Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

РОЗПІЗНАВАННЯ ПІШОХОДІВ НА ЗОБРАЖЕННЯХ ДЛЯ САМОКЕРОВАНИХ АВТОМОБІЛІВ

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи
к.ф-м.н., ст. викладач, Шабінська М.О.

_____

(підпис)

“____” _____ 2020 р.

Виконала студентка
Шлепакова П. Д.
“____” _____ 2020 р.

Київ 2020

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. Кафедри інформатики,
доц., к.ф-м.н.

_____ С.С. Гороховський
(підпис)

“____” _____ 2019 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

Студентці 4 курсу факультету інформатики Шлепаковій Поліні Дмитрівні

ТЕМА: Розпізнавання пішоходів на зображеннях для самокерованих автомобілів

Вихідні дані:
Зміст ТЧ до курсової роботи:
Індивідуальне завдання
Календарний план
Анотація
Вступ
РОЗДІЛ 1: Огляд методів розпізнавання об’єктів на зображеннях
РОЗДІЛ 2: Вирішення задачі розпізнавання пішоходів на зображеннях
Висновки
Список використаної літератури
Додатки

Дата видачі “____” _____ 2019 р. Керівник _____
(підпис)
Завдання отримала _____
(підпис)

Тема: Розпізнавання пішоходів на зображеннях для самокерованих автомобілів

Календарний план виконання роботи:

№	Назва етапу	Термін виконання	Примітка
1.	Отримання завдання на курсову роботу	жовтень 2019	
2.	Вивчення літератури за темою «Методи розпізнавання об’єктів на зображеннях»	12.11.2019	
3.	Написання першого розділу курсової роботи	03.12.2019	
4.	Перегляд роботи з керівником	10.12.2019	
5.	Вивчення предметної області	14.01.2020	
6.	Вибір інструментів та методів вирішення задачі розпізнавання пішоходів	21.01.2020	
7.	Створення, тренування та оцінка моделей для вирішення задачі розпізнавання пішоходів	09.03.2020	
8.	Перегляд результатів з керівником	12.03.2020	
9.	Написання другого розділу курсової роботи	24.03.2020	
10.	Перегляд роботи з керівником	01.04.2020	
11.	Внесення змін до курсової роботи відповідно до зауважень наукового керівника	03.04.2020	
12.	Створення презентації	09.04.2020	

Студентка Шлепакова П. Д.

Керівник Шабінська М.О.

“____” _____

ЗМІСТ

Анотація

У даній курсовій роботі проведено огляд методів розпізнавання об’єктів на зображеннях, зокрема: R-CNN, SPP-net, Fast R-CNN, Faster R-CNN, YOLO, SSD та R-FCN. Розглянуті принципи їх роботи, переваги та недоліки, швидкість та точність розпізнавання.

У другій частині обрано найкращі методи вирішення задачі розпізнавання пішоходів на зображеннях для самокерованих автомобілів та застосовано їх на практиці за допомогою Tensorflow. Було зроблено висновок, що найкращим методом для вирішення даної задачі є Faster R-CNN з генератором ознак Resnet50, який досягає часу розпізнавання у 18 мілісекунд, точності – 88%, повноти – 94%.

Ключові слова: розпізнавання об’єктів на зображеннях, машинне навчання, Tensorflow, R-CNN, SPP-net, Fast R-CNN, Faster R-CNN, YOLO, SSD, R-FCN.

ВСТУП

Самокеровані автомобілі викликають багато інтересу зараз, але також викликають побоювання щодо їх безпечності. Невід'ємною умовою гарантування безпеки самокерованого автомобіля є своєчасне виявлення пішоходів на дорогах. Для оцінки навколишнього середовища самокеровані машини мають певну комбінацію відеокамер, радарів та лідарів. За допомогою методів машинного навчання можна аналізувати відео з камери та розпізнавати різні об'єкти навколо (зокрема, пішоходів).

У даній роботі аналіз кадрів з відео здійснюватиметься незалежно, хоча існують методи, які дозволяють відстежувати зміни між послідовними кадрами. Таким чином, задача зводиться до розпізнавання пішоходів на зображеннях. На сьогоднішній день, існують сотні методів розпізнавання об'єктів на зображеннях, але неможливо визначити найкращий з них, оскільки це сильно залежить від вимог предметної області. Саме через актуальність цієї теми, було вирішено присвятити час її вивченню.

Мета курсової роботи полягає у аналізі наявних методів розпізнавання об'єктів на зображеннях та застосуванню найкращих із них на практиці для вирішення задачі розпізнавання пішоходів.

Для досягнення зазначеної мети потрібно виконати такі завдання:

1. Зробити огляд найкращих та найпопулярніших методів розпізнавання об'єктів на зображеннях.
2. Порівняти характеристики (зокрема, швидкість та точність) розглянутих методів та обрати найкращі з них для поставленої задачі.
3. Застосувати обрані методи для розпізнавання пішоходів на зображеннях та порівняти результати.

Об'єкт дослідження: методи розпізнавання об'єктів на зображеннях.

Методи дослідження: ознайомлення з науковими роботами, статтями, уроками та застосування отриманих знань на практиці.

Джерела дослідження: наукові роботи присвячені методам розпізнавання об’єктів на зображеннях, статті та відповіді на спеціалізованих форумах.

1 Огляд методів розпізнавання об’єктів на зображеннях

Задача розпізнавання об’єктів на зображеннях полягає у знаходженні певних (попередньо визначених класами) об’єктів на зображеннях, визначенні прямокутних обмежувальних рамок та класу для кожного з них. Поява згорткових нейронних мереж (Convolutional Neural Network, CNN) дозволила зробити великий крок у розпізнаванні об’єктів на зображеннях, і майже усі сучасні системи використовують CNN у тій чи іншій формі.

Одною з основних проблем у задачі розпізнавання об’єктів є те, що вона має подвійні пріоритети: визначення класу об’єкта та його положення (обмежувальної рамки). Існують вже натреновані CNN, які здатні класифікувати зображення, але вони припускають, що на зображенні знаходиться один об’єкт, який займає майже всю площу зображення. Тоді як у задачі розпізнавання об’єктів, об’єкт може бути довільного розміру, знаходитися у довільному місці, до того ж, об’єктів на зображенні може бути довільна кількість. Отже, вищезгадані CNN не можна у чистому вигляді застосувати до даної задачі.

Наївним підходом до вирішення цієї проблеми буде створити рамку, яка рухатиметься по зображенню, перебираючи усі можливі позиції об’єкта. Для кожної такої позиції, частина зображення що розташована всередині рамки використовуватиметься як вхідне зображення для певної CNN, що класифікує зображення. Це здається простим рішенням, але є проблема: об’єкти (навіть того ж класу) можуть мати дуже різні розміри та співвідношення сторін на зображенні. Якщо використовувати CNN для класифікації частини зображення у кожній можливій позиції, до того ж, з різними розмірами рамок, алгоритм займатиме забагато часу для його практичного застосування [1].

Рис. 1: Ілюстрація рухомої рамки (зліва) та рамок з різними розмірами та співвідношеннями сторін (справа) [1].

1.1 R-CNN

Щоб вирішити проблему величезної кількості можливих рамок, було запропоновано [7] використати алгоритм селективного пошуку [2]. Селективний пошук має велику повноту, але маленьку точність, тому не може бути використаний для знаходження об’єктів самостійно. Тим не менш, він дозволяє згенерувати обмежену кількість регіонів інтересу (RoI), які містять шукані об’єкти з більшою ймовірністю, ніж випадково обрані регіони.

На першій стадії методу R-CNN (Regions with CNN features) [7], генерується 2000 пропозицій регіонів (рамок) за допомогою селективного пошуку. За отриманими рамками вирізаються відповідні частини зображення, пропорції та розміри кожного з цих зображень змінюються, щоб отримати квадрати фіксованого розміру. Отримані квадратні зображення подаються на вхід CNN, яка на виході видає вектор ознак довжиною 4096.

Після цього отриманий вектор ознак подається на вхід до методу опорних векторів (support vector machine, SVM) [5], щоб класифікувати даний регіон. Для уточнення обмежувальної рамки використовується регресія обмежувальної рамки (bounding box regression) – метод, введений у [7], який передбачає використання CNN, але без останнього sigmoid або softmax рівня.

Натомість, CNN видає на виході 4 значення: x, y, w, h. Координати (x, y) позначають верхній лівий кут обмежувальної рамки, w та h – ширину та висоту відповідно. Незважаючи на те, що алгоритм селективного пошуку надає обмежувальні рамки, цей крок необхідний, тому що рамки можуть бути неточними. Наприклад, регіон може містити лише половину машини, але все одно бути класифікований як машина.

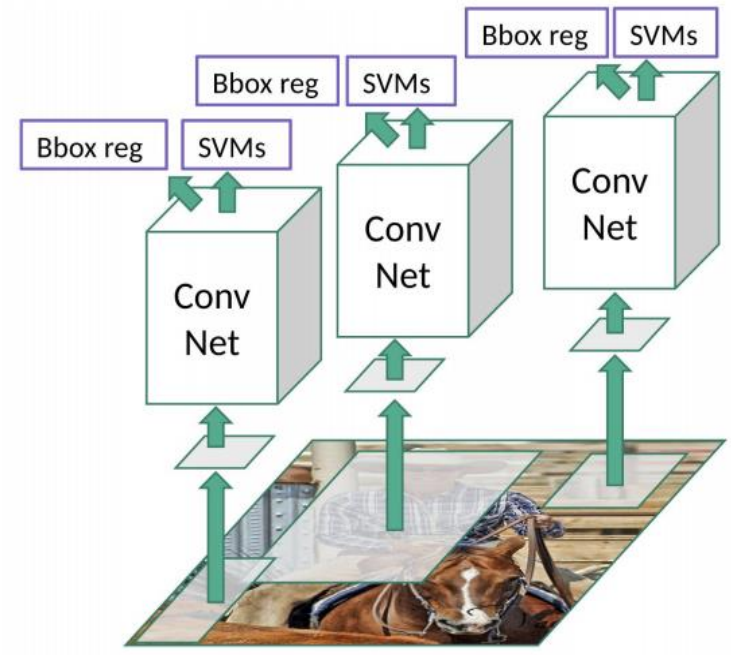

Рис. 2: Ілюстрація алгоритму R-CNN [11].

Але у цього алгоритму є такі проблеми:

- Необхідно дуже багато часу на тренування моделі, оскільки потрібно класифікувати по 2000 регіонів для кожного зображення. До того ж, відбуваються зайві обрахунки: кожна пропозиція регіону обробляється незалежно, навіть якщо певні регіони перетинаються.
- Обробка одного зображення займає 47 секунд [8], отже R-CNN не може бути застосований у багатьох системах реального часу.
- Селективний пошук є фіксованим алгоритмом, і на цій стадії модель не навчається, що може призвести до невдалих пропозицій регіонів.

– CNN, на вхід якій подаються регіони, через свою структуру, приймає на вхід зображення тільки фіксованого розміру. Через це потрібно змінювати розмір та пропорції пропозицій регіонів, що може знизити точність класифікації.
– Тренування має кілька стадій: CNN, SVM та регресор обмежувальної рамки тренуються окремо, і результати одного тренування не впливають на інше.

1.2 SPP-net

Ідея агрегування просторових пірамід (Spatial Pyramid Pooling, SPP) [3] зробила вагомий внесок у еволюцію R-CNN.

В алгоритмі R-CNN використовувалась CNN, яка приймає на вхід зображення тільки одного, фіксованого, розміру. Чому так? CNN складається в основному з двох частин: згорткові (convolutional) рівні та повноз'єднані (fully-connected) рівні. В основу згорткового рівня покладена рухома рамка, тому розмір результуючих карт ознак (feature maps) залежить від розміру вхідних даних. Це означає, що згорткові рівні не потребують зображення фіксованого розміру, і можуть згенерувати карти ознак будь-якого розміру. Але повноз'єднані рівні потребують вхідні дані фіксованого розміру, оскільки вони мають фіксовану кількість вхідних нейронів. Отже, необхідність фіксованих вхідних даних в CNN з'являється тільки через повноз'єднані рівні, які знаходяться на більш глибоких рівнях мережі.

Щоб позбавитися цього обмеження, було запропоновано вставити SPP рівень між згортковими рівнями та повноз'єднаними рівнями. Таку архітектуру мережі було названо SPP-net [3]. SPP рівень агрегує карти ознак таким чином, що навіть для різних початкових розмірів, результати матимуть однаковий розмір. Це дозволяє уникнути обрізання або зміни пропорцій зображення на самому початку, виконуючи агрегування інформації пізніше, після отримання карт ознак.

SPP працює на трьох рівнях: перший рівень має одну комірку, яка містить усі вхідні дані, на другому рівні дані діляться на 4 комірки за допомогою 2×2 решітки, на третьому – 16 комірок за допомогою 4×4 решітки. До кожної комірки застосовується максимізаційне агрегування (max pooling). Наприклад, якщо результат останнього згорткового рівня має 256 каналів, то в результаті агрегування одної комірки отримаємо вектор ознак довжиною 256. Вектори ознак для кожної з комірок конкатенуються і передаються на вхід до повноз'єднаного рівня.

Кількість каналів у результаті останнього згорткового рівня залежить від кількості фільтрів на цьому рівні, що залежить від архітектури мережі. Таким чином, довжина вектору ознак залежить тільки від архітектури мережі, а не від розміру зображення, тож на вхід CNN можна подавати зображення довільного розміру.

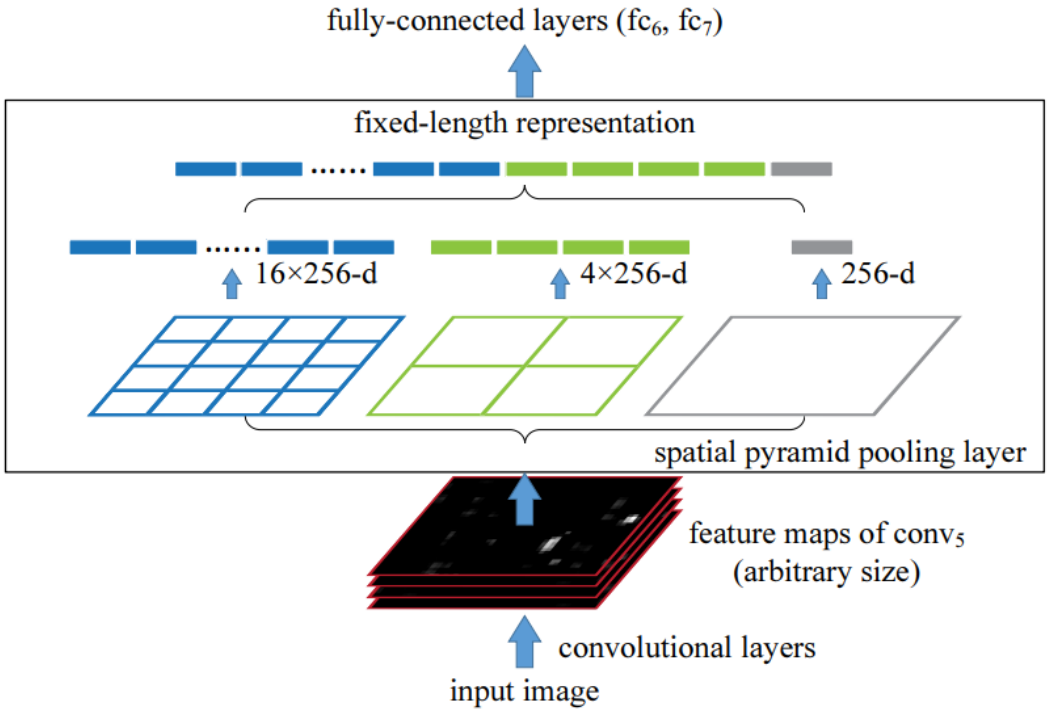

Рис. 3: Структура мережі зі spatial pyramid pooling рівнем. Тут 256 - це кількість каналів у результаті $conv_5$ рівня, а $conv_5$ - останній згортковий рівень [3].

Але SPP допомагає вирішити ще одну суттєву проблему R-CNN: необхідність обробляти кожну пропозицію регіону окремо. До цього, оскільки CNN приймала на вхід тільки зображення фіксованого розміру, потрібно було кожен регіон змінювати під потрібний розмір, та подавати його на вхід окремо. Оскільки з SPP конвертація інформації під необхідний розмір відбувається тільки після згорткових рівнів, можна ділити зображення на регіони вже після згорткових рівнів. Таким чином, згорткові рівні можуть обробити зображення повністю і утворити одну карту ознак, після чого з неї буде отримано 2000 карт ознак – по одній для кожного регіону. Далі до кожної з них буде застосовано SPP та повноз'єднані рівні. Це дозволяє суттєво зменшити складність моделі.

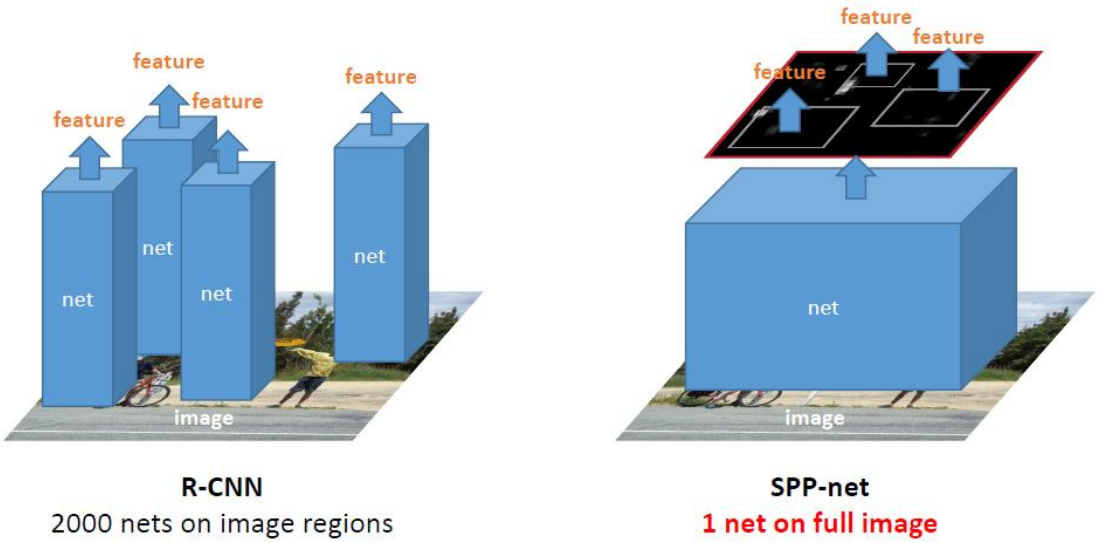

Рис. 4: Порівняння принципу роботи R-CNN та SPP-net [4]. В R-CNN використовується 2000 CNN – по одній для кожного регіону, і з кожної з них отримується карта ознак. В SPP-net використовується одна CNN на все зображення, і з однієї спільної карти ознак отримується 2000 карт ознак для регіонів.

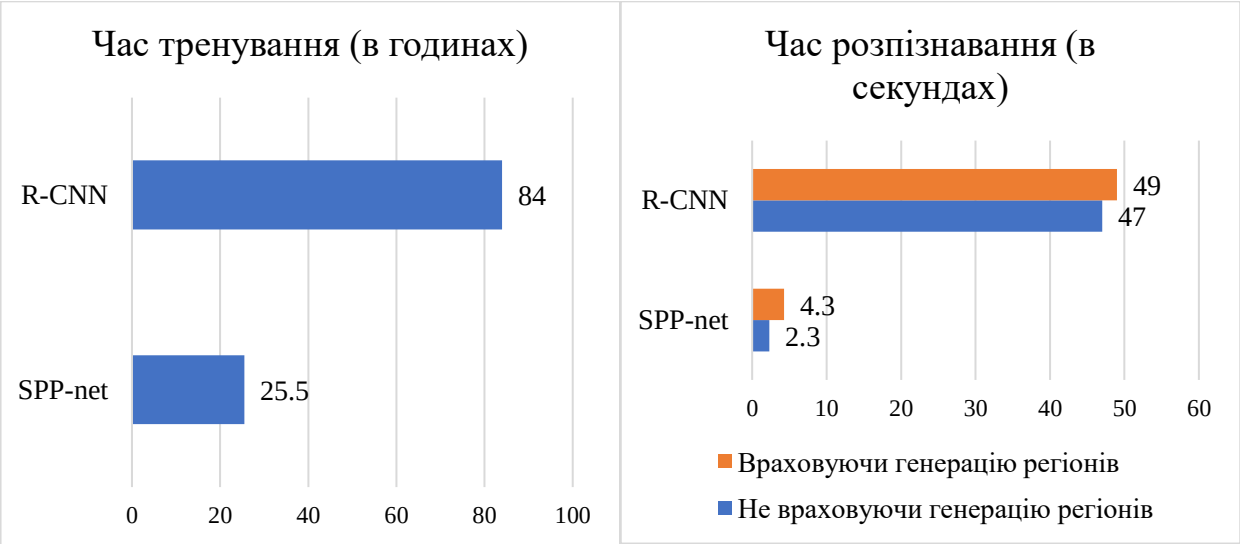

Рис. 5: Порівняння часу тренування і розпізнавання R-CNN та SPP-net [7, 3].

Як ми бачимо з рис. 5, використання SPP допомогло скоротити час тренування більше ніж в 3 рази, а час розпізнавання одного зображення – більше ніж в 11 разів. Також можемо помітити, що в R-CNN генерація пропозицій регіонів займає лише 4% часу розпізнавання, тоді як в SPP-net вона займає вже 47%.

Отже, завдяки використанню SPP вдалося вирішити кілька проблем R-CNN: довге тренування, спровоковане застосуванням CNN до кожного з 2000 регіонів незалежно, та необхідність обрізати або змінювати пропорції зображення. Також з SPP значно скоротився час розпізнавання одного зображення, хоча обробка за 4,3 секунди все одно може бути недостатньо швидкою для багатьох систем реального часу. Втім, все одно залишаються проблеми того, що тренування має кілька стадій, які тренуються окремо і не впливають одна на одну, а також, що селективний пошук є фіксованим алгоритмом, і під час нього не відбувається навчання.

1.3 Fast R-CNN

Fast R-CNN використовує R-CNN та SPP-net як основу, з кількома відмінностями:

– Замість SPP використовується агрегування регіонів інтересу (Region of Interest pooling, RoI pooling). Це спрощена версія SPP з лише одним рівнем – третім. Тобто регіон ділиться на 16 комірок за допомогою 4×4 решітки, до кожної комірки застосовується максимізаційне агрегування, і отримані вектори конкатенуються.
– Замість SVM класифікатора використовується softmax класифікатор. Основною відмінністю цих класифікаторів є їх функція втрат. SVM класифікатор будує гіперплощину у просторі високої або нескінченної вимірності, яка розділятиме точки різних класів, причому так, щоб найближчі до гіперплощини точки знаходились якнайдалі від неї. У якості функції втрат SVM використовує завісні втрати (hinge loss). Softmax передбачає ймовірність належності до певного класу, і в основу функції втрат покладено перехресну ентропію.
– Тренування відбувається в одну стадію завдяки використанню багатозадачної функції втрат. Завдяки цьому, під час тренування оновлюються усі рівні мережі, що дозволяє досягти більшої точності.

Отже, опишемо як працює Fast R-CNN. Нейронна мережа отримує на вхід зображення та набір регіонів інтересу, і створює карту ознак. Використовується агрегування регіонів інтересу, щоб отримати вектор ознак для кожного регіону. Далі кожен з векторів ознак окремо подається на вхід послідовності повноз'єднаних рівнів, після чого модель розгалужується на два вихідні рівні: softmax класифікатор, який видає вектор довжиною C (C – кількість класів) з передбаченими ймовірностями для кожного класу та регресор обмежувальної рамки, який видає по 4 цілі числа (x та y координата верхньої лівої точки, ширина та висота) для кожного з C класів, які позначають межі обмежувальної рамки. Описана модель тренується за допомогою багатозадачної функції втрат, що представляє собою зважену суму функцій втрат для класифікації та визначення обмежувальної рамки.

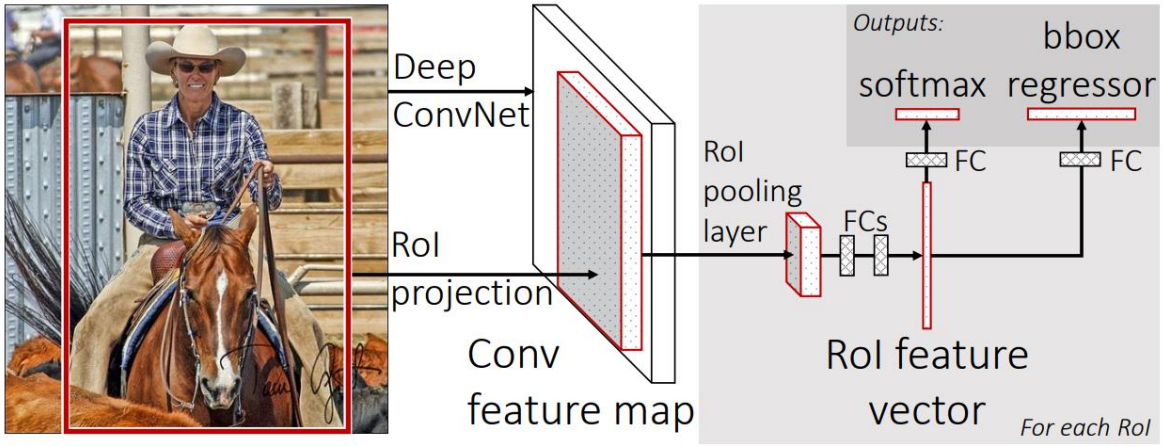

Рис. 6: Ілюстрація архітектури Fast R-CNN [8].

До того ж, Fast R-CNN має більш ефективне тренування, ніж R-CNN та SPP-net, завдяки іншій стратегії вибору регіонів для міні-партій (mini-batch) стохастичного градієнтного спуску (Stochastic Gradient Descent, SGD). Проблемою R-CNN та SPP-net тренування є те, що регіони для міні-партій обираються з різних зображень. До того ж, регіони можуть бути дуже великими, і займати майже все зображення. Це приводить до великої кількості даних, які мають бути пропаговані для одної міні-партії, і до повільного тренування.

У Fast R-CNN вирішили запровадити спільне використання ознак під час тренування. Регіони для міні-партій обираються ієрархічно, тобто спочатку обирається N зображень, а потім з кожного зображення обирається R/N регіонів. Таким чином, регіони, які належать одному зображенню, можуть спільно використовувати пам’ять та обрахунки, що дозволяє значно підвищити ефективність тренування. Наприклад, при $N = 2$ та $R = 128$, запропонована схема тренування приблизно в 64 рази швидше, ніж якщо обирати по одному регіону зі 128 різних зображень (стратегія R-CNN та SPP-net) [8].

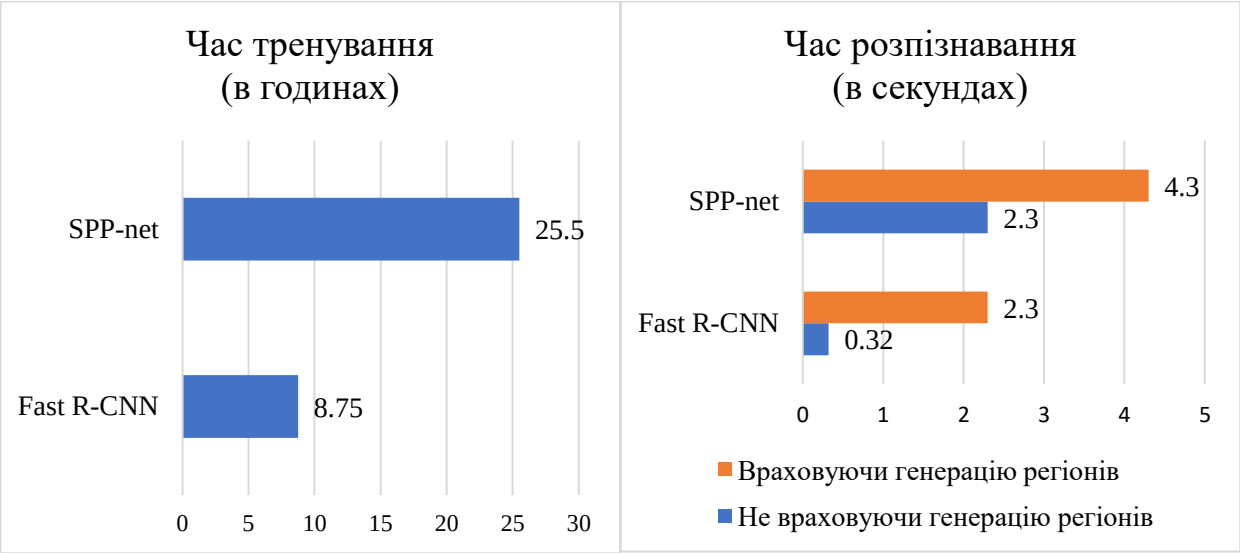

Рис. 7: Порівняння часу тренування і розпізнавання SPP-net та Fast R-CNN [3, 8].

Як ми бачимо з рис. 7, час тренування для Fast R-CNN зменшився в 2,9 разів порівняно з SPP-net, а час розпізнавання одного зображення – в 1,9 разів. Також можемо помітити, що в SPP-net генерація пропозицій регіонів займає 47% часу розпізнавання, тоді як в Fast R-CNN вона займає вже 86%. Таким чином, генерація пропозицій регіонів, яка відбувається за допомогою алгоритму селективного пошуку, стає проблемним місцем в Fast R-CNN.

1.4 Faster R-CNN

Faster R-CNN – це фінальний крок у еволюції R-CNN, який вирішує останню проблему, що залишилась в підході R-CNN: використання селективного пошуку для генерації пропозицій регіонів. Використання селективного пошуку є проблемним через кілька причин:

- Це фіксований алгоритм (модель не навчається на цій стадії), отже він може видавати невдалі пропозиції регіонів, які не можна покращити.
- Як ми бачимо з рис. 7, селективний пошук займає більшу частину часу розпізнавання в Fast R-CNN, і не дає можливості значно прискорити алгоритм.
- Залежність від зовнішнього алгоритму.

У Faster R-CNN було вирішено [9] замість селективного пошуку створити власну мережу для пропозицій регіонів (Region Proposal Network, RPN), яка буде використовувати згорткові рівні спільно з Fast R-CNN детектором, роблячи генерацію регіонів майже безкоштовною з точки зору часу.

Faster R-CNN складається з двох модулів. Перший модуль – це глибока повністю згорткова нейронна мережа, яка пропонує регіони, а другий модуль – це Fast R-CNN детектор, який використовує запропоновані регіони. Повна система є єдиною мережею, яка тренується від початку до кінця за допомогою багатозадачної функції втрат.

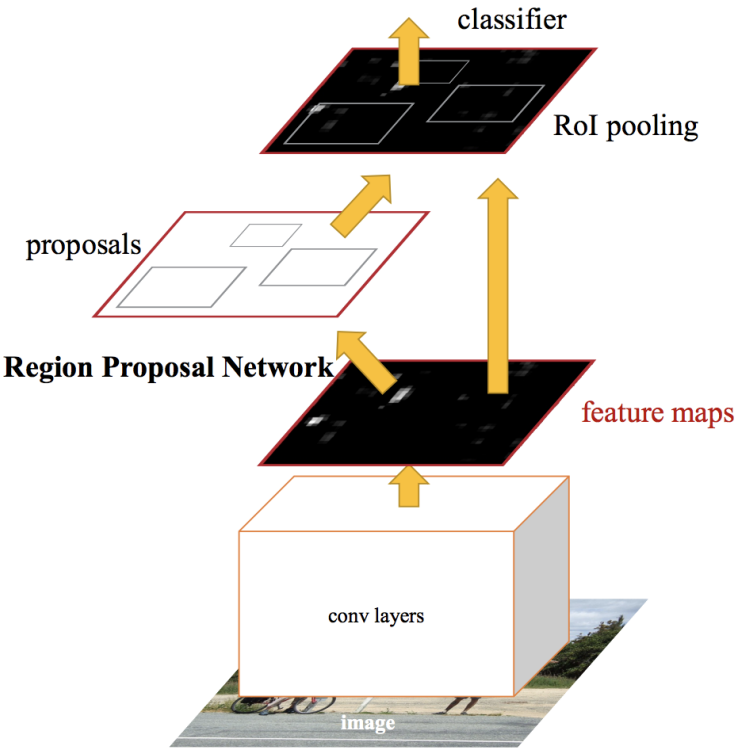

Рис. 8: Ілюстрація архітектури Faster R-CNN [9].

RPN приймає на вхід зображення (будь-якого розміру) та видає набір прямокутних пропозицій регіонів, кожен з них з оцінкою об’єктності. Під об’єктністю мається на увазі ймовірність належності регіону до набору класів, а не до фону [9]. Оскільки RPN має використовувати згорткові рівні спільно з Fast R-CNN детектором, ми припускаємо, що обидві мережі мають певну

кількість спільних згорткових рівнів. Щоб згенерувати пропозиції регіонів, створюється маленька рамка розміром $n \times n$ (наприклад, $n = 3$), яка рухається по карті ознак, створеній останнім спільним згортковим рівнем. На кожній з позицій рамки створюється вектор ознак (наприклад, довжиною 256), після чого модель розгалужується на два повноз'єднані рівні: регресор обмежувальної рамки (reg) та класифікатор рамки (cls).

На кожній позиції рамки одночасно генеруються пропозиції кількох регіонів, і максимальна можлива кількість запропонованих регіонів на одній позиції позначається k. Отже, reg рівень має розмір результату $4k$, що позначає координати обмежувальних рамок регіонів, а рівень cls має розмір результату $2k$, що позначає ймовірність належності кожного регіону до об'єкту або не об'єкту (у [9] було зроблено примітку, що cls рівень реалізовано як двокласний softmax рівень для простоти, але альтернативно можна було використати логістичну регресію, щоб отримати k значень). Вищезгадані k пропозицій регіонів параметризуються відносно k допоміжних рамок, які називаються якорями. Якір центрований відносно позиції рухомої рамки та створюється за допомогою двох значень: розмір і співвідношення сторін. За замовчуванням використовується 3 різні розміри та 3 співвідношення сторін, отже на кожній позиції рухомої рамки створюється $k = 9$ якорів.

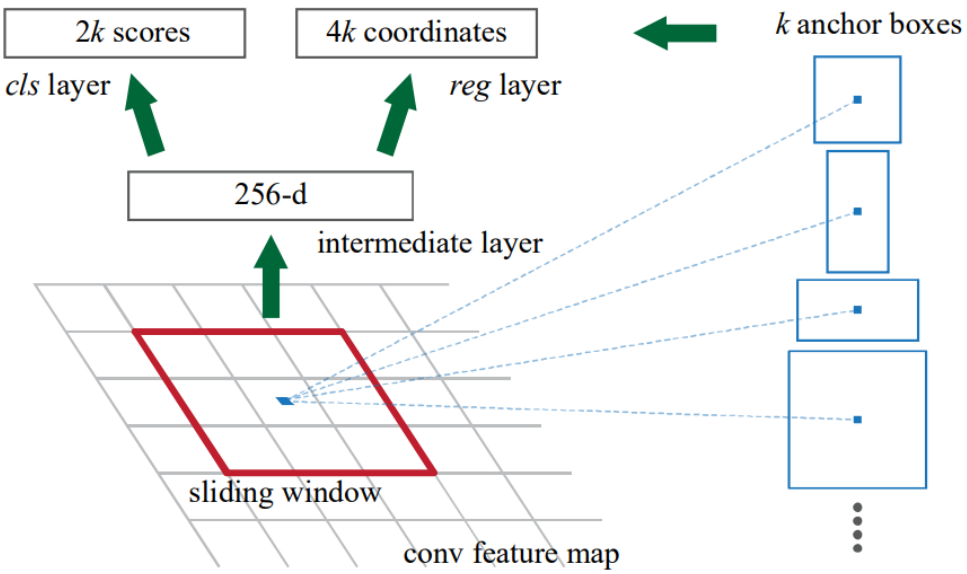

Рис. 9: Ілюстрація архітектури RPN [9].

Для тренування RPN, якорі повинні бути позначені як об'єкт або не об'єкт (позитивна або негативна оцінка). Позитивна оцінка надається якорю у двох випадках: якщо цей якір має найбільше значення коефіцієнту Жаккара (також відомий як Intersection over Union, IoU) з цільовою (ground truth) рамкою або якщо він має $IoU > 0{,}7$ з будь-якою цільовою рамкою. Зазвичай другої умови достатньо, щоб позначити потрібні якорі, але перша умова все одно використовується на випадок, якщо немає якорів, що задовольняють другій умові. Варто зауважити, що одна цільова рамка може надавати позитивні оцінки декільком якорям. Негативна оцінка надається непозитивному якорю, якщо його $IoU < 0{,}3$ для всіх цільових рамок. Якорі, які не позначені ні позитивними, ні негативними оцінками, не впливають на тренування. Рівень reg перетворює позитивні якорі на пропозиції регіонів, та тренується таким чином, щоб наблизити межі пропозицій регіонів до цільових рамок.

Функція втрат для Faster R-CNN враховує такі 4 втрати:

– RPN класифікації (об'єкт чи не об'єкт)
– RPN регресії обмежувальної рамки (рамка пропозиції регіону)

– Fast R-CNN класифікації (фінальна класифікація рамки в один з класів)
– Fast R-CNN регресії обмежувальної рамки (фінальна рамка)

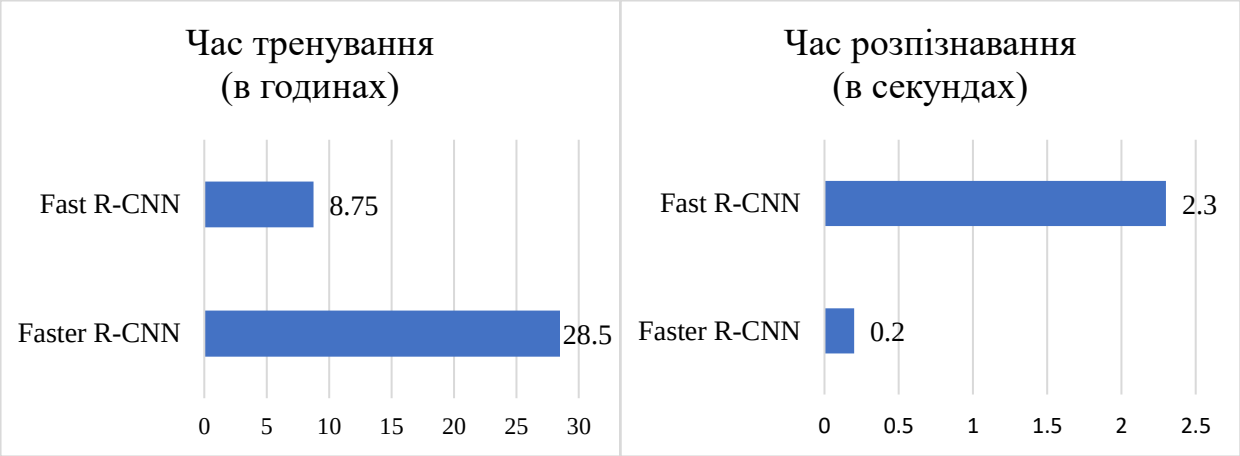

Рис. 10: Порівняння часу тренування і розпізнавання Fast R-CNN та Faster R-CNN [8, 9].

Як ми бачимо з рис. 10, час тренування для Faster R-CNN збільшився більше ніж в 3 рази порівняно з Fast R-CNN, що цілком очікувано, оскільки окрім Fast R-CNN потрібно також натренувати RPN, до того ж таким чином, щоб RPN та Fast R-CNN мали спільні згорткові рівні. Зате час розпізнавання одного зображення зменшився аж в 11,5 разів, що є більш важливим показником, особливо якщо алгоритм використовуватиметься в системах реального часу.

1.5 YOLO

Усі вищеперераховані методи використовували той чи інший метод генерації пропозицій регіонів, який використовувався детектором об’єктів. Такі детектори називають регіональними. Однокрокові детектори використовують одну нейронну мережу і для класифікації, і для локалізації об’єктів, не використовуючи пропозиції регіонів. Одним із них є YOLO (You Only Look Once) [10].

Спочатку зображення ділиться на комірки $S \times S$ решіткою. За знаходження кожного з об'єктів відповідає та комірка, в якій знаходиться центр об'єкта.

Кожна комірка передбачає B обмежувальних рамок та оцінку впевненості для кожної з рамок. Оцінка впевненості відображає наскільки модель впевнена в тому, що рамка містить об'єкт та наскільки точною є рамка. Оцінка впевненості виражається формулою $Pr(Object) * IoU_{pred}^{truth}$. Якщо комірка не містить жодного об'єкта, оцінка має бути нульовою. Інакше оцінка має дорівнювати IoU між передбаченою та цільовою обмежувальними рамками. Кожна рамка репрезентована 5 значеннями: x, y, w, h та оцінка впевненості. Координати (x, y) позначають центр рамки відносно меж комірки. Ширина і висота визначені відносно повного зображення.

Кожна комірка також передбачає C ймовірностей (для кожного з C класів). Ці ймовірності показують які об'єкти можуть міститися в конкретній комірці. Для кожної комірки передбачається тільки один набір ймовірностей за класами, незважаючи на кількість обмежувальних рамок B. Таким чином, передбачення моделі зберігаються у вигляді $S \times S \times (B * 5 + C)$ тензору.

Після цього, ймовірності за класами для кожної комірки множаться на оцінки впевненості для кожної з обмежувальних рамок, що передбачені даною коміркою. В результаті у кожної рамки є C оцінок впевненості (для кожного класу). Ці оцінки демонструють і ймовірність наявності об'єкту певного класу в обмежувальній рамці, і те, наскільки добре передбачена рамка окреслює об'єкт. Далі, за допомогою пригнічення немаксимумів (non-maximum suppression) та фільтрації за порогом оцінки, отримуються фінальні рамки.

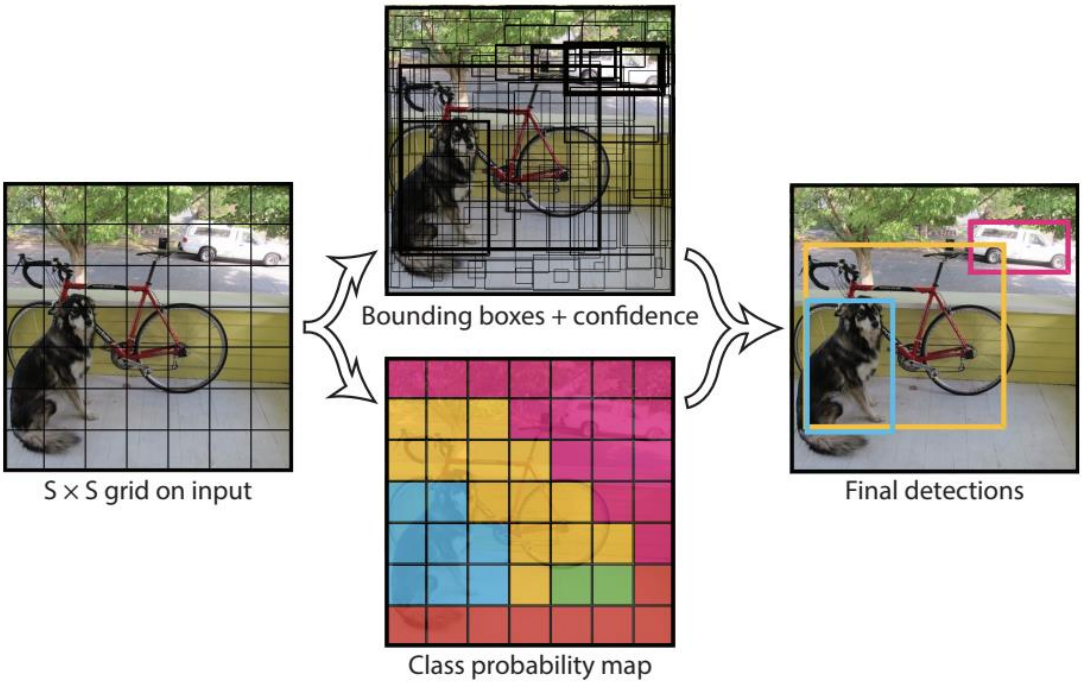

Рис. 11: Ілюстрація алгоритму YOLO [10]. На верхньому зображенні товщина рамки позначає оцінку впевненості (чим більше товщина, тим вище оцінка).

Найбільшою перевагою YOLO є неймовірна швидкість розпізнавання – 45 фреймів за секунду (Frames Per Second, FPS), що в 10 разів швидше за Faster R-CNN. Це означає, що YOLO можна застосувати для обробки відео в реальному часі з затримкою менше ніж 23 мілісекунди.

Існує також швидка версія YOLO зі швидкістю 155 FPS. Fast YOLO відрізняється від YOLO меншою кількістю згорткових рівнів (9 замість 24) та меншою кількістю фільтрів на цих рівнях. Усі інші параметри не відрізняються.

Іншою важливою перевагою є те, що на відміну від методів, що використовують рухомі рамки та пропозиції регіонів, YOLO бачить повне зображення під час тренування і тестування, тому неявно враховує контекст при розпізнаванні об’єктів. Наприклад, Fast R-CNN часто помилково передбачає, що шматки фону є об’єктами через неможливість побачити повну

картину. YOLO робить більш ніж в два рази менше таких помилок, ніж Fast R-CNN [10].

Ще одним плюсом є те, що YOLO вивчає більш узагальнені репрезентації об’єктів. При тренуванні на природних зображеннях та тестуванні на мистецьких роботах, YOLO показав значно кращі результати, ніж інші популярні детектори. Це означає, що YOLO може бути більш стійким до застосування в нових доменах або до непередбачуваних вхідних даних.

На жаль, YOLO має також певні обмеження та недоліки:

– YOLO накладає сильні просторові обмеження на передбачення обмежувальних рамок, оскільки кожна комірка може передбачити тільки 2 рамки (за замовчуванням кількість рамок на комірку $B = 2$). Це обмежує кількість близько розташованих об’єктів, які YOLO може передбачити. Таким чином, YOLO має проблеми з розпізнаванням маленьких об’єктів, які з’являються у групах, наприклад зграї птахів.
– Оскільки модель навчається передбачати обмежувальні рамки з тренувальних даних, у неї можуть виникнути проблеми з розпізнаванням об’єктів з новими чи незвичними співвідношеннями сторін або положеннями. Також модель орієнтується на доволі грубі ознаки, оскільки в архітектурі є багато рівнів, які зменшують розміри даних (тобто менш грубі ознаки можуть просто зникнути в процесі).
– Функція втрат, яка використовується для оцінки якості розпізнавання, однаково штрафує просторові відхилення як у великих, так і у маленьких рамок. Тоді як маленька помилка у маленькій рамці має більший вплив на IoU, ніж така ж помилка у великій рамці. Через це YOLO має проблеми з точною локалізацією об’єктів, особливо маленьких. Також, точність YOLO гірше, ніж у Faster R-CNN (див. рис. 13).

1.6 SSD

Метод YOLO, завдяки своїй швидкості, дозволяє розпізнавати зображення в реальному часі з незначною затримкою, але поступається точністю іншим, більш повільним детекторам. SSD (Single Shot MultiBox Detector) [12] – перший детектор, який не використовує пропозиції регіонів, але досягає не гіршої точності, ніж ті, які використовують.

SSD працює з кількома картами ознак з різними розмірами (наприклад, 8×8 та 4×4 на рис. 12). Для кожної комірки на кожній карті ознак генерується невелика кількість (наприклад, 4) початкових рамок з різними співвідношеннями сторін, центр яких знаходиться в даній комірці. Початкові рамки схожі на якорі, що використовувалися в Faster R-CNN, але на відміну від якорів, початкові рамки застосовуються до кількох карт ознак з різними розмірами, що дозволяє краще дискретно відтворити простір можливих рамок. Для кожної початкової рамки SSD передбачає відхилення рамки та ймовірності для кожного з класів. Наприклад, на рис. 12 передбачення для кота і для собаки були згенеровані з початкових рамок, що знаходились в картах ознак різних розмірів.

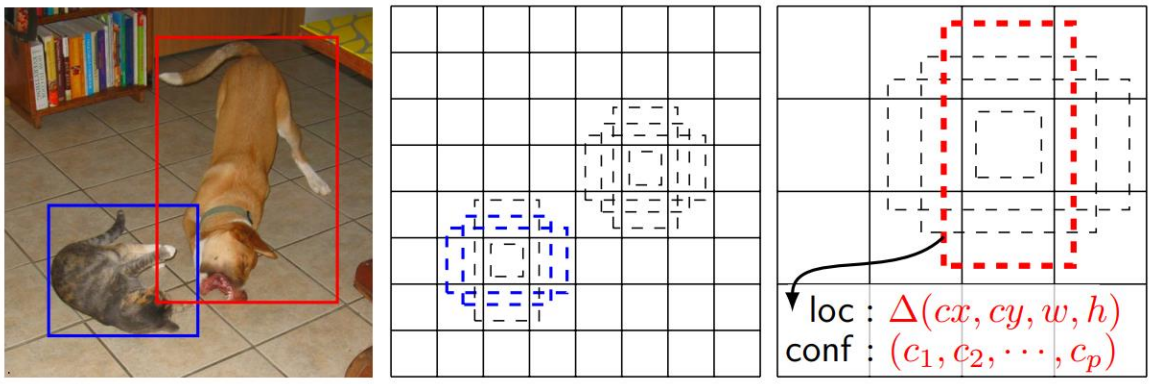

Рис. 12: Зображення з передбаченими обмежувальними рамками та дві карти ознак різних розмірів з початковими рамками, з яких утворились передбачення [12].

SSD базується на згортковій нейронній мережі, яка генерує фіксовану кількість обмежувальних рамок та класових ймовірностей для цих рамок, за якою слідує крок пригнічення немаксимумів, щоб отримати фінальні результати. Початкові рівні мережі базуються на стандартній архітектурі для класифікації зображень (але її обрізають, щоб прибрати рівні класифікації), яка називається базовою мережею. У оригінальній реалізації SSD використовується мережа VGG-16 як базова, але можливе використання інших мереж [12]. Після обрізаної базової мережі слідують згорткові рівні. З кожним рівнем розмір вихідних карт ознак зменшується, щоб забезпечити передбачення різних розмірів. Кожен з доданих рівнів (або навіть рівень всередині базової мережі) може згенерувати фіксовану кількість передбачень, використовуючи набір згорткових фільтрів. Для карти ознак розмірами $m \times n$ з p каналами ядро фільтра матиме розмір $3 \times 3 \times p$. Фільтр може передбачати класову ймовірність або відхилення рамки.

Під час тренування необхідно визначити які з початкових рамок відповідають цільовим рамкам, щоб відповідно до цього натренувати мережу. Це називається кроком встановлення відповідностей. Спочатку, кожній цільовій рамці ставиться у відповідність початкова рамка з найбільшим IoU. Після цього, початкові рамки ставляться у відповідність тим цільовим рамкам, з якими у них IoU дорівнює щонайменше $0{,}5$. Схожий підхід застосовується до якорів у Faster R-CNN, але з більшим порогом – $0{,}7$. Менший поріг дозволяє мережі створити більше передбачень з початкових рамок, що перетинаються між собою, і обрати найкраще тільки після отримання оцінки, що покращує точність.

Ті початкові рамки, які відповідають хоча б одній цільовій рамці, називають позитивними, інші – негативними. Після кроку встановлення відповідностей, більшість початкових рамок стають негативними, особливо коли початкових рамок багато. Це створює суттєвий дисбаланс між позитивними і негативними тренувальними прикладами. Тому замість того,

щоб використовувати усі негативні приклади, їх сортують за спаданням значення функції втрат для класифікації і беруть таку кількість з початку списку, щоб співвідношення між негативними та позитивними прикладами було не більше ніж 3:1. Результатом цього є більш швидке та стабільне тренування.

Функція втрат, яка використовується для тренування SSD, є зваженою сумою втрат локалізації та класифікації.

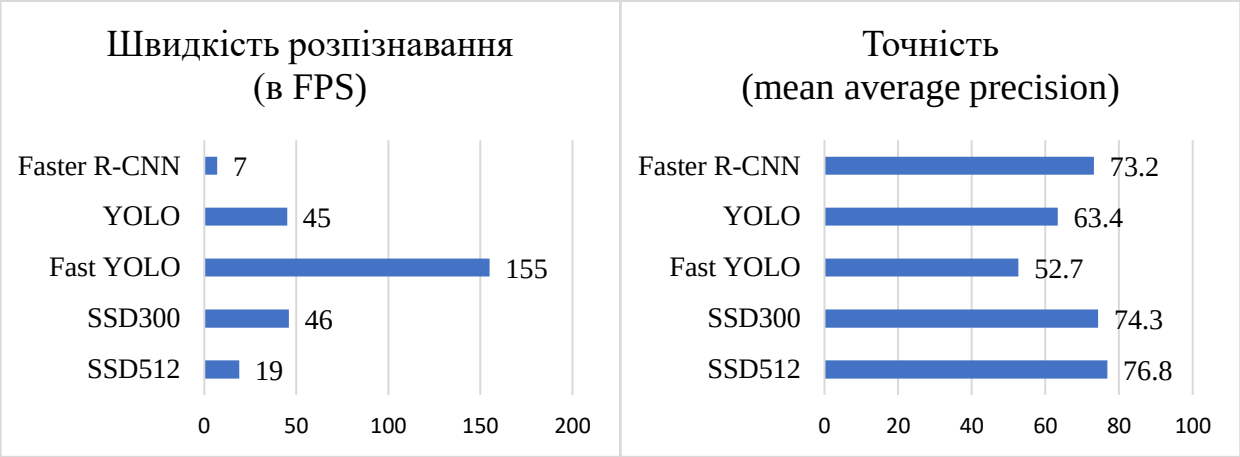

Рис. 13: Порівняння швидкості (FPS) та точності (mAP) детекторів Faster R-CNN, YOLO та SSD [9, 10, 12]. SSD300 та SSD512 – SSD, який працює із зображеннями розмірами 300 × 300 та 512 × 512 відповідно.

Як ми бачимо, SSD300 досягає трохи кращої точності, ніж Faster R-CNN, при цьому не поступаючись у швидкості YOLO. Зі швидкістю 46 FPS SSD300 можна застосовувати для розпізнавання в реальному часі, отримуючи при цьому точність більше 70%. Такої характеристики не має жоден з вищезгаданих детекторів. При роботі з більшим розміром зображення, SSD512 досягає найкращої точності з усіх перелічених методів, маючи при цьому швидкість близьку до швидкості реального часу. Найшвидшим є Fast YOLO, випереджуючи інші детектори як мінімум у кілька разів, але при цьому він має найгіршу точність – ледве вище 50%. Таким чином, з перелічених методів, SSD300 має найкращий баланс між швидкістю і точністю. Якщо швидкість

реального часу необов’язкова, то можна отримати невеликий приріст у точності за допомогою SSD512.

1.7 R-FCN

У Faster R-CNN моделі для кожного зображення зазвичай існує багато регіонів, які перетинаються між собою. Після RPN рівня, ознаки для класифікатора рамки обраховуються окремо для кожного регіону. Це приводить до зайвих обрахунків, якщо регіони перетинаються. Метою R-FCN (Region-based Fully-Convolutional Network) [13] є попередній обрахунок ознак для класифікатора рамок на всьому зображенні. Тоді для створення передбачень для кожного з регіонів, достатньо розглянути відповідні шматки попередньо обрахованих ознак.

У моделі R-FCN повноз’єднані рівні замінені на згорткові рівні. Останній згортковий рівень генерує набір k^2 чутливих до позиції карт оцінок для кожного з класів, отже на виході отримується $k^2(C+1)$ каналів, де C – кількість класів (+ 1 для фону). Набір k^2 карт оцінок відповідає $k \times k$ решітці, яка описує відносні позиції. Наприклад, при $k = 3$, 9 карт оцінок описують такі позиції: верхня ліва комірка, верхня середня комірка, верхня права комірка, …, нижня середня комірка, нижня права комірка.

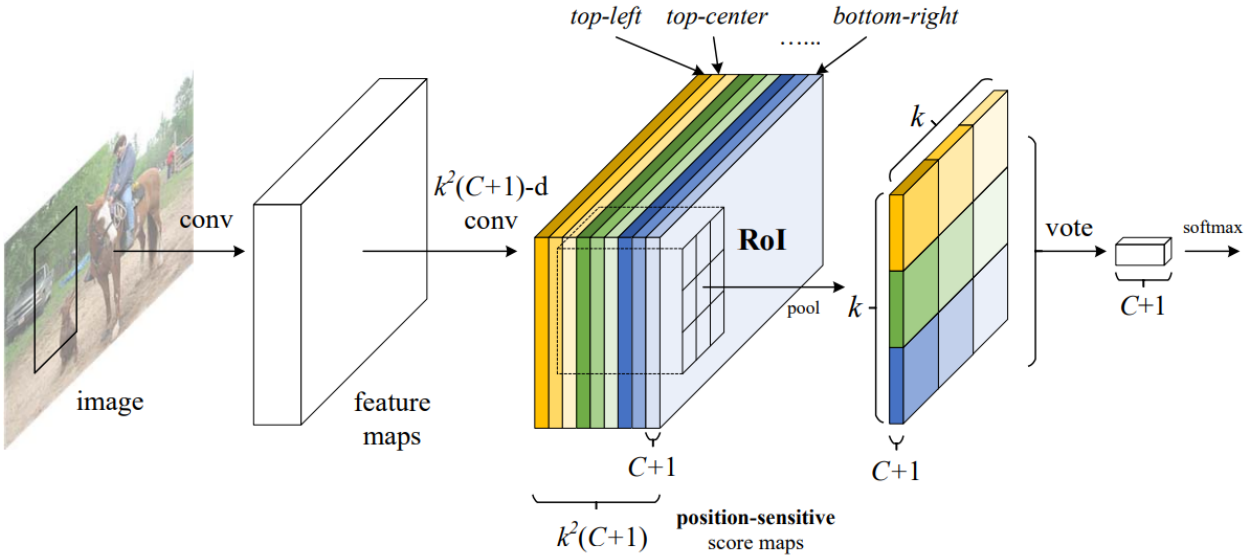

Рис. 14: Ілюстрація основної ідеї роботи R-FCN, де k = 3 [13].

Після отримання чутливих до позиції карт ознак кожен регіон розділяється на комірки $k \times k$ решіткою. Наприклад, якщо розмір зображення $w \times h$, то кожна комірка матиме розмір $\approx \frac{w}{k} \times \frac{h}{k}$. Всередині кожної з комірок виконується усереднювальне агрегування (average pooling) на відповідній карті оцінок. Отримується k^2 чутливих до позиції оцінок. Наприклад, щоб отримати значення верхньої лівої оцінки, береться карта ознак для верхньої лівої позиції, з неї вирізається комірка, що за розмірами та позицією відповідає верхній лівій комірці регіону, та до вирізаної частини застосовується усереднювальне агрегування.

Після цього, до k^2 чутливих до позиції оцінок застосовується голосування, в результаті якого отримується вектор розміром $C + 1$ для кожного регіону. За замовчуванням голосування представляє собою усереднення оцінок. Далі застосовується softmax, щоб отримати класові ймовірності, які використовуватимуться для обрахунку перехресної ентропії під час тренування та для вибору результуючого класу під час розпізнавання.

Така ж стратегія застосовується для регресії обмежувальної рамки. Після останнього згорткового рівня, модель розгалужується на два сусідні рівні: вищеописаний рівень класифікації та рівень регресії обмежувальної рамки. Останній теж використовує набір чутливих до позиції карт оцінок, але їх кількість $4k^2$. Таким чином, для кожного регіону отримується вектор розміру $4k^2$, до якого застосовується голосування, щоб отримати вектор розміру 4, який позначає обмежувальну рамку. Заради простоти, у даному методі, на відміну від інших, передбачається тільки одна рамка для кожного регіону, тобто одна рамка застосовується для кожного з класів, хоча передбачення окремої рамки для кожного класу також можливе [13].

Основною перевагою методу R-FCN є те, що кількість пропозицій регіонів майже не впливає на його швидкість, оскільки для кожного регіону використовуються попередньо обраховані карти оцінок і до них

застосовуються операції агрегування та голосування, які є майже безкоштовними з точки зору часу. Це означає, що можна збільшувати кількість пропозицій регіонів (для покращення точності) без погіршення швидкості, чого не вийде зробити, наприклад, з Faster R-CNN.

1.8 Компроміс між швидкістю та точністю

Як було показано вище, існує декілька методів розпізнавання об’єктів, які забезпечують високу точність та/або швидкість обробки. Втім, неможливо однозначно визначити який із них найкращий, оскільки це залежить від специфіки предметної області, в якій буде застосовуватися розпізнавання об’єктів. Наприклад, для автономних автомобілів необхідне розпізнавання в реальному часі з мінімальною затримкою, тоді як в галузі медицини точність може бути набагато важливішою за швидкість.

Розуміючи пріоритети предметної області обрати найоптимальніший метод має бути нескладно. Але є інша проблема. Хоча автори праць про методи розпізнавання об’єктів і описують результати експериментів та зазначають точність і швидкість методу, складно точно порівняти результати експериментів з різних праць, оскільки вони могли проводитись в різних умовах, на різних машинах та датасетах. До того ж, автори праць можуть удаватися до деяких хитрощів, щоб продемонструвати переваги свого методу, наприклад використання зображень іншого розміру, щоб підвищити точність або швидкість, або зміна інших параметрів мереж.

Щоб вирішити цю проблему Google Research організував експеримент [14], метою якого було протестувати кілька методів розпізнавання об’єктів в однакових умовах. В експерименті використовувалося 3 архітектури: Faster R-CNN, SSD та R-FCN. Кожна праця, присвячена певному методу, зазвичай пропонує використання конкретного генератора ознак (feature extractor), конкретного розміру зображень та конкретної кількості пропозицій регіонів (для методів, які їх використовують).

Ці параметри також впливають на результати експериментів. Тому в [14] кожна з трьох архітектур була реалізована та протестована з різними комбінаціями цих параметрів. Тестування усіх комбінацій відбувалось на одному датасеті – Microsoft COCO (Common Objects in Context) [15]. Результат було продемонстровано у вигляді графіка.

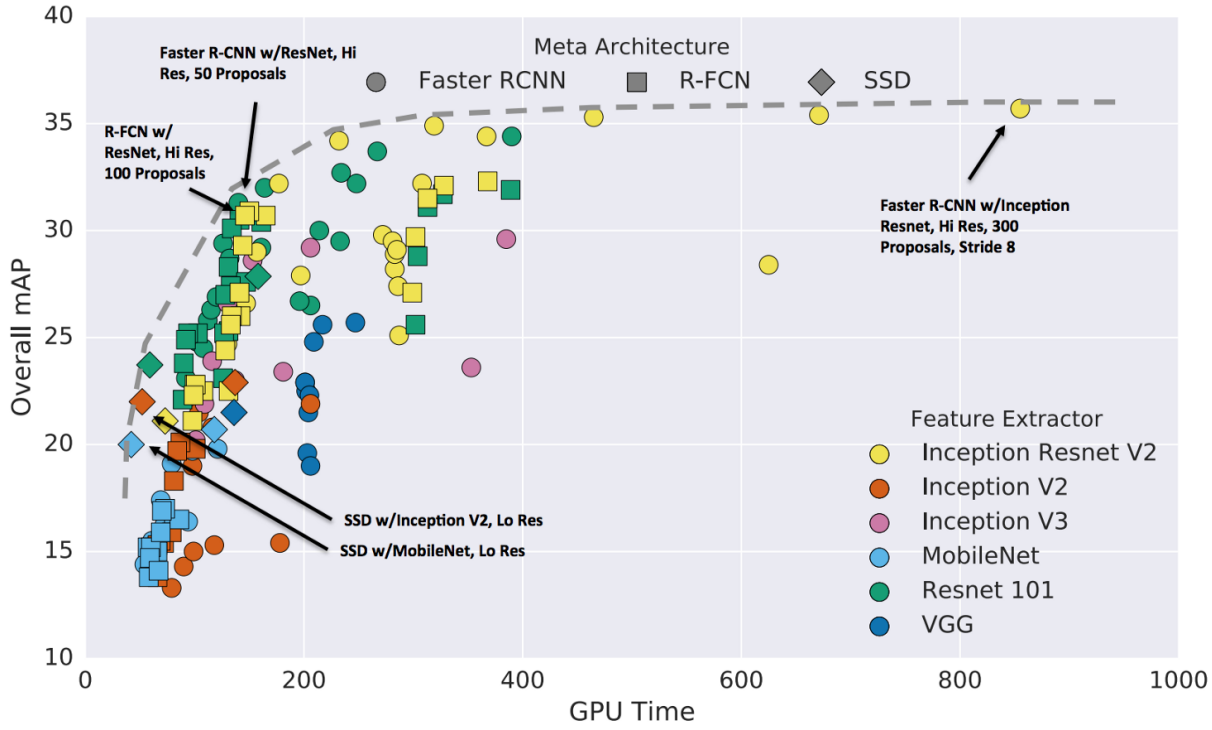

Рис. 15: Швидкість vs точність. Форма точки на графіку позначає архітектуру, а колір - генератор ознак. Кожна пара архітектури та генератору ознак може бути представлена кількома точками через різницю в розмірі вхідних зображень, кількості регіонів і т.д. [14]

На рис. 15 автори [14] визначають кілька цікавих точок: найшвидші, найточніші та найоптимальніші з точки зору швидкості та точності:

– Найшвидшими (а точніше найточнішими з найшвидших) є дві комбінації: SSD з Inception V2 та SSD з MobileNet, обидва з низькою роздільною здатністю вхідного зображення.
– Найточнішою комбінацією є Faster R-CNN з Inception Resnet, високою роздільною здатністю вхідних зображень та з 300 пропозиціями регіонів.

– Найоптимальнішими (такими, що покращення точності неможливе без погіршення швидкості і навпаки) є дві комбінації: R-FCN з Resnet та 100 пропозиціями регіонів і Faster R-CNN з Resnet та 50 пропозиціями, обидва з високою роздільною здатністю вхідних зображень.

Також з рис 15. видно, що загалом найкращі точності досягаються тільки за допомогою Faster R-CNN, а з точки зору швидкості (при не найгіршій точності) лідером є SSD. Вибір генератора ознак сильно впливає на точність та швидкість Faster R-CNN і R-FCN, але майже не впливає на SSD.

Іншим важливим спостереженням є те, що роздільна здатність вхідного зображення має велике значення. Зменшення розмірів зображення вдвічі по ширині та висоті в середньому погіршує точність на 15,88%, але також в середньому покращує швидкість на 27,4% [14].

2 Вирішення задачі розпізнавання пішоходів на зображеннях

2.1 Опис задачі

Задача розпізнавання пішоходів на зображеннях полягає у знаходженні прямокутних рамок, кожна з яких позначатиме межі одного пішохода. З кожною з рамок також асоціюється число – відсоток впевненості в тому, що об’єкт всередині рамки є пішоходом.

Оскільки у даній роботі ця задача розглядається в контексті застосування для самокерованих машин, існують певні умови та обмеження до алгоритму. Щоб уникнути аварійних ситуацій, система керування машиною повинна мати змогу вчасно «зрозуміти» поточну ситуацію на дорозі та зреагувати на неї. Час реакції системи визначається двома факторами: кадрова частота та затримка. Кадрова частота визначає наскільки часто сенсори можуть подавати дані на вхід процесору (наприклад, скільки кадрів в секунду може передавати камера, щоб процесор встиг їх обробити). Затримка визначає наскільки швидко система реагує на дані від сенсора.

Водії реагують на події на дорозі з різною швидкістю в залежності від того, чи очікують вони дану подію та від дії яку потрібно виконати, але найменший можливий час реакції водія – 100-150 мілісекунд [16]. Щоб забезпечити кращу безпеку, самокеровані машини мають реагувати швидше за водіїв, тобто затримка має бути не більше 100 мілісекунд. Крім того, самокеровані машини мають часто оновлювати своє представлення про дорожні умови. Кадрова частота має бути високою, на випадок якщо дорожні умови кардинально змінились на поточному кадрі порівняно з попереднім. Реакція самокерованої машини на зміну дорожніх умов також має бути швидшою за реакцію водія, отже оновлення представлення має відбуватись хоча б кожні 100 мілісекунд, що означає кадрова частота має бути не менше 10 FPS [16]. Таким чином, розпізнавання об’єктів на зображенні має

відбуватись швидше, ніж за 100 мілісекунд, щоб залишити якомога більше часу планувальним та контрольним модулям для реакції на ці об’єкти [17].

2.2 Вибір методу

У якості інструменту для вирішення задачі було обрано Tensorflow. Tensorflow – відкрита програмна бібліотека для машинного навчання, розроблена Google. Було обрано саме Tensorflow, тому що на його базі створено колекцію найуспішніших моделей для розпізнавання об’єктів, які вже попередньо натреновані на великому датасеті (більшість на COCO). Ці моделі можна одразу використовувати для розпізнавання об’єктів, якщо цільові класи присутні в датасеті, на якому була натренована модель. Але найбільшою перевагою є те, що можна дотренувати наявну модель для своїх цілей, використовуючи трансферне навчання.

Трансферне навчання дозволяє використати досвід накопичений при вирішенні однієї задачі для вирішення іншої схожої задачі. Глибокі нейронні часто мають мільйони параметрів (вагів), і потребують величезну кількість даних та ресурсів для успішного тренування з нуля. Це може бути проблемою, оскільки не завжди є можливість зібрати достатню кількість даних для потрібної предметної області або забезпечити потрібні ресурси. Трансферне навчання вирішує цю проблему: можна спочатку натренувати модель на великому загальному датасеті (наприклад COCO, який має 330 000 зображень і 80 класів об’єктів), а потім дотренувати на більш специфічному, можливо меншому, датасеті. Оскільки модель навчатиметься не з нуля, можна значно скоротити час та ресурси для тренування.

Існують різні стратегії, за допомогою яких можна дотренувати модель. В даній роботі, моделі будуть тренуватися від початку до кінця як зазвичай, але ваги будуть ініціалізовані кінцевими вагами попередньо натренованої моделі. Ця стратегія називається тонким налаштуванням (fine tuning).

Оскільки для самокерованих машин дуже важлива швидкість, було вирішено спробувати застосувати метод SSDLite з генератором ознак MobileNet V2 [18] для розпізнавання пішоходів. Цей метод є одним з найшвидших серед наявних на момент написання роботи (усі швидкості вимірювалися на датасеті COCO). Хоча й існують два методи, які обробляють зображення на 1 мілісекунду швидше за SSDLite з MobileNet V2, вони мають точності нижче на 2% і 4%. Обраний метод завдячує своїй швидкості генератору ознак MobileNet V2, який оптимізовано під мобільні девайси з обмеженими ресурсами, а також оптимізованому SSD під назвою SSDLite. SSDLite оптимізовано завдяки заміні звичайних згорток на відокремлювані згортки: спочатку поглибинна (depthwise), а потім поточкова (pointwise) згортка [18]. Поглибинна згортка не змінює глибину вхідних даних і виконується за допомогою фільтрів з глибиною ядра 1. Наприклад, якщо вхідні дані мають глибину d, буде застосовано d фільтрів з глибиною 1: по одному фільтру на кожен рівень даних. Поточкова згортка використовує фільтри з розмірами 1×1 та глибиною, що дорівнює глибині вхідних даних. Результат відокремлюваної згортки такий же, як і звичайної, але використовує значно менше операцій множення, відповідно відокремлювана згортка швидше. Недоліком відокремлюваної згортки є те, що вона зменшує кількість параметрів мережі, і якщо у мережі і так небагато параметрів, це може заважати навчанню моделі. Але це не є проблемою для SSD.

Оскільки найшвидші методи поступаються у точності більш повільним, було вирішено також спробувати застосувати один із найоптимальніших методів і порівняти результати з найшвидшим. Під найоптимальнішими методами маються на увазі такі, що покращення точності неможливе без погіршення швидкості і навпаки. Можна виділити два таких методи: R-FCN з Resnet101 і Faster R-CNN з Resnet50 (число після Resnet вказує на кількість рівнів мережі). На датасеті COCO вони мають однакову точність, але Faster R-CNN швидше на 3 мілісекунди, тому було вирішено застосувати його.

Зазвичай Faster R-CNN не розглядається як метод, який може підійти для застосування в реальному часі, але при зменшенні кількості рівнів мережі Resnet до 50 замість 101 Faster R-CNN працює значно швидше, зберігаючи непогану точність.

2.3 Опис датасету

Першим кроком у навчанні нейронної мережі є збір даних. Для даної задачі потрібні зображення зняті камерою, встановленою на машині, яка рухається дорогами. Оскільки моделі, які будуть використовуватися, покладаються на навчання з учителем, також потрібно, щоб зображення були розмічені: тобто щоб для кожного пішохода на зображеннях були координати обмежувальної рамки.

Цим умовам відповідає ETH Pedestrian Dataset [19]. Датасет містить більше 5 500 зображень та більше 45 000 анотованих пішоходів. Частина зображень були зняті за допомогою камери встановленої на машині, що рухається міськими дорогами, інша частина – за допомогою камери встановленої на дитячій колясці, яка рухається тротуарами. Зображення мають розмір 640×480 та були зняті з частотою 13-14 кадрів на секунду.

ETH надає зображення датасету у форматі .png, а анотації – у форматі .idl. У таких форматах зручно переглядати дані, але це не найефективніший спосіб читання даних для моделі. Tensorflow має власний бінарний файловий формат – TFRecord, який може містити в собі і зображення, і анотації до них. Бінарні файли займають менше місця на диску, а також швидше завантажуються в пам’ять. До того ж, TFRecord формат оптимізовано для використання разом з Tensorflow. Завдяки цьому, використання даного файлового формату може значно покращити ефективність тренування моделі, і саме його рекомендовано використовувати для зберігання датасетів для Tensorflow.

Існують скрипти, які генерують TFRecord файли на основі даних у кількох популярних форматах (наприклад, у форматі, який використовує COCO датасет або Pascal VOC датасет). На жаль, формат, який використовує ETH датасет, відрізняється від форматів найпопулярніших датасетів, тому скрипт для генерації TFRecord файлу потрібно було написати самостійно.

2.4 Навчання моделі SSDLite з MobileNet V2

Алгоритм SSD генерує початкові рамки за набором розмірів та співвідношень сторін. У реалізації за замовчуванням визначено 5 значень співвідношень сторін, які обрано таким чином, щоб відобразити співвідношення сторін більшості об’єктів. Під час тренування розміри та координати початкових рамок можуть змінюватися, щоб краще підійти під об’єкт, але чим ближче початкова рамка до цільової, тим стабільніше та швидше відбуватиметься тренування. Оскільки в даній роботі потрібно розпізнати тільки один клас об’єктів – пішоходів, було вирішено змінити співвідношення значень сторін встановлені за замовчуванням, щоб початкові рамки були близькими до можливих цільових рамок. В середньому, обмежувальна рамка для пішохода матиме співвідношення сторін 0,41 [20]. Але було б неправильно визначити тільки одне співвідношення сторін, оскільки пішохід може необов’язково відповідати цьому середньому значенню або навіть може знаходитись не у вертикальному положенні. Тому було визначено такі значення співвідношень сторін: 0,4; 0,5; 0,7; 1; 2. На практиці такі значення співвідношень сторін дійсно забезпечили кращу середню точність, ніж значення за замовчуванням.

Крім цього, за замовчуванням в алгоритмі SSD розмір усіх вхідних зображень змінюється до 300×300. Щоб уникнути викривлення або обрізання зображення, було зроблено так, щоб розмір усіх вхідних зображень змінювався до 640×480 (що відповідає розміру зображень з датасету).

Тренування мережі з такими налаштуваннями зайняло 5 годин (оскільки використовувалось трансферне навчання, тренування не зайняло багато часу). Час розпізнавання для одного зображення – 39 мілісекунд (25 FPS), середня точність (mean average precision, mAP [21]) – 84%. Ця швидкість вкладається у ліміт часу реакції самокерованої машини (100 мілісекунд). Але у даному випадку лише розпізнавання пішоходів займатиме майже половину виділеного часу реакції, а хотілося б також надати більше часу іншим процесам. До того ж, отримана швидкість повільніша за ту, яка досягається на COCO датасеті з реалізацією за замовчуванням, що підказує, що її можна покращити.

Причиною гіршої швидкості є те, що модель приймала вхід зображення більшого розміру: 640×480 замість 300×300 в реалізації за замовчуванням. Розмір зображення суттєво впливає на швидкість алгоритмів розпізнавання об’єктів, але також впливає на середню точність розпізнавання. Отже, зі зменшенням розміру зображення для досягнення кращої швидкості погіршиться середня точність.

У другому варіанті моделі будемо змінювати розмір зображень до 320×240 – половина від попереднього розміру по ширині і висоті. Оскільки вхідні зображення меншого розміру, можна також збільшити розмір партії. Тренувальні приклади оброблюються партіями для швидшого тренування. Для минулої моделі розмір партії був 16, для цієї спробуємо 28. Завдяки цьому вдалось зменшити час тренування, тепер воно займає менше 4 годин.

Час розпізнавання на одному зображенні – 14 мілісекунд (69 FPS), а середня точність – 72%. Отже, завдяки зменшенню розміру зображення вдвічі по ширині та висоті вдалось зменшити час розпізнавання в 2,8 разів, але при цьому дійсно погіршилась середня точність – на 12%.

Втім, одного значення середньої точності недостатньо для того, щоб повноцінно оцінити якість роботи моделі. Основною метою розпізнавання

пішоходів для самокерованих машин є не пропустити пішохода. Отже, важливо мінімізувати кількість випадків, коли на зображенні є пішоход, якого не розпізнала модель. Будемо вважати, що модель розпізнала пішохода якщо для відповідної цільової рамки існує фінальна рамка, з якою у неї $IoU \geq 0{,}5$.

У такому випадку можна оцінити модель за допомогою метрик точності та повноти. Точність покаже яка частка розпізнаних моделлю пішоходів дійсно є пішоходами, а повнота покаже яку частку від загальної кількості пішоходів було виявлено моделлю. Середня точність, яку ми використовували до цього, показує площу під кривою точності та повноти [21], отже можна сказати, що вона включає в себе обидві метрики. Точність та повнота взаємозалежні, і при збільшенні одного значення зменшиться інше. Для певних задач обидві метрики однаково важливі, але у випадку розпізнавання пішоходів важливішою є повнота. Чим більше повнота, тим менше пішоходів не помічає модель. У даному випадку можна трохи пожертвувати точністю заради збільшення повноти, оскільки краще зреагувати на пішохода, якого насправді не існує, ніж не зреагувати на існуючого.

В останній версії моделі маємо точність 97%, а повноту – 76%. Отже точність дуже хороша: лише у 3% випадків модель розпізнає щось, що не є пішоходом. Але повнота не є задовільною: 24% пішоходів модель просто не бачить. Кожна обмежувальна рамка згенерована моделлю супроводжується відсотком впевненості, який показує наскільки модель впевнена в тому, що об’єкт всередині рамки є пішоходом і в тому, що рамка є точною. Оскільки відсоток впевненості може бути дуже низьким, перед підрахунком значень вище було відфільтровано усі рамки з впевненістю нижче 50%. Якщо понизити цю межу, то можна очікувати, що кількість обмежувальних рамок зросте, і хоча б частина нових рамок буде окреслювати пішоходів, які раніше були проігноровані, відповідно повнота зросте (але інша частина рамок окреслюватиме не пішоходів, через що точність впаде). Але сильно понижувати цю межу не варто, оскільки тоді точність буде занадто низькою.

За допомогою експериментів було виявлено, що встановлення межі на 25% допомагає досягнути балансу між точністю та повнотою: точність – 87%, повнота – 85%. Таким чином, 15% пішоходів залишаються нерозпізнаними. Судячи з тестових прикладів, більшість нерозпізнаних пішоходів можна розділити на три категорії:

– перекриті іншими пішоходами чи об’єктами (тобто система все одно зреагує на об’єкти попереду);
– пішоходи в групах, наприклад, два пішоходи, які стоять поряд, можуть мати одну спільну обмежувальну рамку;
– пішоходи маленького розміру, які знаходяться далеко.

Рис. 16: Приклади розпізнавання за допомогою SSDLite з MobileNet V2 з відсотком впевненості від 50% (зліва) та від 25% (справа).

2.5 Навчання моделі Faster R-CNN з Resnet50

Модель Faster R-CNN покладається на передбачення пропозицій регіонів за допомогою RPN, і важливим параметром є кількість пропозицій. Зазвичай використовується 300, але для пришвидшення алгоритму можна

використовувати низьке значення пропозицій регіонів, наприклад 50. Низьке значення також понижує середню точність алгоритму. З кожної пропозиції можна отримати тільки одну обмежувальну рамку, відповідно кількість пропозицій також обмежує кількість виявлених об’єктів. Через це, у випадку розпізнавання пішоходів на дорозі, низьке значення пропозицій регіонів може бути недоліком не лише через погіршення точності, а й через те, що обмежує кількість пішоходів, які можуть бути розпізнані. Пішоходів на дорозі може бути більше кількох десятків, при цьому частина пропозицій регіонів може бути невдалою і не містити пішоходів, отже 50 є занадто низьким значенням кількості пропозицій регіонів. За допомогою експериментів будо виявлено, що 300 є оптимальним значенням.

На відміну від SSD, Faster R-CNN не потребує зображень фіксованого розміру, і може працювати з будь яким розміром. Втім, щоб уникнути занадто великих зображень (може нашкодити швидкості) або занадто маленьких (може нашкодити точності), накладаються обмеження на мінімальну та максимальну довжину сторін. За замовчуванням мінімальне значення 600, а максимальне – 1024. Це означає, що зображення з датасету розміром 640×480 будуть збільшені. Втім, зменшення мінімальної довжини сторони до 480 (щоб не змінювався розмір вхідних зображень) не призвело до покращення швидкості, тому було залишено значення за замовчуванням.

Розмір партії прикладів для тренування було встановлено на 1, тому що інакше не вистачало пам’яті. Через це тренування зайняло більше часу, ніж для SSDLite – більше 8 годин. Час розпізнавання на одному зображенні – 18 мілісекунд (55 FPS), а середня точність – 96%. Отже, порівняно з SSDLite, час розпізнавання більше всього на 4 мілісекунди, але при цьому середня точність збільшилась аж на 24%.

При встановленні межі впевненості на 50% отримуємо точність 92%, і повноту теж 92%. Отже, порівняно з SSD, точність нижче на 5%, але при цьому повнота, яка важливіша для нашої задачі, вище аж на 16%. Отже, 8%

розпізнаних об’єктів не є пішоходами, і 8% пішоходів не були розпізнані. Спробуємо зменшити кількість не розпізнаних пішоходів, понизивши межу впевненості. При пониженні межі до 30% досягається баланс: точність – 88%, повнота – 94%. Отже, вийшло скоротити кількість нерозпізнаних пішоходів на 2% ціною зменшення точності на 4%.

Аналогічно з SSDLite, нерозпізнані пішоходи зазвичай або перекриті іншими об’єктами, або знаходяться в групах, або маленького розміру. Рамки, які містять не пішоходів, містять зазвичай інші об’єкти (стовпи, дорожні знаки, сумки, частини пішоходів або машин), і набагато рідше – фон.

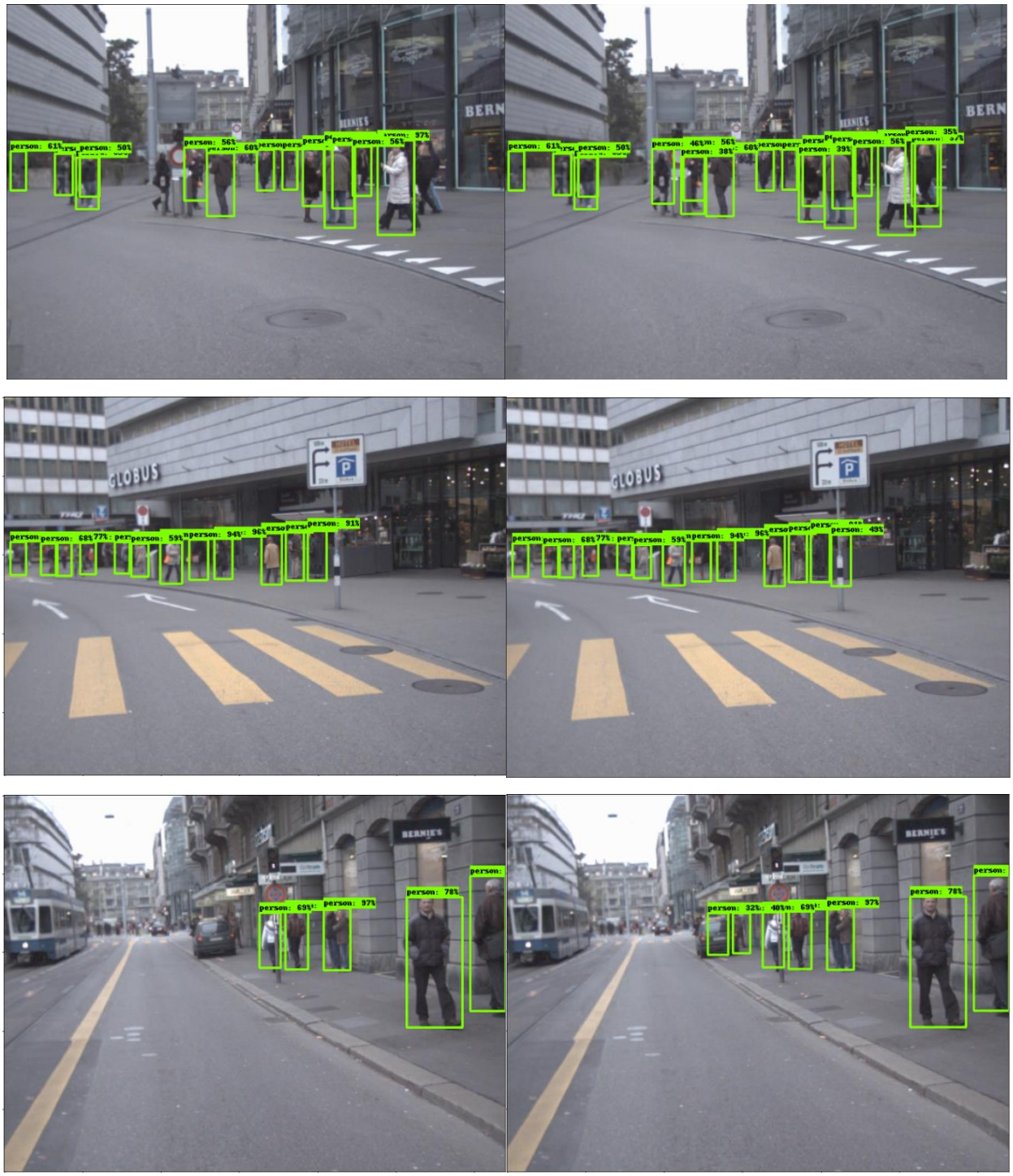

Рис. 17: Приклади розпізнавання за допомогою Faster R-CNN з Resnet50 з відсотком впевненості від 50% (зліва) та від 30% (справа).

2.6 Висновки

Для вирішення задачі розпізнавання пішоходів на зображеннях було вирішено спробувати застосувати два методи: SSDLite з MobileNet V2 та Faster R-CNN з Resnet50. Обидва методи забезпечили гарну швидкість

розпізнавання, яка задовольняє умови задачі. Час розпізнавання SSDLite трохи краще: 14 мілісекунд порівняно з 18 у Faster R-CNN. При цьому, якість розпізнавання Faster R-CNN набагато краще: точність – 88%, повнота – 94%, тоді як у SSDLite: точність – 87%, повнота – 85%. Можна помітити, що у Faster R-CNN не тільки краще і точність, і повнота, а й кращий баланс між ними. При майже однаковій точності, повнота Faster R-CNN набагато вище, ніж у SSDLite.

Таким чином, метод Faster R-CNN з Resnet50 краще підходить для вирішення задачі розпізнавання пішоходів на зображеннях.

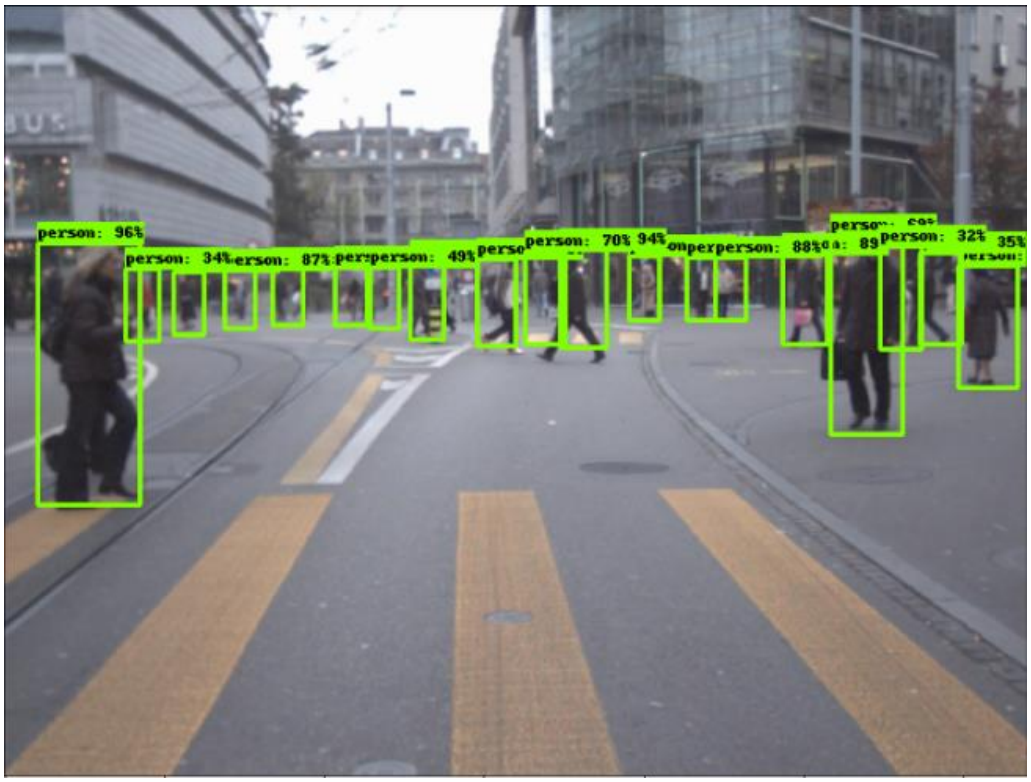

Рис. 18: Приклад розпізнавання пішоходів на тестовому зображенні за допомогою Faster R-CNN з Resnet50.

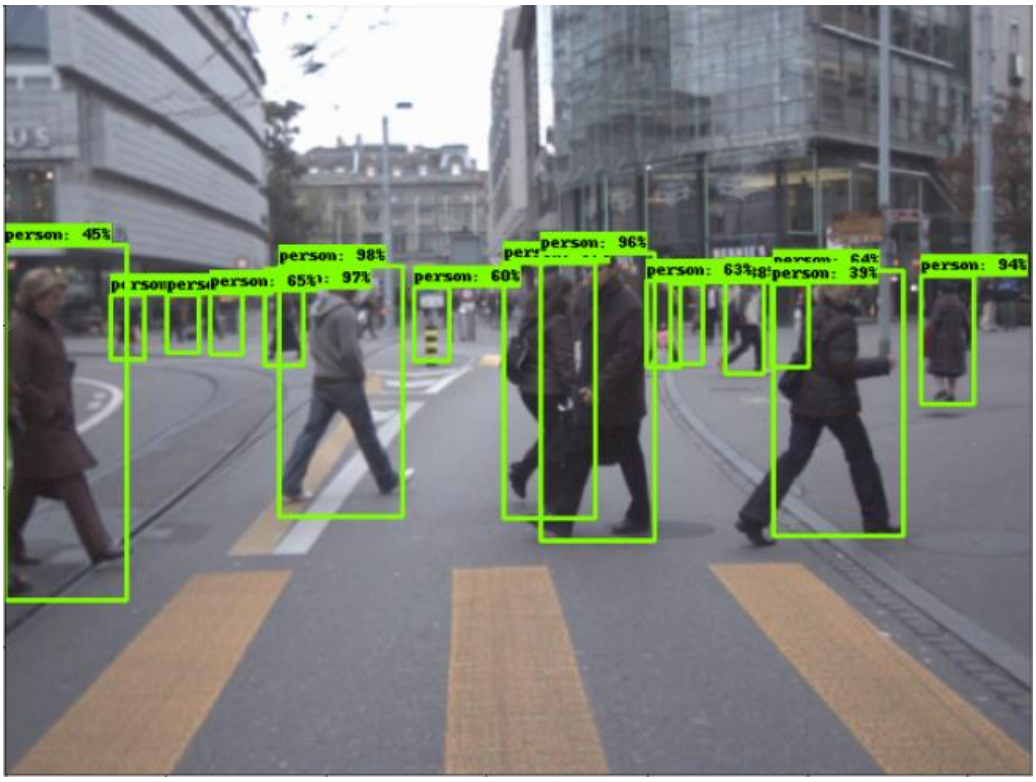

Рис. 19: Приклад розпізнавання пішоходів на тестовому зображенні за допомогою Faster R-CNN з Resnet50.

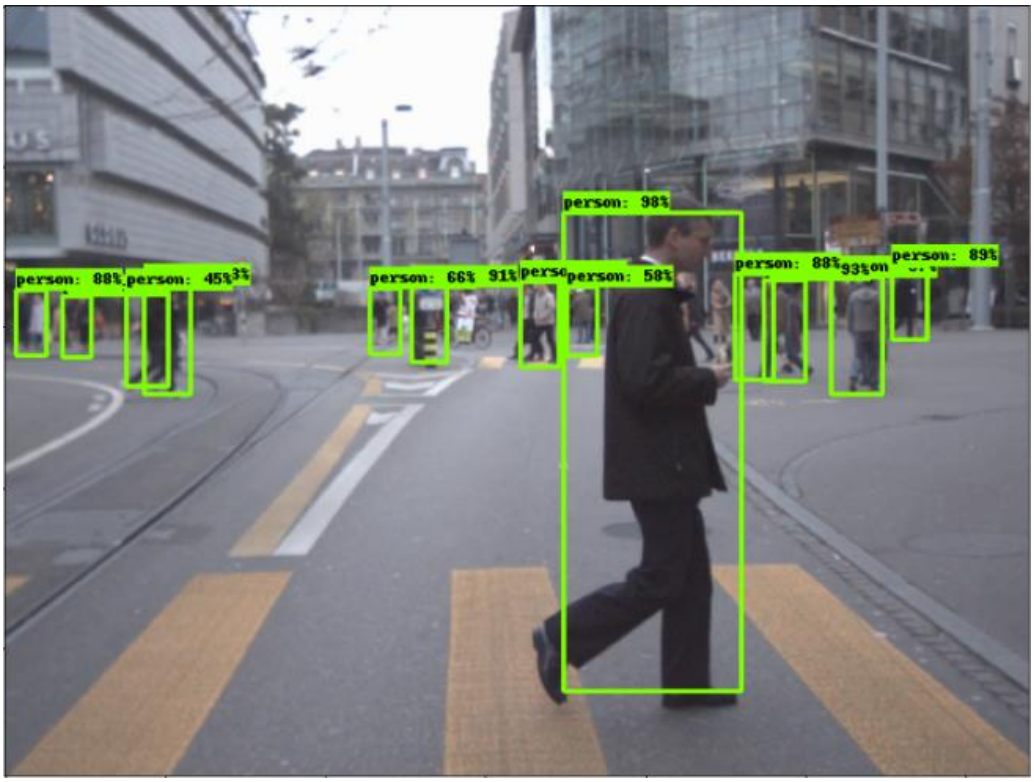

Рис. 20: Приклад розпізнавання пішоходів на тестовому зображенні за допомогою Faster R-CNN з Resnet50.

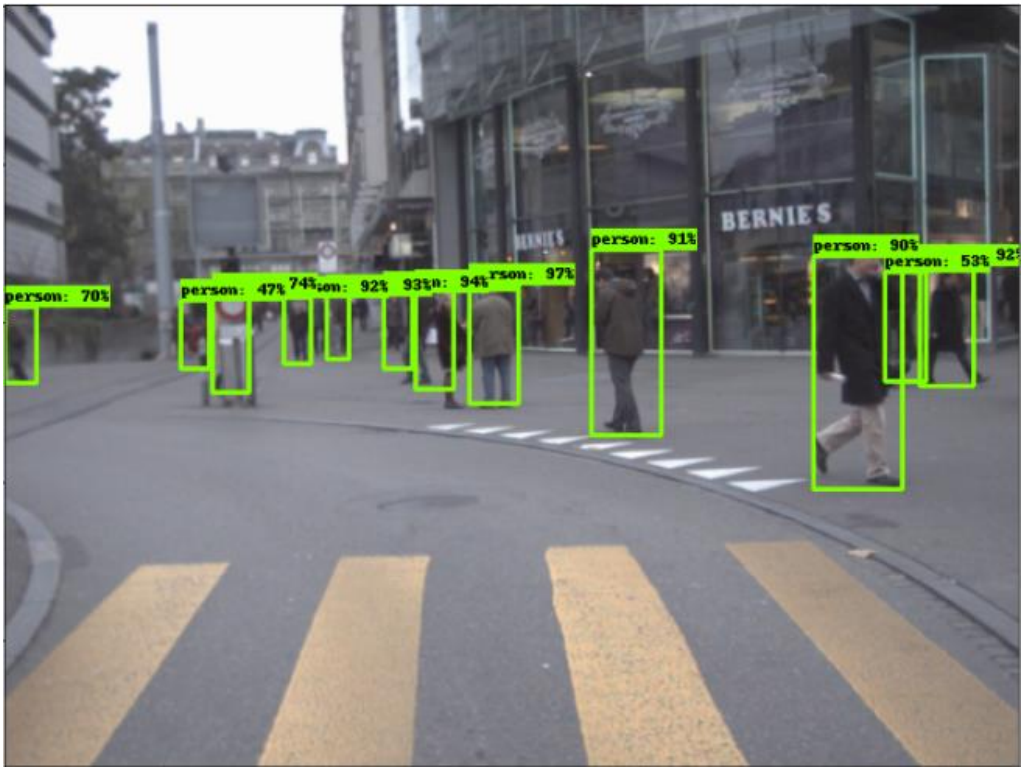

Рис. 21: Приклад розпізнавання пішоходів на тестовому зображенні за допомогою Faster R-CNN з Resnet50.

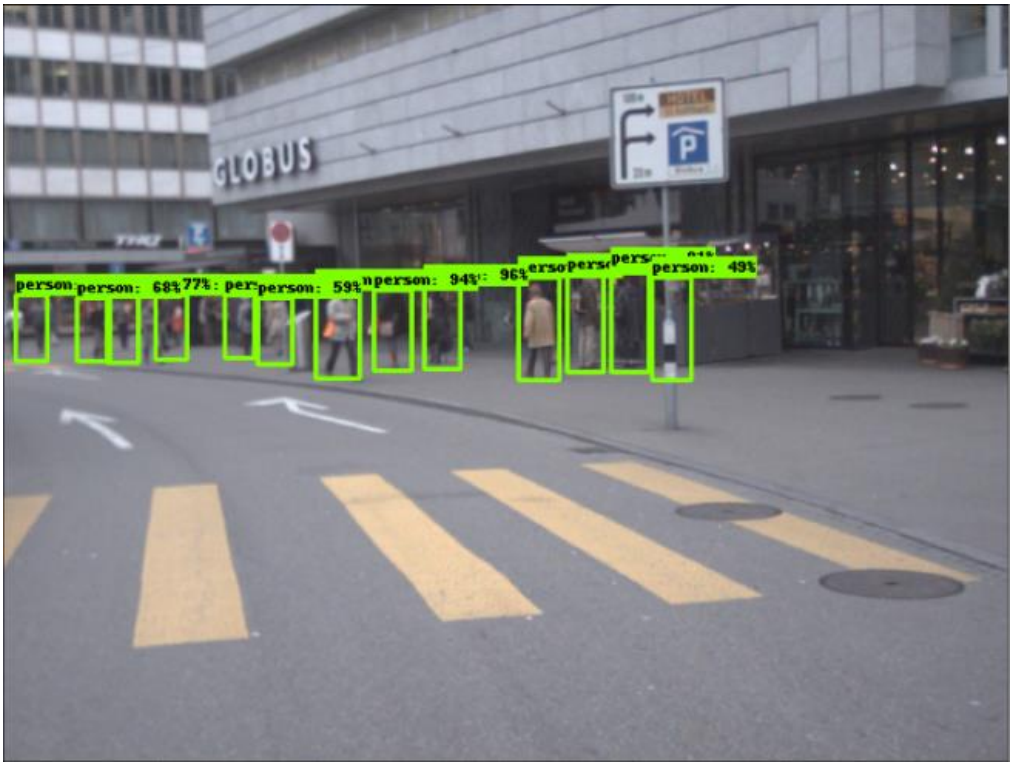

Рис. 22: Приклад розпізнавання пішоходів на тестовому зображенні за допомогою Faster R-CNN з Resnet50.

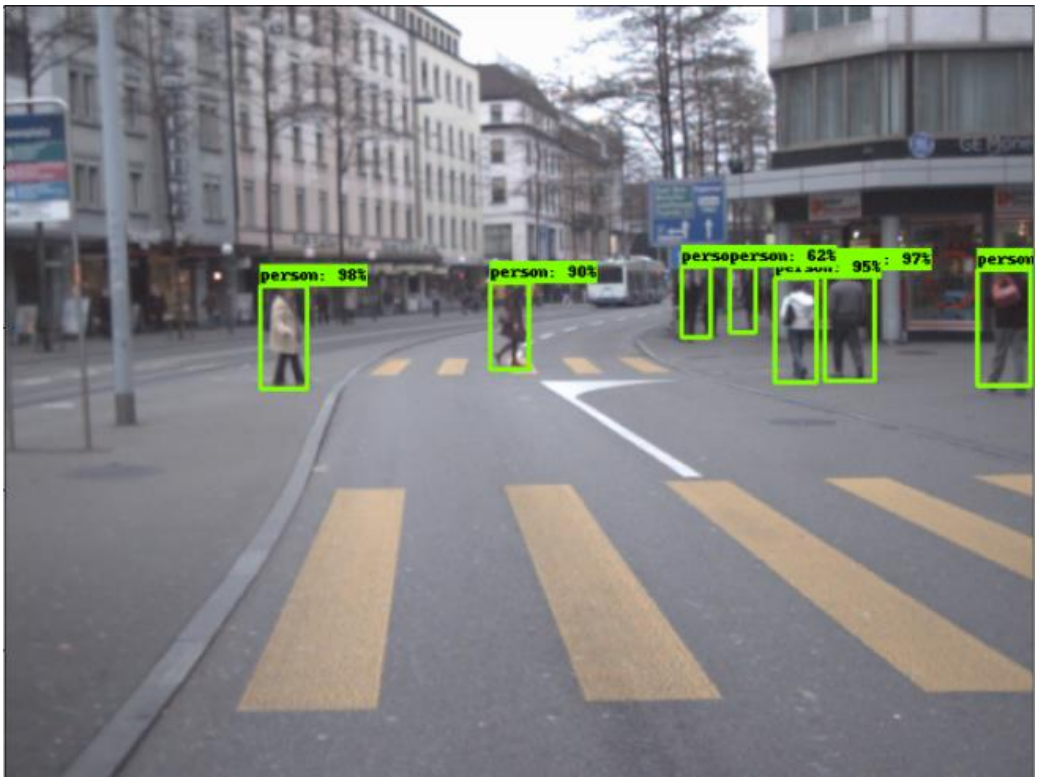

Рис. 23: Приклад розпізнавання пішоходів на тестовому зображенні за допомогою Faster R-CNN з Resnet50.

Висновки

Першим етапом даної роботи був огляд методів розпізнавання об’єктів на зображеннях. Розглянуті методи можна розділити на дві категорії: методи, що покладаються на пропозиції регіонів та однокрокові методи (яким не потрібні пропозиції регіонів). До першої категорії відносяться методи R-CNN, SPP-net, Fast R-CNN, Faster R-CNN та R-FCN. До другої категорії відносяться YOLO та SSD. В основному, методи, що покладаються на пропозиції регіонів, є точними, але повільними і рідко досягають швидкості реального часу, тоді як однокрокові методи дуже швидкі, але не можуть перевершити методи першої категорії за точністю.

На другому етапі роботи було проведено аналіз предметної області, визначені обмеження до алгоритму, обрано датасет та найкращі методи для вирішення задачі розпізнавання пішоходів для самокерованих автомобілів. Було визначено, що обробка одного зображення має відбуватись швидше, ніж за 100 мілісекунд, та мають оброблятися як мінімум 10 кадрів за секунду. Дані для тренування та тестування моделей було вирішено взяти з ETH Pedestrian Dataset. Зображення для датасету були зняті на міських дорогах та тротуарах, всього там 5 500 зображень, для яких анотовано більше 45 000 пішоходів. Для тренування моделей було вирішено використати бібліотеку Tensorflow, тому що вона має моделі, які попередньо натреновані на великих загальних датасетах, завдяки чому можна використати трансферне навчання для тренування власних моделей, витративши менше часу та ресурсів. Наостанок, було обрано два методи для вирішення задачі розпізнавання пішоходів: SSDLite з MobileNet V2 та Faster R-CNN з Resnet50. Перший є одним з найшвидших, тоді як другий досягає балансу між швидкістю та точністю.

На третьому етапі даної роботи було застосовано на практиці вищезгадані два методи для вирішення задачі розпізнавання пішоходів на зображеннях. Для кожного з методів було натреновано моделі з різними значеннями параметрів, щоб визначити найкращу комбінацію параметрів та

створити найкращу модель. В результаті, для кожного з методів було отримано значення точності, повноти та часу розпізнавання. За допомогою SSDLite з MobileNet V2 вдалося досягнути часу розпізнавання у 14 мілісекунд (можна обробляти 69 кадрів на секунду), точності – 87%, повноти – 85%. Для Faster R-CNN з Resnet50, час – 18 мілісекунд (55 кадрів на секунду), точність – 88%, повнота – 94%. Було зроблено висновок, що Faster R-CNN з Resnet50 є кращим методом вирішення поставленої задачі, оскільки від досягає кращих значень точності, повноти та кращого балансу між ними, при цьому маючи час розпізнавання всього на 4 мілісекунди повільніше, ніж SSDLite з MobileNet V2.

Результати:

1. Було проведено огляд методів розпізнавання об’єктів на зображеннях та порівняння їх характеристик.
2. Було створено дві моделі – SSDLite з генератором ознак MobileNet V2 та Faster R-CNN з Resnet50 – для вирішення задачі розпізнавання пішоходів на зображеннях для самокерованих автомобілів.
3. Було оцінено характеристики вищезгаданих моделей у контексті даної задачі та обрано найкращу з них.

Список використаної літератури

1. https://medium.com/analytics-vidhya/beginners-guide-to-object-detection-algorithms-6620fb31c375
2. J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers , A.W.M. Smeulders. Selective Search for Object Recognition. IJCV 2012. Режим доступу: https://koen.me/research/pub/uijlings-ijcv2013-draft.pdf
3. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. ECCV 2014. Режим доступу: https://arxiv.org/pdf/1406.4729.pdf
4. https://miro.medium.com/max/1910/1*n4LE9idyGJX_efOsS-FNvw.png
5. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
6. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
7. Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR 2014. Режим доступу: https://arxiv.org/pdf/1311.2524.pdf
8. Ross Girshick. Fast R-CNN. ICCV 2015. Режим доступу: https://arxiv.org/pdf/1504.08083.pdf
9. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS 2015. Режим доступу: https://arxiv.org/pdf/1506.01497.pdf
10. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. CVPR 2016. Режим доступу: https://arxiv.org/pdf/1506.02640v5.pdf
11. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf
12. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single-Shot MultiBox

Detector. ECCV 2016. Режим доступу: https://arxiv.org/pdf/1512.02325.pdf

13. Jifeng Dai, Yi Li, Kaiming He, Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. NIPS 2016. Режим доступу: https://arxiv.org/pdf/1605.06409.pdf
14. Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. CVPR 2017. Режим доступу: https://arxiv.org/pdf/1611.10012.pdf
15. http://cocodataset.org/
16. Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach Md E. Haque, Lingjia Tang, Jason Mars. The Architectural Implications of Autonomous Driving: Constraints and Acceleration. ASPLOS 2018. Режим доступу: https://web.eecs.umich.edu/~shihclin/papers/AutonomousCar-ASPLOS18.pdf
17. Ekim Yurtsever, Jacob Lambert, Alexander Carballo, Kazuya Takeda. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. T-IV 2019. Режим доступу: https://arxiv.org/pdf/1906.05113.pdf
18. Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. CVPR 2018. Режим доступу: https://arxiv.org/pdf/1801.04381.pdf
19. https://data.vision.ee.ethz.ch/cvl/aess/dataset/
20. https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06
21. https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173