

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук
(повна назва)

Кафедра _____ Штучного інтелекту
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____ другий (магістерський)

Виявлення вільних місць для паркування за допомогою
техніки комп'ютерного зору
(тема)

Виконав:
студент 2 курсу, групи СШІм-19-1
Селезень А. К.
(прізвище, ініціали)

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

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

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник к.т.н. Узлов Д. Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри

(підпис) В.О. Філатов
(прізвище, ініціали)

2020 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-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _____ Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_____» _____ 20 ____ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _____ Селезень Анастасії Костянтинівні
(прізвище, ім'я, по батькові)

1. Тема роботи _____ Виявлення вільних місць для паркування за допомогою техніки комп'ютерного зору

затверджена наказом університету від 30 жовтня 20 20 р. № 1497ст

2. Термін подання студентом роботи до екзаменаційної комісії 16 грудня 20 20 р.

3. Вихідні дані до роботи _____ Науково-технічні публікації, дані відомих наукових проектів щодо розробки нейронних мереж для розпізнавання об'єктів та їх використання для вирішення проблеми пошуку вільного місця для паркування автомобіля, дані статей, результати експериментальних досліджень. Документація мови програмування Python та бібліотек для розробки та тренування нейронних мереж Tensorflow та Keras.

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі та формалізована постановка задачі

2) Техніки комп'ютерного зору для розпізнавання образів

3) Імітаційне моделювання

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

Рисунок 1 – Архітектура мережі YOLOv3

Рисунок 2 – Архітектура мережі Faster R-CNN

Рисунок 3 – Алгоритм роботи мережі Mask R-CNN

Рисунок 4 – Результати виявлення об'єктів мережею Faster RCNN на наборі даних PKLot

Рисунок 5 – Результати виявлення об'єктів мережею Mask RCNN на наборі даних PKLot

Рисунок 6 – Результати виявлення об'єктів мережею Faster RCNN на наборі даних CNRPark

Рисунок 7 – Результати виявлення об'єктів мережею Mask RCNN на наборі даних CNRPark

Рисунок 8 – Результати виявлення вільного місця мережею YOLO

Рисунок 9 – Результати виявлення вільного місця мережею Faster RCNN

Рисунок 10 – Результати виявлення вільного місця мережею Mask RCNN

Рисунок 11 – Heat map для некоректно визначених об'єктів мережею Mask RCNN

Рисунок 12 – Результати виявлення об'єктів самостійно-навченою мережею Mask RCNN

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	Узлов Д. Ю.		
	.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі і постановка завдання	01.11.2020 – 04.11.2020	виконано
2	Аналіз проблеми пошуку вільних місць паркування	05.11.2020 – 7.11.2020	виконано
3	Аналіз нейронних мереж для розпізнавання об'єктів	8.11.2020 – 13.11.2020	виконано
4	Аналіз існуючих проблем даної предметної галузі	14.11.2020 – 18.11.2020	виконано
5	Розробка методу пошуку вільних місць паркування	19.11.2020 – 23.11.2020	виконано
6	Експериментальне моделювання та навчання моделі	24.11.2020 – 26.11.2020	виконано
7	Тестування і опрацювання імітаційної моделі	27.11.2020	виконано
8	Написання пояснювальної записки	28.11.2020 – 1.12.2020	виконано
9	Попередній захист	12.12.2020	виконано
10	Захист перед ЕК	16.12.2020	виконано

Дата видачі завдання 1 листопада 2020 р.

Студент _____
(підпис)

Керівник роботи _____
(підпис) _____
(посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 100 с., 1 табл., 53 рис., 26 джерел.

АЛГОРИТМ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, КОМП'ЮТЕРНИЙ ЗІР, ОБМЕЖУВАЛЬНА РАМКА, ПАРКУВАННЯ, РОЗПІЗНАВАННЯ ОБРАЗІВ, ФУНКЦІЯ ВТРАТ.

Об'єкт дослідження — виявлення вільних місць для паркування автомобіля.

Мета роботи — розробка алгоритму виявлення вільних місць для паркування автомобіля.

Методи дослідження базуються на теорії обчислювального інтелекту, а саме на методах теорії штучних згорткових нейронних мереж, а також обробка та аналіз отриманих результатів.

Здійснено моделювання процесу розпізнавання об'єктів на зображеннях та відео з використанням архітектур нейронних мереж Faster RCNN, Mask RCNN та YOLO.

На основі результатів виконаних досліджень розроблено алгоритм розпізнавання об'єктів, а саме визначення вільних та зайнятих місць для паркування.

РЕФЕРАТ

Пояснительная записка: 100 с., 1 табл., 53 рис., 26 источников.

АВТОСТОЯНКА, АЛГОРИТМ, КОМПЬЮТЕРНОЕ ЗРЕНИЕ, ОГРАНИЧИТЕЛЬНАЯ РАМКА, РАСПОЗНАВАНИЯ ОБРАЗОВ, СВЕРТОЧНАЯ НЕЙРОННАЯ СЕТЬ, ФУНКЦИЯ ПОТЕРЬ.

Объект исследования – выявление свободных мест для парковки автомобиля.

Цель работы – разработка алгоритма выявления свободных мест для парковки автомобиля.

Методы исследования базируются на теории вычислительного интеллекта, а именно на методах теории искусственных сверточных нейронных сетей, а также обработка и анализ полученных результатов.

Осуществлено моделирование процесса распознавания объектов на изображениях и видео с использованием архитектур Faster RCNN, Mask RCNN и YOLO.

На основе результатов выполненных исследований разработан алгоритм распознавания объектов, а именно определения свободных и занятых мест для парковки.

ABSTRACT

Explanatory note: 100 pages, 1 table, 53 figures, 26 sources.

ALGORITHM, BOUNDING BOX, CAR PARKING, COMPUTER VISION, CONVOLUTIONAL NEURAL NETWORKS, IMAGE RECOGNITION, LOSS FUNCTION.

The object of the research is the identification of vacant parking spaces.

The purpose of the work is to develop an algorithm for identifying vacant parking spaces.

The research methods are based on the theory of computational intelligence, specifically on the methods of the theory of artificial convolutional neural networks, as well as the processing and analysis of the obtained results.

The modeling of the process of object recognition in images and video was carried out using the Faster RCNN, Mask RCNN and YOLO architectures.

Based on the results of the research carried out, an object recognition algorithm has been developed, namely, the determination of vacant and occupied parking spaces.

ЗМІСТ

Перелік умовних позначень, скорочень і термінів	8
Вступ.....	9
1 Аналіз предметної галузі та формалізована постановка задачі	11
1.1 Загальний опис та існуючі методи вирішення задачі пошуку вільних місць для паркування.	11
1.2 Дротові системи на основі датчиків (wired sensor-based system).....	14
1.3 Бездротові магнітні сенсорні (wireless magnetic sensor-based).....	16
1.4 Системи на основі зображень або камер (image or camera-based systems) ..	19
1.5 Приклади існуючих систем «розумної парковки»	21
1.6 Постановка задач дослідження.....	23
2 Техніки комп'ютерного зору для розпізнавання образів.....	24
2.1 Комп'ютерний зір.....	24
2.2 Виявлення, розпізнавання об'єктів	29
2.3 YOLO. Архітектура.....	36
2.4 Fast R-CNN. Архітектура.....	53
2.5 Faster R-CNN. Архітектура.....	56
2.6 Mask R-CNN. Архітектура	58
3 Імітаційне моделювання	66
3.1 Використані відкриті набори даних	66
3.2 Опис та результати експериментів.....	69
Висновки	96
Перелік джерел посилання	97
Додаток А Відомість атестаційної роботи магістра.....	99

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Комп'ютерний зір – це міждисциплінарна наукова область, яка займається питаннями того, як комп'ютери можуть отримати розуміння на високому рівні за допомогою цифрових зображень чи відео;

Розпізнавання об'єктів – одна з найвідоміших і широко досліджуваних тем у галузі машинного зору, що займається виявленням та локалізацією деяких об'єктів, таких як людина, машина, автобус, ложка тощо, на зображенні;

Fast RCNN – Fast Region Based Convolutional Neural Networks – швидка регіональна глибока згорткова нейронна мережа;

Faster RCNN – Faster Region Based Convolutional Neural Networks – швидша регіональна глибока згорткова нейронна мережа;

RCNN – Region Based Convolutional Neural Networks – регіональна глибока згорткова нейронна мережа;

Mask RCNN – Mask Region Based Convolutional Neural Networks – регіональна глибока згорткова нейронна мережа з маскою;

YOLO – You only look once – «ви дивитесь лише один раз», нейронна мережа для виявлення об'єктів.

ВСТУП

Однією з важливих проблем сучасності є збір та обробка даних. Обсяги інформації зростають дуже швидко, і тому важливим є вибір методів для її опрацювання [1]. Саме це питання висвітлюється у розділах інтелектуального аналізу даних і машинного навчання.

Для власників автомобілів важливою проблемою в мегаполісах є пошук місць для паркування.

Пошук вільного місця для паркування в перевантаженому районі чи великій стоянці вимагає багато часу і турбує водіїв, особливо в години пік або коли цільова стоянка переповнена або майже заповнена. Крім того, відволікання уваги на пошук місця для паркування є однією з основних причин аварій на автостоянках, оскільки водії зосереджені на пошуку вільного місця і часто нехтують спостерігати за іншими водіями та пішоходами, які перебувають у русі. Така ситуація небезпечна, коли на автостоянці затори автомобільним рухом та пішоходами. Інтелектуальна система виявлення вільних місць для паркування може зробити цю процедуру більш приємнішою та позбавити водіїв значної частини вище вказаних клопотів, оперативно надаючи водіям точну інформацію про наявність місця на автостоянці.

Сьогодні вже існують різні підходи до вирішення цієї задачі. У багатьох існуючих системах використовуються сенсорні методи, такі як ультразвукові та інфрачервоні датчики. Але цей тип систем може вимагати великих витрат на встановлення та обслуговування. Тому в останні роки зростає інтерес до використання систем, що базуються на комп'ютерному зорі, завдяки його високим характеристикам та дешевим рішенням. Ці рішення можуть охоплювати велику кількість місць для паркування з мінімальною кількістю камер, на додаток до багатьох інших послуг, які можуть бути надані, такі як керівництво водієм та відеоспостереження. Система виявлення порожнього місця для паркування, яка може автоматично визначати порожні місця для паркування та

спрямовувати користувачів транспортного засобу до нього, є дуже бажаною, заощадить багато часу, грошей та зусиль [2]. У цьому контексті моя робота спрямована на впровадження системи виявлення вільних місць для паркування на основі технік комп'ютерного зору. Вони можуть надати інтелектуальне рішення, яке надійно підраховує загальну кількість вільних місць на стоянці, точно вказує їх місце розташування та виявляє зміни статусу в режимі реального часу. Тому пропонується система з використанням камери, або камер, яка використовувала б алгоритми комп'ютерного зору для виявлення вільних місць для паркування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ФОРМАЛІЗОВАНА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальний опис та існуючі методи вирішення задачі пошуку вільних місць для паркування.

Пошук порожніх місць для паркування є загальною проблемою в усьому світі в густонаселених містах. Ніхто не любить кружляти на автостоянках, шукаючи неіснуючі порожні місця для паркування, і ніхто не любить їздити, дивлячись на обидві сторони на наявність вільного місця, коли вони насправді всі зайняті. Люди витрачають в середньому 8–10 хвилин, у пошуках місця для паркування автомобіля. На це припадає близько 30% транспортних потоків у містах і сприяє перевантаженню транспорту в години пік. Багато часу та зусиль можна було б заощадити, якби інформація про паркування щодо наявності вільних місць могла би бути доступна за допомогою телефону водія або будь-яким іншим способом, доступним для будь-кого.

Щоб полегшити цю проблему та заощадити час та зусилля на пошук вільного місця для паркування, на сьогодні розроблюються спеціальні інформаційні системи відстеження вільних та зайнятих місць для паркування. Такі системи вимагають точної та сучасної інформації щодо заповнюваності місць для паркування, щоб забезпечити користувачів надійними вказівками щодо вільних місць. Такі системи мають різні технологічні підходи до вирішення цієї задачі. Але, більшість сучасних систем виявлення вільних місць для паркування використовують прості датчики для фіксування факту в'їзду та виїзду на стоянку, які не можуть визначити, де є вільні місця для паркування. Більше того, нинішні системи виявлення місць для паркування автомобілів не можуть визначити, коли більше ніж одне місце для паркування зайнято великогабаритним транспортним засобом, або погано припаркованим автомобілем [2].

Існуючі системи виявлення зайнятості місця для паркування використовують елементарні датчики відключення (trip sensors) у місцях в'їзду та виїзду з стоянок. Нажаль, цей тип системи виходить з ладу, коли транспортний засіб займає більше одного місця або коли на стоянці є різні типи паркувальних місць або нерівні місця. Нещодавно були проведені деякі дослідження щодо вдосконалення систем виявлення місць для паркування. Систематизовані системи маршрутування автостоянок (systematized car-park routing systems) можуть допомогти водіям миттєво і без проблем отримати оптимальне місце для паркування. Але існуючі підходи, які визначають вільні місця для паркування, або є дорогими через вимоги до обладнання для кожної стоянки, або не забезпечують повномасштабних рекомендацій щодо заповнення.

Система виявлення порожнього місця для паркування, яка може автоматично визначати порожні місця для паркування та спрямовувати користувачів транспортного засобу до нього, є дуже бажаною, заощадить багато часу, грошей та зусиль.

Як правило, інтелектуальні системи паркування отримують інформацію про доступні місця для паркування в певному географічному районі, і механізм реального часу розміщує транспортні засоби на вільних місцях. Використання точних датчиків, збору даних у режимі реального часу та мобільних телефонних систем бронювання, які дозволяють людям заздалегідь зарезервувати стоянку або дуже точно передбачають, де вони, ймовірно, знайдуть місце для паркування своїх транспортних засобів, – деякі критичні фактори, які слід враховувати в розвитку системи. Застосовуючи таку систему, як розумна парковка, таким чином, зменшуються викиди автомобілів у міських центрах, зменшуючи потребу людей у непотрібному об'їзді міських кварталів, які шукають місце для паркування. Це також дозволяє столицям ретельно керувати паркувальним запасом. Така система допомагає одній з величезних проблем з керуванням автомобілем у міських районах, таких як пошук порожніх місць для паркування та контроль за незаконним паркуванням.

Існуючі системи відстеження вільних та зайнятих місць для паркування класифікуються на чотири категорії на основі методів та технологій виявлення:

- системи на основі лічильників (counter-based systems);
- дротові системи на основі датчиків (wired sensor-based system);
- бездротові магнітні сенсорні (wireless magnetic sensor-based);
- системи на основі обробки зображень або відео з камер (image or camera-based systems).

Системи на базі лічильників покладаються на датчики на в'їзді та виїзді з паркінгу. Системи, що базуються на лічильниках, можуть надавати інформацію лише про загальну кількість вільних місць, а не направляти водіїв до точного розташування паркувальних місць, і такі системи не можуть застосовуватися до вуличних паркувальних майданчиків та житлових паркувальних місць [5].

Дротові та бездротові магнітні сенсорні системи покладаються на ультразвукові, інфрачервоні світла або бездротові магнітні датчики, встановлені на кожному паркувальному місці. Обидві системи застосовуються в практичному комерційному використанні, особливо в приміщеннях, таких як великі торгові центри. Однак такі методи вимагають встановлення дорогих датчиків на додаток до процесорних блоків та приймачів-передавачів для бездротових технологій. Системи на основі датчиків користуються високим ступенем надійності, але їх висока вартість встановлення та обслуговування обмежує їх використання для широкого застосування. Датчики в більшості бездротових систем мікроконтрольовані і, як правило, також включають мультисенсорні елементи управління. Вони ефективніші, ніж дротова система датчиків, але є дуже дорогими, ніж більшість інших систем. Деякі загальні приклади цих систем включають використання розширеної мережевої архітектури арбалета, яка використовує анізотропний магніторезистивний датчик магнітного поля разом з мікропроцесорним приймачем [3].

Порівняно з сенсорними системами, технології з використанням відеокамер відносно економічні, оскільки обидві функції загального

спостереження та виявлення зайнятості стоянок можуть виконуватися одночасно.

Переваги використання систем на основі камер у порівнянні з іншими існуючими системами потрійні. По-перше, немає потреби в додатковій інфраструктурі, за умови, що установа оснащена камерами відеоспостереження, що охоплюють місця для паркування. По-друге, камерні системи забезпечують точне розташування вільних місць для паркування, що є необхідною умовою для навігації транспортних засобів до вільних місць для паркування. По-третє, методи з використанням камер дуже легко можуть бути застосовні до вуличних та житлових місць для паркування.

1.2 Дротові системи на основі датчиків (wired sensor-based system)

Дротова система на основі датчиків – це мережева система датчиків та виконавчих механізмів, що базується на інфраструктурі, де кожне з місць для паркування в зоні паркування діє як окремий вузол. Ультразвукові датчики відстані розміщені на кожному вузлі. Ці датчики підключені до контролерів (наприклад, Arduino Duemilanove) у входних воротах за допомогою дротових з'єднань. Кілька вузлів датчика підключені до одного Arduino.

Плати Arduino підключені до системи біля входних воріт. У системі є чотири блоки камер, два біля в'їзних воріт і два на виході, камери на в'їзних і вихідних воротах мають спеціальний кут, щоб зафіксувати зображення обличчя водія та номерний знак транспортних засобів при прибутті та виїзді транспортних засобів та відправити їх до системи біля входних воріт для обробки за допомогою дротового з'єднання. Система на затворі також підключена до базової станції, якій вона надсилає постійні дані, звіт для моніторингу. Ця базова станція перебуває під наглядом людей і забезпечує втручання людей, коли це потрібно під час невідповідності даних на вихідних воротах. Динамік і принтер

також підключені за допомогою дротів до системи біля входних воріт, що забезпечує вказівки користувачеві до стоянки.

На в'їзді, а також на виїзді, де транспортні засоби повинні стояти для їхніх зйомок, позначена конкретна область. Ці демаркаційні регіони розгортаються за допомогою датчиків тиску, які виявляють присутність автомобіля в демаркаційній зоні та спрацьовують, активуючи камери для зйомки зображень. Ультразвукові датчики на вузлах не тільки допомагають оновлювати базу даних вільних та зайнятих місць, але і під час виходу з стоянки вони посилають сигнали про звільнення місця, спрацьовування здійснюється через контролери та світлодіоди від цього місця до виходу світяться, щоб вести транспортні засоби до виїзних воріт стоянки [8].

Кожне місце оснащено ультразвуковим датчиком відстані, який використовується для виявлення зайнятості місць, тобто незалежно від того, яке конкретне місце вільне чи зайняте. Зв'язок між датчиком та мікроконтролерами на входних воротах є дротовим, оскільки система відповідає архітектурі, що базується на інфраструктурі, і ми знаємо про всю область, що нас цікавить. Таким чином, дротове з'єднання є доречним, надійним та економічно вигідним. Ультразвуковий датчик, розміщений у прорізах, забезпечує точне вимірювання відстані до 3 метрів. Датчик працює, передаючи ультразвуковий спалах і забезпечуючи вихідний імпульс, який відповідає часу, необхідному для повернення ехо-сигналу до датчика. Відстань до цілі можна легко розрахувати, вимірявши ширину імпульсу. У цьому сценарії ціллю є автомобіль, припаркований у кожному місці, і відстань вимірюється та заповнюваність визначається на основі заздалегідь визначеного порогового значення.

Значення датчиків агрегуються у входних воротах, де розміщені мікроконтролери. Потім мікроконтролери передають дані базовій станції за допомогою послідовного зв'язку.

Датчики тиску також використовуються в системі на входних і вихідних воротах. Ці датчики підключені до аналогового штифта для вимірювання

вихідної напруги датчика. Такі датчики розгортаються в окреслених районах на в'їзді та виїзді для виявлення присутності транспортного засобу.

Світлодіоди розміщені на підлозі паркувальної зони, щоб направити всі виїзні машини, світяться світлодіоди від джерела виїзного автомобіля до пункту призначення, тобто вихідних воріт. Автомобіль може просто слідувати за прикріпленими світлодіодами, щоб дістатися до виїзних воріт без будь-яких затримок та незручностей. Гучномовець, підключений до в'їзних воріт, дає вказівки всім автомобілям, що в'їжджають, використовуючи вказівки, згенеровані системою, до призначеного вільного місця через синтетичний голос тексту в мову. Поряд із голосовими вказівками друкований чек із вказівками щодо напрямку та номером місця також автоматично генерується за допомогою принтера.

Така модель – це ефективна, економічна автоматизація системи паркування, яка забезпечує керівництво напрямком та безпеку, економить час та людські зусилля.

1.3 Бездротові магнітні сенсорні (wireless magnetic sensor-based)

Бездротові сенсорні мережі носять спеціальний характер, використовуючи велику кількість сенсорних модулів, які взаємодіють з центральною станцією нагляду (Central Supervisory Station – CSS). Датчик, який працює від акумулятора, має обмежені можливості обчислень та зв'язку. Кілька параметрів навколишнього середовища можна відстежувати, шляхом взаємодії мікровузла з різними датчиками. Бездротові сенсорні мережі мають багато переваг перед своїм дротовим аналогом, такі як гнучкість, властивий інтелект, нижча вартість, швидке розгортання та більше точок зондування, особливо в районах, які неможливо підключити за допомогою дротів. Завдяки цим перевагам WSN знайшли свій шлях у різних сферах застосування, таких як управління об'єктами, як управління дорожнім рухом та паркуванням, охорона здоров'я, моніторинг

навколишнього середовища, інтелектуальні будівлі, програми ліквідації наслідків стихійних лих тощо.

Система управління паркуванням автомобілів з використанням бездротових сенсорних мереж – це економічно ефективно та енергоефективно. Кожен елемент – це пристрій, що працює від акумулятора, оснащений лише одним пасивним датчиком зовнішнього освітлення для виявлення присутності або відсутності автомобіля. Датчик руху також передає таку інформацію, як час, коли машина була припаркована, а також стан справності акумуляторів. Радіомодулю вузла дозволяється бути неактивним через деякі рівні проміжки часу, тим самим роблячи систему енергоефективною. Така система повністю автоматизована і не вимагає присутності людини у точці входу або виходу. Система не тільки відображає статус доступності місця, але також надсилає таку інформацію, як позначка відведеного місця, час паркування, платіжна інформація та деталі направлення на мобільний телефон користувача за допомогою служби коротких повідомлень (SMS). Як правило, для того, щоб прив'язати місце для паркування до конкретного транспортного засобу, мікровузел датчика повинен бути присутнім як на транспортному засобі, так і на місці для паркування. Але існують модифікації, за допомогою яких можна вирішити задачу розміщуючи датчик лише біля стоянок. Це може бути досягнуте з використанням безкоштовної функції вхідних SMS, присутніх на всіх мобільних телефонах. Користувача просять ввести свій номер мобільного, коли він заходить на автостоянку. Потім номер використовується для позначення користувача на паркувальному місці. Це зменшить кількість мікродатчиків на 50%. Крім того, включення функції SMS допомагає уникнути використання паперових або пластикових карток, які в даний час використовуються для паркування та виставлення рахунків [4].

Структурна схема системи показана на рисунку 1.1. Дизайн системи відповідає ієрархічній архітектурі. Кожен з датчиків датчика розміщений у паркувальному гнізді. Сукупність таких куців утворює кластер і спілкується з

головою кластера. З точки зору апаратного забезпечення, кластерні головки ідентичні частинам, але відрізняються за функціональністю. Ці глави кластера передають інформацію в CSS (central supervisory station), центральний наглядовий пункт.

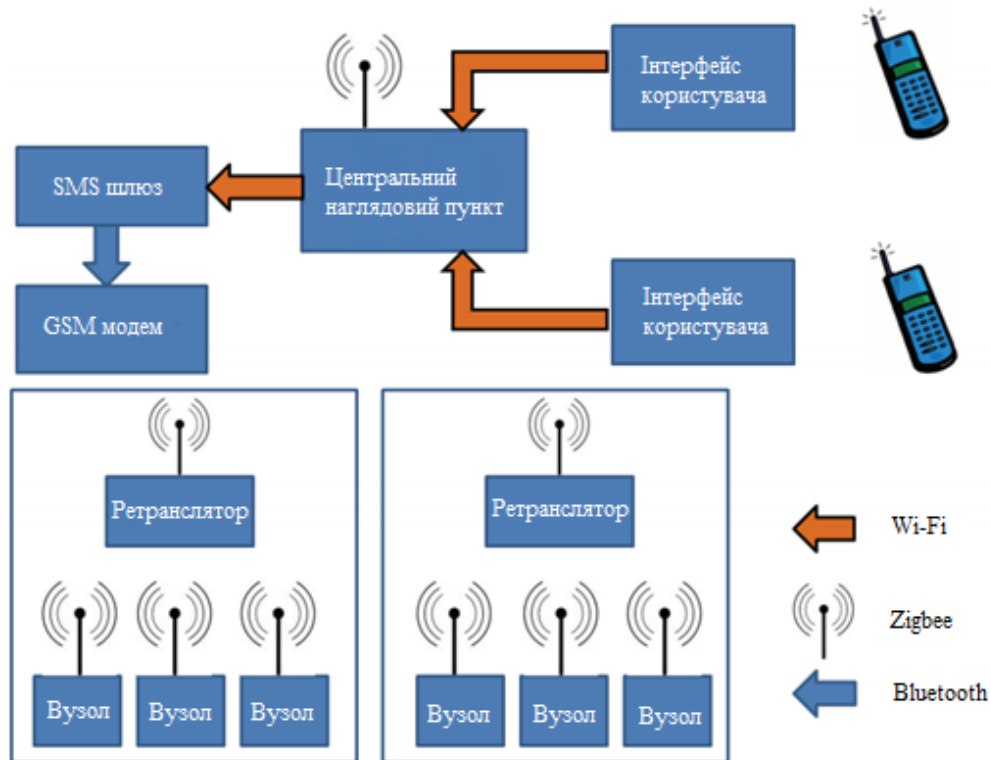


Рисунок 1.1 – Структурна схема системи з бездротовою сенсорною мережею

У цій системі прийнята ієрархічна схема спілкування. Це дозволяє долати енергетичні проблеми, які зазвичай зустрічаються в стратегії багатокаскадної маршрутизації. Набір мікровузлів згрупований у кластер. Потім ці набори, так само як і окремі вузли (сенсори), пересилають свої дані до призначеної головки кластера, розміщеної на зручній відстані як від CSS. Головка кластера просто повторно передає дані, які вона отримує від блоків, до CSS. Голова кластера з точки зору апаратного забезпечення ідентична частинам, розміщеним на стоянках, але функціонує набагато більше, як CSS. CSS – це потужний процесор,

здатний підтримувати базу даних та інтерфейси з GSM-модемом. Голова кластера та CSS живляться від мережі змінного струму.

Для виявлення присутності, або відсутності транспортного засобу на місці для паркування використовується метод вимірювання кількості навколишнього світла за допомогою датчика освітленості. Щоб світло падало лише в нормальному напрямку, датчик розміщується в центрі паркувального місця у фізичному корпусі під поверхнею землі. Фізичний корпус також гарантує захист датчика від фізичних пошкоджень. Коли транспортний засіб рухається над місцем, кількість навколишнього світла зменшується навколо ділянки. Використовуючи цей принцип та порівнявши інтенсивність світла з відповідним порогом, можна визначити, чи стоїть транспортний засіб над місцем. Для встановлення порогового значення світла на основі поточних світлових умов використовується алгоритм адаптивного порогового значення. Датчик контролює та реєструє показання світла протягом певного інтервалу часу. Записані значення усереднюються і порівнюються з попереднім порогом. Це робить систему розумною і дозволяє їй адаптуватися до різних умов освітлення навколо неї [9].

1.4 Системи на основі зображень або камер (image or camera-based systems)

У літературі пропонуються різні підходи до виявлення місця зайнятості на парковці. Використання алгоритму для порівняння еталонного зображення та наборів вхідних даних для обчислення транспортного засобу до площі пікселів місця для паркування за допомогою аналізу основних компонентів. Навчання байєсівського класифікатора для перевірки виявлення транспортних засобів з використанням кутів, країв та характеристик вейвлетів. Використання комбінації виявлення точок характеристики автомобіля та класифікацію кольорової гістограми. Інформаційна система заповнення місць для паркування автомобілів (COINS) інтегрує вдосконалені методи обробки зображень,

включаючи висівання, пошук меж, виявлення об'єктів та виявлення країв для надійного виявлення місця зайнятості. ParkLotD використовує крайові функції для виявлення місць на паркуванні. Також в деяких системах використовують байєсівські рамки, засновані на 3D-моделі паркувальних місць для виявлення місць, які можуть працювати день і ніч. Використовуються індивідуальні нейронні мережі, які навчені визначати заповнюваність паркування на основі вилучених візуальних характеристик з місць для паркування. Виявляється заповнюваність, комбінуючи фонове віднімання, використовуючи суміш Гаусса для виявлення та відстеження транспортних засобів, а також для створення тимчасової карти для виявлення місця паркування та залишення транспортних засобів. Навчають класифікатори SVM щодо різних текстурних особливостей та покращують ефективність виявлення за допомогою ансамблів SVM. Проводять аналіз траєкторії, використовуючи відео в режимі реального часу та тимчасові розбіжності на зображеннях, щоб визначити, зайняте чи вільне місце для паркування. Методи, згадані вище, базуються на ручно створених особливостях (таких як краї, колір, текстура) та відніманні фону, що робить ці методи сприйнятливими до різних погодних умов та змін освітленості [7].

CNN (Lecun et al., 1998) – це алгоритм машинного навчання, який використовує локальну просторову інформацію в зображенні та вивчає ієрархію все більш складних функцій, що автоматизує процес побудови ознак. Нещодавно фреймворки, засновані на CNN, досягли найсучаснішої точності в класифікації зображень та виявленні об'єктів. Розробляються децентралізовані рішення для візуального виявлення зайнятості місця для паркування з використанням глибокого CNN та розумних камер. Науковці тренують та допрацьовують мініатюрну версію AlexNet, mAlexNet для двійкової класифікації та повідомляють про точність 90,7% для процесу навчання передачі. Подібну роботу проводили в 2017, де автори розширюють набір даних CNRPark та порівнюють результати mAlexNet з AlexNet. Результати вказують на досягнуту точність передачі знань для AlexNet та mAlexNet в межах 90,52–95,60%

та 82,88– 95,28% відповідно. Інші науковці використовують каскадний класифікатор Haar-AdaBoosting для виявлення транспортних засобів на АЗС та підтвердження справжніх позитивних даних із глибоким CNN.

Підводячи підсумок, у літературі є чіткі докази того, що навчання функцій глибоким CNN перевершує звичайні методи, використовуючи ручно створені функції для виявлення зайнятості паркувальних місць з точки зору точності, надійності та навчання передачі. Однак усі вищезазначені системи на базі CNN вдосконалюють існуючі попередньо навчені мережі, що є додатковим етапом навчання, що вимагає додаткових зусиль.

1.5 Приклади існуючих систем «розумної парковки»

AUTOPARK – сенсорна, автоматизована, безпечна та ефективна система керування паркуванням. AUTOPARK має архітектуру сенсорної мережі на основі інфраструктури та складається з різних модулів, які взаємодіють між собою, щоб забезпечити безпечну та повністю автоматизовану систему паркування автомобілів в Індії.

Система складається з чотирьох камер для забезпечення безпеки на в'їзних і вихідних воротах. Усі камери підключені до комп'ютера який збирає дані з камер, а також місця для паркування. Вхідні ворота мають дві камери, одну сфокусовану для зйомки зображення обличчя людини, а другу для ідентифікації номерного знаку. Зображення зберігаються в базі даних для порівняння під час виходу.

Так само під час виїзду зображення людини та номерний знак транспортного засобу знову фіксується та порівнюється з уже наявними зображеннями в базі даних. Таким чином система забезпечує безпеку та запобігає розкраданню транспортних засобів. Неподалік є базова станція, за допомогою якої працює людський помічник, і до нього звертаються у разі будь-

якого невідповідності під час виїзду зі стоянки, щоб негайно допомогти людині, щоб уникнути нещасних випадків або затримок.

Система була протиставлена звичайній паркувальній системі щодо часу та людських зусиль, які вона економить. У звичайній системі паркування автомобіль, що в'їжджає, не має інформації про будь-яке вільне місце для паркування, доступне в зоні паркування. Він шукає вільне місце для паркування, яке рухається в різних напрямках, щоб дістатися до вільного місця, це призводить до більш тривалого часу пошуку у випадках, коли зона паркування дуже зайнята або сама зона паркування дуже велика. Ця система AUTOPARK допомагає зменшити час та зусилля на пошук, надаючи інформацію, а також вказівки щодо напрямку до вільного місця для паркування [3].

Інша розумна система паркування використовує бездротові сенсорні мережі.

Ідея цієї системи полягає в тому, що коли користувач заходить на автостоянку, біля входу буде клавіатура та дисплей. Водій набирає номер свого мобільного телефону за допомогою клавіатури. Після успішного введення телефонного номера на моніторі відображатимуться ідентифікатори найближчого порожнього місця для паркування, час в'їзду та напрямок маршруту, а також вони будуть відправлені на мобільний телефон користувача за допомогою SMS. Для того, щоб включити функцію SMS, до CSS підключений GSM-модем. Шлюз SMS на основі Java у CSS забезпечує основну функціональність, що використовується командами для надсилення SMS.

Коли водій паркує машину у призначеному місці, таймер запускається в шварті, яка знаходиться в цьому місці. Датчик повідомляє CSS про те, що місце на даний момент зайняте. CSS оновить базу даних з інформацією про зайнятість, а також відповідний номер мобільного телефону користувача. Це робиться для унікальної ідентифікації місця для стоянки автомобіля.

Коли водій виходить із зони паркувального місця, таймер негайно у тому місці паркування зупиняється. Дані таймера передаються в CSS. CSS перевіряє

свою базу даних, витягує поле номера мобільного телефону, що відповідає отриманому ідентифікатору, і надсилає SMS користувачеві. SMS буде містити інформацію про те, як довго стояв транспортний засіб, та суму рахунку. На той час, коли водій дійде до виїзду з автостоянки, він отримує платіжну інформацію через SMS [4].

1.6 Постановка задач дослідження

Ґрунтуючись на вищевикладеному, формально завдання дослідження зводиться до наступного:

- провести аналіз існуючих методів обчислювального інтелекту для вирішення задачі виявлення вільних місць для паркування;
- розглянути поняття та задачі комп'ютерного зору;
- розглянути архітектуру та методи навчання нейронних мереж для завдання розпізнавання об'єктів;
- розробити частину системи відстеження вільних місць для паркування за допомогою технік комп'ютерного зору;
- провести аналіз розробленої системи на підбити підсумки.

2 ТЕХНІКИ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ РОЗПІЗНАВАННЯ ОБРАЗІВ

2.1 Комп'ютерний зір

Комп'ютерний зір є одним з найпопулярніших напрямків теми глибокого навчання. Він розташований на перетині багатьох дисциплін, які включають інформатику, фізику, математику, техніку та психологію. Маючи такий широкий спектр предметів, багато експертів вважають, що все це наближає людську популяцію ближче до області штучного інтелекту. Через складність комп'ютерного зору вибір правильної моделі може бути досить складним завданням.

Комп'ютерний зір фокусується на відтворенні частин складності системи людського зору та наданні комп'ютерам можливості ідентифікувати та обробляти об'єкти на зображеннях та відео так само, як це роблять люди. Донедавна комп'ютерний зір працював лише в обмежених можливостях.

Завдяки досягненням у галузі штучного інтелекту та нововведенням у галузі глибокого навчання та нейронних мереж, за останні роки ця галузь змогла зробити значні стрибки і перевершити людей у виконанні деяких завдань, пов'язаних з виявленням та маркуванням об'єктів.

Одним із рушійних факторів зростання комп'ютерного зору є обсяг даних, які сьогодні створюються, які потім використовуються для навчання та вдосконалення комп'ютерного зору.

Поряд із величезною кількістю візуальних даних, понад 3 мільярди зображень обмінюються в мережі щодня, обчислювальна потужність, необхідна для аналізу даних, тепер є доступною. Оскільки поле комп'ютерного зору розросталося за допомогою нових апаратних засобів та алгоритмів, зростав і рівень точності ідентифікації об'єкта. Менш ніж за десять років сучасні системи досягли точності 99 від 50 відсотків, що робить їх більш точними, ніж люди, швидко реагуючи на візуальні дані. Ранні експерименти з комп'ютерним зором

розпочалися ще в 1950-х роках, і вперше його було комерційно застосовано для розрізнення набраного та рукописного тексту до 1970-х років, сьогодні програми для комп'ютерного зору зросли в геометричній прогресії.

Комп'ютерний зір займається питанням того, як комп'ютери можуть отримати розуміння на високому рівні за допомогою цифрових зображень чи відео. З точки зору інженерії, він прагне зрозуміти та автоматизувати завдання, які може виконувати зорова система людини.

Завдання комп'ютерного зору включають методи отримання, обробки, аналізу та розуміння цифрових зображень, а також вилучення великих розмірів даних із реального світу з метою отримання числової або символічної інформації, наприклад у формах рішень. Розуміння в цьому контексті означає перетворення зорових образів (введення сітківки) в описи світу, які мають сенс для процесів мислення і можуть викликати відповідні дії. Це розуміння зображення можна розглядати як роз'єднання символічної інформації з даними зображень за допомогою моделей, побудованих за допомогою геометрії, фізики, статистики та теорії навчання.

Наукова дисципліна комп'ютерного зору стосується теорії штучних систем, що витягують інформацію із зображень. Дані зображення можуть приймати різні форми, такі як відео послідовності, види з декількох камер, багатовимірні дані із 3D-сканера або медичного скануючого пристрою. Технологічна дисципліна комп'ютерного зору прагне застосувати свої теорії та моделі для побудови систем комп'ютерного зору.

Багато популярних програм комп'ютерного зору передбачають спробу розпізнати об'єкти на фотографіях; наприклад:

- класифікація об'єктів: яка категорія об'єкта на цій фотографії;
- ідентифікація об'єкта: який тип даного об'єкта є на цій фотографії;
- перевірка об'єкта: чи є об'єкт на фотографії;
- виявлення об'єктів: де саме знаходяться об'єкти на фотографії;

- виявлення орієнтира об'єкта: які ключові моменти мають об'єкти на фотографії;

- сегментація об'єкта: які пікселі належать об'єкту на зображенні;

- розпізнавання об'єктів: які об'єкти на цій фотографії та де вони.

Інші методи аналізу, крім розпізнавання, включають:

- аналіз руху відео, використовує комп'ютерне бачення для оцінки швидкості руху об'єктів у відео або самої камери;

- сегментація зображень, алгоритми розділяють зображення на кілька наборів подань;

- реконструкція сцени, алгоритм створює 3D-модель сцени, введenu через зображення або відео;

- відновлення зображень, такі шуми, як розмиття, видаляються з фотографій за допомогою фільтрів на основі машинного навчання.

Будь-яка інша програма, яка передбачає розуміння пікселів за допомогою програмного забезпечення, може бути сміливо позначена як комп'ютерне бачення.

В даній роботі освітлюється підтема комп'ютерного зору – розпізнавання об'єктів. А саме виявлення вакантних місць для паркування автівки.

Виявлення або розпізнавання об'єктів – це комп'ютерна технологія, пов'язана з комп'ютерним зором та обробкою зображень, яка займається виявленням випадків семантичних об'єктів певного класу, наприклад, людей, будівель чи автомобілів, у цифрових зображеннях та відео. Добре досліджені області виявлення об'єктів включають виявлення обличчя та виявлення пішоходів. Розпізнавання об'єктів застосовується у багатьох областях комп'ютерного зору, включаючи пошук зображень та відеоспостереження.

До появи глибокого навчання завдання, які міг виконувати комп'ютерний зір, були дуже обмеженими і вимагали багато ручного кодування та зусиль розробників та людських операторів [10].

Машинне навчання передбачало інший підхід до вирішення проблем із комп'ютерним зором. Завдяки машинному навчанню розробникам більше не потрібно вручну кодувати кожне правило в своїх програмах бачення. Натомість вони програмували «функції», менші програми, які могли виявляти конкретні візерунки на зображеннях. Потім вони використовували статистичний алгоритм навчання, такий як лінійна регресія, логістична регресія, дерева рішень або підтримка векторних машин (SVM) для виявлення шаблонів та класифікації зображень та виявлення об'єктів у них. Машинне навчання допомогло вирішити багато проблем, які були історично складними для класичних засобів та підходів до розробки програмного забезпечення. Також популярними методами машинного навчання для вирішення задачі комп'ютерного зору є:

- метод Віоли-Джонса (Viola-Jones object detection) – алгоритм, що дозволяє виявляти об'єкти на зображеннях в реальному часі. Його запропонували Паул Віола і Майкл Джонс в 2001 році. Хоча алгоритм може розпізнавати об'єкти на зображеннях, основним завданням при його створенні було виявлення осіб;

- масштабнезалежне перетворення ознак, або SIFT (Scale-invariant feature transform) – алгоритм, який виявляє і описує локальні ознаки зображення, застосовується для розпізнавання образів, побудови карт для навігації роботів, 3D-реконструкції, розпізнавання жестів, відстеження об'єктів та ін. Алгоритм був опублікований Девідом Лоу у 1999 р. і запатентований в США Британо-колумбійським університетом;

- гістограма напрямлених градієнтів (histogram of oriented gradients, HOG) – дескриптор ознак, який використовується в комп'ютерному зорі і обробці зображень з метою розпізнавання об'єктів. Метод підраховує напрямки градієнтів в локальних точках зображення. Він близький до гістограми орієнтованих границь, SIFT дескриптора, та значення форми, але відрізняється тим, що обраховується в щільній сітці рівномірно розташованих клітин та для підвищення точності використовує локальну нормалізацію контрасту.

У свою чергу глибоке навчання забезпечило принципово інший підхід до машинного навчання. Глибоке навчання покладається на нейронні мережі, функцію загального призначення, яка може вирішити будь-яку проблему, представлену на прикладах. Коли нейронній мережі надають багато позначених прикладів конкретного виду даних, вона зможе виділити загальні закономірності між цими прикладами та перетворити їх у математичне рівняння, яке допоможе класифікувати майбутні частини інформації.

Наприклад, для створення програми для розпізнавання облич із глибоким навчанням потрібно лише розробити або вибрати заздалегідь побудований алгоритм і навчити його на прикладах облич людей, яких він повинен виявити. Враховуючи достатню кількість прикладів, нейронна мережа зможе виявляти обличчя без подальших вказівок щодо особливостей чи вимірювань.

Поглиблене навчання є дуже ефективним методом комп'ютерного зору. У більшості випадків створення хорошого алгоритму глибокого навчання зводиться до збору великої кількості позначених навчальних даних та настройки таких параметрів, як тип і кількість шарів нейронних мереж та епохи тренувань. Порівняно з попередніми типами машинного навчання, глибоке навчання одночасно легше і швидше розробляти та застосовувати.

Більшість сучасних програм комп'ютерного зору, такі як виявлення раку, самокеровані машини та розпізнавання обличчя, використовують глибоке навчання. Поглиблене навчання та глибокі нейронні мережі перейшли з концептуальної сфери у практичні програми завдяки доступності та досягненню апаратних та хмарних обчислювальних ресурсів. Найбільш популярними методами глибокого навчання для вирішення задачі комп'ютерного зору є такі нейронні мережі:

- region-based convolutional network, наприклад, R-CNN та її модифікації Fast R-CNN, Mask R-CNN, cascade R-CNN;
- детектор MultiBox з одним знімком, Single Shot MultiBox Detector (SSD);

- нейронна мережа уточнення одного пострілу для виявлення об'єктів Single-Shot Refinement Neural Network for Object Detection (RefineDet);
- you only look once (YOLO);
- retina-Net;
- деформовані згорткові мережі.

2.2 Виявлення, розпізнавання об'єктів

Виявлення об'єктів (Object Detection) – одна з найвідоміших і широко досліджуваних тем у галузі машинного зору. Ця галузь займається виявленням та локалізацією деяких об'єктів, таких як людина, машина, автобус, ложка тощо, на зображенні. Цього можна досягти, намалювавши обмежувальне поле навколо даного конкретного цільового класу.

Постановка проблеми виявлення об'єктів може бути сформульована так: визначення розташування об'єктів та класів, до яких воно належить. Для виконання цього завдання будь-який метод виявлення об'єктів, а саме глибоке та «не таке глибоке» навчання можна сформулювати у три задані кроки, що зображені на рисунку 2.1.

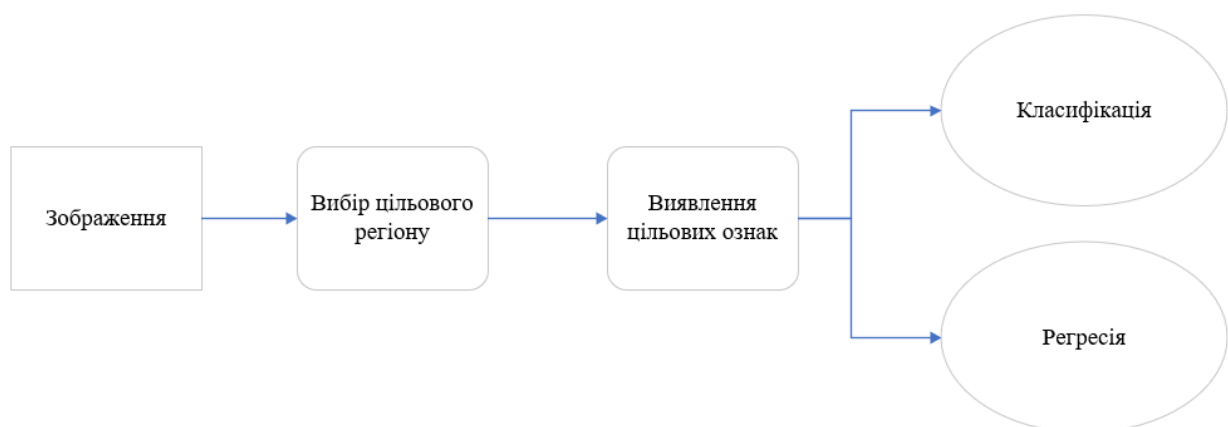


Рисунок 2.1 – Алгоритм виявлення об'єкта

Виконання алгоритму починається з того що на вхід подається зображення, після чого відбувається його обробка та так званий вибір цільового регіону.

Вибір регіону традиційним методом в основному здійснюється методом грубої сили розсувного вікна на зображенні. Це розсувне вікно має фіксований розмір і форму. Він ковзає по зображенню і отримує різні частини вхідного зображення (crops).

Складним моментом може бути те, що на зображенні багато об'єктів, які потрібно класифікувати, вони мають різне співвідношення сторін, розміри та положення на зображенні. Пошук ідеального вікна для кожного об'єкта на зображенні дуже виснажливий в обчислюванні і створює занадто багато зайвих вікон, що може ще більше уповільнити роботу даного алгоритму. Отже, якщо взяти фіксовану кількість розмірів та вікон шаблонів і провести по зображенню, це зменшить час обробки зображення, але не враховуватиме той самий об'єкт у різних масштабах.

Після вибору цільового регіону відбувається вилучення функції цілей (Feature Extraction of Targets).

Видобуток ознак (Feature Extraction) – це головний етап даного алгоритму. Отримавши різні частини вхідного зображення (crops) з вищевказаного кроку, тепер потрібно проаналізувати та вивчити семантичне та візуальне зображення кожного об'єкта на зображенні. Цього можуть дескриптори функцій (місцеві та глобальні), такі як дескриптори функцій SIFT, HoG та Harr-Like. Ці дескриптори можуть бути налаштовані для різних об'єктів загалом і можуть дати деякі дуже перспективні результати. Але через мінливість зовнішнього вигляду об'єкта внаслідок шуму, масштабу, освітлення, оклюзії стає дуже громіздким проектування та налаштування дескрипторів об'єктів кожного об'єкта вручну.

Після етапу вилучення ознак об'єктів відбувається класифікація, або регресія.

Завершальним кроком даного алгоритму є класифікація існуючих об'єктів з різних частин вхідного зображення за допомогою отриманих значень дескриптора ознак та присвоєння об'єкта деякому класу. Одночасно малюється обмежувальна рамка навколо об'єкта на даному зображенні [10]. Більшість відомих методів класифікації, що використовуються, – це Support Vector Machines, AdaBoost, Random Forest та ін.

Цим моделям потрібно набагато більше інформації про клас, тому для отримання хороших результатів потрібне клопітливе налаштування. Наприклад, SVM, як правило, не підтримує дискримінацію класу ймовірностей. Отже, стає дуже складно застосувати цей алгоритм у багатокласовій класифікації. Цим методам також погано вдається узагальнення даних, тобто SVM, як правило, дуже погано працює з даними, що містять шум, і мають точки перекриття даних.

Але після появи архітектури CNN та глибокої нейронної мережі стало зручніше та надійніше заповнювати прогалини, які є в традиційних алгоритмах виявлення об'єктів.

Завдяки наявності великої кількості даних і «глибшій» архітектурі нейронних мереж, все більше і більше складних функцій засвоюються автоматично, що допомагає заповнити прогалину, з якою стикається етап вилучення функції цілей.

Крім того, завдяки різноманітним навчальним підходам, з'явилася можливість дізнатися більш інформативні уявлення об'єктів, усуваючи проблему, вивчення функції об'єкта вручну.

Як показано, на рисунку 2.2 на даний момент доступні два типи методів виявлення об'єктів.

- двоступеневі детектори: на основі пропозиції регіону;
- одноразові детектори: на основі регресії.

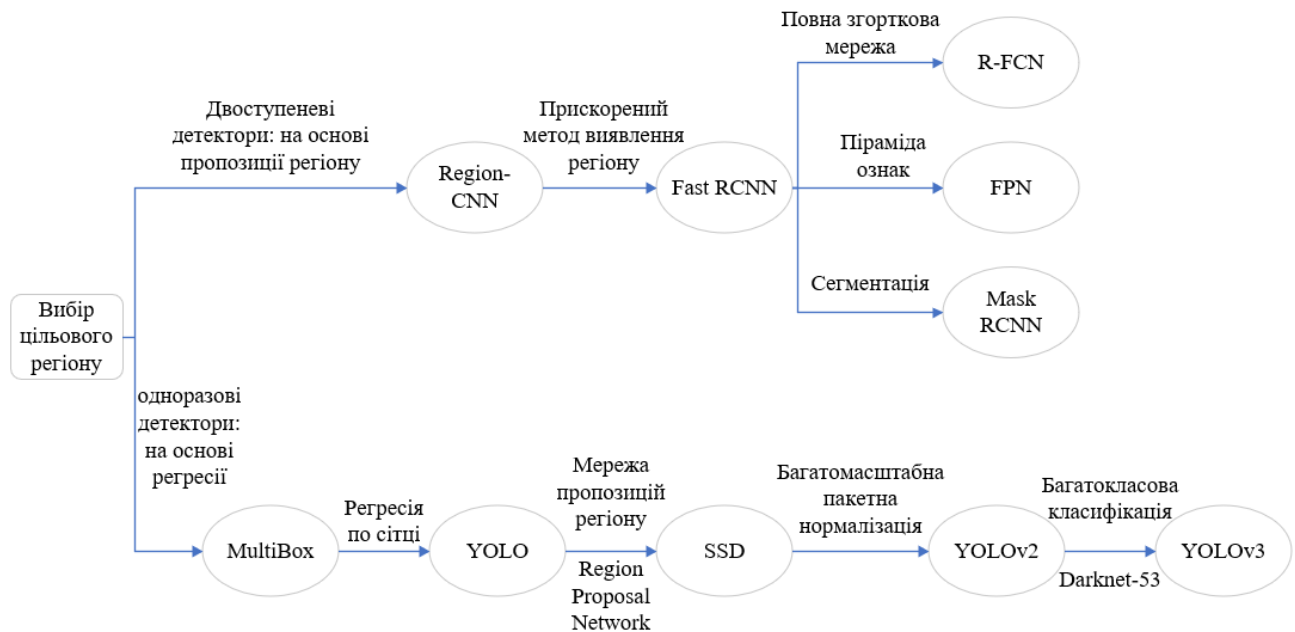


Рисунок 2.2 – Різні типи архітектур та методів виявлення об’єктів

Отже, розглянемо детальніше кожен з цих типів.

Перший етап – вибір цільового регіону. Двоступеневі детектори починають з вибіркового пошуку. В традиційних методах вибору регіону, замість використання надлишкових слайдів вікон, використовується піксельний підхід. Цей метод має справу із об’єднанням подібних пікселів на основі інформації про текстуру за допомогою структури даних набору об’єднань (Merge-Set Data Structure). Це також відоме як сегментація Super Pixel і може бути здійснено за допомогою алгоритму Graph-Cut.

Але є й мінус цього методу. Отримавши пропозиції, вони надходять у CNN для вилучення функцій. Якщо отримано 500 пропозицій, кожна з них потім подається у просту згорткову нейронну мережу для подальшого вилучення характеристик. Вони навчаються та роблять висновки дуже повільно через перекриття областей та надмірне виділення функцій для всіх пропозицій. Цей метод вперше був використаний у R-CNN.

Fast RCNN (видалення надлишкових прямих передач у CNN). Для вирішення вищевказаного методу, замість того, щоб щоразу передавати патч ROI

на CNN, спочатку використовується екстрактор функцій для всього зображення. Потім використовується метод екстрактора регіонів, такий як вибіркового пошуку, і витягуються патчі згенерованих карт функцій [14]. Цей метод допомагає скоротити зайві прямі проходи кожного патча та допомагає різко скоротити час обробки.

Мережі регіональних пропозицій (Region Proposal Networks). Процес навчання та вилучення висновків з використанням алгоритму вибіркового пошуку дуже трудомістке, оскільки воно працює на центральному процесорі. Тому неможливо запустити цей алгоритм у режимі реального часу. Щоб вирішити цю проблему, в дію вступають регіональні мережі пропозицій. Ці мережі навчені наскрізно, використовуючи легкий CNN для генерування ROI на картах функцій, замість використання необроблених зображень великих розмірів. Завдяки навчальній особливості цієї мережі та її налаштуванню гіперпараметрів, вона може генерувати більшу кількість ROI за дуже короткий час. Це було вперше представлено в Faster RCNN.

Що стосується одноразових детекторів, то цей крок пропускається, оскільки ці детектори не залежать від пропозицій регіону (Region Proposal), він передбачає обмежену фіксовану кількість пропозицій на даний момент часу із зображення та безпосередньо зазнає глобальної регресії, або класифікації, прямо переходячи від пікселів зображення до координат обмежуючого поля та ймовірностей класу. Такі типи моделей, або мереж надзвичайно швидкі, але за рахунок зменшення точності.

Наступний етап – вилучення функції цілей. Для двоступінчастих та одноразових детекторів вилучення ознак – це метод вилучення низькорівневого прихованого подання зображення. Ця інформація корисна через свій невеликий розмір і містить лише корисну інформацію, яка допомагає зменшити простір пошуку. Іноді цей модуль використовується заздалегідь у глибоких мережах.

Прихована карта, отримана з головних шарів мережі, надалі використовується на етапі вибору цільової області. Кожен з шарів призначений

для виконання конкретних завдань, а деякі з них є вдосконаленими версіями останнього. Деякі модулі екстрактора функцій – VGG16, GoogleNet, ResNet, DarkNet 53, різні варіанти FCN тощо.

Останній етап – регресія та класифікація. Для двоступінчастих та одноразових детекторів завершальним кроком до виявлення об'єкта є класифікація та локалізація обмежувальної рамки. Цей крок, як правило, базується та модифікується за допомогою різних комбінацій функцій втрат, включаючи регресійні втрати та класифікаційні втрати. Різні мережі мають різні варіації останньої функції втрати, але основні функції, що використовуються, наведено нижче.

Нарешті кінцевий вихід екстрактора ознак використовується для обчислення витрат (loss), які зворотно розповсюджуються для коригування локалізованих значень та ймовірностей класу. Ці модулі в загальних рисах також відомі як головний класифікатор (Classifier Head) та головний регресор (Regressor Head).

Розглянемо деякі функції втрат, які використовуються в Regressor Head:

- середньоквадратична втрата помилок, або втрата норми L2 (MSE);
- середня абсолютна помилка (MAE);
- втрата Губера (Huber Loss).

Втрата MSE є однією з найбільш часто використовуваних функцій втрат. Це сума квадратної відстані між цільовою змінною та прогнозованою змінною.

MAE – це ще одна функція втрат, яка використовується для Regressor Head. Це сума різниці між абсолютними значеннями цілі та прогнозованою змінною.

Втрата Губера менш чутлива до відхилень даних, ніж втрата квадратних помилок. Вона також диференціюється на 0. Це абсолютна помилка, яка стає квадратичною, коли помилка мала. Наскільки маленькою повинна бути ця помилка, щоб зробити її квадратичною, залежить від гіперпараметра δ , який можна налаштувати. Втрати Губера наближаються до MAE, коли $\delta \sim 0$, і до MSE, коли $\delta \sim \infty$.

Найбільш поширеною функцією втрат, що використовується в класифікаторі, є перехресні ентропійні втрати.

Перехресні ентропійні втрати або логарифмічні втрати вимірюють ефективність класифікаційної моделі, вихід якої становить значення ймовірності від 0 до 1. Втрати перехресних ентропій збільшуються в міру відхилення передбачуваної ймовірності від фактичної мітки. Отже, прогнозування ймовірності 0,012, коли фактична мітка спостереження дорівнює 1, буде поганим і призведе до великого значення втрат. Ідеальна модель мала б логарифмічну втрату 0.

Отже, можна зробити висновок, що виявлення об'єктів широко застосовується у різних сферах, таких як спостереження, безпека, криміналістика, автоматизовані системи автомобілів. Завдяки такому типу чутливих випадків використання, надзвичайно важливо, щоб детектори працювали з великою швидкістю і в той же час забезпечували дуже хорошу точність. Деякі труднощі, з якими стикається виявлення, або розпізнавання об'єктів, такі:

- виявлення в різних масштабах. Однією з найпоширеніших проблем є об'єкт, який виявляється в одному масштабі, може бути виявлений в інших менших, або більших масштабах. Тож для екстрактора ознак стає важливим узагальнити функції, які можна використовувати для будь-якого масштабу. Для цього FPN, також відомі пірамідальні мережі, які допомагають виділяти функції в будь-якому масштабі (малому, середньому та великому). Цей тип функціональних екстракторів широко використовується у більшості детекторів об'єктів;

- навчання для різної роздільної здатності зображення: Ще одним пунктом загального виявлення об'єкта є тренування на кожному вхідному зображенні. Більшість регресорів та класифікаторів – це повністю зв'язані шари. Тому зміна розміру зображення під час роботи неможлива. Мережа, навчена на одному розмірі, може не дати хороших результатів для іншого. Повністю згорткові

мережі є рішенням для вирішення цієї проблеми. Замість шарів FC, FCN слідує за звивистими шарами 1x1 Regressor та Classifier Head;

– швидкість та точність: Однією з найбільших проблем, з якою стикаються є фактор швидкості. Розгортання цих складних детекторів на дешевому вбудованому пристрої є основною проблемою, яка може збалансувати як аспекти швидкості, так і точності. Отже, відкрите дослідження продовжує створення мережі, яка має пристойну швидкість роботи в режимі реального часу, таку як YOLOv3, і точність однаково порівняно з різними сучасними детекторами, такими як Mask RCNN [16];

– дисбаланс класів. Дисбаланс класів робить мережу упередженою до вивчення додаткової довідкової інформації та впливає на точність. Щоб подолати цю проблему, деякі комбінації збільшення числа прикладів міноритарного класу та видалення прикладів мажоритарного класу виконуються на наборах даних, щоб генерувати рівне співвідношення позитивних та негативних вибірок.

2.3 YOLO. Архітектура

YOLO – це досить новий підхід до виявлення та розпізнавання об'єктів на зображеннях, або відео. Проблема виявлення об'єктів тут, на відміну від більш ранніх розробок, визначається як проблема регресії для просторово відокремлених обмежувальних рам і пов'язаних з ними ймовірностей класів.

У порівнянні з іншими мережами класифікації пропозицій регіонів (Fast RCNN), які виконують виявлення пропозицій різних регіонів і, таким чином, виконують прогнозування кілька разів для різних регіонів на зображенні, архітектура YOLO більше схожа на FCNN (повна згорткова нейронна мережа) і передає зображення ($n \times n$) один раз через FCNN і на вихід отримує ($m \times m$) передбачення. YOLO – це доволі проста мережа. Єдина згорткова мережа

одночасно передбачає кілька обмежувальних блоків та ймовірності класів для цих блоків.

YOLO тренується на повноцінних зображеннях і безпосередньо оптимізує ефективність виявлення. Ця уніфікована модель має ряд переваг перед традиційними методами виявлення об'єктів.

По-перше, YOLO надзвичайно швидка мережа. Оскільки розпізнавання в ній визначається як проблема регресії, тож вона не потребує складних послідовностей дій. Необхідно просто запустити нейронну мережу на новому зображенні під час тестування, щоб передбачити виявлення та розпізнати об'єкти на зображенні. Базова мережа працює зі швидкістю 45 кадрів в секунду без пакетної обробки на графічному процесорі Titan X, а швидка версія працює зі швидкістю більше 150 кадрів в секунду. Це означає, що за допомогою цієї мережі можна обробляти потокове відео в режимі реального часу із затримкою менше 25 мілісекунд. Крім того, YOLO досягає більш ніж удвічі середньої точності серед інших систем реального часу.

По-друге, YOLO робить міркування в цілому щодо зображення, коли робить прогнози. На відміну від ковзних вікон та методів, що базуються на пропозиціях регіонів, YOLO бачить все зображення під час навчання та тестування, тому неявно кодує контекстну інформацію про класи, а також їх зовнішній вигляд. Fast R-CNN, найпопулярніший метод виявлення, інколи плутає фонові плями на зображенні з об'єктами, оскільки не бачить більшого контексту. YOLO робить у двічі менше фонових помилок порівняно з Fast R-CNN.

По-третє, YOLO вивчає узагальнюючі уявлення об'єктів. Коли тренується на природних зображеннях та тестується на ілюстраціях, YOLO значно перевершує найкращі методи виявлення, такі як DPM (Deformable parts models) та R-CNN. Оскільки YOLO є узагальненим, він рідше руйнується при застосуванні до нових доменів або несподіваних входів.

Виявлення об'єктів відбувається наступним чином. Мережа використовує функції з усього зображення, щоб передбачити кожну обмежувальну рамку. Вона також передбачає усі обмежувальні рамки у всіх класах для зображення одночасно. Це означає, що мережа міркує про повне зображення та всі об'єкти на зображенні. Дизайн YOLO забезпечує наскрізне навчання та швидкість у реальному часі, зберігаючи високу середню точність.

Також ця мережа ділить вхідне зображення на сітку $S \times S$. Якщо центр об'єкта потрапляє в комірку сітки, ця комірка сітки відповідає за виявлення цього об'єкта.

Кожна комірка сітки передбачає B (деяке число) обмежувальних рамок та оцінок достовірності для цих рамок. Ці показники впевненості відображають, наскільки модель впевнена в тому, що рамка містить об'єкт, а також наскільки точним вона вважає це передбачення. Формально визначають впевненість як $Pr(\text{Об'єкт}) * IOU$. Якщо в цій рамці не існує жодного об'єкта, оцінка достовірності повинна бути нульовою. В іншому випадку показник впевненості дорівнював би перетину над об'єднанням (IOU) між передбачуваним полем та основною істиною.

Кожна обмежувальна рамка складається з 5 передбачень: x , y , w , h та впевненості. Координати (x, y) представляють центр вікна відносно меж комірки сітки. Ширина та висота передбачаються щодо всього зображення. Нарешті, прогноз достовірності представляє IOU між передбачуваним та будь-яким основним вікном істини. Кожна клітинка сітки також передбачає імовірності умовного класу C , $Pr(\text{Клас}_i | \text{Об'єкт})$. Ці ймовірності зумовлені осередком сітки, що містить об'єкт. Мережа передбачає лише один набір ймовірностей класів на клітинку сітки, незалежно від кількості вікон.

Під час тестування умовні ймовірності класу та індивідуальні прогнози довіри коробки перемножуються, що дає оцінки достовірності класу для кожного поля. Ці оцінки кодують як ймовірність появи цього класу у вікні, так і те, наскільки передбачуване поле відповідає об'єкту.

YOLO передбачає кілька обмежувальних рамок для кожної комірки сітки. Під час навчання лише одна рамка відповідає за кожен об'єкт. Призначається один провісник відповідальним за прогнозування об'єкта, на основі якого передбачення має найвищий IOU з основною істиною. Це призводить до спеціалізації між провісниками обмежувальної рамки. Кожен провісник (preditor) покращує прогнозування певних розмірів, пропорцій або класів об'єкта, покращуючи загальну оцінку точності.

YOLO накладає сильні просторові обмеження на передбачення обмежувальних рам, оскільки кожна клітинка сітки передбачає лише дві рами і може мати лише один клас. Це просторове обмеження обмежує кількість сусідніх об'єктів, які модель може передбачити. Модель бореться з дрібними предметами, які з'являються групами, наприклад, зграями птахів.

Оскільки модель вчиться передбачати обмежувальні рамки на основі даних, вона намагається узагальнити об'єкти в нових або незвичних пропорціях або конфігураціях. Ця модель також використовує відносно грубі функції для прогнозування обмежувальних рам, оскільки задана архітектура має кілька шарів зменшення вибірки з вхідного зображення.

Нарешті, поки модель тренується на функції втрат, яка наближає ефективність виявлення, функція втрат обробляє помилки однаково в малих обмежувальних рамах та великих обмежувальних рамах. Невелика помилка у великій рамці, як правило, є доброякісною, але невелика помилка в маленькій рамці має набагато більший вплив на IOU. Тож, головне джерело помилок – неправильні локалізації [11].

До цього моменту мова йшла про першу версію мережі YOLO. Але також є її модифікація YOLOv2, або YOLO9000. Покращена модель, YOLOv2, є сучасною у виконанні стандартних завдань виявлення, таких як PASCAL VOC та COCO. Використовуючи новий, багатомасштабний метод навчання, одна і та ж модель YOLOv2 може працювати з різними розмірами, пропонуючи легкий компроміс між швидкістю та точністю. При 67 FPS YOLOv2 отримує 76,8 mAP

на VOC 2007. При 40 FPS YOLOv2 отримує 78,6 mAP, перевершуючи найсучасніші методи, такі як швидший RCNN з ResNet та SSD, при цьому працює значно швидше. Нарешті, пропонується метод спільного тренування з виявлення та класифікації об'єктів. За допомогою цього методу одночасно навчається YOLO9000 на наборі даних виявлення COCO та наборі даних класифікації ImageNet. Таке спільне навчання дозволяє YOLO9000 прогнозувати виявлення для класів об'єктів, які не мають маркованих даних виявлення. Перевірено такий підхід у завданні виявлення ImageNet. YOLO9000 отримує 19,7 mAP на наборі перевірки виявлення ImageNet, незважаючи на наявність даних виявлення лише для 44 з 200 класів. У 156 класах, що не належать до COCO, YOLO9000 отримує 16,0 mAP. Але YOLO може виявити більше 200 класів; він передбачає виявлення понад 9000 різних категорій об'єктів. І це все ще працює в реальному часі. Тому пропонується новий метод для використання великої кількості класифікаційних даних, які вже існують, і використання його для розширення сфери застосування сучасних систем виявлення. Цей метод використовує ієрархічний погляд на класифікацію об'єктів, що дозволяє поєднувати різні набори даних разом.

Також пропонується спільний алгоритм навчання, який дозволяє тренувати детектори об'єктів як на даних виявлення, так і на класифікації. Метод використовує розмічені зображення виявлення, щоб навчитися точно локалізувати об'єкти, тоді як він використовує класифікаційні зображення, щоб збільшити свій словниковий запас та надійність.

За допомогою цього методу навчається YOLO9000, детектор об'єктів у реальному часі, який може виявити понад 9000 різних категорій об'єктів. Спочатку вдосконалюється базова система виявлення YOLO, щоб створити YOLOv2, найсучасніший детектор реального часу. Потім використовується метод комбінації наборів даних та спільний алгоритм навчання, щоб навчити модель для більш ніж 9000 класів від ImageNet, а також дані виявлення від COCO.

YOLO страждає від багатьох недоліків щодо сучасних систем виявлення. Аналіз помилок YOLO порівняно з Fast R-CNN показує, що YOLO допускає значну кількість помилок локалізації. Крім того, YOLO має відносно низький рівень відкликання порівняно з регіональними методами, що базуються на пропозиціях. Таким чином, має сенс зосереджитися головним чином на поліпшенні запам'ятовування та локалізації при збереженні точності класифікації. Комп'ютерний зір, як правило, спрямовано на збільшення, тобто поглиблення мережі. Краща продуктивність часто залежить від навчання великих мереж або об'єднання декількох моделей разом. Однак з YOLOv2 – це більш точний детектор, який все ще є швидким. Замість того, щоб масштабувати мережу, вона спрощується, а потім подання робиться простішим у засвоєнні.

Нормалізація патча (Batch Normalization). Нормалізація патча призводить до значного поліпшення конвергенції, або збіжності, одночасно усуваючи потребу в інших формах регуляризації. Додаючи нормалізацію патча на всіх згорткових шарах в YOLO, отримуємо поліпшення mAP на 2%. Нормалізація патча також допомагає впорядкувати модель. За допомогою пакетної нормалізації можливо усунути відсівання (dropout) з моделі, не стикаючись з перенавчанням.

Класифікатор високої роздільної здатності (High-Resolution Classifier). Усі найсучасніші методи виявлення використовують класифікатор, попередньо навчений на ImageNet. Починаючи з AlexNet, більшість класифікаторів працюють із вхідними зображеннями менше 256×256 . Оригінальний YOLO навчає мережу класифікаторів у форматі 224×224 і збільшує роздільну здатність до 448 для виявлення. Це означає, що мережа повинна одночасно переключитися на виявлення навчальних об'єктів і налаштуватись на нову роздільну здатність.

Для YOLOv2 спочатку точно налаштовується класифікаційна мережа з повною роздільною здатністю 448×448 протягом 10 епох на ImageNet. Це дає мережі час, щоб налаштувати свої фільтри для кращої роботи при введенні з більшою роздільною здатністю. Потім так само точно налаштовується отримана

мережа на виявлення, розпізнавання об'єктів. Ця мережа класифікації з високою роздільною здатністю дає збільшення mAP майже на 4%.

Конволюційний шар з опорними обмежувальними рамками, або вікнами (Convolutional With Anchor Boxes). YOLO передбачає координати обмежувальних рамок безпосередньо за допомогою повністю з'єднаних шарів на вершині екстрактора згорткових функцій. Замість прямого прогнозування координат Faster R-CNN прогнозує обмежувальні рамки за допомогою підібраних вручну пріоритетів. Використовуючи лише згорткові шари, регіональна мережа пропозицій (RPN) у Faster R-CNN передбачає зміщення та конфіденційність для опорних обмежувальних рамок. Оскільки шар прогнозування є згортковим, RPN передбачає ці зміщення в кожному місці на карті об'єктів. Прогнозування зсувів замість координат спрощує проблему та полегшує навчання мережі.

Повністю видаляються зв'язані шари з YOLO і використовуються опорні обмежувальні рамки, щоб передбачити обмежувальні поля. Спочатку усувається один шар об'єднання, щоб зробити вихід згорткових шарів мережі з більш високою роздільною здатністю. Також скорочується мережа, щоб працювати з 416 вхідними зображеннями замість 448×448 . Це робиться, оскільки потрібна непарна кількість місць на карті об'єктів, щоб була одна центральна рамка. Об'єкти, особливо великі об'єкти, як правило, займають центр зображення, тому добре мати одне розташування прямо в центрі, щоб передбачити ці об'єкти, а не чотири місця, які знаходяться поруч. Конволюційні шари YOLO зменшують вибірку зображення в 32 рази, тому, використовуючи вхідне зображення 416, отримуємо вихідну карту функцій 13×13 .

Коли переходимо до обмежувальних рамок, також відокремлюється механізм передбачення класу від просторового розташування і замість цього прогнозується клас і об'єктивність для кожної опорної обмежувальної рамки. Розглядаючи YOLO, цільове передбачення все ще передбачає IOU істинного

значення та пропонованої обмежувальної рамки, а передбачення класу передбачають умовну ймовірність цього класу, враховуючи наявність об'єкта.

За допомогою опорних обмежувальних рамок отримуємо невелике зниження точності. YOLO передбачає лише 98 обмежувальних рамок на зображення, але з опорними обмежувальними рамками така модель передбачає більше тисячі. Без опорних обмежувальних рамок така проміжна модель отримує 69,5 mAP при відкликанні 81%. З опорними обмежувальними рамками така модель отримує 69,2 mAP при відкликанні 88%. Незважаючи на те, що mAP зменшується, збільшення відкликання означає, що ця модель ще має потенціал для вдосконалення.

Кластери розмірності (Dimension Clusters). Існує дві проблеми під час використання опорних обмежувальних рамок з YOLO. По-перше, розміри рамки вибираються вручну. Мережа може навчитися належним чином коригувати рамки, але якщо вибрати кращі пріоритети для мережі, можна полегшити навчання мережі прогнозувати хороші виявлення.

Замість того, щоб вибрати пріоритети вручну, можна запустити кластеризацію k-середніх значень на обмежувальних полях навчального набору для автоматичного пошуку хороших пріоритетів. Якщо використовуються стандартні k-засоби з евклідовою відстанню, більші поля створюють більше помилок, ніж менші. Однак, пріоритети повинні призводити до хороших балів IOU, що не залежить від розміру рамки.

Прогнозування точного місцезнаходження (Direct location prediction). При використанні опорних обмежувальних рамок з YOLO стикаємося з другою проблемою: нестабільністю моделі, особливо під час ранніх ітерацій. Більша частина нестабільності походить від передбачення (x, y) місць для коробки. У регіональних мережах пропозицій мережа передбачає значення t_x і t_y , а координати центра (x, y) обчислюються як $x = (t_x * w_a) - x_a$ та $y = (t_y * h_a) - y_a$.

Наприклад, передбачення $t_x = 1$ змістить поле вправо на ширину опорного вікна, передбачення $t_x = -1$ змістить його вліво на ту саму величину. Це формулювання є необмеженим, тому будь-яке опорне вікно може опинитися в будь-якій точці зображення, незалежно від того, яке місце передбачало поле. При випадковій ініціалізації модель потребує тривалого часу для стабілізації до прогнозування розумних зсувів. Замість прогнозування зсувів можна дотримуватися підходу YOLO і прогнозувати координати розташування щодо розташування рамки сітки. Це обмежує основну істину від 0 до 1. Використовується логістична активація, щоб обмежити прогнози мережі, що потрапляють у цей діапазон. Мережа передбачає 5 обмежувальних полів у кожній рамці на вихідній карті функцій. Мережа передбачає 5 координат для кожного обмежувального вікна, t_x , t_y , t_w , t_h та t_o . Якщо рамка зміщена від верхнього лівого кута зображення на (c_x, c_y) , а обмежувальне поле має ширину та висоту p_w, p_h , тоді прогнози відповідають:

$$b_x = \sigma(t_x) + c_x, \quad (2.1)$$

$$b_y = \sigma(t_y) + c_y, \quad (2.2)$$

$$b_w = p_w e^{t_w}, \quad (2.3)$$

$$b_h = p_h e^{t_h}, \quad (2.4)$$

$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o), \quad (2.5)$$

де b_x, b_y, b_w, b_h – передбачені координати об’єкту;

t_x, t_y, t_w, t_h та t_o – координати для кожного обмежувального вікна;

c_x, c_y – координати центру обмежувальної рамки.

Оскільки прогнозування розташування обмежується, параметризації легше засвоїти, роблячи мережу більш стабільною. Використання кластерів розмірності поряд із безпосереднім прогнозуванням розташування центру обмежувальної рамки покращує YOLO майже на 5% у порівнянні з версією з опорними вікнами.

Дрібнозернисті особливості (Fine-Grained Features). Цей модифікований алгоритм YOLO передбачає виявлення на карті об'єктів 13×13 . Незважаючи на те, що цього достатньо для великих об'єктів, він може отримати користь від більш тонких деталей для локалізації менших об'єктів. Faster R-CNN та SSD запускають мережі пропозицій на різних картах функцій у мережі, щоб отримати діапазон розмірностей. Тут застосовується інший підхід, просто додаючи прохідний шар, який приносить елементи попереднього шару з роздільною здатністю 26×26 .

Прохідний шар поєднує функції вищої роздільної здатності з функціями низької роздільної здатності шляхом розміщення сусідніх об'єктів у різні канали замість просторових розташувань, подібно до відображення ідентичності в ResNet. Це перетворює карту об'єктів $26 \times 26 \times 512$ у карту об'єктів $13 \times 13 \times 2048$, яку можна об'єднати з оригінальними об'єктами. Цей детектор працює поверх цієї розширеної карти функцій, щоб він мав доступ до дрібних деталей. Це дає незначне збільшення продуктивності на 1%.

Багатомасштабне навчання (Multi-Scale Training). Оригінальний YOLO використовує вхідну роздільну здатність 448×448 . З додаванням опорних вікон роздільна здатність змінилася на 416×416 . Однак, оскільки така модель використовує лише згорткові та об'єднуючі шари, її можна змінювати на льоту. Щоб YOLOv2 був надійним для роботи на зображеннях різного розміру, необхідно навчити цьому модель.

Замість того, щоб фіксувати розмір вхідного зображення, просто змінюється мережа кожні кілька ітерацій. Кожні 10 партій така мережа випадковим чином вибирає новий розмір зображення. Оскільки модель зменшує

вибірки в 32 рази, необхідно вибирати з таких кратних 32: {320, 352, ..., 608}. Таким чином, найменший варіант – 320×320 , а найбільший – 608×608 . Змінюємо розмір мережі до цього виміру і продовжуємо навчання.

Цей режим змушує мережу навчитися добре передбачати різні вхідні виміри. Це означає, що одна і та ж мережа може передбачати виявлення з різною роздільною здатністю. Мережа працює швидше при менших розмірах, тому YOLOv2 пропонує легкий компроміс між швидкістю та точністю.

При низькій роздільній здатності YOLOv2 працює як дешевий, досить точний детектор. При 288×288 він працює зі швидкістю більше 90 кадрів в секунду з mAP, майже таким же хорошим, як Fast R-CNN. Це робить його ідеальним для менших графічних процесорів, відео з високою частотою кадрів або декількох відеопотоків.

При високій роздільній здатності YOLOv2 – це сучасний детектор із 78,6 mAP на VOC 2007, який все ще працює над швидкістю в реальному часі.

Більшість фреймворків виявлення об'єктів покладаються на VGG-16 як основний екстрактор функцій. VGG-16 – це потужна, точна мережа класифікації, але вона є надто складною. Конволюційні шари VGG-16 вимагають 30,69 мільярда операцій з плаваючою комою для одного проходу над одним зображенням з роздільною здатністю 224×224 .

Структура YOLO використовує власну мережу, засновану на архітектурі Googlenet. Ця мережа швидша за VGG-16, використовує лише 8,52 мільярда операцій для прямого проходу. Однак її точність дещо гірша, ніж VGG16. Для одинарного обрізання, із роздільною здатністю 224×224 , спеціальна модель YOLO отримує 88,0% ImageNet порівняно з 90,0% для VGG-16.

Була запропонована нову модель класифікації, яка буде використана як основа YOLOv2 – Darknet-19. Ця модель базується на попередній роботі над проектуванням мережі, а також загальних знаннях у цій галузі. Подібно до моделей VGG, використовуються переважно фільтри 3×3 і подвоюється кількість каналів після кожного кроку об'єднання. Після роботи над мережею в

мережі (Network in Network – NIN) доцільно використовувати загальне середнє об'єднання для прогнозування, а також фільтри 1×1 для стиснення представлення функцій між 3×3 шарами згортання. Використовується пакетна нормалізація для стабілізації навчання, прискорення конвергенції, збіжності та регулювання моделі.

Модель, яка називається Darknet-19, має 19 згорткових шарів та 5 шарів максимального сполучення. Повний опис шарів наведено на рисунку 2.4.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Рисунок 2.4 – Архітектура Darknet-19

Darknet-19 вимагає лише 5,58 мільярда операцій для обробки зображення, але при цьому досягається 72,9% точності топ-1 та 91,2% топ-5 в ImageNet.

Нарешті, найсучаснішою модифікацією є мережа YOLOv3. YOLOv2 використовував власну глибоку архітектуру darknet-19, спочатку 19-шарову мережу, доповнену ще 11 шарами для виявлення об'єктів. Завдяки 30-шаровій архітектурі YOLOv2 часто має труднощі з виявленням дрібних об'єктів. Це було пов'язано з втратою дрібнозернистих функцій, оскільки шари зменшували вибірку вводу. Щоб виправити це, YOLOv2 використовував відображення ідентичності, об'єднуючи карти об'єктів із попереднього шару для захоплення об'єктів низького рівня [12].

Однак в архітектурі YOLOv2 все ще бракувало деяких найважливіших елементів, які зараз є основними елементами більшості найсучасніших алгоритмів. Немає залишкових блоків, відсутні пропускні з'єднання та відсутність вибірки. YOLOv3 містить усе це.

По-перше, YOLO v3 використовує варіант Darknet, який спочатку має 53-шарову мережу, навчену на Imagenet. Для завдання виявлення на нього укладено ще 53 шари, що дає нам 106-шарову повністю згорнуту базову архітектуру для YOLOv3. Це причина повільності YOLOv3 порівняно з YOLOv2. Зараз архітектура YOLO v3 виглядає так як зображено на рисунку 2.5.

Нова архітектура може похвалитися залишковими пропусками з'єднань та підвищеною вибіркою. Найбільш помітною особливістю v3 є те, що він здійснює виявлення в трьох різних масштабах. YOLO – це повністю згорнута мережа, і її кінцевий результат генерується шляхом застосування ядра 1×1 на карті об'єктів. У YOLOv3 виявлення здійснюється шляхом застосування 1×1 ядер виявлення на картах об'єктів трьох різних розмірів у трьох різних місцях мережі.

Вигляд ядра виявлення становить:

$$1 \times 1 \times (B \times (5 + C)), \quad (2.6)$$

де B – кількість обмежувальних полів, які може передбачити клітинка на карті об'єктів;

«5» – це 4 атрибути обмежувального поля та одна довіра об'єкта;

C – кількість класів.

У YOLOv3, що навчається на COCO, $B = 3$ і $C = 80$, тому розмір ядра становить $1 \times 1 \times 255$. Карта об'єктів, вироблена цим ядром, має однакову висоту і ширину попередньої карти функцій і має атрибути виявлення вздовж глибини, як описано вище.

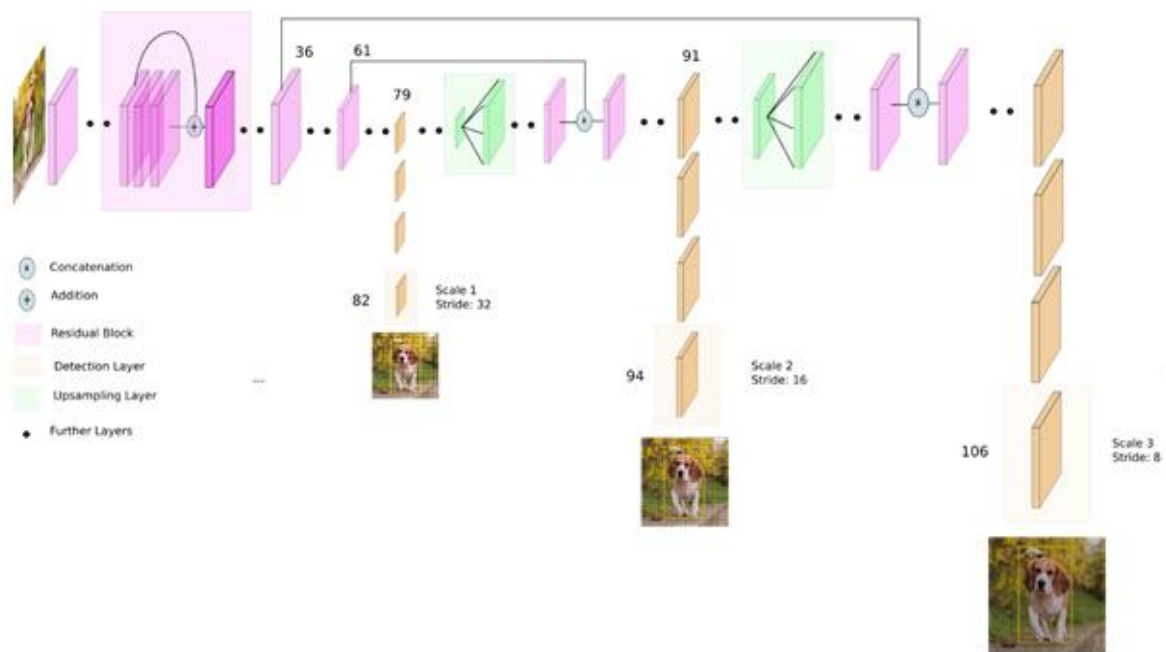


Рисунок 2.5 – Архітектура мережі YOLOv3

Важливо зазначити, що крок мережі або рівень визначається як відношення, за допомогою якого він зменшує вибірку вводу. Тобто, доцільно припустити, що маємо вхідне зображення розміром 416×416 .

YOLOv3 робить прогнозування за трьома шкалами, які точно даються шляхом зменшення розмірів вхідного зображення на 32, 16 та 8 відповідно.

Перше виявлення здійснюється 82-м шаром. Для перших 81 шарів зображення знімається мережею, таким чином, що 81-й шар має крок 32. Якщо маємо зображення 416×416 , результуюча карта об'єктів буде мати розмір 13×13 .

Одне виявлення зроблено тут за допомогою ядра виявлення 1×1 , що дає карту особливостей виявлення $13 \times 13 \times 255$.

Потім карта об'єктів із шару 79 піддається декільком згортковим шарам, перш ніж бути відібрана вдвічі до розмірів 26×26 . Потім ця карта об'єктів об'єднується вглиб із картою об'єктів із шару 61. Потім об'єднані карти об'єктів піддали декілька згорткових шарів 1×1 , щоб сплавити риси з попереднього шару (61). Потім друге виявлення здійснюється 94-м шаром, отримуючи карту функцій виявлення $26 \times 26 \times 255$.

Знову повторюється подібна процедура, коли карта об'єктів з шару 91 піддається декільком згортковим шарам, перш ніж бути об'єднана глибиною з картою об'єктів із шару 36. Як і раніше, кілька згорткових шарів 1×1 слідує для злиття інформації з попереднього шару (36). На фінальному етапі на 106-му шарі, отримуємо карту функцій розміром $52 \times 52 \times 255$.

Також YOLOv3 краще виявляє дрібні предмети. Виявлення на різних шарах допомагає вирішити проблему виявлення дрібних предметів, на що має часті скарги YOLOv2. Шар з вибіркою, об'єднаний з попередніми шарами, допомагає зберегти дрібнодисперсні особливості, які допомагають виявляти дрібні предмети.

Шар 13×13 відповідає за виявлення великих об'єктів, тоді як шар 52×52 виявляє менші об'єкти, а шар 26×26 виявляє середні об'єкти.

Вибір опорних вікон (anchor boxes). YOLOv3, загалом використовує 9 опорних вікон. Три на кожну шкалу. Якщо тренувати YOLO на власному наборі даних, слід використати кластеризацію K-Means для створення 9 опорних вікон.

Потім, розташувати опорні вікна в порядку зменшення розміру. Призначити три найбільші вікна для першої шкали, наступні три для другої шкали і останні три для третьої.

Більше обмежувальних рамок на зображення. Для вхідного зображення однакового розміру YOLOv3 передбачає більше обмежуючих рамок, ніж YOLOv2. Наприклад, при рідній роздільній здатності 416×416 YOLOv2

передбачав $13 \times 13 \times 5 = 845$ рамок. У кожній клітинці сітки (grid cell) було виявлено 5 обмежувальних рамок за допомогою 5 вікон (anchors).

З іншого боку, YOLOv3 передбачає обмежувальні рамки в 3 різних масштабах. Для того самого зображення 416×416 кількість передбачуваних обмежувальних рамок становить 10647. Це означає, що YOLOv3 передбачає 10-кратну кількість вікон, передбачених YOLOv2. Очевидно, чому це повільніше, ніж YOLOv2. У кожному масштабі кожна сітка може передбачити 3 обмежувальні рамки, використовуючи 3 анкери. Оскільки існує три шкали, кількість опорних вікон, що використовуються, становить 9,3 для кожної шкали.

Також зміни відбулися й в функції витрат. Остання частина функції витрат в YOLOv2 – це помилки у квадраті, тоді як у YOLOv3 вони були замінені умовами помилок з перехресною ентропією. Іншими словами, надійність об'єктів та прогнозування класу в YOLOv3 тепер прогнозуються за допомогою логістичної регресії.

Поки тренується детектор, для кожної істинної обмежувальної рамки призначається передбачена обмежувальна рамка, вікно (anchor) якої максимально перекривається з істинною обмежувальною рамкою.

Також, YOLOv3 тепер виконує багатозначну класифікацію об'єктів, виявлених на зображеннях.

Раніше в YOLO автори використовували softmax оцінки класу і вважали клас з максимальним балом класом об'єкта, що міститься в обмежувальній рамці. Це було змінено в YOLOv3.

Класи softmaxing спираються на припущення, що класи взаємовиключні, або простими словами, якщо об'єкт належить одному класу, то він не може належати іншому. Це чудово працює в наборі даних COCO.

Однак, коли у є такі класи, як Person та Women у наборі даних, то вищенаведене припущення не вдається. Це є причиною того, що автори YOLO утримались від softmaxing. Натомість кожен бал класу прогнозується за допомогою логістичної регресії, а поріг використовується для прогнозування

кількох міток для об'єкта. Класи, оцінки яких перевищують цей поріг, присвоюються в обмежувальній рамці [13].

Отже, YOLOv3 – хороший детектор. Він швидкий, він досить точний. Не такий точний COCO average AP, між показниками від 0,5 до 0,95 IOU. Але дуже має добрі показники за старою метрикою виявлення 0.5 IOU.

2.4 Fast R-CNN. Архітектура

Росс Гіршік вирішив деякі недоліки R-CNN, щоб побудувати швидший алгоритм виявлення об'єктів, і він отримав назву Fast R-CNN. Підхід подібний до алгоритму R-CNN. Але, замість того, щоб подавати регіональні пропозиції CNN, подається вхідне зображення на CNN, щоб сформувати згорнуту карту функцій. За допомогою згорткової карти об'єктів визначається область пропозицій і перетворює їх на квадрати, а за допомогою шару об'єднання RoI вони переформуються у фіксований розмір, щоб можна було подати в повністю зв'язаний шар. З вектора функцій RoI використовується шар softmax, щоб передбачити клас запропонованої області, а також значення зміщення для обмежувальної рамки.

Причина, по якій Fast R-CNN швидший, ніж R-CNN, полягає в тому, що не потрібно щоразу подавати пропозиції 2000 регіонів до згорткової нейронної мережі. Натомість операція згортки виконується лише один раз для кожного зображення, і на ньому генерується карта об'єктів.

Вхідне зображення та декілька цільових областей (RoI) вводяться в повністю згорнуту мережу. Кожен RoI об'єднується у карту об'єктів фіксованого розміру, а потім відображається у вектор об'єкта повністю пов'язаними шарами (FC). Мережа має два вихідні вектори на RoI: імовірності softmax та зміщення регресії обмежувального блоку для кожного класу. Архітектура навчена наскрізно з багатозадачними втратами. Рисунок 2.6 ілюструє архітектуру Fast R-CNN.

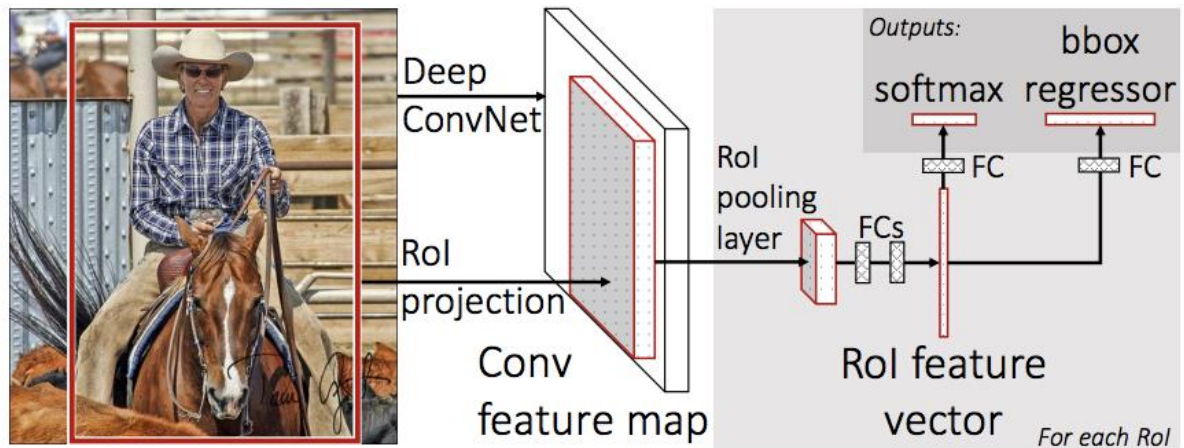


Рисунок 2.6 – Архітектура Fast R-CNN

Fast R-CNN приймає на вхід ціле зображення та набір пропозицій об'єктів. Спочатку мережа обробляє ціле зображення за допомогою декількох згорткових (conv) шарів та шарів максимального об'єднання (max pooling), щоб створити конволюційну карту об'єктів. Потім для кожної пропозиції об'єкта шар об'єднання регіону, що цікавить (RoI), витягує вектор об'єкта фіксованої довжини з карти об'єктів. Кожен вектор ознак подається у послідовність повністю зв'язаних (fc) шарів, які, нарешті, розгалужуються на два вихідних шари братів: один, який виробляє оцінки ймовірності softmax для K класів об'єктів плюс загальнодоступний клас фону, а інший шар виводить чотири реальних значення числа для кожного з K -класів об'єктів. Кожен набір з 4 значень кодує уточнені обмежувальні позиції для одного з класів K .

Шар об'єднання RoI використовує максимальне об'єднання для перетворення об'єктів у будь-якій дійсній області, що цікавить, у невелику карту об'єктів із фіксованою просторовою протяжністю $H \times W$ (наприклад, 7×7), де H і W – гіперпараметри шару, які є незалежно від будь-якого конкретного RoI. У цьому випадку RoI – це прямокутне вікно на конволюційній карті об'єктів conv.

Кожен RoI визначається чотирма кортежами (r, c, h, w) , що визначає його лівий верхній кут (r, c) та його висоту та ширину (h, w) .

RoI max pooling працює шляхом поділу вікна $h \times w$ RoI на сітку $H \times W$ з підвікон приблизного розміру $h/H \times w/W$, а потім об'єднує значення в кожному підвікні у відповідну комірку вихідної сітки. Пулінг застосовується незалежно до кожного каналу карти об'єктів, як у стандартному max pooling. Шар RoI – це просто приватний випадок просторового шару об'єднання пірамід, що використовується в SPPnets, в якому існує лише один рівень піраміди.

Винахідники мережі експериментували з трьома попередньо навченими мережами ImageNet, кожна з п'ятьма max pooling шарами та між п'ятьма та тринадцятьма шарами conv. Коли попередньо навчена мережа ініціалізує мережу Fast R-CNN, вона зазнає трьох перетворень.

По-перше, останній max pooling шар замінюється шаром об'єднання RoI, який налаштовано шляхом встановлення H і W як сумісних з першим повністю підключеним шаром мережі (наприклад, $H = W = 7$ для VGG16).

По-друге, останній повністю підключений шар мережі та softmax (які були навчені для 1000-напрямної класифікації ImageNet) замінюються двома описаними раніше шарами (повністю підключеним шаром та softmax над категоріями $K+1$ та регресорами обмежувальної коробки, характерними для категорії).

По-третє, мережа модифікована для отримання двох входів даних: списку зображень та списку RoIs на ці зображення.

Тренування всіх ваг мережі із зворотним розповсюдженням є важливою здатністю Fast R-CNN. З'ясуємо, чому SPPnet не може оновити ваги нижче шару об'єднання просторової піраміди.

Першопричиною є те, що зворотне розповсюдження через рівень SPP є вкрай неефективним, коли кожен навчальний зразок (тобто RoI) походить з іншого зображення, саме так тренуються мережі R-CNN та SPPnet.

Неефективність впливає з того, що кожен RoI може мати дуже велике сприйнятливим поле, рамку, яке часто охоплює все вхідне зображення.

Оскільки прямий прохід повинен обробити все рецептивне поле, вхідні дані для навчання великі (часто ціле зображення).

Тож, пропонується більш ефективний метод навчання, який використовує переваги спільного використання функцій під час навчання. У навчанні Fast RCNN міні-партії стохастичного градієнтного спуску (SGD) відбираються ієрархічно, спочатку шляхом вибірки N зображень, а потім вибірки R/N RoI з кожного зображення.

Критично важливо, що RoI з одного зображення обмінюються обчисленнями та пам'яттю в прямому та зворотному проходах. Внесення малого N зменшує обчислення міні-партії. Наприклад, при використанні $N = 2$ та $R = 128$, запропонована схема навчання приблизно на 64 рази швидша, ніж вибірка одного RoI зі 128 різних зображень (тобто стратегія R-CNN та SPPnet).

Одне занепокоєння щодо цієї стратегії полягає в тому, що вона може спричинити повільну конвергенцію тренувань, оскільки RoI з того самого образу співвідноситься. Здається, це занепокоєння не є практичним питанням, і досить хороших результатів можна досягнути із $N = 2$ та $R = 128$, використовуючи менше ітерацій SGD, ніж R-CNN.

На додаток до ієрархічної вибірки, Fast R-CNN використовує спрощений навчальний процес з одним етапом тонкої настройки, який спільно оптимізує класифікатор softmax та регресори обмежувальної рамки, а не навчає softmax класифікатор, SVM та регресори у три окремі етапи [15].

2.5 Faster R-CNN. Архітектура

Обидва вищезазначені алгоритми (R-CNN та Fast R-CNN) використовують вибіркового пошуку для з'ясування пропозицій регіону. Вибірковий пошук – це повільний і трудомісткий процес, що впливає на продуктивність мережі. Тому

Шаоцин Рен придумав алгоритм виявлення об'єктів, який виключає вибіркового алгоритму пошуку і дозволяє мережі вивчати регіональні пропозиції. Архітектура мережі Faster R-CNN представлена на рисунку 2.7.

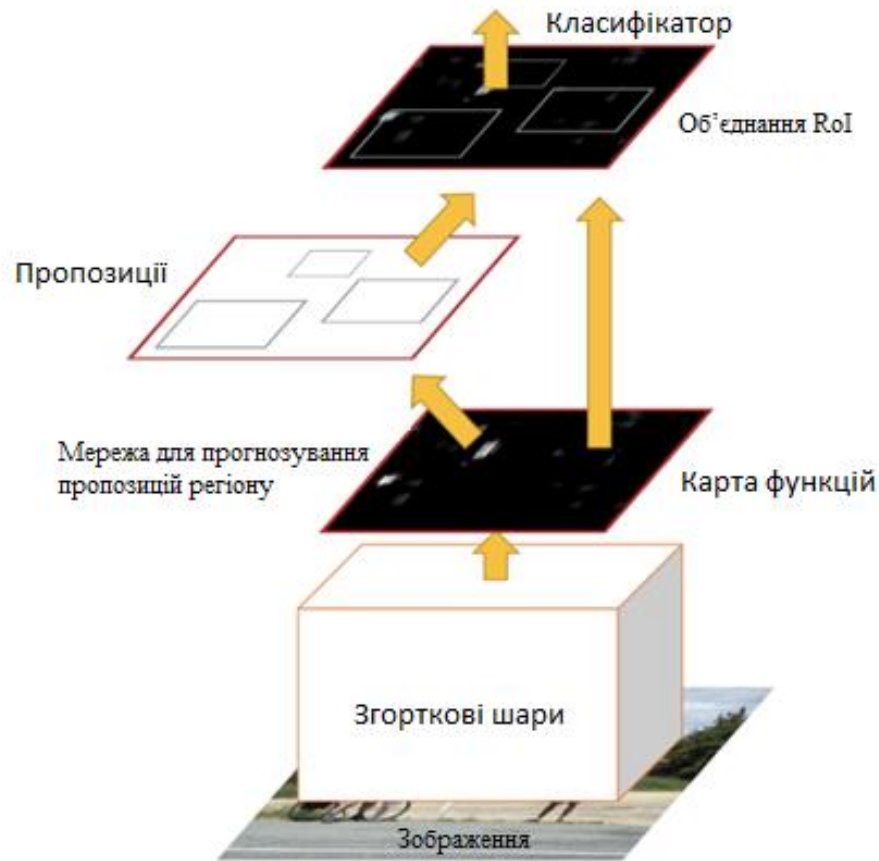


Рисунок 2.7 – Архітектура мережі Faster R-CNN

Подібно до Fast R-CNN, зображення подається як вхід до згорткової мережі, яка забезпечує згорнуту карту функцій. Замість використання алгоритму вибіркового пошуку на карті об'єктів для ідентифікації пропозицій регіону використовується окрема мережа для прогнозування пропозицій регіону. Потім прогнозовані пропозиції регіонів переробляються за допомогою шару об'єднання RoI, який потім використовується для класифікації зображення в межах запропонованої області та прогнозування значень зміщення для

обмежувальних рам. Порівняння швидкості роботи мереж при виконанні задачі детекції об'єктів показано на рисунку 2.8.

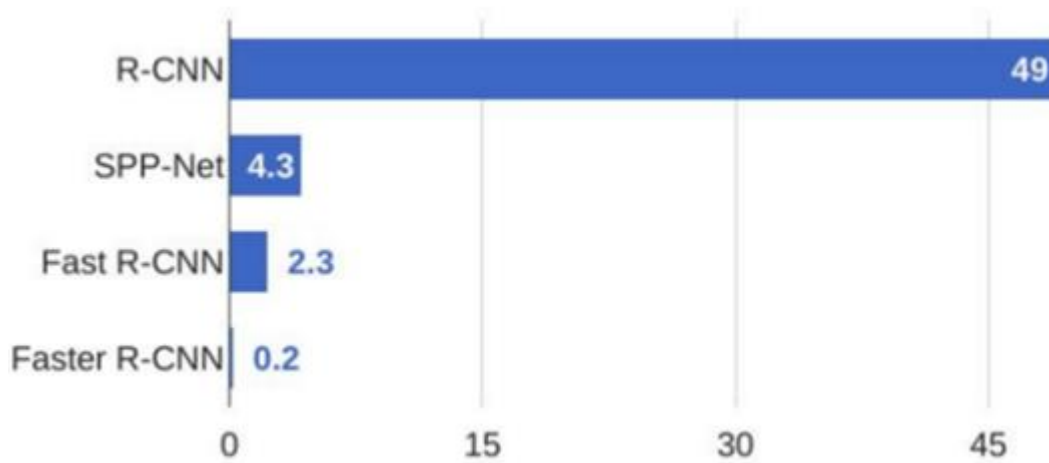


Рисунок 2.8 – Порівняння швидкості роботи мереж при виконанні задачі детекції об'єктів

З наведеного графіку видно, що Faster R-CNN набагато швидше, ніж його попередники. Тому його навіть можна використовувати для виявлення об'єктів у реальному часі [14].

2.6 Mask R-CNN. Архітектура

Метод, який називається Mask R-CNN, розширює Faster R-CNN, додаючи гілку для прогнозування масок сегментації в кожному регіоні, що цікавить (RoI), паралельно з існуючою гілкою для класифікації та регресії обмежувальної рамки. Гілка маски – це невеликий FCN, що застосовується до кожного RoI, передбачаючи сегментаційну маску з пікселя в піксель. Mask R-CNN просто впровадити та навчити завдяки швидшій структурі R-CNN, яка полегшує широкий спектр гнучких архітектурних конструкцій. Крім того, гілка маски

лише додає невеликі обчислювальні накладні витрати, що забезпечує швидку систему та швидкі експерименти.

В принципі, Mask R-CNN – це інтуїтивно зрозуміле розширення Faster R-CNN, але правильна побудова гілки маски є критичним завданням для отримання гарних результатів. Найголовніше, Faster R-CNN був розроблений не для вирівнювання пікселя в піксель між входами та виходами мережі. Це найбільш очевидно в тому, як RoIPool, фактична основна операція для відвідування екземплярів, виконує грубе просторове квантування для вилучення ознак. Щоб виправити розбіжність, пропонується простий шар без квантування, який називається RoIAlign, який зберігає точні просторові місця. Незважаючи на незначну зміну, RoIAlign має великий вплив: він покращує точність маски на відносно від 10% до 50%, демонструючи більший виграш за більш суворих показників локалізації. По-друге, надзвичайно важливо розділити маску та передбачення класів: передбачається двійкова маска для кожного самостійного класу, без конкуренції між класами, і покладаємось на гілку класифікації RoI мережі, щоб передбачити категорію. На відміну від цього, FCN зазвичай виконують мультикласову класифікацію за пікселем, яка поєднує сегментацію та класифікацію, і на основі експериментів такий підхід працює погано.

Mask R-CNN перевершує всі попередні сучасні одномодельні результати щодо завдання сегментації екземплярів COCO, включаючи інженерні роботи переможця конкурсу 2016 року. Як побічний продукт, такий метод також відрізняється завданням виявлення об'єкта COCO. В експериментах з абляцією оцінюється кілька основних примірників, що дозволяє продемонструвати її стійкість та проаналізувати вплив основних факторів.

Такі моделі можуть працювати на графічному процесорі приблизно на 200 мс на кадр, а навчання на COCO займає один-два дні на одному 8-графічному процесорі. Швидке тренування та тестування, а також гнучкість та точність фреймворку допоможуть і полегшать майбутні дослідження сегментації екземплярів.

Mask R-CNN концептуально проста мережа: Faster R-CNN має два виходи для кожного об'єкта-кандидата, мітку класу та зміщення обмежувальної рамки; до цього додається третя гілка, яка виводить маску об'єкта. Таким чином, маска R-CNN є природною та інтуїтивно зрозумілою ідеєю. Але додатковий вивід маски відрізняється від виходів класу та вікна, що вимагає виділення набагато більш тонких просторових макетів об'єкта. Далі описуються ключові елементи Mask R CNN, включаючи вирівнювання між пікселями, яке є основним відсутнім фрагментом в Fast/Faster R-CNN.

Faster R-CNN складається з двох етапів. На першому етапі, який називається регіональна мережа пропозицій (RPN), пропонуються рамки обмеження об'єктів-кандидатів. Другий етап, який, по суті є реалізацією Fast R-CNN, витягує функції за допомогою RoIPool з кожного поля кандидатів та виконує класифікацію та регресію обмежувальної рамки. Функції, що використовуються на обох етапах, можуть бути спільними для швидшого висновку.

Mask R-CNN приймає ту саму двоступеневу процедуру з однаковим першим етапом (який є RPN). На другому етапі, паралельно з прогнозуванням класу та обмежувальної рамки, Mask R-CNN також виводить двійкову маску для кожного RoI. Це відрізняється від найновіших систем, де класифікація залежить від прогнозів маски. Такий підхід слідує духу Fast R-CNN, який паралельно застосовує обмежувальну класифікацію та регресію, що, як виявилось, значно спростило багатоступеневий конвеєр оригінального R-CNN.

Формально, під час навчання, визначається функція втрат декількох завдань на кожному відібраному RoI як:

$$L = L_{cls} + L_{box} + L_{mask}, \quad (2.7).$$

де L_{cls} – класифікаційна функція втрат;

L_{box} – функція втрат передбачення обмежувальної рамки;

L_{mask} – функція втрат передбачення маски об'єкта.

Гілка маски має розмірний вивід Km^2 для кожного RoI, який кодує K бінарних масок з роздільною здатністю $m \times m$, по одній для кожного з класів K . Для цього застосовується функція сигмоїд на піксель і визначається L_{mask} як середня двійкова втрата перехресної ентропії. Для RoI, пов'язаного з істинним класом k , L_{mask} визначається лише на k -й масці (інші виходи маски не сприяють втраті).

Таке визначення L_{mask} дозволяє мережі створювати маски для кожного класу без конкуренції між класами. Покладаємось на спеціальну гілку класифікації, щоб передбачити мітку класу, яка використовується для вибору вихідної маски. Це відокремлює маску та передбачення класу. Це відрізняється від загальноприйнятої практики при застосуванні FCN для семантичної сегментації, яка, як правило, використовує per-pixel softmax та мультиноміальну перехресну ентропію. У цьому випадку змагаються маски між класами; у випадку коли використовуються per-pixel sigmoid та двійковій втраті вони цього не роблять. Експериментально доведено, що це формулювання є ключовим для хороших результатів сегментації екземплярів.

Представлення маски: маска кодує просторовий макет вхідного об'єкта. Таким чином, на відміну від міток класів або зсувів рамок, які неминуче згортаються на короткі вихідні вектори повністю зв'язаними (fc) шарами, витяг просторової структури масок може бути вирішений природним шляхом відповідності між пікселями та пікселями, що забезпечується згортаннями.

Зокрема, передбачається маска $m \times m$ від кожної RoI за допомогою FCN. Це дозволяє кожному шару у гілці маски підтримувати явний просторовий макет об'єкта $m \times m$, не згортаючи його у векторне представлення, у якому відсутні просторові розміри. На відміну від попередніх методів, які вдаються до шарів fc для прогнозування маски, таке повністю згорткове подання вимагає меншої кількості параметрів і є більш точним, як продемонстрували експерименти.

Така поведінка пікселів до пікселів вимагає, щоб функції RoI, які самі по собі є невеликими картами об'єктів, були добре вирівняні, щоб точно зберігати явне просторове відповідність за пікселем. Це спонукає розробити наступний рівень RoIAlign, який відіграє ключову роль у прогнозуванні маски.

RoIAlign – це стандартна операція з вилучення невеликої карти об'єктів (наприклад, 7×7) з кожного RoI. RoIPool спочатку квантує RoI з плаваючим числом до дискретної деталізації карти об'єктів, потім цей квантований RoI поділяється на просторові біни (spatial bins), які самі квантуються, і, нарешті, значення характеристик, охоплені кожним біном, агрегуються, як правило, шляхом максимального об'єднання (max pooling). Квантування виконується, наприклад, на неперервній координаті x шляхом обчислення $[x / 16]$, де 16 – крок карти об'єктів, а $[\cdot]$ – округлення, так само квантування виконується при поділі на біни (наприклад, 7×7).

Ці квантування вносять розбіжності між RoI та вилученими ознаками. Хоча це може не вплинути на класифікацію, яка є надійною для невеликих перекладів, вона має великий негативний вплив на прогнозування масок з точністю до пікселів.

Для вирішення цієї проблеми пропонується рівень RoIAlign, який усуває жорстке квантування RoIPool, належним чином вирівнюючи витягнуті функції з вхідними даними. Пропонована зміна проста: уникається будь-яке квантування меж або бінів RoI (тобто використовує $x / 16$ замість $[x / 16]$). Також використовуємо білінійну інтерполяцію для обчислення точних значень вхідних характеристик у чотирьох регулярно відібраних місцях у кожному RoI та узагальнюємо результати, використовуючи максимальне, або середнє значення). Слід зазначити, що результати не є чутливими до точних місць відбору проб або до кількості відібраних точок, якщо квантування не проводиться.

Тож, RoIAlign призводить до значних поліпшень. Також порівнюючи з операцією RoIWarp, на відміну від RoIAlign, RoIWarp не враховував проблему вирівнювання і був реалізований як квантування RoI так само, як RoIPool. Отже,

хоча RoIWarp також приймає дволінійну передискретизацію, вона виконується нарівні з RoIPool, як доведено експериментами, демонструючи вирішальну роль вирівнювання.

Mask R-CNN (регіональна згорткова нейронна мережа) – це двоступенева структура: перша стадія сканує зображення та генерує пропозиції (області, які можуть містити об'єкт). А другий етап класифікує пропозиції та створює обмежувальні рамки та маски. Обидва етапи пов'язані з структурою магістралі (backbone structure). Алгоритм роботи мережі Mask R-CNN показано на рисунку 2.9.

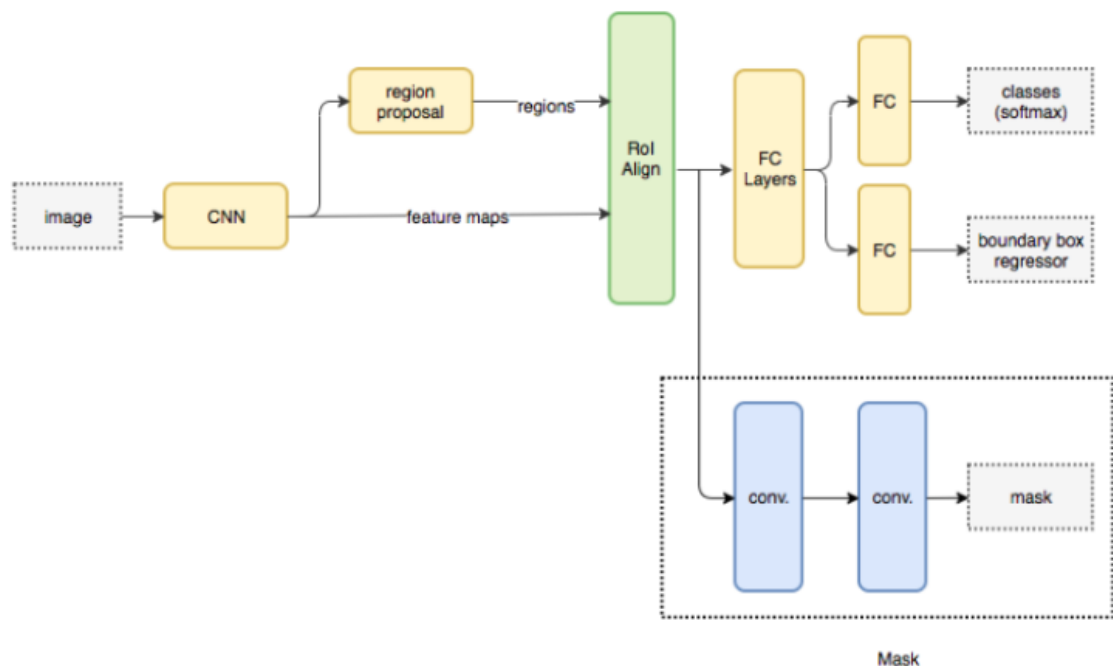


Рисунок 2.9 – Алгоритм роботи мережі Mask R-CNN

Магістраль – це стандартна згорткова нейронна мережа (як правило, ResNet50 або ResNet101), яка служить екстрактором функцій. Ранні шари виявляють елементи низького рівня (краї та кути), а пізніші шари послідовно виявляють елементи більш високого рівня (автомобіль, людина, небо).

Проходячи через магістральну мережу, зображення перетворюється з $1024 \times 1024 \times 3$ (RGB) на функціональну карту форми $32 \times 32 \times 2048$. Ця карта функцій стає вхідною інформацією для наступних етапів.

Хоча хребет, описаний вище, чудово працює, його можна вдосконалити. Feature Pyramid Network (FPN) була запроваджена тими ж авторами Mask R-CNN як розширення, яке може краще представляти об'єкти в різних масштабах.

FPN покращує стандартну піраміду вилучення особливостей, додаючи другу піраміду, яка бере високорівневі характеристики з першої піраміди і передає їх нижчим шарам. Це дозволяє функціям на кожному рівні мати доступ як до нижчого, так і до більш високого рівня. Структура мережі Mask R-CNN показана на рисунку 2.10.

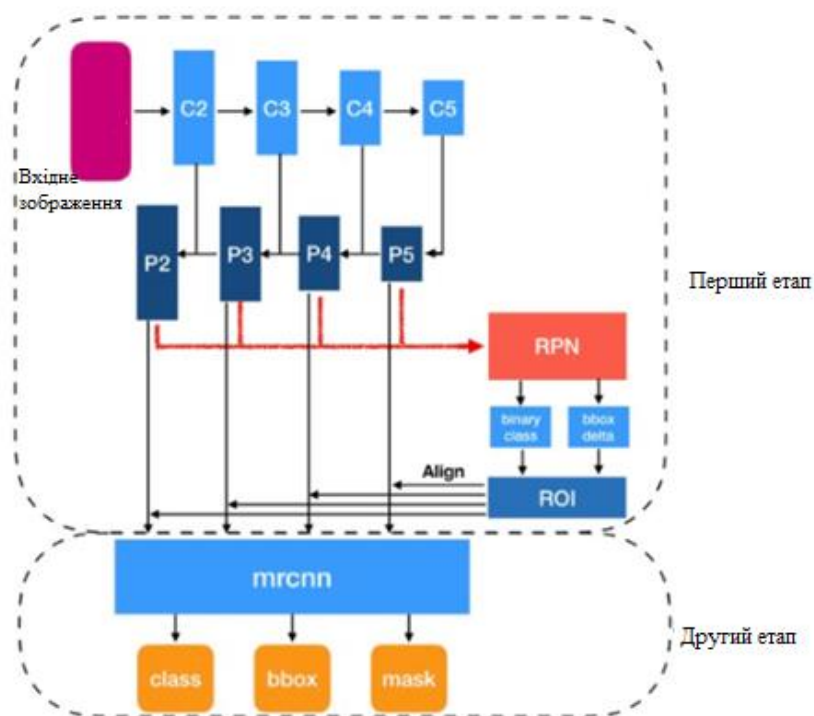


Рисунок 2.10 – Структура мережі Mask R-CNN

Перший етап: легка нейронна мережа, яка називається мережею регіональних пропозицій (RPN), сканує всі FPN зверху до знизу та пропонує

регіони, які можуть містити об'єкти. Незважаючи на те, що сканування карти об'єктів є ефективним способом, йому потрібен метод прив'язки об'єктів до місця його вихідного зображення. Рішення називається анкори, або опорні вікна (anchors). Анкори – це набір рамок із заздалегідь визначеними місцями та масштабами щодо зображень. Істинні класи (на цьому етапі класифікуються лише двійкові файли об'єктів або фону) та обмежувальні рамки призначені окремим анкорам. Оскільки прив'язки з різними масштабами прив'язуються до різних рівнів карти об'єктів, RPN використовує ці прив'язки, щоб з'ясувати, з чого карта об'єктів повинна отримати об'єкт і який розмір його обмежувальної рамки.

Другий етап: процедура виглядає схожою на RPN, єдині відмінності полягають у тому, що без допомоги анкорів, цей етап використовував ROIAAlign, щоб знайти відповідні ділянки карти об'єктів, і існує гілка, що генерує маски для кожного об'єкта на рівні пікселів. ROIAAlign, в якому вони беруть вибірку карти об'єктів у різних точках та застосовують білінійну інтерполяцію.

Найцікавіше, про Mask R-CNN, це те, що можна фактично змусити різні шари в нейронній мережі вивчати функції з різними масштабами, точно як анкери та ROIAAlign, замість того, щоб розглядати шари як чорний ящик [16].

3 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

3.1 Використані відкриті набори даних

У роботі використано декілька відкритих наборів даних, а саме PKLot Dataset, CNRPark+EXT Dataset, та Udacity Self Driving Car Dataset.

Набір даних PKLot містить 12 417 зображень автостоянок та 695 899 зображень паркованих місць, сегментованих з них, які були перевірені та позначені вручну. Всі зображення були отримані на стоянках Федерального університету Парани (UFPR) та Папського католицького університету Парани (PUCPR), які розташовані в Курітібі, Бразилія. Протокол, який використовується для побудови набору даних PKLot, складається з трьох етапів, як описано нижче.

Процес отримання зображень було визначено так – зображення знімалися з 5-хвилинним інтервалом часу протягом більше 30 днів за допомогою недорогої повноцінної камери високої чіткості (Microsoft LifeCam, HD-5000), розташованої у верхній частині будівлі для мінімізації можливої оклюзії між сусідніми транспортними засобами. Основною метою було отримання зображень за різних погодних умов (похмурих, сонячних та дощових періодів), реєструючи кожні 5 хв зміни навколишнього середовища. Така установка дозволяє знімати послідовності зображень, що демонструють високу мінливість щодо освітленості, спричиненої змінами погоди. Наприклад, за короткий період можна спостерігати невеликий дощ, сильний дощ та умови після дощу. На жаль, у наборі немає нічних знімків, оскільки освітлення, доступне на стоянках, було недостатнім для отримання якісних зображень. Отримані зображення зберігались у кольоровому форматі JPEG із стисненням без втрат (якість 100%) з роздільною здатністю 1280 на 720 пікселів. Вони були організовані у три підмножини з іменами UFPR04, UFPR05 та PUCPR. Перші два містять зображення різних видів тієї самої стоянки, знятих з 4-го та 5-го поверхів будівлі

UFPR. Останній набір даних містить зображення, зняті з 10-го поверху адміністративної будівлі PUCPR. Можна спостерігати деякі виклики, пов'язані з цим набором даних, такі як: сонячні зображення, що представляють переекспоновані машини та тіні, викликані деревами; або зображення, отримані під сильним дощем, які виглядають як нічні зображення через відсутність природного світла.

Для кожного зображення стоянки було створено файл XML, що містить положення та ситуацію (вільну чи зайняту) кожного місця для паркування. Розроблено інтерактивний інструмент для маркування зображень. Такий інструмент дозволяє візуалізувати кожне зображення та визначити межі кожного місця для паркування (представлені точками багатокутника), а також ситуацію (вільну чи зайняту). Різні підпапки використовувались для ручної класифікації зображень відповідно до спостережуваних погодних умов (похмурий, сонячний чи дощовий період) [18].

Ось короткий підсумок того, що робить цей набір даних цікавим для спільноти дослідників комп'ютерного зору:

- зображення, що охоплюють різні кліматичні умови (сонячний, дощовий та похмурий періоди), були зроблені при неконтрольованому освітленні;
- знімки були зроблені з різних паркінгів, що представляють виразні особливості;
- камери розташовувались на різних висотах;
- зображення демонструють різноманітний тип проблем, таких як наявність тіней, надмірний вплив сонячного світла, слабе світло в дощові дні, різниця в перспективі тощо;
- зображення транспортних засобів є типовими для комерційної системи спостереження, тобто камера розташована високо над транспортними засобами, що робить виявлення ще більш вимогливим.

CNRPark, складається з зображень, отриманих з розумних камер, розміщених у двох різних місцях, з різними точками зору та різними

перспективами стоянки дослідницького району Національної дослідницької ради (CNR) у Пізі.

Набір даних CNRPark містить зображення, зроблені в різні дні, з різними освітленими умовами, а також включає ситуації оклюзії та тіні, які ускладнюють завдання виявлення зайнятості. Запропонований набір даних було вручну та вичерпно анотовано та надано науковому співтовариству для визначення нових алгоритмів виявлення місць на автостоянці.

CNRPark + EXT – це набір даних для візуального виявлення заповнюваності паркінгів із приблизно 150 000 позначених зображень вільних та зайнятих паркувальних місць, побудованих на стоянці з 164 паркувальними місцями. CNRPark + EXT розширює CNRPark, попередній набір даних, який складається з 12 000 зображень, зібраних у різні дні липня 2015 року з 2 камер. Додаткову підмножину, яка називається CNR-EXT, складають зображення, зібрані з листопада 2015 року по лютий 2016 року за різних погодних умов 9 камерами з різними перспективами та кутами зору. CNR-EXT фіксує різні ситуації освітленості та включає часткові схеми оклюзії через перешкоди (дерева, ліхтарні стовпи, інші машини) та часткові або глобальні тіньові машини.

CNRPark + EXT складається з двох підмножин, зібраних під час дослідження. Далі в таблиці 3.1 повідомляються деталі обох підмножин разом із попереднім переглядом зібраних даних .

Таблиця 3.1 – Деталі даних з набору CNRPark + EXT

Підмножина	Кількість камер	Дати зйомки	Погодні умови	Кількість зображень
CNRPark	2	Липень 2015	Сонячно	242
CNR- EXT	9	Листопад 2015 – Лютий 2016	Сонячно, похмуро, дощі	4081

CNRPark також складається з сильно закритих просторів (майже повністю покритих деревами та ліхтарними стовпами), які не входять до набору сегментованих просторів PKLot; до того ж зображення беруться з нижньої точки зору щодо PKLot, що призводить до більшої кількості оклюзій через сусідні транспортні засоби [19]. Ці аспекти роблять класифікацію PKLot простішим викликом щодо CNRPark, який демонструє більшу мінливість між зображеннями.

Стратегія, що використовується для вибору зображень для складання навчальних та тестових наборів, визначає, що зображення одного дня можуть належати лише одному з цих наборів. Це дозволяє уникнути того, щоб у навчальних та тестових комплектах одночасно могли з'являтися фотографії, пов'язані з одним і тим же автомобілем, який годинами стояв на одному і тому ж просторі, на яких показані лише легкі варіації.

Також для додаткової перевірки мереж використовувалися випадкові зображення та відео різних паркування, що були знайдені на просторах інтернету.

3.2 Опис та результати експериментів

Для проведення експериментів було обрано три стратегії. Перша стратегія полягає в навчанні нейронної мережі на виявлення двох об'єктів, а саме вільного і зайнятого місця паркування. Для порівняння було використано дві архітектури нейронних мереж – Faster RCNN та Mask RCNN. При навчанні на наборі даних PKLot виникла проблема з розміткою зображень, що на першому етапі тренування вилилося в некоректній поведінці при тестуванні мережі. Проблема полягала в тому що оригінальна розмітка мала такий параметр як кут нахилу обмежувальної рамки. В свою чергу задача виявлення об'єктів потребує щоб усі обмежувальні рамки були під прямим кутом. Цю проблему було вирішено шляхом математичного перерахування координат обмежувальної рамки. На

рисунках 3.1 – 3.6 можна побачити результати тренування мереж з використанням правильної розмітки на наборі даних PKLot.

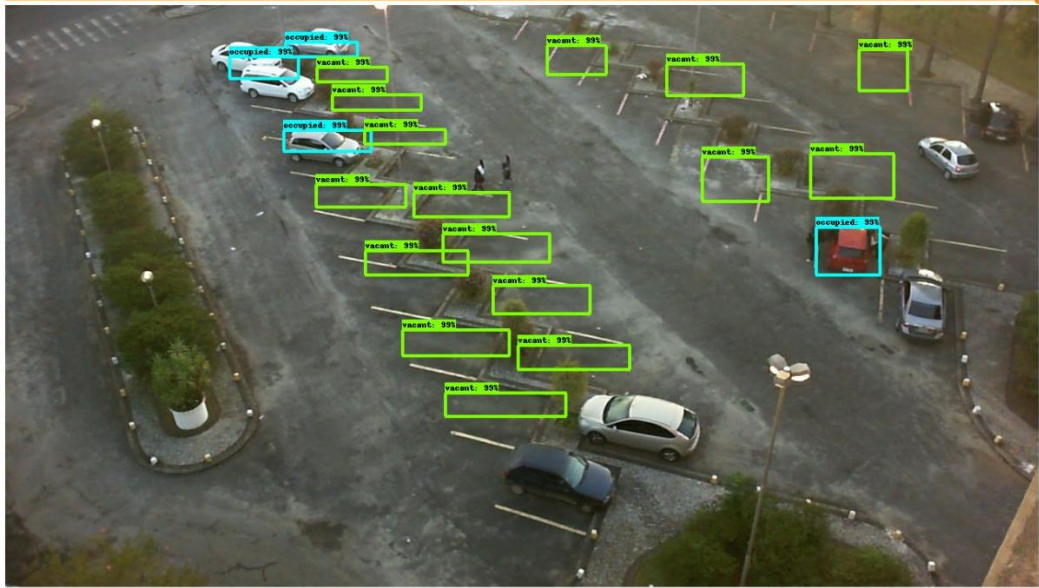


Рисунок 3.1 – Результати виявлення об’єктів мережею Faster RCNN на наборі даних PKLot

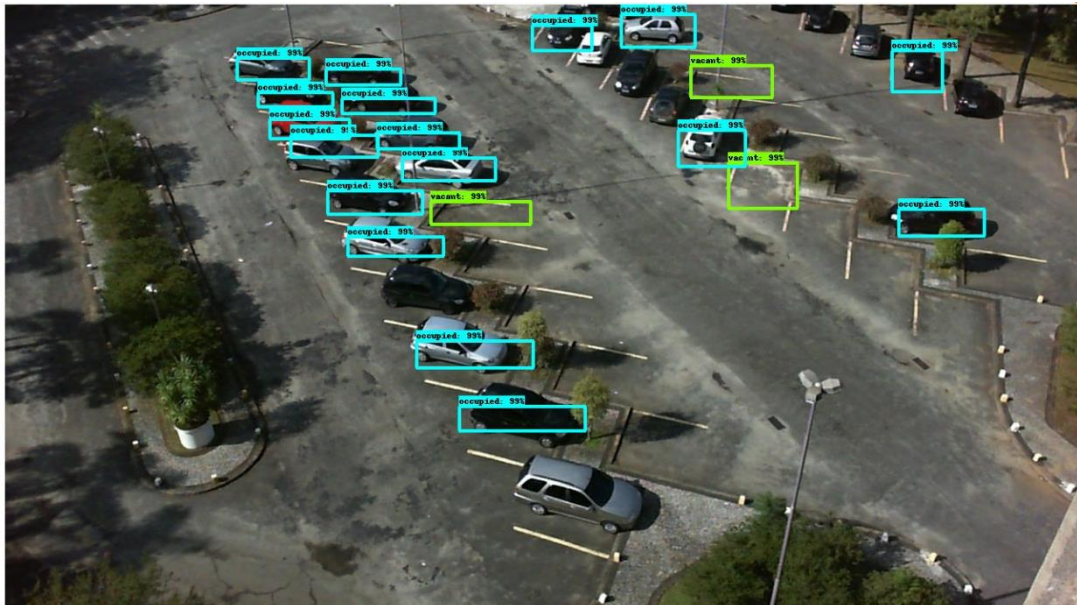


Рисунок 3.2 – Результати виявлення об’єктів мережею Faster RCNN на наборі даних PKLot



Рисунок 3.3 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних PKLot

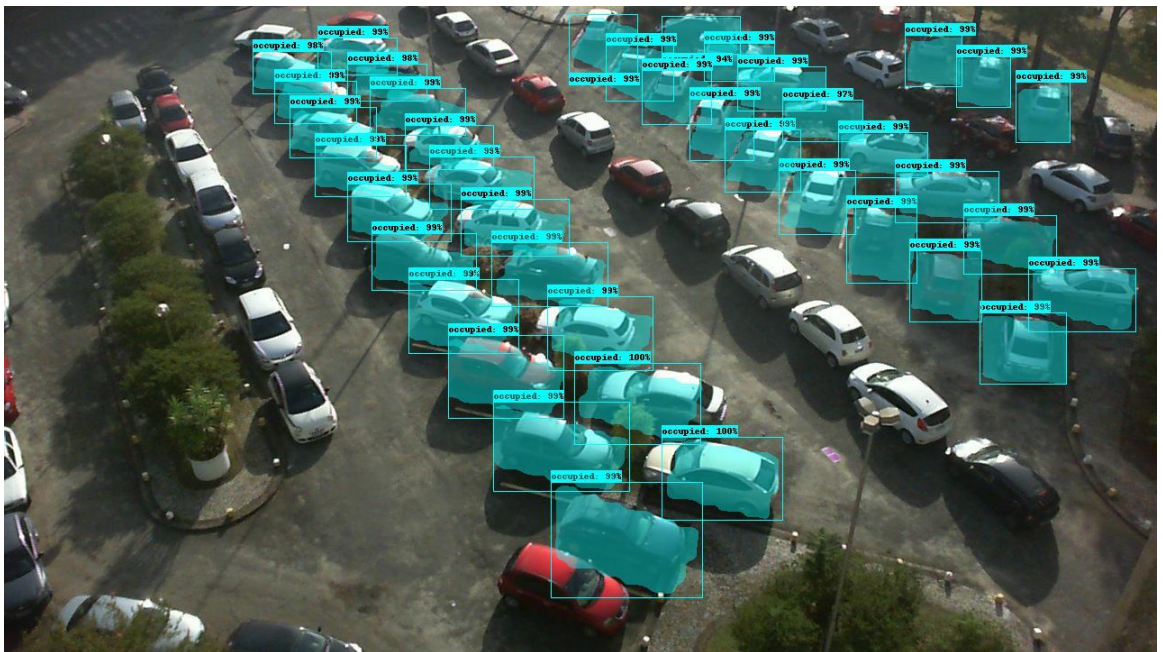


Рисунок 3.4 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних PKLot



Рисунок 3.4 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних PKLot

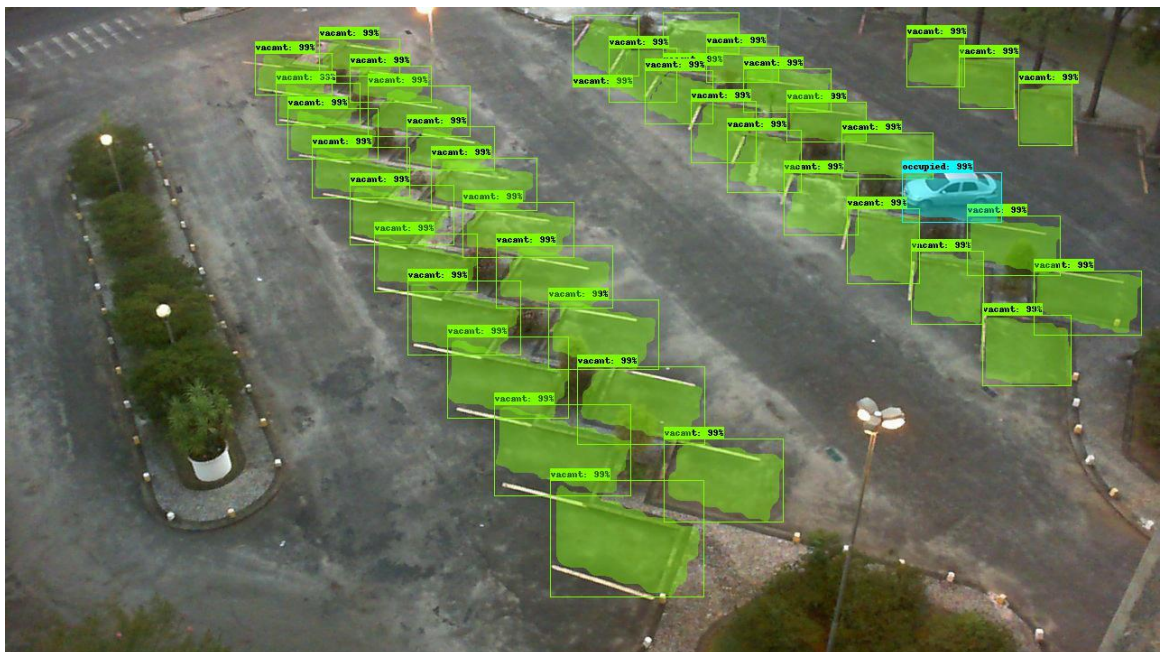


Рисунок 3.5 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних PKLot

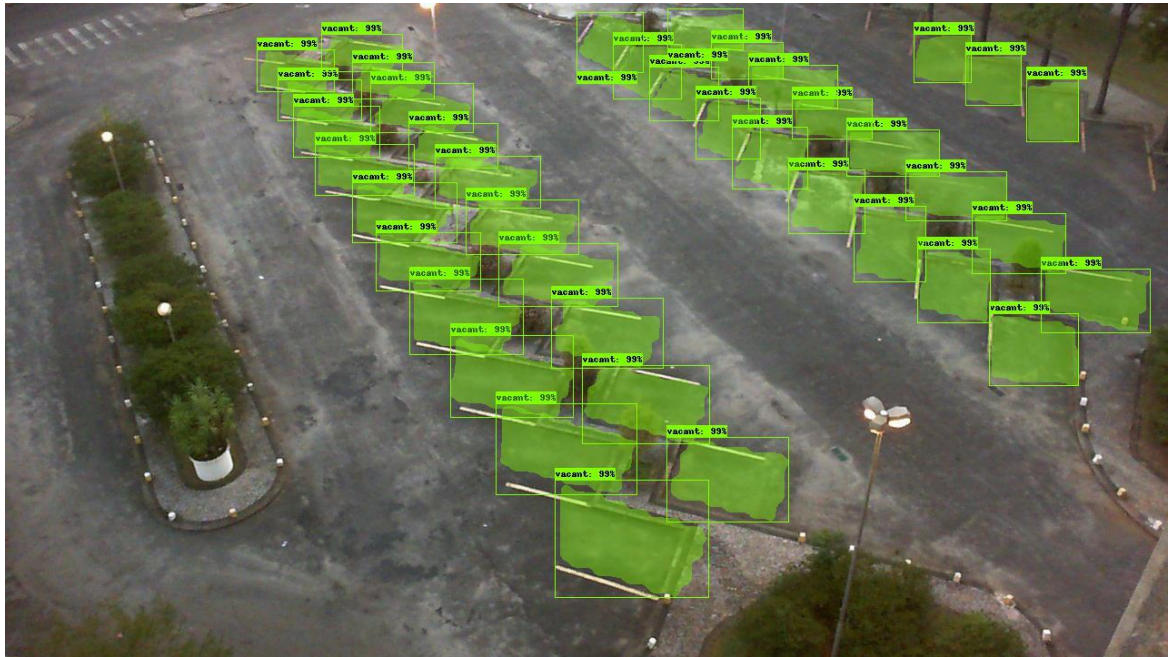


Рисунок 3.6 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних PKLot

Обидві мережі навчалися з стандартними значеннями параметрів налаштування $\text{learning_rate} = 0.001$ та $\text{steps_per_epoch} = 1000$, $\text{iou_threshold} = 0.7$. Точність виявлення становила 98,9% для зайнятих місць та 97,9% для порожніх місць, для нейронної мережі Faster RCNN. Та 99,7% для зайнятих місць та 98,9% для порожніх місць, для нейронної мережі Mask RCNN. Але, як можна побачити з рисунків мережа Mask RCNN змогла визначити всі місця на тестових даних, на відміну від мережі Faster RCNN, отже показала себе краще.

На рисунках 3.7 – 3.22 можна побачити результати тренування мереж на наборі даних CNRPark. Як видно, на цьому наборі даних обидві моделі дали гарні результати, при тих самих налаштуваннях параметрів. Набор даних CNRPark має більше різних ракурсів паркувань, що сприяло попередженню перенавчання моделі, та спонукало дійсно вивчити важливі риси об’єктів. Точність виявлення об’єктів становила 98 – 99% і для мережі Faster RCNN, і для мережі Mask RCNN.

Тож було вирішено перевірити навчені моделі на так званих «реальних» даних, – випадкових зображеннях, взятих з інтернету.

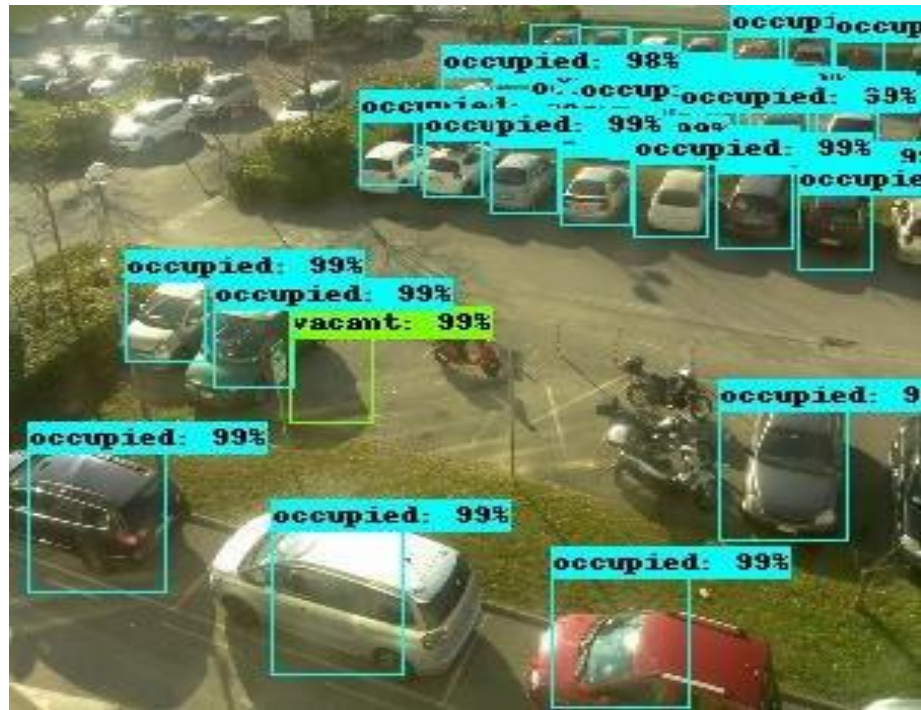


Рисунок 3.7 – Результати виявлення об’єктів мережею Faster RCNN на наборі даних CNRPark



Рисунок 3.8 – Результати виявлення об’єктів мережею Faster RCNN на наборі даних CNRPark



Рисунок 3.9 – Результати виявлення об’єктів мережею Faster RCNN на наборі даних CNRPark

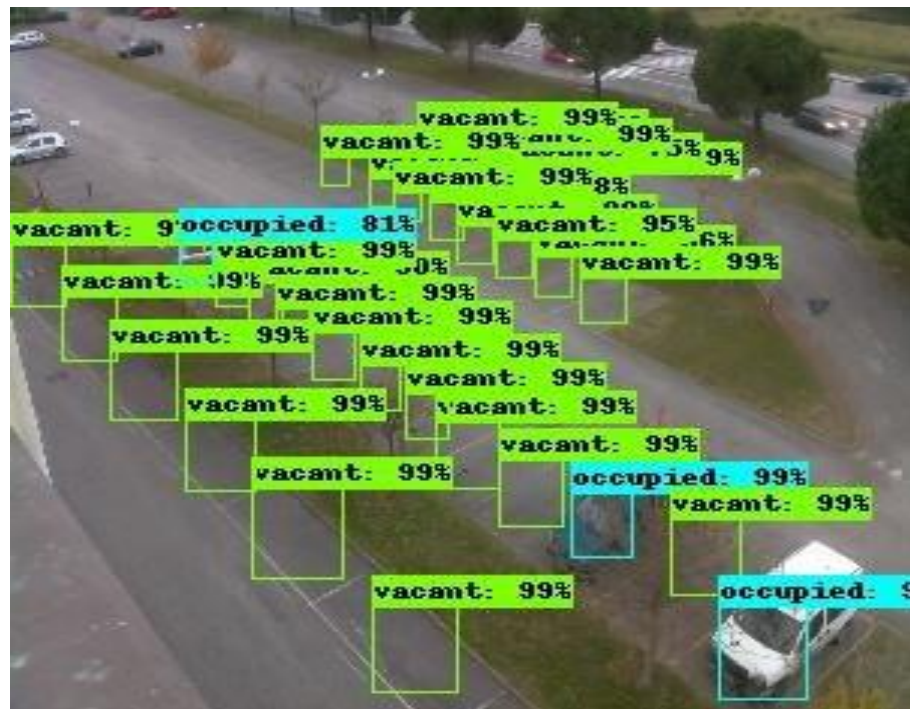


Рисунок 3.10 – Результати виявлення об’єктів мережею Faster RCNN на наборі даних CNRPark

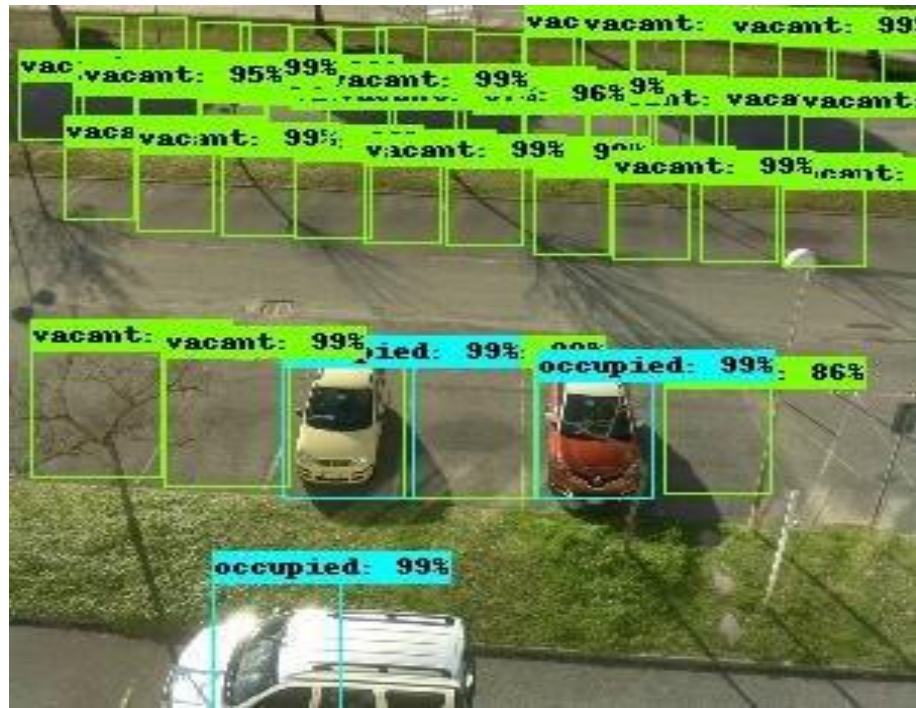


Рисунок 3.13 – Результати виявлення об’єктів мережею Faster RCNN на наборі даних CNRPark



Рисунок 3.14 – Результати виявлення об’єктів мережею Faster RCNN на наборі даних CNRPark



Рисунок 3.15 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних CNRPark

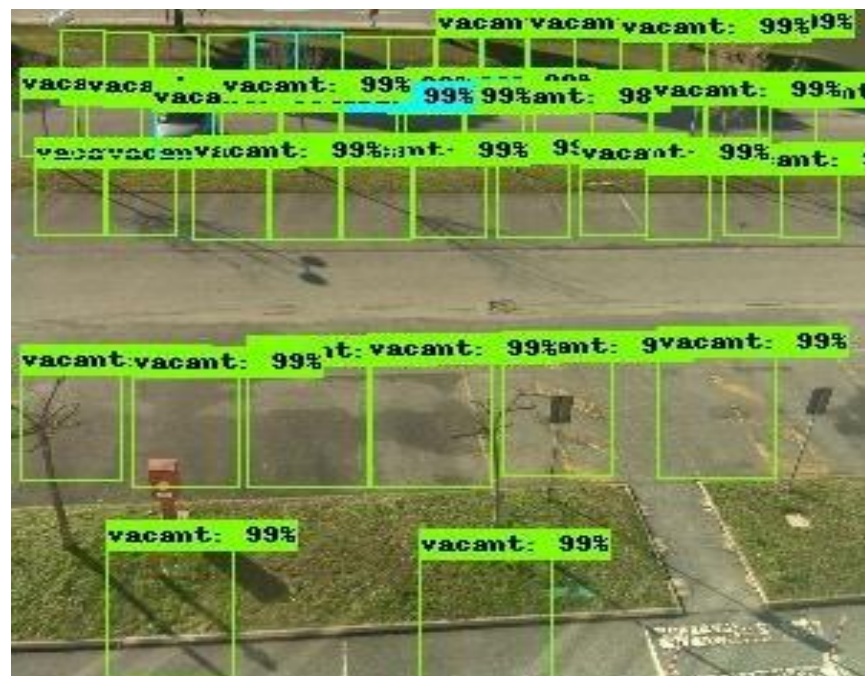


Рисунок 3.16 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних CNRPark



Рисунок 3.17 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних CNRPark



Рисунок 3.18 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних CNRPark



Рисунок 3.19 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних CNRPark



Рисунок 3.20 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних CNRPark

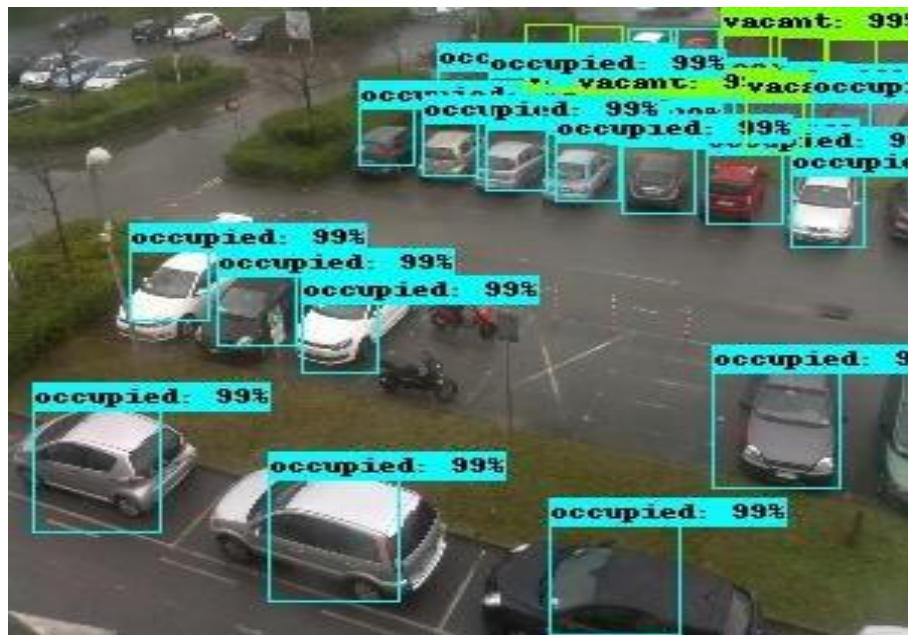


Рисунок 3.21 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних CNRPark

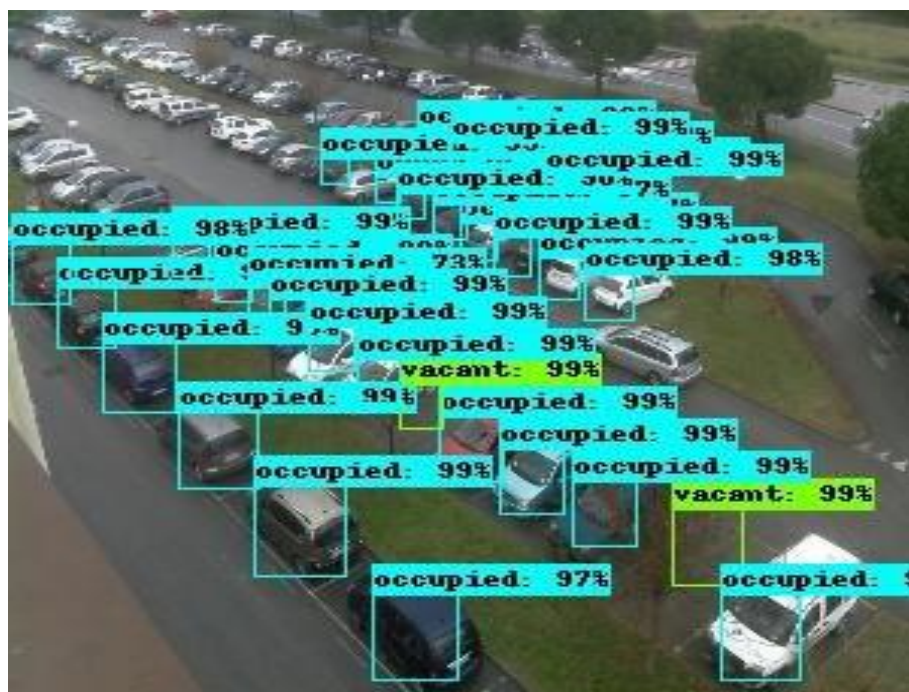


Рисунок 3.22 – Результати виявлення об’єктів мережею Mask RCNN на наборі даних CNRPark

На рисунках 3.23 – 3.28 можна побачити результати тестування мереж на випадкових зображеннях з інтернету. Можна побачити, що обидві мережі справляються з реальними даними значно гірше ніж з тренувальними, але знову Mask RCNN трохи краще ніж Faster RCNN, бо виділяє більше об'єктів.



Рисунок 3.23 – Результати виявлення об'єктів мережею Faster RCNN на випадкових зображеннях



Рисунок 3.24 – Результати виявлення об'єктів мережею Faster RCNN на випадкових зображеннях

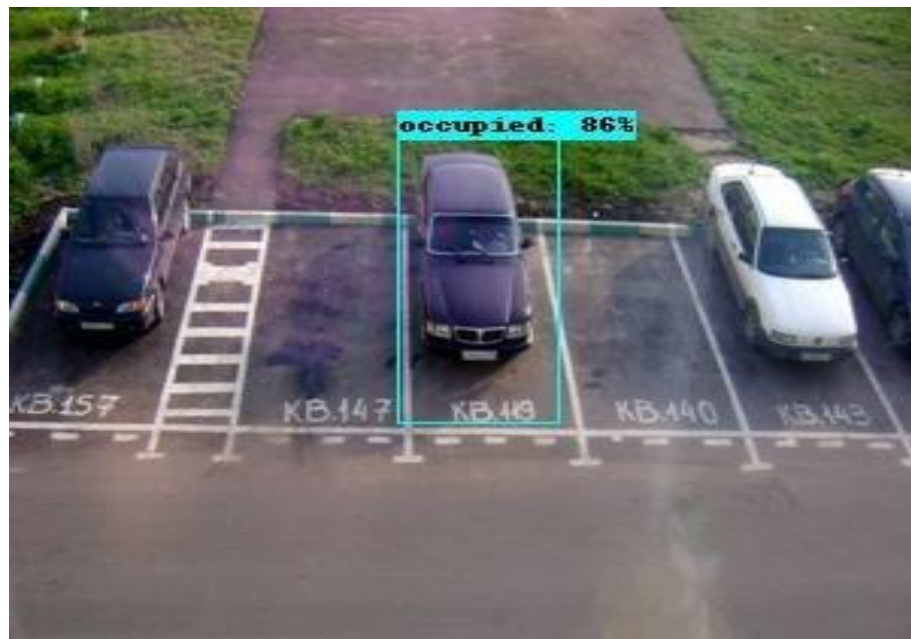


Рисунок 3.25 – Результати виявлення об’єктів мережею Faster RCNN на випадкових зображеннях



Рисунок 3.26 – Результати виявлення об’єктів мережею Mask RCNN на випадкових зображеннях



Рисунок 3.27 – Результати виявлення об’єктів мережею Mask RCNN на випадкових зображеннях



Рисунок 3.28 – Результати виявлення об’єктів мережею Mask RCNN на випадкових зображеннях

Тож, другим сценарієм була ідея, що полягала в використанні заздалегідь навченої моделі на виявлення автомобілів.

Вхідним сигналом алгоритму є відеопотік із звичайної веб-камери, в даному випадку відео з інтернету. Кожен кадр відео проходить через алгоритм, по одному кадру за раз.

Першим кроком у алгоритм є виявлення всіх можливих місць для паркування у відеокадрі. Очевидно, потрібно знати, які частини зображення є місцями для паркування, перш ніж виявляти, які місця для паркування вільні.

Другим кроком є виявлення всіх автомобілів у кожному кадрі відео. Це дозволить відстежувати рух кожного автомобіля від рами до рами.

Третій крок – визначити, які місця для паркування в даний час зайняті автомобілями, а які ні. Це вимагає поєднання результатів першого та другого кроків.

Однією з ідей може бути пошук паркоматів та припущення, що біля кожного лічильника є місце для паркування. Але при такому підході є деякі ускладнення. По-перше, не кожне місце для паркування має автомат для паркування. А по-друге, знання місця розташування автомата для паркування не говорить, де саме знаходиться місце для паркування.

Інша ідея полягає у створенні моделі виявлення об'єкта, яка шукає хеш-позначки місця для паркування, намальовані на дорозі. Але такий підхід теж не надійний. Перш за все, маркери ліній паркування можуть бути невеликі і їх важко побачити здалеку, тому їх також буде важко виявити за допомогою комп'ютера. А по-друге, вулиця повна всіляких не пов'язаних між собою ліній та позначок. Буде важко відокремити, які лінії є місцями для паркування, а які лініями – розділювачами смуг або пішохідними переходами.

Взагалі, місце для паркування – це просто місце, де автостоянка тривалий час. Інша ідея полягає в тому, що дійсними місцями для паркування є лише місця, де містяться автомобілі, що не рухаються:

Отже, якщо можна виявити машини і з'ясувати, які з них не рухаються між кадрами відео, тоді можна зробити висновок про місце розташування паркувальних місць.

Тож перші декілька кадрів відео слугують для визначення паркувальних місць – кожний виявлений автомобіль це місце для паркування. Наступний крок визначити чи вільне місце. Отже, якщо припустити, що кожна з виявлених обмежувальних рамок являє собою місце для паркування, можливо, рамка може бути частково зайнята автомобілем, навіть коли місце порожнє. Спосіб виміряти, наскільки два об'єкти перекриваються, щоб перевірити наявність вільних рамок – це міра, що, називається Intersection Over Union або IoU. IoU обчислюється шляхом знаходження кількості пікселів, де два об'єкти перекриваються, і ділення його на кількість пікселів, охоплених обома об'єктами.

Це дасть розуміння того, наскільки обмежувальна рамка автомобіля перекриває обмежувальну рамку місця для паркування. Завдяки цьому можна легко з'ясувати, чи є машина на парковці чи ні. Якщо показник IoU низький, як 0,2, це означає, що автомобіль насправді не займає більшу частину місця для паркування. Але якщо міра висока, як 0,6, це означає, що автомобіль займає більшу частину площі місця для паркування, тому можна припустити, що простір зайнятий.

Для тестування такого підходу було використано три архітектури мереж – YOLO, Faster RCNN, Mask RCNN, попередньо навчені ваги для MS COCO, та таймлапс відео з паркування.

MS COCO (Microsoft Common Objects in Context) – це багатоомасштабний набір даних для виявлення та сегментації об'єктів, що має більше ніж 200 тисяч розмічених зображень та 80 класів об'єктів, враховуючи людей, автомобілі, тварини та інші.

Програмний код реалізації функції визначення вільного місця паркування, поданий у лістингу 3.1.

Лістинг 3.1 – Програмний код функції визначення вільного місця паркування

```
car_boxes = get_car_boxes(r['rois'], r['class_ids'])
overlaps = mrcnn.utils.compute_overlaps(parked_car_boxes, car_boxes)
free_space = False
for parking_area, overlap_areas in zip(parked_car_boxes, overlaps):
    max_IoU_overlap = np.max(overlap_areas)
    y1, x1, y2, x2 = parking_area
    if max_IoU_overlap <= 0.2:
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 3)
        free_space = True
    else:
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 1)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, f"{max_IoU_overlap:0.2}", (x1 + 6, y2 -
6), font, 0.3, (255, 255, 255))
```

Результати роботи алгоритму виявлення об'єктів та визначення вільного місця можна побачити на зображеннях 3.29 – 3.32. Значення, що зображені в обмежувальних рамках – є мірою IoU.

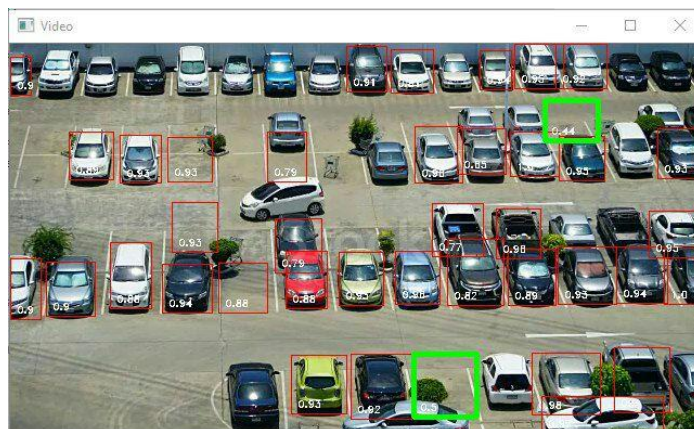


Рисунок 3.29 – Результати роботи алгоритму виявлення об'єктів та визначення вільного місця мережею YOLO

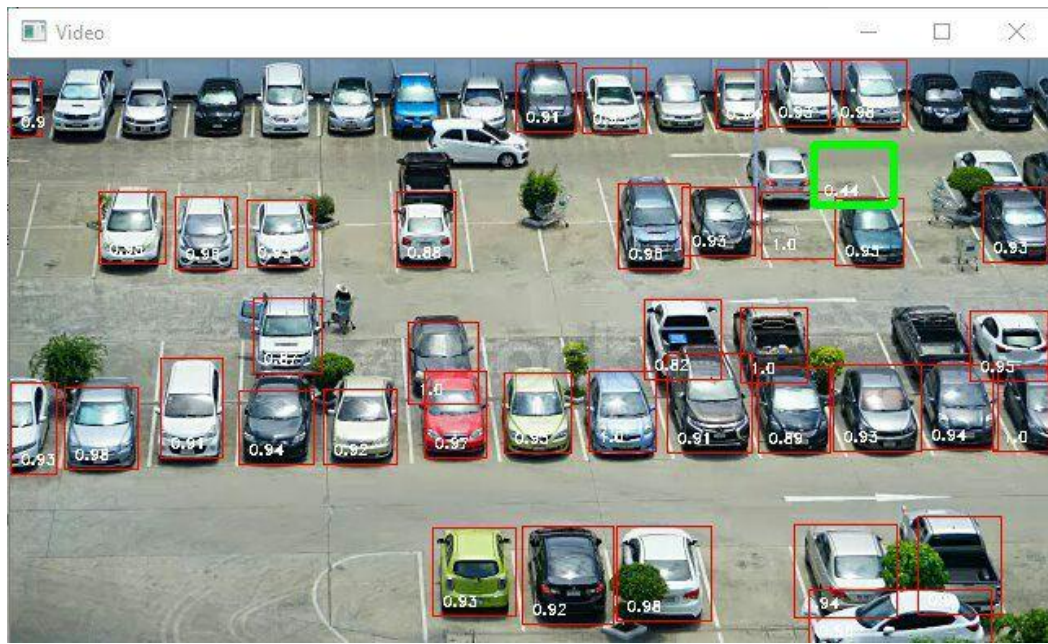


Рисунок 3.30 – Результати роботи алгоритму виявлення об’єктів та визначення вільного місця мережею YOLO

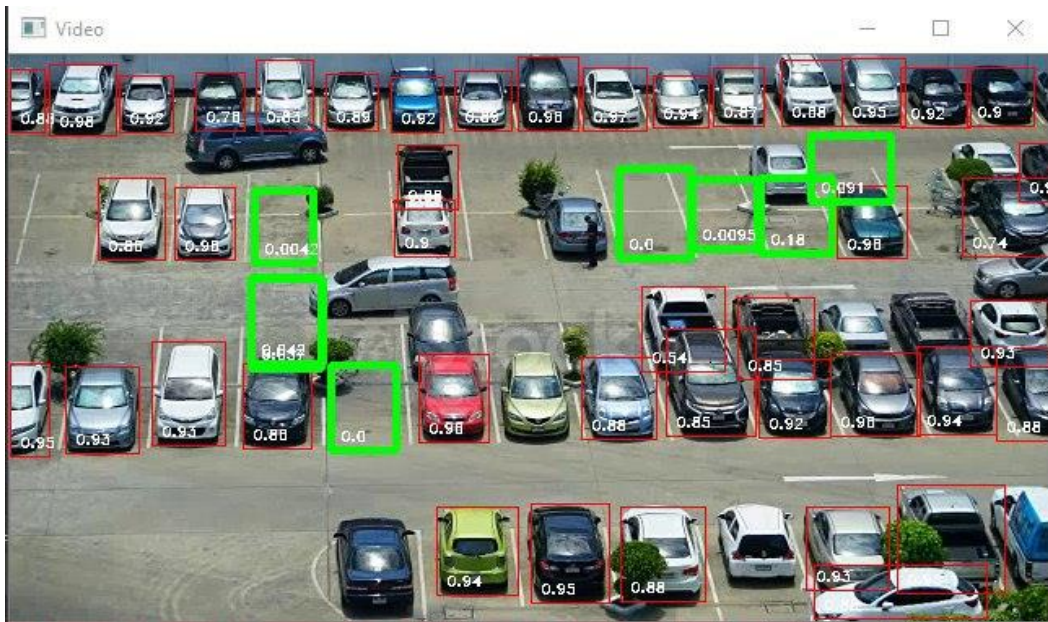


Рисунок 3.31 – Результати роботи алгоритму виявлення об’єктів та визначення вільного місця мережею Faster RCNN

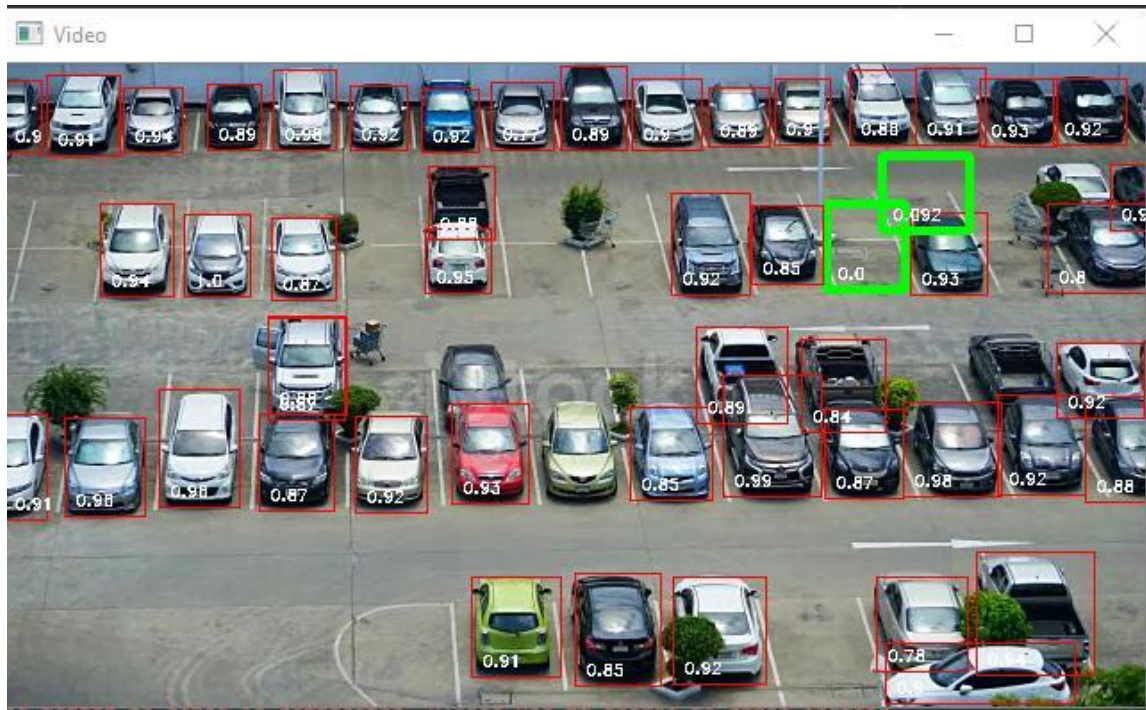


Рисунок 3.32 – Результати роботи алгоритму виявлення об’єктів та визначення вільного місця мережею Mask RCNN

З зображень 3.31 та 3.32 видно що мережі Faster RCNN та Mask RCNN дали досить схожий результат, але при аналізі метрик точності все ж таки можна зробити висновок, що Mask RCNN справляється з задачею краще, адже визначає машини з більшою точністю, – 90%, ніж Faster RCNN, – 85– 88%. В свою чергу мережа з архітектурою YOLO пропустила досить багато об’єктів, що є поганим результатом. Але використання попередньо навчених моделей теж має свої недоліки. При аналізі мережі Mask RCNN було визначено, що деякі машини вона визначає як інші предмети, а не як машини. Приклади показано на рисунках 3.33 та 3.34. Проаналізувавши теплові карти (heat maps), що зображено на рисунках 3.35 та 3.36, можна побачити, що мережа не виділяє достатньо деталей для визначення об’єкта як машини, але досить гарно виділяє сидіння, що дозволяє зробити висновок, хоча й з досить низькою точністю – 69%, що це стілець.



Рисунок 3.33 – Результати виявлення об’єктів попередньо-навченою мережею Mask RCNN



Рисунок 3.34 – Результати виявлення об’єктів попередньо-навченою мережею Mask RCNN

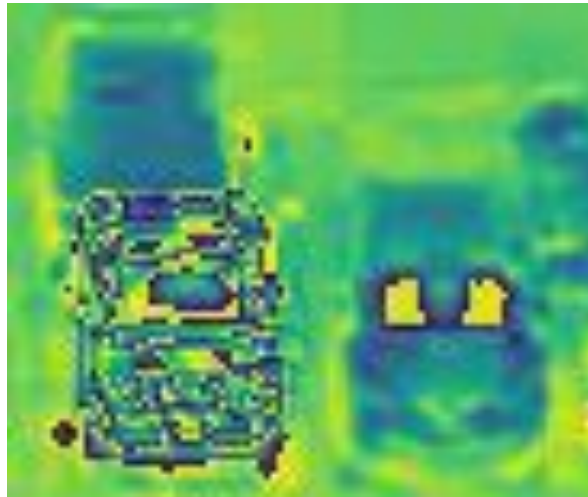


Рисунок 3.35 – Heat map для некоректно визначених об’єктів попередньо-навченою мережею Mask RCNN



Рисунок 3.36 – Heat map для некоректно визначених об’єктів попередньо-навченою мережею Mask RCNN

Та нарешті, останньою ідеєю було самостійно навчити мережу для визначення автомобілів. В умовах обмеженого часу та обчислювальних ресурсів було прийнято рішення провести експеримент лише з однією мережею, Mask RCNN, бо вона показала кращі результати на попередніх етапах. Результати можна побачити на рисунках 3.37 – 3.39.

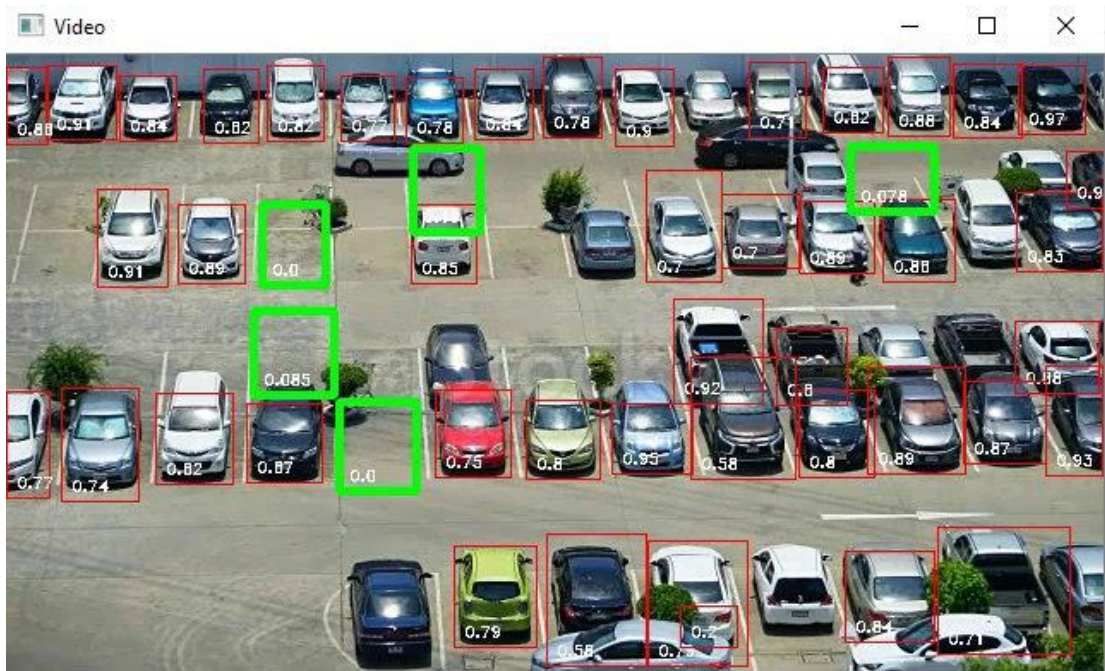


Рисунок 3.37 – Результати виявлення об'єктів самостійно-навченою мережею
Mask RCNN

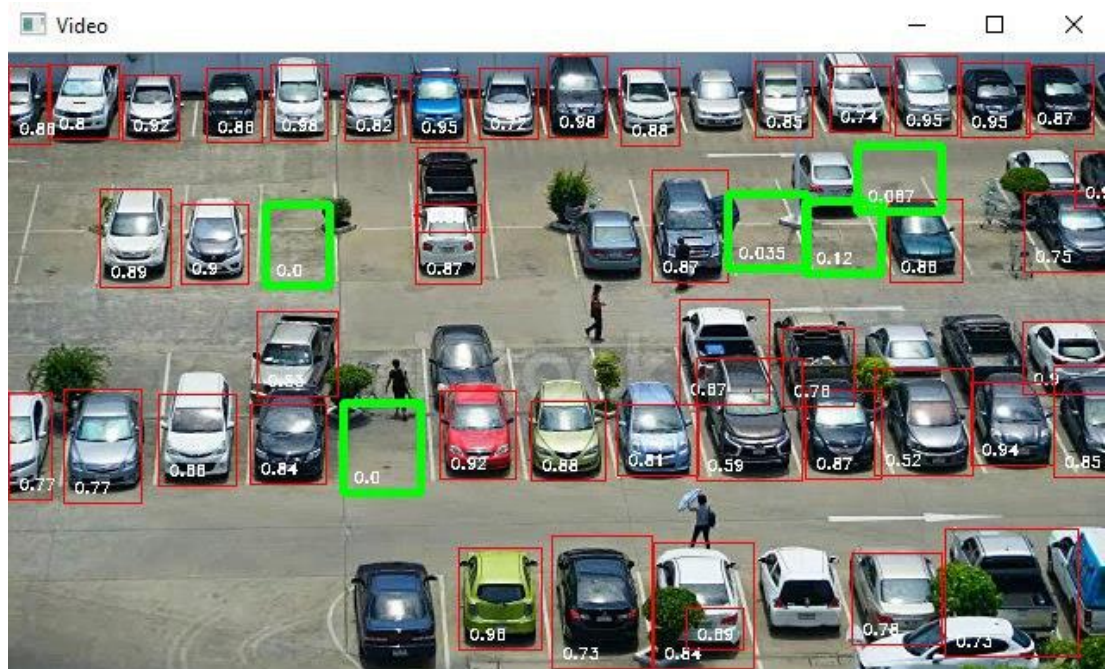


Рисунок 3.38 – Результати виявлення об'єктів самостійно-навченою мережею
Mask RCNN

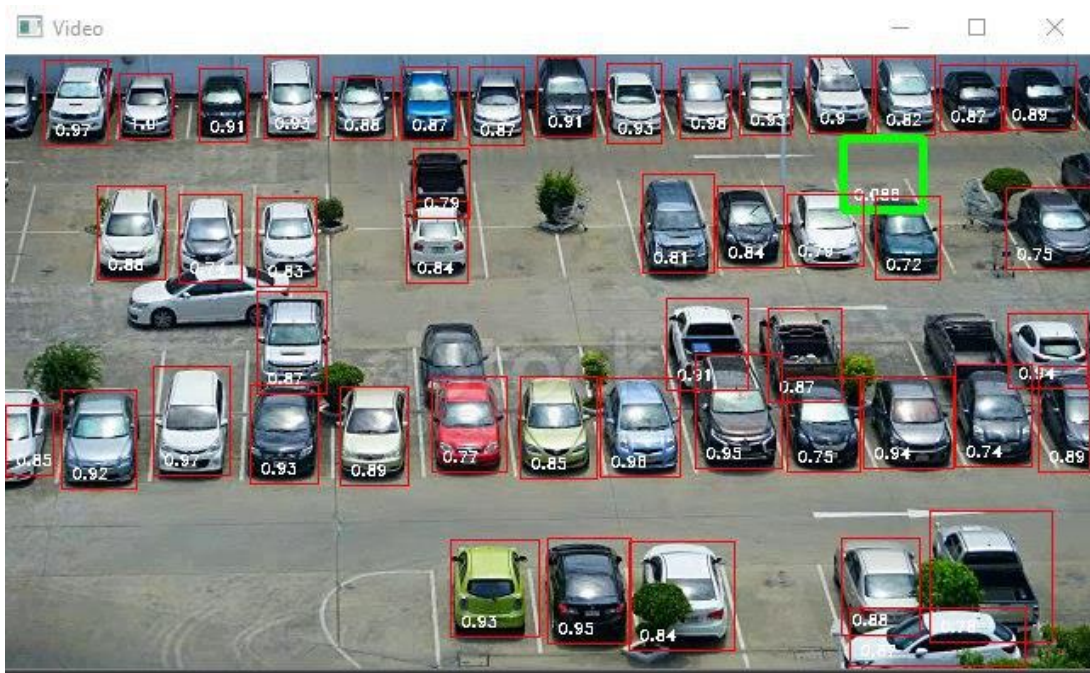


Рисунок 3.39 – Результати виявлення об’єктів самостійно-навченою мережею Mask RCNN

Модель почала визначати машини значно краще, з вищою точністю – 95%.

Модель була навчена не з нуля, а на основі вже відомих їй вагів COCO, тобто використовувався підхід так званого трансферного навчання. Навчання відбувалося на відкритих даних з набору Udacity Self Driving Car Dataset [21], тривалістю 15 епох. Навіть за такої малої кількості епох можна побачити значні поліпшення в точності виявлення автомобілів, тож можна зробити висновок, що збільшення кількості різноманітних даних та епох може, ще дати деякі покращення.

Також, якщо міркувати про ідеальний алгоритм, то він би застосовував відеоряд з паркування за попередній день для визначення паркувальних місць. Ряд проходить через нейронну мережу, яка визначає автомобілі та фіксує координати його знаходження. Таким чином, зафіксовані координати будуть слугувати розміткою для наступного дня.

Але, найпростіший підхід – це жорстке кодування розташування кожного місця для паркування в програмі вручну, замість того, щоб намагатися автоматично виявити місця для паркування. Але тоді, накладається обмеження на переміщення камери, бо доведеться ще раз жорстко кодувати місця паркування вручну.

Було проведено експеримент на заздалегідь розміченому відео. Результати можна побачити на рисунках 3.40 – 3.42.



Рисунок 3.40 – Результати виявлення об’єктів самостійно-навченою мережею Mask RCNN на розміченому відео

Як видно з зображень інколи відбуваються помилкові виявлення вільного або зайнятого місця, це залежить від різних факторів, таких як кут нахилу камери, освітлення паркування, погодних умов, та нарешті точність детекції нейронною мережею. Тож, алгоритм має ще місце для подальшого покращення.

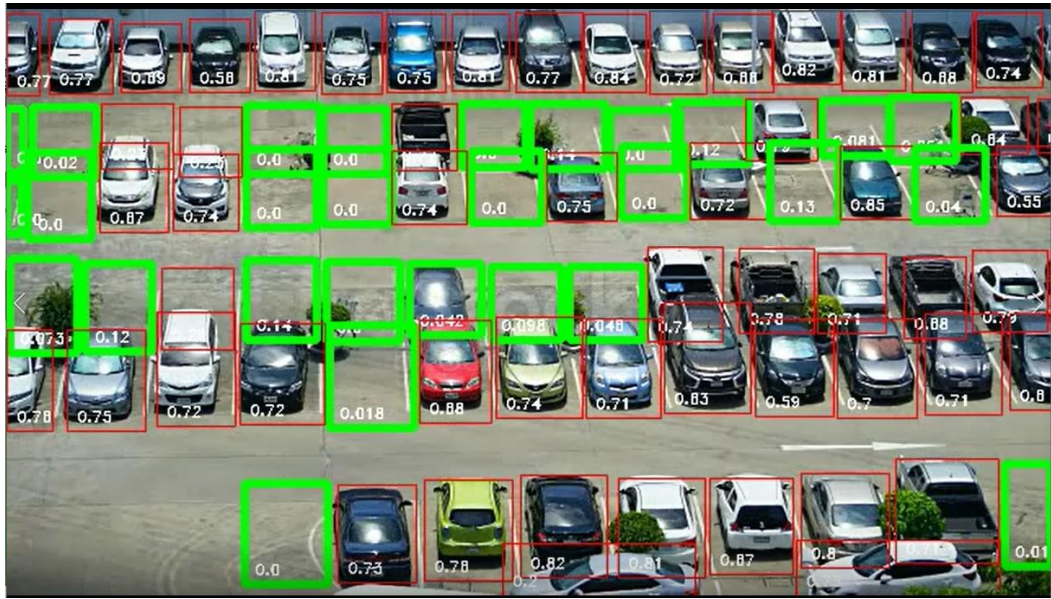


Рисунок 3.41 – Результати виявлення об’єктів самостійно-навченою мережею Mask RCNN на розміченому відео

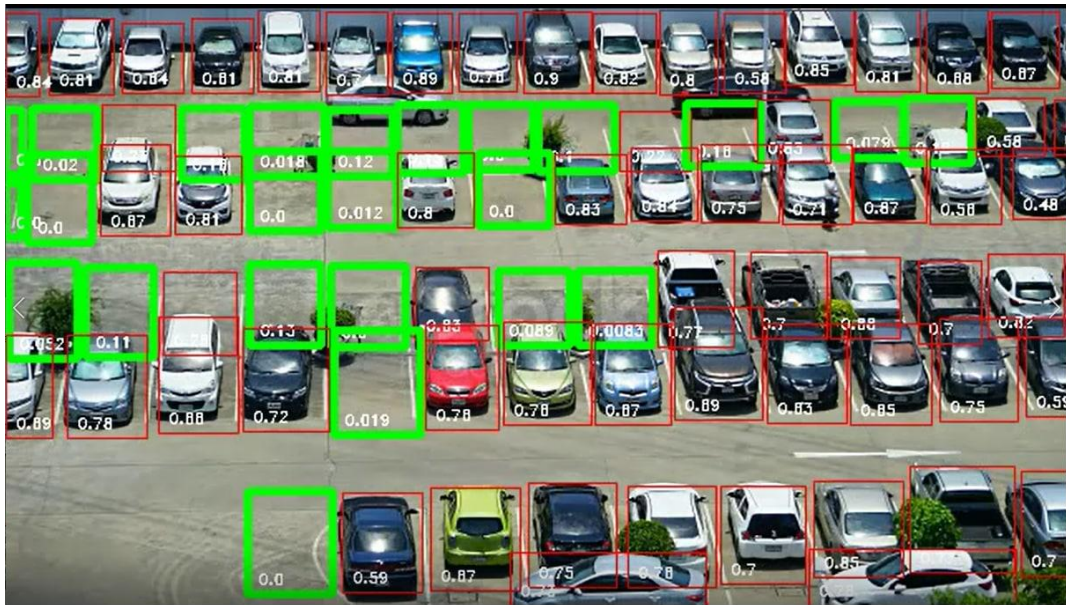


Рисунок 3.42 – Результати виявлення об’єктів самостійно-навченою мережею Mask RCNN на розміченому відео

ВИСНОВКИ

В ході проходження передатестаційної практики була проаналізована предметна галузь машинного та глибокого навчання – комп’ютерний зір та використання її технік для вирішення задачі виявлення вільних місць для паркування автівки.

Були розглянуті основні поняття і методи вирішення задачі виявлення об’єктів на зображеннях та їх реалізація за допомогою штучних нейронних мереж. Вивчено різноманітні модифікації згорткових штучних нейронних мереж, які отримали назву region-based convolutional network і алгоритми класичного машинного навчання для виявлення об’єктів. Також проаналізовано їх структуру і можливості.

У даній роботі були розглянуті алгоритми мереж Fast RCNN, Faster RCNN, Mask RCNN, а також інші варіанти – YOLO. Були виділені ключові особливості усіх алгоритмів і описано їх структуру та методи навчання.

В усіх трьох експериментах вдалося досягти точності виявлення об’єктів більше ніж 80%. В останньому експерименті з використанням архітектури мережі Mask RCNN точність сягнула 95%.

Були виявлені певні проблеми проаналізованих алгоритмів, головні з яких це виявлення об’єктів в різних масштабах, навчання для різної роздільної здатності зображення, швидкість та точність, дисбаланс класів.

Надані у роботі матеріали можливо використовувати з призначення для подальшого розвитку та вдосконалення алгоритму вирішення проблеми виявлення вільних місць для паркування автівки, та в навчальному процесі університету.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Найдич А. Большие данные: насколько они большие? URL: <http://compress.ru/article.aspx?id=23469> (дата звернення: 12.11.2020).
2. Maggo S., Aswani R. AUTOPARK: A Sensor Based, Automated, Secure and Efficient Parking Guidance System. IOSR Journal of Computer Engineering, 2013. Vol. 8. P. 47-56.
3. Abboud W.N., Automation of the Parking Industry: A Strategic and Managerial Overview. Massachusetts Institute of Technology, 1994.
4. Benson J. P., O'Donovan T., O'Sullivan P, Barton J., Murphy A., O'Flynn B., Car-Park Management using Wireless Sensor Networks. University College Cork (UCC). Tyndall National Institute. Cork, Ireland, 2004.
5. Bagula A., Castelli L., Zennaro M. On the Design of Smart Parking Networks in the Smart Cities: An Optimal Sensor Placement Model. Sensors, 2015.
6. Vlahogianni E.I, Kepaptsoglou K. A Real-Time Parking Prediction System for Smart Cities, Journal of Intelligent Transportation Systems, 2015.
7. True N. Vacant parking space detection in static images. University of California, San Diego, 2007.
8. Chunhe Y., Jilin L. A type of sensor to detect occupancy of vehicle berth in car park. 7th International Conference on Signal Processing, 2014. Vol. 3.
9. Jeffrey J., Patil R. G., Kumar S., Didagi Y., Bapat J., Das D. Smart Parking System Using Wireless Sensor Networks. 6th International Conference on Sensor Technologies and Applications, 2012. P. 306-311.
10. Klette R. Concise Computer Vision: An Introduction into Theory and Algorithms. London, 2014. 441 p.
11. Divvala S., Girshick R., Farhad A., Redmon J. You Only Look Once: Unified, Real-Time Object Detection, 2016.
12. Farhad A., Redmon J. YOLO9000: Better, Faster, Stronger, 2017.
13. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. University

of Washington, 2018.

14. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2016.

15. Girshick R., Fast R-CNN. IEEE International Conference on Computer Vision (ICCV), 2015.

16. He K., Gkioxari G., Dollar P., Girshick R. Mask R-CNN. Facebook AI Research (FAIR), 2016.

17. Paulo R.L. de Almeida, Luiz S. Oliveira, Alceu S. Britto Jr., Eunelson J. Silva Jr., Alessandro L. Koerich. PKLot – A Robust Dataset for Parking Lot Classification. Expert Systems with Applications, 2015.

18. Parking Lot Database. URL: <https://web.inf.ufpr.br/vri/databases/parking-lot-database/> (дата звернення: 20.11.2020).

19. CNRPark+EXT. A Dataset for Visual Occupancy Detection of Parking Lots. URL: <http://cnrpark.it/> (дата звернення: 21.11.2020).

20. Amato G., Carrara F., Falchi F. Car Parking Occupancy Detection Using Smart Camera Networks and Deep Learning. Pisa, Italy, 2016.

21. Udacity Self Driving Car Dataset. URL: <https://public.roboflow.com/object-detection/self-driving-car> (дата звернення: 21.11.2020).

22. Mask RCNN source code. URL: https://github.com/matterport/Mask_RCNN (дата звернення: 22.11.2020).

23. Tensorflow models, source code. URL: <https://github.com/tensorflow/models> (дата звернення: 23.11.2020).

24. TensorFlow Object Detection API. URL: https://github.com/tensorflow/models/tree/master/research/object_detection (дата звернення: 23.11.2020).

25. TensorFlow 2 Object Detection API tutorial. URL: <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/> (дата звернення: 24.11.2020).

26. Литвин В.В. Методи та засоби інженерії даних та знань. Львів: Магнолія, 2012. 241 с.