням комп'ютерного зору. Вирішення даних завдань вимагає всебічного аналізу існуючих методів виявлення об'єктів на зображенні.

Існує багато методів, які дозволяють вирішити дану задачу. А саме методи які аналізують зображення у реальному часі, систему кодування відеопотоку, семантично аналізують його, шукають об’єкти, класифікують їх та ін. Виявлення об'єктів на зображенні є актуальним завданням комп'ютерного зору. Вирішення даних завдань вимагає всебічного аналізу існуючих методів виявлення об'єктів на зображенні.

За останні роки досягнуто значного прогресу в області виявлення об'єктів з використанням згорткових нейронних мереж. Сучасні детектори на основі цих мереж, такі як R-FCN, Faster R-CNN, Multibox, SDD та YOLO стали досить швидкими для використання в споживчих продуктах та для роботи на мобільних та вбудовуваних пристроях.

Отже, технології визначення об’єктів є актуальною темою. Вона торкається різноманітних галузей, починаючи з космічних розробок для слідкування за станом планети, закінчуючи органами безпеки, що використовують їх для регулювання злочинності. В даний час ще не створено оптимальний алгоритм виявлення об’єктів, який би підходив до усіх поставлених задач. Існує кілька провідних бібліотек, яким вдається розпізнавати образи на картинці, шукати ідентичні особи в мережі. Проте мета пошуку місцезнаходження об'єкта або його ідентифікації не досягнуто.

При реалізації виявляються наступні недоліки:

- відсутність доступного технічного обладнання. Камери, що передають дуже якісне зображення, необхідне для обробки системи, мають високу вартість. Відповідно, дуже мала частина суб'єктів світової економіки може собі

дозволити такі витрати. Часто вони є недоцільними. До того ж сама система розпізнавання об’єкті є досить витратною;

- проблеми низькоякісного обладнання зйомки, фактори освітленості, погодні умови, температурний режим.
- способи приховання зовнішність від камер – головні убори, окуляри, макіяж, перука, борода, вуса тощо. Способи подолання даних факторів на сьогоднішній день не знайдено.
- відсутність ресурсів у компаній-розробників. Для розробки інтелектуальних систем високого рівня потрібні великі інвестиції, вкласти які готовий кожен, оскільки відомий результат розробки.

Основна перевага методів, заснованих на машинному навчанні – відсутність необхідності попереднього виділення ознак шуканих об'єктів. Такі методи використовують переважно згорткові нейронні мережі та навчаються у ході своєї роботи. Для методів, заснованих на глибокому навчанні, потрібно попередньо виділити ознаки об'єкта, з якими алгоритм в подальшому порівнює ознаки, виділені у нового об'єкта. Таким чином, методи з використанням глибокого навчання краще використовувати у випадках, коли шуканих предметів невелика кількість або їх ознаки можна визначити. Методи з використанням машинного навчання підійдуть для ситуацій, коли об'єкти дуже різноманітні, або вони ще не визначені.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ВИЯВЛЕННЯ ОБ’ЄКТІВ

2.1 Методи виявлення об’єктів на зображені

Метою даної роботи є дослідження методів ефективності виявлення об’єктів. Існує багато методів для вирішення цієї задачі. Кожні з них мають свої переваги та недоліки, але найефективнішими методами для обробки зображення в реальному часі є методи YOLO та SSD.

Метод YOLO. В його основу йде розбиття картинки на рівномірну сітку S×S і для кожної комірки видавати чи є в ній центр якогось об'єкта, якого він класу і його точний прямокутник. Робота даного алгоритму починається з обробки вхідного зображення, побудови піраміди шарів на розбивання їх на сітку розміром SxS. Кожна сітка складає блок, який окремо де тестується на наявність самих об’єктів. Кожен об’єкт аналізується на знаходження центру та його класу. Після підтвердження класу, усі блоки додаються та аналізуються на збіги. Після аналізу, на дисплеї об’єкт індефікується та обводитися прямокутником. Алгоритм роботи зображено на рисунку 2.1.

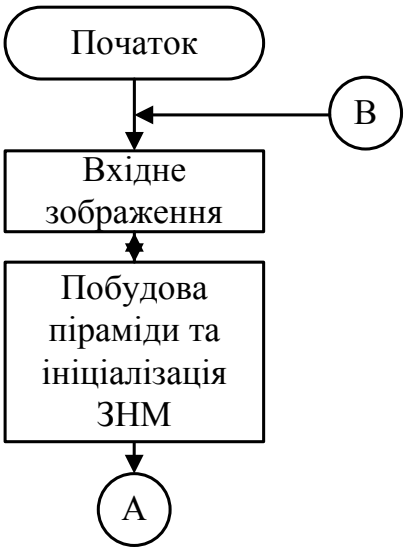

Рисунок 2.1 – Блок-схема алгоритму роботи методу YOLO

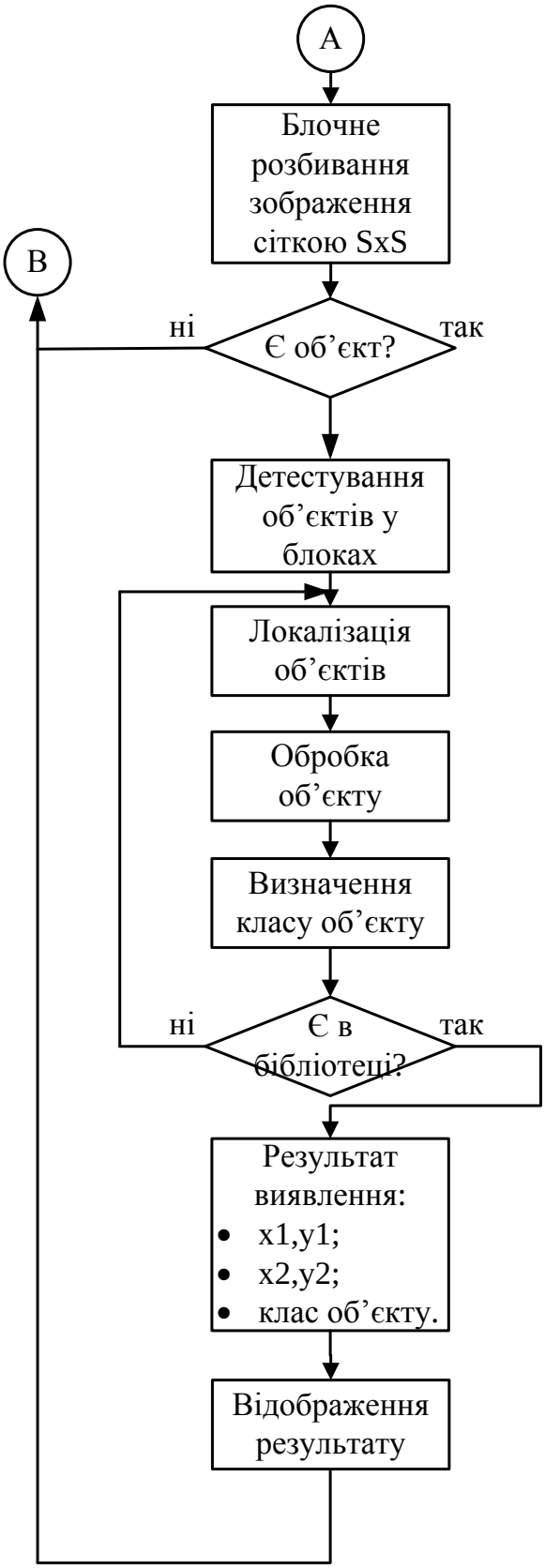

Рисунок 2.2 – Продовження рисунку 2.2

Для кожної комірки віддається набір виду: ⟨x,y,w,h,p⟩,

Де x,y - відносні координати центру об'єкта всередині комірки;

w,h - розміри об'єкта;

p - це впевненість мережі, що об'єкт є та його прямокутник заданий правильно.

Формально це ймовірність наявності центру об'єкта в комірці помножена на IoU (перетин ділене на об'єднання) областей реального об'єкта і передбаченого прямокутника x, y, w, h.

Візьмемо приклад, що зображений на рисунку 2.3.

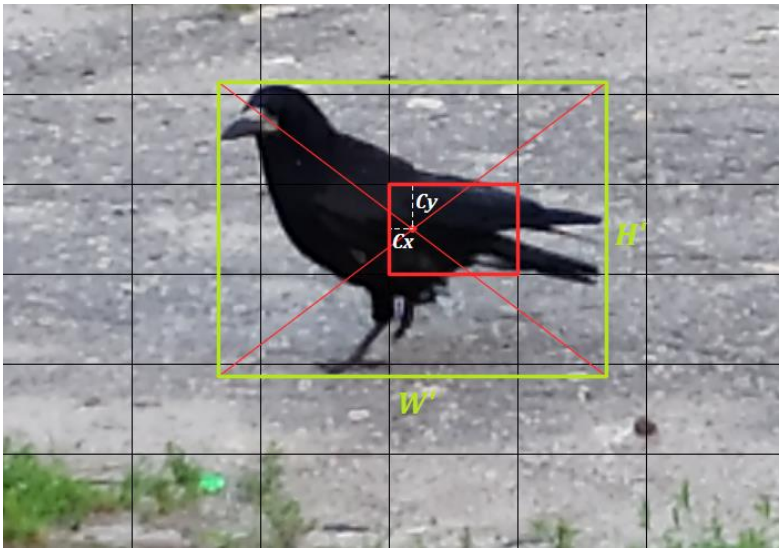

Рисунок 2.3 – Приклад роботи методу YOLO

Позначимо, що розмірами зображення є W,H. Візьмемо сітку розміром S×S, тоді розмір комірки буде $W_c = {}^{W}/_{S}, H_c = {}^{H}/_{S}$. Центр нашого об’єкту (в системі координат: (C'_x, C'_y)) потрапляє до комірки (3:2), розміри об'єкта: (W′,H′). Центр об'єкта нам потрібен у координатах від лівого верхнього кута комірки, а це буде:

$$C_x = C_{\grave{x}} - 3 \cdot W_c,$$

$$C_y = C_y^{`} - 2 \cdot H_c, \tag{2.1}$$

Якщо об'єкт на зображенні один, то всі комірки крім однієї (3:2) будуть утримувати нулі. А для комірки, де центр нашої ворони, замість нулів у тензорі буде:

1. П’ятірка $\langle \hat{x}, \hat{y}, \hat{w}, \hat{h}, 1 \rangle$:

$$\begin{aligned} \hat{x} &= {C_x}/{W_c}, \\ \hat{y} &= {C_y}/{H_c}, \\ \hat{w} &= {W^{`}}/{W}, \\ \hat{h} &= {H^{`}}/{H} \end{aligned} \tag{2.2}$$

2. Вектор ймовірностей для класів: $\hat{p}$=(0,…,0,1,0,…,0) - з одиницею дома класу птиці і нулями інших.

Штрафна функція логічним чином розбивається на суму штрафів по всіх комірках зображення:

$$\mathcal{L} = \sum_{i=1}^{S \cdot S} \mathcal{L}(i) \tag{2.3}$$

Осередки у нас двох типів: які містять і не містять центр об'єкта, і штрафувати їх треба по-різному. Формалізуємо цю відмінність і введемо індикаторну функцію:

$$x_i = \begin{cases} 1, & i - \text{я комірка, що містить об}'\text{єкт} \\ 0, & i - \text{я комірка, що не містить об}'\text{єкт} \end{cases}$$

Де i=1,…, (S·S).

Для комірок у яких є об'єкт, позначимо його прямокутником:

$$\widehat{R_\iota} = \left(\hat{x}_i, \widehat{y_\iota}, \widehat{w_\iota}, \widehat{h_\iota}\right) \tag{2.4}$$

При цьому для кожної комірки у нас є провісники, кожен з яких видає свій прямокутник. Позначимо прямокутник від j-го провісника в i-ї комірки:

$$R_{ij} = \left(x_{ij}, y_{ij}, w_{ij}, h_{ij}\right) \tag{2.5}$$

Якщо комірка містить центр об'єкта, то мережа повинна правильно передбачити клас цього об'єкта, тому першим складником у штрафній функції комірки буде штраф за класифікацію:

$$\mathcal{L}_{class}(i) = x_i \cdot \sum_{cl \in classes} (p_i(cl) - \widehat{p_\iota}(cl))^2 \tag{2.6}$$

Де χi - залишає це доданок тільки для комірок, де є об'єкт.

pi(cl),cl=1,…,C - передбачена ймовірність приналежності об'єкта класу cl.

$\widehat{p}_i$(cl) - одиниця для дійсного класу в який розмічений об'єкт у комірці, і нуль всім інших класів.

Якщо комірка містить центр об'єкта, то "кращий" провісник (той у якого максимальний IoU передбаченого прямокутника з розміченим) повинен добре передбачити координати центру та розміри прямокутника, решта провісників штрафувати не будемо:

$$\mathcal{L}_{coord}(i) = \sum_{j=1}^{B} x_{ij} \cdot \mathcal{L}_{coord}(i,j) \tag{2.7}$$

Штраф кращому провіснику задається як:

$$\mathcal{L}_{coord}(i,j) = \left(x_{ij} - \widehat{x_\iota}\right)^2 + \left(y_{ij} - \widehat{y_\iota}\right)^2 + \left(\sqrt{w_{ji}} - \sqrt{\widehat{w_\iota}}\right)^2 + \left(\sqrt{h_{ji}} - \sqrt{\widehat{h_\iota}}\right)^2 \qquad (2.8)$$

Оскільки χ_{ij} не нуль, тільки для комірок, які містять об'єкт, $L_{coord}(i)$ ненульова, також тільки для таких комірок. Також квадратне коріння в розмірах введено, щоб однакові помилки розмірів для маленьких і великих прямокутників штрафувалися по-різному

За впевненість будемо штрафувати по-різному у випадку, якщо в осередку є центр об'єкта і якщо ні. Якщо є, штраф вважаємо як:

$$\mathcal{L}_{conf}(i) = \sum_{j=1}^{B} x_{ij} \cdot (C_{ij} - \widehat{C_{\iota j}})^2 \qquad (2.9)$$

Де C_{ij} - це впевненість, яку видає мережа для і-ої комірки j-м провісником, а $\widehat{C_{\iota j}}$- це IoU між реальним прямокутником і тим, що видав провісник.

Якщо в осередку немає центру об'єкта, то штрафуємо:

$$\mathcal{L}_{noobj}(i) = (1 - x_i) \cdot \sum_{j=1}^{B} \left(C_{ij}\right)^2 \qquad (2.10)$$

Об'єднаємо всі штрафи в одну штрафну функцію для комірок:

$$\mathcal{L}(i) = \mathcal{L}_{class}(i) + \lambda_{coord} \cdot \mathcal{L}_{coord}(i) + \mathcal{L}_{conf}(i) + \lambda_{noobj} \cdot \mathcal{L}_{noobj}(i) \qquad (2.11)$$

Оскільки осередків багато, а об'єктів малюнку зазвичай мало, то автори вибирають λnoobj=0.5, щоб мережа вирішила, що дешевше завжди видавати, що

об'єктів немає. Так само помилка на координатах зазвичай мала (самі координати та розміри у нас із відрізка [0,1]), тому обираємо λcoord=5.

Перевагами даного методу є :

- швидкість визначення об’єкту в режимі реального часу;
- принцип роботи методу передбачає введення відразу всього зображення, яке проходить через згорткову нейронну мережу лише один раз.

Недоліком є:

- згіршення якісті зображення.

Для того щоб вирішить цей недолік, візьмемо ще один метод визначення об’єктів – SSD.

Метод SSD використовується для опису архітектур, у яких використовується одна згорткова нейрона мережа для безпосереднього передбачення розташування областей та їх класів, без застосування другого етапу класифікації. У цьому вся методі на виході нейронної мережі формується кілька тисяч прогнозів для можливих регіонів розташування об'єктів різної форми в різних масштабах, далі з допомогою придушення немаксимумов відбувається вибір кількох найімовірніших областей. Така єдина структура, одночасно з урахуванням різних масштабів зображення забезпечила методу SSD найвищі показники за швидкістю та якістю виявлення об'єктів порівняно з іншими сучасними підходами.

Робота методу SSD базується на фіксованому наборі прямокутників, які перевіряють наявність об’єкта на кожному з них. Припустимо, у нас є деяка карта особливостей m×n, яка отримана з одного зі згорткових шарів нейронної мережі. Пройдемося по ній згорткою з ядром 3×3, яка на виході видавала 4+Cl каналів, де Cl - кількість класів .

Тобто ми розбиваємо наше зображення сіткою, тому що кожна особливість на виході згорткового шару вбирає інформацію про пікселі квадрата у вихідному зображенні і значить може детектувати об'єкт, що знаходиться в цьому квадраті.

Чим більш ранній шар ми використовуємо, щоб забрати карту особливостей, тим більший її розмір (тобто m і n) і тим менші за розміром об'єкти ми зможемо детектувати. Вірно і зворотне, якщо ми детектуємо об'єкти на карті особливостей з ядром фіксованого розміру 3×3, то щоб вміти знаходити великі об'єкти на зображенні, треба брати карти особливостей з останніх шарів нейронної мережі. Алгоритм роботи даного методу зображено на рисунку 2.4.

Виявлення об’єктів починається з:

1 Вихідне зображення проходить через ряд згорткових шарів, що в результаті дає набір карт ознак для різних масштабів (наприклад, 19x19, 10x10, 5x5 і т.д.).

2. У кожній точці кожної карти ознак застосовується згортковий фільтр розміру 3*3 для отримання безлічі прямокутників, що описують.

3 Для кожного прямокутника одночасно оцінюються просторове зміщення ймовірність знаходження об'єкта

4. У процесі навчання справжні описуючі об'єкти прямокутники зіставляються передбаченими для виключення пізніх виявлень.

На відміну від R-CNN, де у регіонах-кандидатах є хоча б мінімальна ймовірність знаходження об'єкта, в SSD крок фільтрації регіонів відсутня. В результаті генерується набагато більша кількість описують прямокутників на різних масштабах в порівнянні з R-CNN, і більша частина не містить об'єкт. З метою вирішення даної проблеми в SSD, по-перше, використовується придушення не максимумів об'єднання схожих один на одного прямокутників у один. По-друге, використовується техніка hard negative mining, згідно з якою на кожній ітерації навчання використовується лише частина негативних прикладів, у SSD відношення числа негативних прикладів до позитивних дорівнює 3 до 1.

Вибір регіонів-кандидатів та класифікація виконуються одночасно: при заданій кількості класів C кожен описує прямокутник пов'язаний з (4+C)-мірним вектором, який містить 4 координати та ймовірності для всіх класів. На останньому етапі застосовується функція softmax класифікації об'єктів.

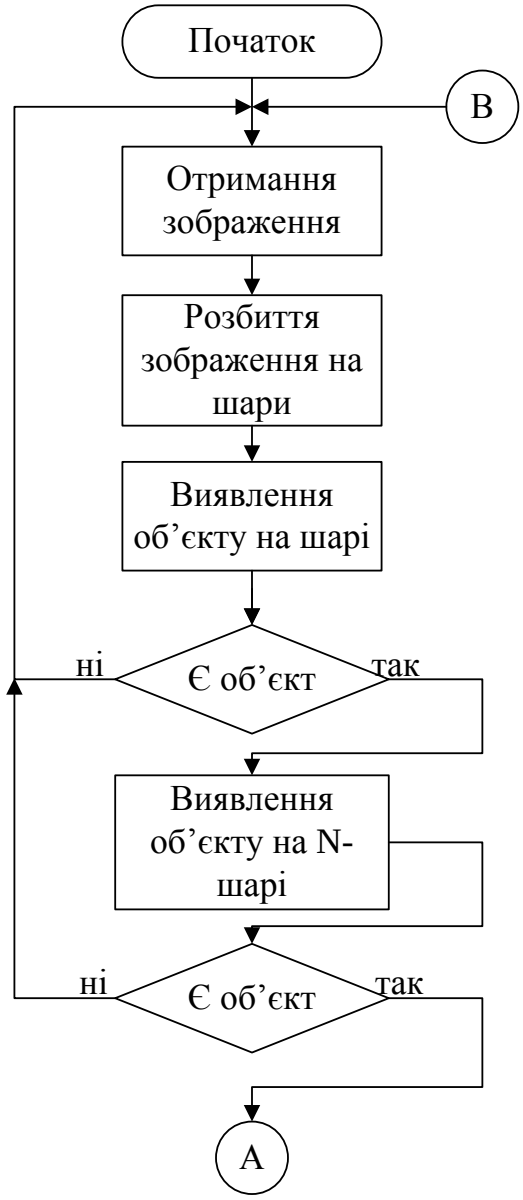

Рисунок 2.4 – Блок-схема алгоритму виявлення об’єктів

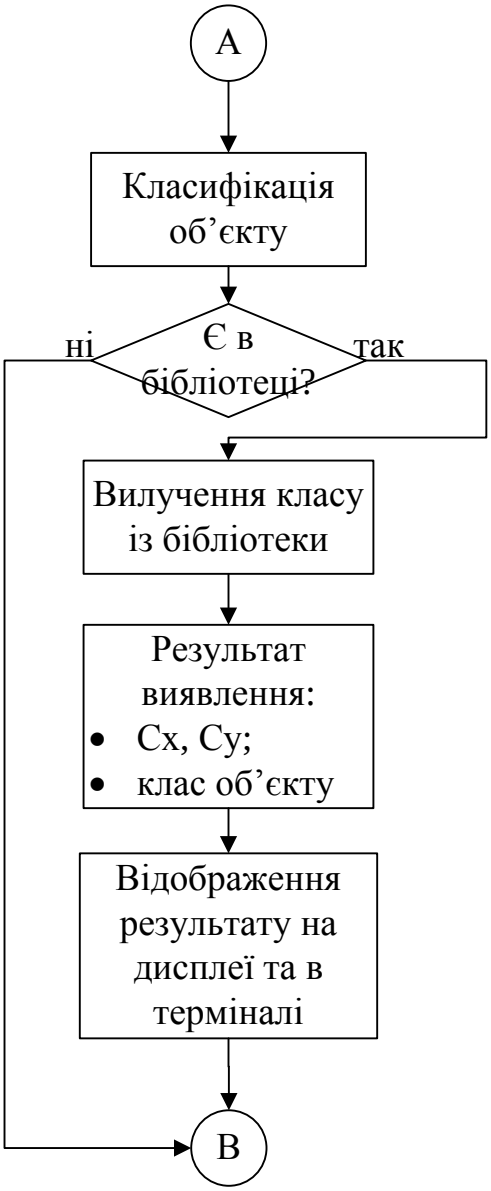

Рисунок 2.5 – Продовження рисунку 2.4

Для кожного об'єкта, розміченого на зображенні (рисунок 2.6), може бути кілька провісників, від яких ми готові і хочемо отримати опис об'єкту. Введемо індикаторну функцію x_{ij}, яка дорівнює одиниці, якщо i-ий анкер має IoU більше 0.5 з j-м об'єктом на зображенні, і нуль якщо ні. При цьому оскільки для одного реального об'єкта може бути кілька анкерів маємо на увазі, що взагалі кажучи $\sum_i x_{ij} \geq 1$

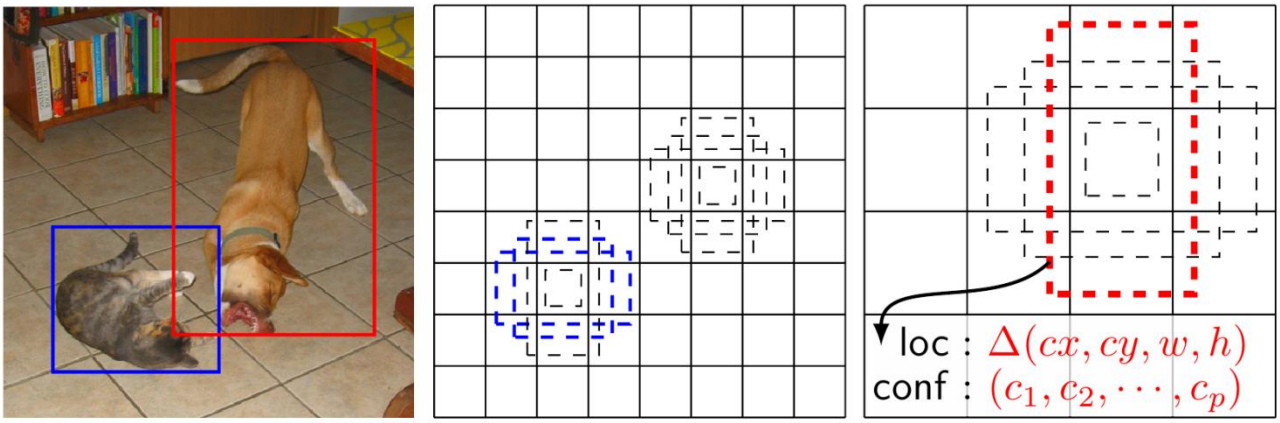

Рисунок 2.6 – Приклад роботи методу SSD

Перерахуємо всі анкери, для яких знайшлися реальні об'єкти:

$$N = \sum_i x_{ij} \tag{2.12}$$

Загальна функція втрат цілі є зваженою сумою втрати локалізації (loc) і втрати довіри (conf):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{2.13}$$

де N – кількість відповідних блоків за замовчуванням. Якщо N = 0, wet встановлює втрату на 0. Локалізаційна втрата є втратою плавної L1 між параметрами прогнозованого боксу (l) і основного поля істинності (g). Регресуємо до зміщень для центру (cx; cy) рамки за замовчуванням (d), а також для її ширини (w) і висоти (h).

$\mathcal{L}_{class}$ - відповідає за правильне визначення класу об'єкта і підсумовується по багатьох анкерах.

$\mathcal{L}_{loc}$ - це сума за всіма анкерами, яким зіставлено об'єкт, тобто $\chi_{ij}\neq 0$ і підсумовуються штрафи за помилки визначення прямокутника об'єкта.

Допустимо ви вибрали m=6 шарів з яких будемо забирати карти особливостей для детектування об'єктів задамося мінімальним Smin і максимальним Smax масштабами об'єктів, що детектуються. Ми будемо шукати об'єкти різних масштабів, використовуючи різні карти особливостей, для мінімального масштабу візьмемо карту особливостей з найближчого шару до входу мережі, потім з наступного і т.д. Якщо у нас вибрано m шарів, то рівномірно розкидавши з масштабів отримаємо:

$$S_k = S_{min} + \frac{S_{max} - S_{min}}{m-1} \cdot (k-1), k = 1, \dots, m \tag{2.14}$$

Обираємо $S_{min} = 0{,}1$ та $S_{max} = 0{,}95$.

Також для кожного масштабу, щоб вибирати не тільки квадратні анкери, задамося набором аспектів $A = \{a_r | r = 1, \dots, T\}$, тоді розміри анкерів будуть обчислюватися за формулами:

$$\begin{aligned} w_k^r &= S_k\sqrt{a_r}, \\ h_k^r &= \frac{S_k}{\sqrt{a_r}} \end{aligned} \tag{2.15}$$

Обираємо набір аспектів $A = \left\{1,2,3,\frac{1}{2},\frac{1}{3}\right\}$ та для a_r = 1 додають на кожний шар анкери з додатковім масштабом $S_k^{`} = \sqrt{S_k S_k + 1}$. Таким чином мі маємо по 6 анкерів для кожної особливості в карті.

Позиції анкерів обираються просто. Якщо мі прив’язуємо квадрати до шару з особливістю розміру m×n, то центри квадратів будуть у точках:

$$\left(\frac{i+0.5}{m}\right), \left(\frac{j+0.5}{n}\right), i = 0.1, \dots, m; j = 0.1, \dots, n \quad (2.16)$$

Архітектура SSD є найбільш підходящою для обробки зображень в режимі реального часу (особливо при використанні мереж MobileNet), але необхідно враховувати, що високі вимоги до точності не завжди можуть бути дотримані.

Цей метод вирішив проблему попереднього, а саме має такі переваги:

- швидкість визначення об’єкту в режимі реального часу;
- один прохід через згорткову нейрону мережу;
- пошук об’єктів на шарах, що не погіршує якість вхідного зображення. Тобто більше ящиків за замовчуванням призводить до більш точного виявлення, хоча це впливає на швидкість;
- наявність MultiBox на декількох шарах також призводить до кращого виявлення завдяки тому, що детектор працює з функціями з декількома дозволами;
- SSD змішує об'єкти зі схожими категоріями (наприклад, тварини). Це, ймовірно, тому що місця є спільними для кількох класів;
- SSD-500 (варіант із найвищою роздільною здатністю, що використовує вхідні зображення 512x512) досягає найкращого значення MAP на Pascal VOC2007 при 76,8%, але за рахунок швидкості, де його частота кадрів падає до 22 кадрів за секунду. SSD-300, таким чином, є кращим компромісом з 74,3 мАП при 59 кадрах в секунду.

До недоліків відноситься те, що SSD дає найгіршу продуктивність для невеликих об'єктів, оскільки вони можуть відображатися не на всіх картах об'єктів. Збільшення роздільної здатності вхідного зображення полегшує цю проблему, але не вирішує її повністю

2.2 Побудова алгоритму виявлення об’єктів на платформі Raspberry Pi

Отже, заберемо до купи усі оглянуті алгоритми та методи розв’язання даної задачі і покажемо загальну схему алгоритму виявлення об’єктів на зображені.

Крок 1. Пропускаємо вихідне зображення через ряд згорткових шарів та отримуємо набір карт ознак для різних масштабів.

В результаті аналізу існуючих методів та аналізу їх переваг та недоліків ддля реалізації системи обираємо метод SSD300. Основна перевага даної версії методу полягає в тому, що вхідне зображення приводиться до розміру 300x300 пікселів. Що у свій час покращує час на обробку даного зображення.

Крок 2. Застосовуємо згортковий фільтр розміром 3*3 для кожної точці кожної карти ознак для отримання прямокутників.

Крок 3. Для кожного прямокутника одночасно оцінюється просторове зміщення ймовірність знаходження об'єкта

Крок 4. У процесі навчання справжні описуючі об'єкти прямокутники зіставляються передбаченими для виключення пізніх виявлень.

Як модель для навчання виявлення об’єктів використовуємо модель COCO.

Загальні об'єкти в контексті (COCO) - це велика колекція зображень, які були позначені та позначені дослідниками з Microsoft, Facebook та низки університетів. Він містить понад 200 000 зображень приблизно з 90 категоріями об'єктів.

Моделі виявлення або класифікації об'єктів можуть бути навчені на наборі даних COCO, щоб дати нам відправну точку для розпізнавання повсякденних об'єктів, таких як люди, автомобілі, велосипеди, чашки, ножиці, собаки, кішки тощо.

Крок 5. Порівняємо клас об’єкту на зображенні з класом об’єкту моделі COCO.

Крок 6. Об’єднуємо усі шари зображення та придушуємо немаксимуми.

Враховуючи велику кількість блоків, згенерованих під час прямого проходу SSD під час виведення, важливо скоротити більшу частину обмежувальних блоків, застосовуючи метод, відомий як не максимальне придушення: коробки з порогом втрати достовірності менше Конектикут (наприклад, 0,01) і менш ніж (наприклад, 0 ,45) відкидаються, і тільки верхні прогнози зберігаються. Це гарантує, що у мережі зберігаються лише найімовірніші прогнози, а гучніші - видаляються.

Крок 7. Виводимо результат.

Отже роботу SSD можна описати таким чином (рисунок 2.7).

Також важливою частиною системи виявлення об’єктів є саме платформа, на якій вона і буде реалізована. Raspberry Pi – це одноплатний комп'ютер від Raspberry Pi Foundation, який працює на базі процесору ARM Cortex-A72 1,5 ГГц.

Мікроконтролери можуть одночасно виконувати лише одне завдання і добре з цим справляються. А одноплатні комп'ютери виконують програми в рамках операційної системи (найчастіше Linux), мають більшу продуктивність і широкі мультимедійні можливості.

Даний комп’ютер має багато переваг, основні з них:

- мобільність та можливість використовувати як самостійну систему, так і вбудовану. Тобто даний комп’ютер можна з легкістю використовувати як частину роботу;
- міцність процесору;
- можливість вдосконалення апаратури.

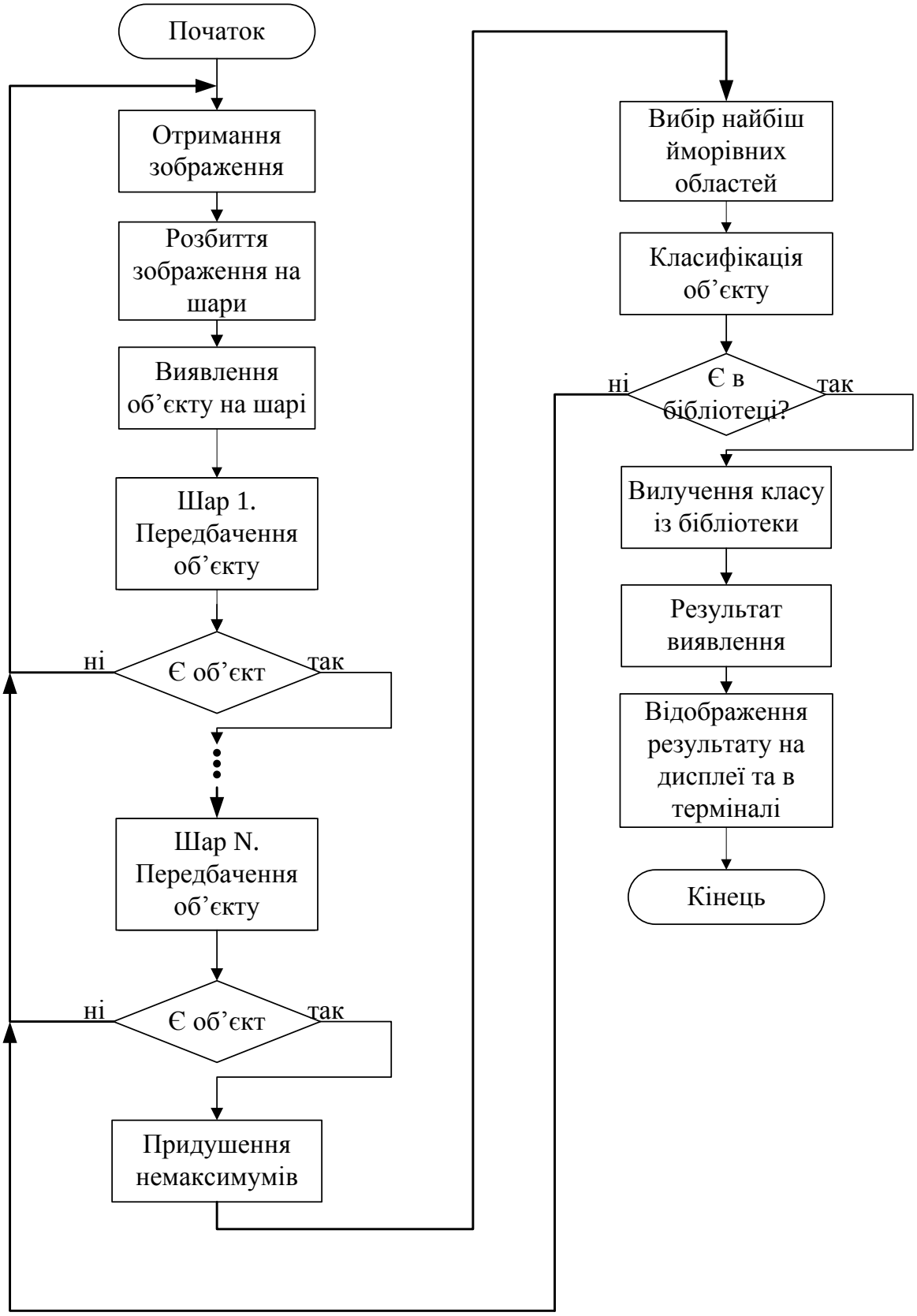

Рисунок 2.7 – Блок-схема алгоритму роботи методу виявлення об’єктів на платформі Raspberry Pi

2.3 Програмне забезпечення

Реалізація методу SSD здійснена на мові програмування Python 3 з використанням бібліотек глибокого навчання Tensor Flow та Keras.

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.

Для початку роботи нам необхідно підключити всі бібліотеки.

```
import os
import argparse
import cv2
import numpy as np
import sys
import time
from threading import Thread
import importlib.util
```

Для роботи з обробкою потокового відео, що буде надходити в веб-камери необхідно визначити клас.

```
class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self,resolution=(640,480),framerate=30):
```

Ініціалізація камери та потоку зображення.

```
        self.stream = cv2.VideoCapture(0)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC,
cv2.VideoWriter_fourcc(*'MJPG'))
        ret = self.stream.set(3,resolution[0])
        ret = self.stream.set(4,resolution[1])
```

Програма захоплює кадр із камери за допомогою OpenCV, змінює розмір кадру до 300x300 пікселів і передає отриманий тензор TensorFlow Lite. Функція invoke () повертає список виявлених об'єктів у кадрі, оцінку достовірності кожного об'єкта та координати їх рамок, що обмежують. Програма бере ці координати і малює зелений прямокутник навколо об'єкта перед відображенням користувача.

Початок роботи починається с отримання кадру з відео-потоку та зміни його до потрібних розмірів.

```
frame1 = videostream.read()

frame = frame1.copy()
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
frame_resized = cv2.resize(frame_rgb, (width, height))
input_data = np.expand_dims(frame_resized, axis=0)
```

Далі виконується фактичне виявлення, запустивши модель із зображенням як вхідним

```
interpreter.set_tensor(input_details[0]['index'],input_data)
interpreter.invoke()
```

Отримання результату виявлення.

Координати обмежувальної рамки виявлених об'єктів:

```
boxes = interpreter.get_tensor(output_details[0]['index'])[0]
```

Індекс класу виявлених об'єктів:

```
classes = interpreter.get_tensor(output_details[1]['index'])[0]
```

Впевненість виявлених об'єктів

```
scores = interpreter.get_tensor(output_details[2]['index'])[0]
```

При виявленні об’єктів у відео потоці, вони будуть обведені у прямокутник. Також інтерпретатор може повертати координати, які знаходяться за межами розмірів зображення, потрібно змусити їх знаходитися в межах зображення за допомогою max() і min()

```
ymin = int(max(1,(boxes[i][0] * imH)))
xmin = int(max(1,(boxes[i][1] * imW)))
ymax = int(min(imH,(boxes[i][2] * imH)))
xmax = int(min(imW,(boxes[i][3] * imW)))

cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)
```

Далі до прямокутника малюється етикетка, який вказує ім’я об’єкту. Воно розташоване в масиві "labels" та знаходиться за допомогою індексу класу.

```
object_name = labels[int(classes[i])]
label = '%s: %d%%' % (object_name, int(scores[i]*100))
```

Отримується розмір шрифту.

```
labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2)
```

Перевірка, що малюнок етикетки не занадто близько до верхньої частини вікна

```
label_ymin = max(ymin, labelSize[1] + 10)
cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED)
cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2)
```

Висновки до другого розділу

В даному пункті був складений алгоритми роботи системи виявлення об’єктів. Була написана реалізація даного алгоритму, показане програмне забезпечення розроблене на платформі Raspberry Pi.

Було проаналізовано два методи виявлення об’єктів на зображенні у реальному часі, визначені їх переваги та недоліки та був обран метод, на основі якого була розроблена система. Метод SSD є найефективнішим для реалізації потрібної задачі. Він затрачує найменше часу на аналіз зображення.

3 МОДЕЛЮВАННЯ

3.1 Моделювання системи та перевірка ефективності

Для оцінки якості виявлення об’єктів використовуються такі метрики як міра перетину найдених та еталонних прямокутників, які містять об’єкти (Intersecion, I), повнота (Recall, R) та точність (Precision, P) виявлення об’єктів.

Міра перетину найдених та еталонних прямокутників I показує, як чітко згорткова нейрона мережа знайшла прямокутних відносно прямокутника еталонної розмітки.

$$I = \frac{S_i}{S_j + S_{gt} - S_I}, \tag{3.1}$$

Де S_i - площа перетину істинного та обчисленого прямокутника;

S_j - площа найденого алгоритмом прямокутника;

S_{gt} - площа еталонного прямокутника (grounded truth).

Повнота R показує чуткість алгоритму до помилок 2-го роду, тобто пропусків, та дорівнює відношенню кількості правильно знайдених об’єктів к кількості цих об’єктів в еталонній розмітці.

$$\mathrm{R} = \frac{\mathrm{t_p}}{\mathrm{t_p} + \mathrm{f_n}}, \tag{3.2}$$

Де $\mathrm{t_p}$ - істино-позитивні (true positives) – це ті об’єкти, що очікували побачити та отримати на виході;

$\mathrm{f_n}$ - хибно-негативні (false negatives) – об’єкти, які ми очікували побачити, але алгоритм їх не визначив (пропуски).

Точність P показує чуткість алгоритму до помилок 1-го роду, тобто хибним спрацьовуванням та дорівнює відношенню кількості правильно знайдених об’єктів до загально кількості знайдених алгоритмом прямокутників.

$$P = \frac{t_p}{t_p + f_p}, \tag{3.3}$$

Де f_p - хибно-позитивні (false positives) об’єкти – такі,яких на виході бути не повинно, але алгоритм їх помилково повернув на виході (хибні спрацьовування).

Для перевірки ефективності візьмемо 3 методи: метод YOLO, SSD300 та SSD300, створений на платформі Rasberry Pi (далі SSD300_Raspberry). Параметри для оцінки ефективності розраховуємо по формулам 3.1-3.3.

Під час проведення експерименту оброблялося 100 зображень. Для побудови графіків змінюється пороговий коефіцієнт в алгоритмі виявлення об'єктів у діапазоні від 0 до 1 з кроком 0,01. Під пороговим коефіцієнтом розуміється мінімальне значення оцінки ймовірності, у якому буде прийнято рішення про виявлення об'єкта. На рисунках 3.1-3.3 представлені графіки залежностей точності, повноти та міри перетину від заданого порога.

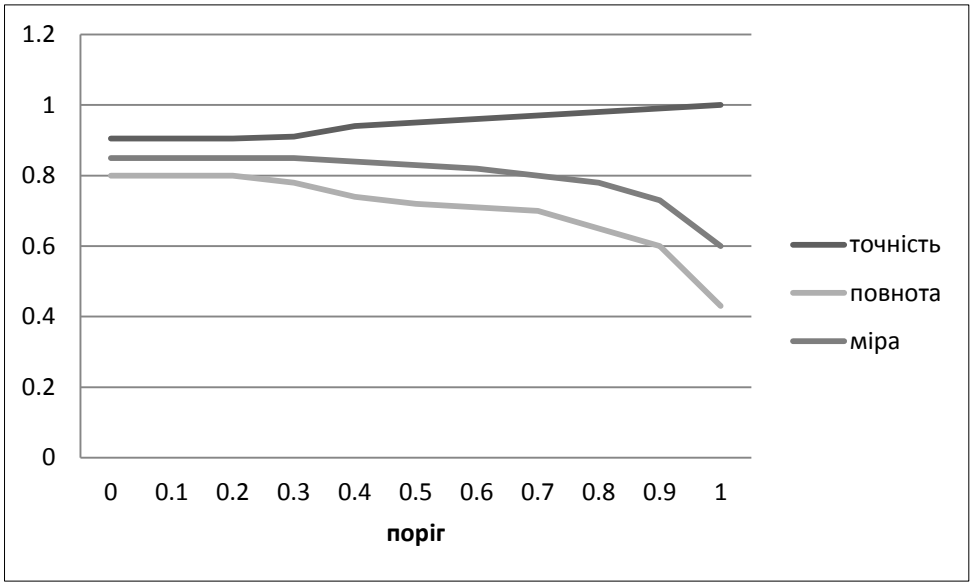

Рисунок 3.1 – Залежностей точності, повноти та міри перетину від заданого порога при використанні методу YOLO

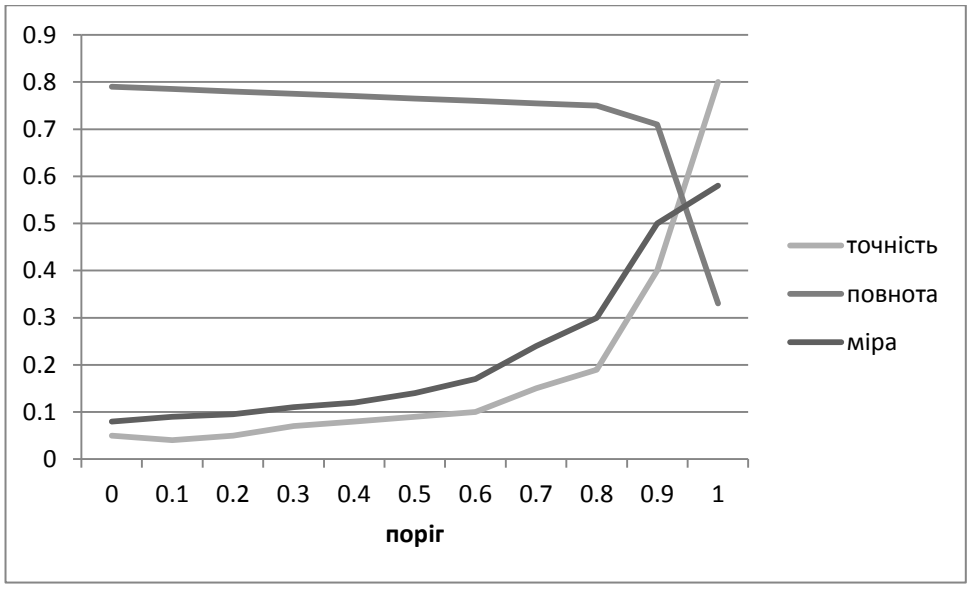

Рисунок 3.2 – Залежностей точності, повноти та міри перетину від заданого порога при використанні методу SSD300

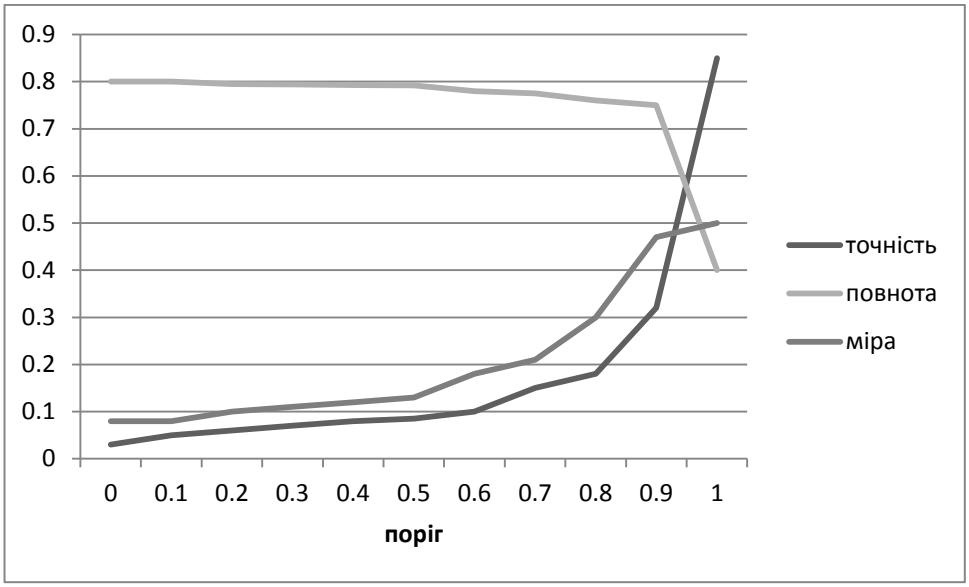

Рисунок 3.3 – Залежностей точності, повноти та міри перетину від заданого порога при використанні методу SSD300_Raspberry

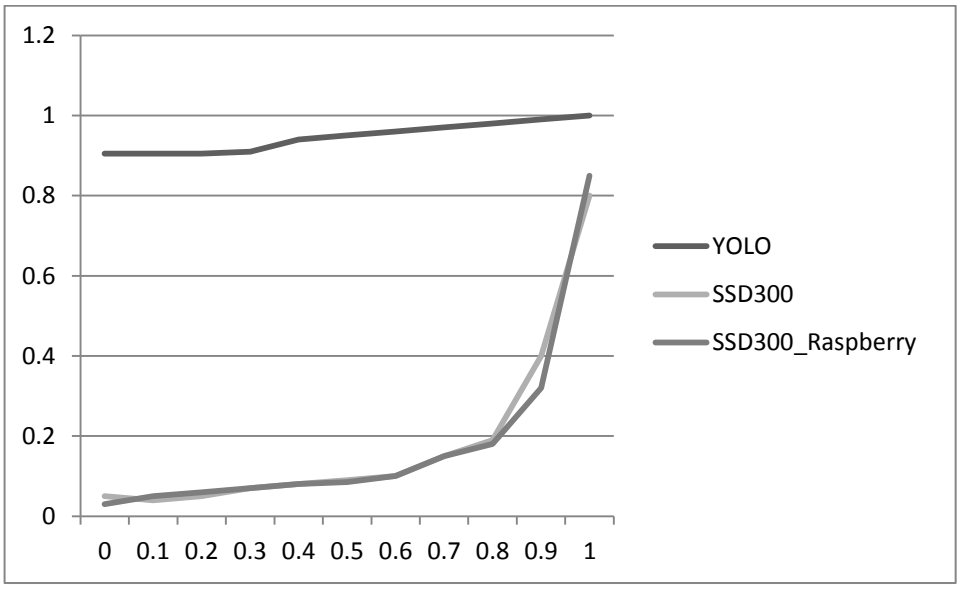

Рисунок 3.4 – Залежностей точності спрацьовування методів: YOLO , SSD300 та SSD300_Raspberry від заданого порога

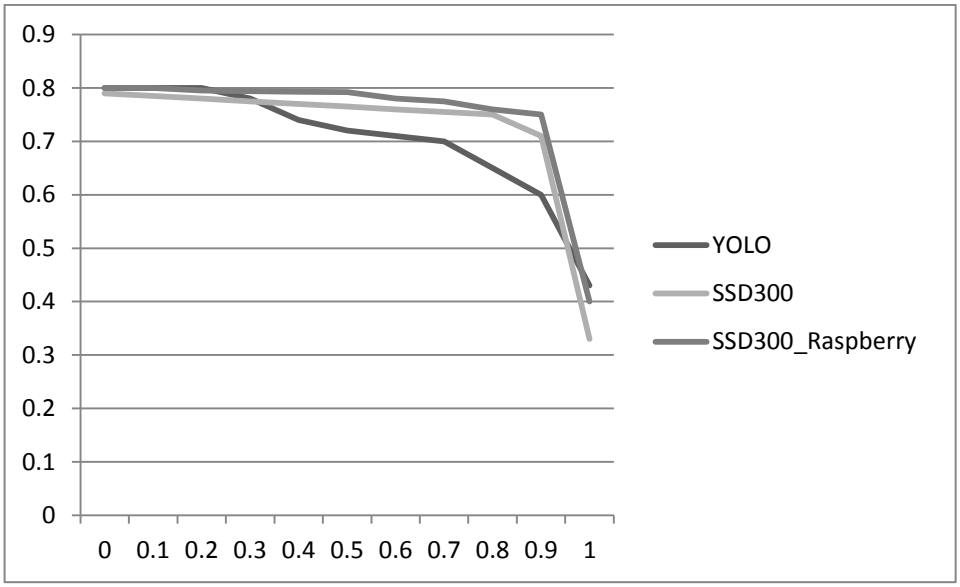

Рисунок 3.5 – Залежностей повноти спрацьовування методів: YOLO , SSD300 та SSD300_Raspberry від заданого порога

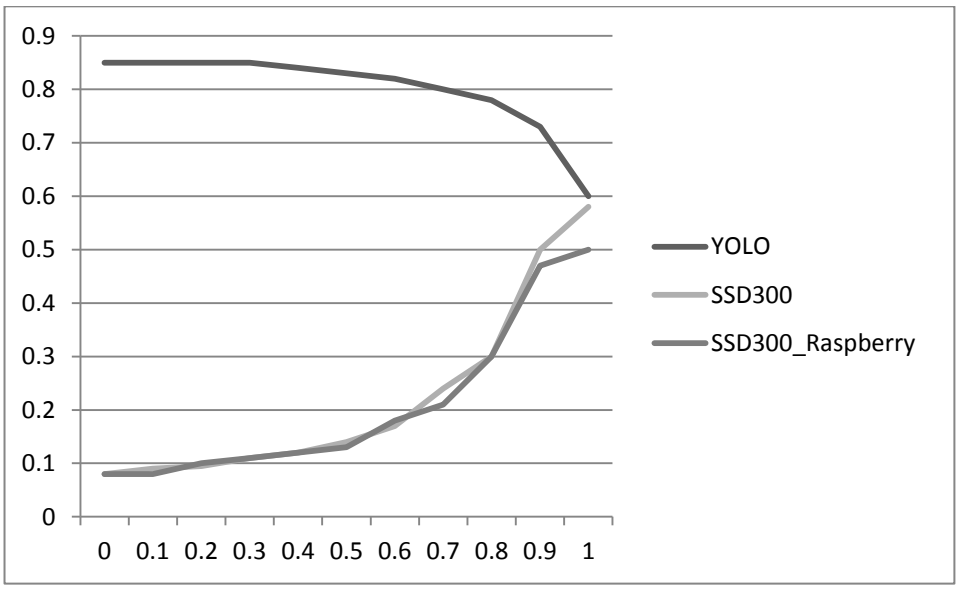

Рисунок 3.6 – Залежностей міри перетину найдених та еталонних прямокутників при використанні методів: YOLO , SSD300 та SSD300_Raspberry від заданого порога

Сегментація екземплярів – це новий напрямок. До епохи глибокого навчання йому було багато наборів даних, оскільки алгоритми були недостатньо хороші. В наші дні в основному використовуються MSCOCO, PASCAL, CityScape та всілякі набори даних, які включають завдання, яке називається сегментацією екземплярів. Основна ідея сегментації екземплярів полягає в тому, що ми повинні сегментувати кожен екземпляр кожної категорії, щоб, два об’єкта на зображенні мали ту ж мітку категорії.

Для оцінки якості роботи системи проводиться розрахунок метрики mAP (mean Average Precision), яка являє собою середнє значення AP по всьому класи об'єктів. AP (Average Precision) – середнього значення максимальної точності за різних значень повноти. Як інтегральні оцінки якості роботи детекторів використовувалися площа під графіком залежності точності від повноти (AUC – area under curve) та mAP. Названі показники представлені в табл. 3.1.

Таблиця 3.1 – Результати перевірки ефективності

Метод	Характеристики		
	AUC	mAP, %	Час, мс
YOLO	0,882	66,4	76
SSD300	0,573	72,1	58
SSD300_Raspberry	0,534	75,1	56

В результаті проведення оцінки якості роботи системи було виявлено, що метод SSD300_Raspberry на 0,2 мс спрацьовує швидкіше, а також на 3% точніше детектує об’єкти. З цього можна зробити висновки, що система на платформі raspberry Pi краще підходить для реалізації системи, що розпізнає об’єкти у реальному часі.

3.2 Тестування системи

Тестування – процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином.

Працездантість або працездатний стан — стан виробу (машини, деталі), при якому він спроможний виконувати певні функції при збереженні значень параметрів в межах, заданих нормативно-технічною документацією та/або конструкторською документацією.

Тестування працездатності системи розпізнавання образів відбувається на платформі Raspberry Pi. В термінал вводиться команда, яка організовує запуск та роботу файлу, який відповідає за роботу програми. Камера зчитує отримане зображення, обробляє його та виводить на дисплей результат.

Перевірка працездатності інтерфейсу відбувається через термінал. В якій запускається файл TFLite_detection_webcam.py. Після запуску файлу ніяких помилок не було виявлено, це зображено на рисунку 3.7.

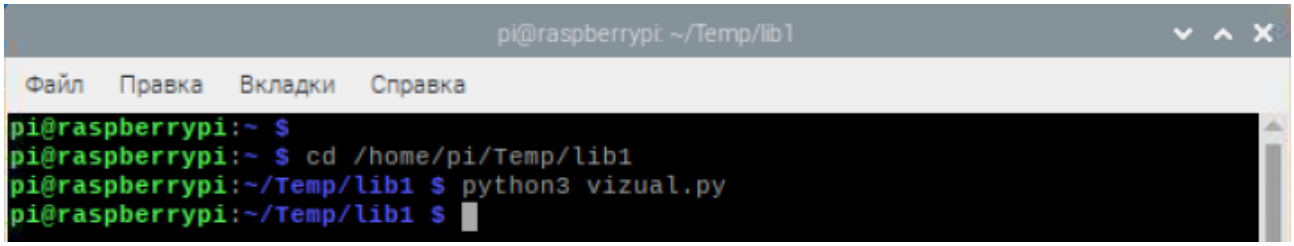

Рисунок 3.7 – Початок роботи програми в терміналі

Під час виконання програми камера постійно зчитує зображення на відео-потоці та обробляє його. Отримані результати вона передає у термінал, це зображено на рисунку 3.8.

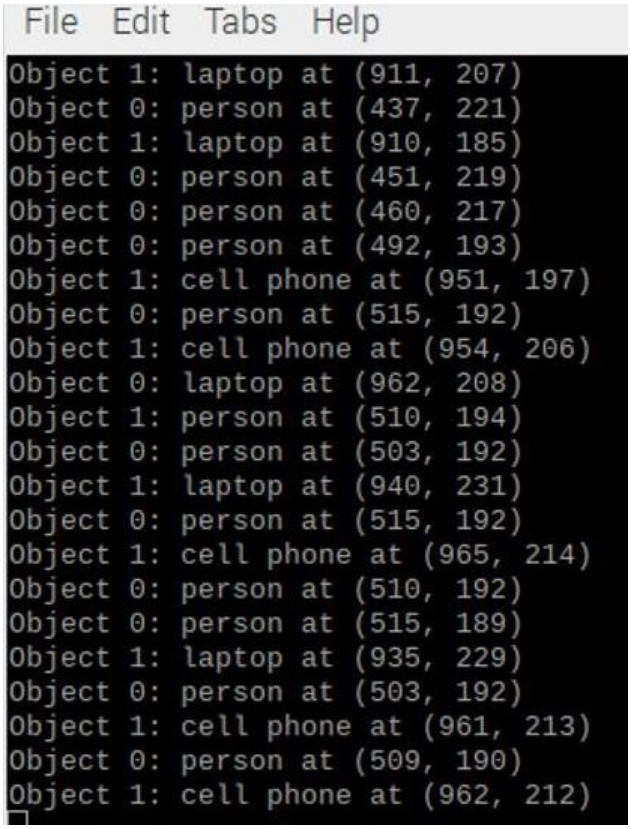

Рисунок 3.8 – Результат виконання програми в терміналі

Також одночасно з цим передається на дисплей, що показано на рисунку 3.9.

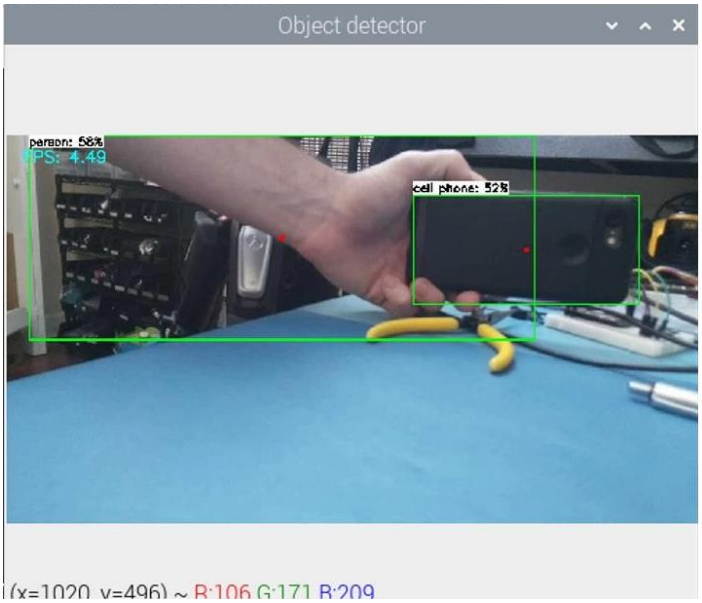

Рисунок 3.9 – Результат виконання програми на дисплеї

Висновки до третього розділу

В даному пункті була протестована система на працездатність та перевірена на ефективність.

По результатам можна сказати, що розроблений алгоритм на 3% точніше розпізнає об’єкти та 0,2 мс швидкіше.

ВИСНОВКИ

В результаті дослідження були проаналізовані методи виявлення об’єктів та засоби їх підвищення. Була розроблена система виявлення об’єктів на платформі Raspberry Pi.

Виявлення об'єктів – це технологія, яка відноситься до галузі комп'ютерного зору та цифрової обробки зображень. Її завдання – виявлення на цифровому зображенні чи відео об'єктів певного виду (живі істоти, машини, будівлі).

Існує багато методів, які дозволяють вирішити дану задачу. А саме методи які аналізують зображення у реальному часі, систему кодування відеопотоку, семантично аналізують його, шукають об’єкти, класифікують їх та ін. Виявлення об'єктів на зображенні є актуальним завданням комп'ютерного зору. Вирішення даних завдань вимагає всебічного аналізу існуючих методів виявлення об'єктів на зображенні.

За останні роки досягнуто значного прогресу в області виявлення об'єктів з використанням згорткових нейронних мереж. Сучасні детектори на основі цих мереж, такі як R-FCN, Faster R-CNN, Multibox, SDD та YOLO стали досить швидкими для використання в споживчих продуктах та для роботи на мобільних та вбудовуваних пристроях.

До основних результатів роботи відносяться:

- був розроблений алгоритм роботи системи виявлення об’єктів на платформі Raspberry Pi;
- був проведений аналіз методів виявлення об’єктів;
- була перевірена ефективність методів.

Також в результаті проведення дослідження було виявлено, що найефективнішим методом виявлення об’єктів на даний час є метод Single-Shot

MultiBox Detector (SSD). Даний метод має найбільшу швидкість обробки зображення, що також дозволяє використовувати його в режимі реального часу. Необхідно затвердити, що у даного методу є мінус – це зменшення якості зображення. Для вирішення цієї проблеми необхідно покращити якість обладнання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 1. Історія розвитку машинного навчання. : веб-сайт. URL : https://hi-news.ru /technology/trendy-mashinnoe-obuchenie.html

2. Дасиопулу, Стаматия и др.« Обнаружение семантических видеообъектов с помощью знаний ». IEEE Transactions on Circuits and Systems for Video Technology 15.10 (2005): 1210–1224.

3. Л. Шапиро, Дж. Стокман. Компьютерное зрение = Computer Vision. — М. : Бином. Лаборатория знаний, 2006

4. Комп'ютерний зір : веб-сайт. URL: https://znaimo.com.ua/Комп_ютерне_зір

5. Kirichenko L., Radivilova T., Zinkevich I. Comparative Analysis of Conversion Series Forecasting in E-commerce Tasks: веб-сайт. URL: https://link.springer.com/ chapter/10.1007/978-3-319-70581-1_16

6. Машинне навчання : веб-сайт. URL: https://uk.wikipedia.org/wiki/Машинне_навчання

7. Ciresan, Dan; Ueli Meier; Jonathan Masci; Luca M. Gambardella; Jurgen Schmidhuber (2011). Flexible, High Performance Convolutional Neural Networks for Image Classification. Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two 2. - p.1237–1242

8. A Gentle Introduction to Object Recognition with Deep Learning. : веб-сайт . URL: https://machinelearningmastery.com/object-recognition-with-deep-learning/.

9. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR. (2016)

10. How RetinaNet works? : веб-сайт. URL: https://developers.arcgis.com/python/guide/how-retinanet-works/

11. How single-shot detector (SSD) works? : веб-сайт. URL: https://developers.arcgis.com/python/guide/howssd-works/

12. Распознавание образов : веб-сайт. URL: http://ai-news.ru/raspoznavanie obrazov.html

13. Программа Deepface – определяет лица почти как живой человек : веб-сайт. URL: http://www.sciencedebate2008.com/deepface-defines-a-personas-a-living-person/

14. TensorFlow : веб-сайт. URL: https://www.tensorflow. org/

15. Распознаем лица на фото с помощью Python и OpenCV : веб-сайт. URL: https://habrahabr. ru/post/301096/

16. Веб-камера, Node.js и OpenCV: делаем систему распознавания лиц : веб-сайт. URL: https:// habrahabr.ru/company/ruvds/blog/335770/

17. Методы обнаружение объектов на изображении. URL: https://libeldoc.bsuir.by/bitstream/123456789/43782/1/Verkhov_Metody.pdf

18. Верхов, К.А. Обнаружение объектов на изображении с использованием машинного обучения/ К.А. Верхов // Новые информационные технологии в научных исследованиях: материалы XXV Юбилейной Всероссийской научно-технической конференции студентов, молодых ученых и специалистов; Рязань: ИП Коняхин А.В. (Book Jet), 2020 – С. 226-227.

ДОДАТОК А

Лістинг програми

```
import os
import argparse
import cv2
import numpy as np
import sys
import time
from threading import Thread
import importlib.util

class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self,resolution=(640,480),framerate=30):
        # Initialize the PiCamera and the camera image stream
        self.stream = cv2.VideoCapture(0)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC,
cv2.VideoWriter_fourcc(*'MJPG'))
        ret = self.stream.set(3,resolution[0])
        ret = self.stream.set(4,resolution[1])
       (self.grabbed, self.frame) = self.stream.read()
        self.stopped = False
    def start(self):
        Thread(target=self.update,args=()).start()
        return self
    def update(self):
        while True:
```

```
            if self.stopped:
                self.stream.release()
                return
            (self.grabbed, self.frame) = self.stream.read()
        def read(self):
            return self.frame
        def stop(self):
            self.stopped = True
    parser = argparse.ArgumentParser()
    parser.add_argument('--modeldir', help='Folder the .tflite file is located in',
                        required=True)
    parser.add_argument('--graph', help='Name of the .tflite file, if different than detect.tflite',
                        default='detect.tflite')
    parser.add_argument('--labels', help='Name of the labelmap file, if different than labelmap.txt',
                        default='labelmap.txt')
    parser.add_argument('--threshold', help='Minimum confidence threshold for displaying detected objects',
                        default=0.5)
    parser.add_argument('--resolution', help='Desired webcam resolution in WxH. If the webcam does not support the resolution entered, errors may occur.',
                        default='1280x720')
    parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to speed up detection',
                        action='store_true')
    args = parser.parse_args()
    MODEL_NAME = args.modeldir
    GRAPH_NAME = args.graph
```

```
        LABELMAP_NAME = args.labels
        min_conf_threshold = float(args.threshold)
        resW, resH = args.resolution.split('x')
        imW, imH = int(resW), int(resH)
        use_TPU = args.edgetpu
        pkg = importlib.util.find_spec('tflite_runtime')
        if pkg:
            from tflite_runtime.interpreter import Interpreter
            if use_TPU:
                from tflite_runtime.interpreter import load_delegate
        else:
            from tensorflow.lite.python.interpreter import Interpreter
            if use_TPU:
                from tensorflow.lite.python.interpreter import load_delegate
        if use_TPU:
            if (GRAPH_NAME == 'detect.tflite'):
                GRAPH_NAME = 'edgetpu.tflite'
        CWD_PATH = os.getcwd()
        PATH_TO_CKPT                                                    =
os.path.join(CWD_PATH,MODEL_NAME,GRAPH_NAME)
        PATH_TO_LABELS                                                  =
os.path.join(CWD_PATH,MODEL_NAME,LABELMAP_NAME)
        with open(PATH_TO_LABELS, 'r') as f:
            labels = [line.strip() for line in f.readlines()]
        if labels[0] == '???':
            del(labels[0])
        if use_TPU:
            interpreter = Interpreter(model_path=PATH_TO_CKPT,
                                      experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
```

```
    print(PATH_TO_CKPT)
else:
    interpreter = Interpreter(model_path=PATH_TO_CKPT)
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]
floating_model = (input_details[0]['dtype'] == np.float32)
input_mean = 127.5
input_std = 127.5
frame_rate_calc = 1
freq = cv2.getTickFrequency()
videostream = VideoStream(resolution=(imW,imH),framerate=30).start()
time.sleep(1)
cv2.namedWindow('Object detector', cv2.WINDOW_NORMAL)

while True:
    t1 = cv2.getTickCount()
    frame1 = videostream.read()
    frame = frame1.copy()
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_resized = cv2.resize(frame_rgb, (width, height))
    input_data = np.expand_dims(frame_resized, axis=0)
    if floating_model:
        input_data = (np.float32(input_data) - input_mean) / input_std
    interpreter.set_tensor(input_details[0]['index'],input_data)
    interpreter.invoke()
```

```
        boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box coordinates of detected objects
        classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected objects
        scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected objects
        for i in range(len(scores)):
            if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):
                ymin = int(max(1,(boxes[i][0] * imH)))
                xmin = int(max(1,(boxes[i][1] * imW)))
                ymax = int(min(imH,(boxes[i][2] * imH)))
                xmax = int(min(imW,(boxes[i][3] * imW)))

                cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)
                        object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
                label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
                labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
                label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window
                cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw white box to put label text in
                cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text
                xcenter = xmin + (int(round((xmax - xmin) / 2)))
                ycenter = ymin + (int(round((ymax - ymin) / 2)))
```

```
                cv2.circle(frame, (xcenter, ycenter), 5, (0,0,255), thickness=-1)
                print('Object ' + str(i) + ': ' + object_name + ' at (' + str(xcenter) + ', ' +
str(ycenter) + ')')

            cv2.putText(frame,'FPS:
{0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,2
55,0),2,cv2.LINE_AA)
            cv2.imshow('Object detector', frame)
            t2 = cv2.getTickCount()
            time1 = (t2-t1)/freq
            frame_rate_calc= 1/time1
            if cv2.waitKey(1) == ord('q'):
                break
        cv2.destroyAllWindows()
        videostream.stop()
```