НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи і методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки»

на тему: «Аналіз та ідентифікація автомобільного трафіку відносно транспортного засобу»

Виконав:
студент IV курсу, групи КА-75
Нікітін Владислав Олегович _____

Керівник:
Савастьянов В. В. _____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А. Є. _____

Консультант з економічного розділу:
доцент, к.е.н. Рощина Н. В. _____

Рецензент:
доцент, к.т.н Терентьєв О. М. _____

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

Київ – 2021 рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп'ютерні науки»
Освітня програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«26» травня 2021 р.

ЗАВДАННЯ
на дипломну роботу студенту
Нікітіну Владиславу Олеговичу

1. Тема роботи «Аналіз та ідентифікація автомобільного трафіку відносно транспортного засобу», керівник роботи Севастьянов Володимир, затверджені наказом по університету від «26» травня 2021 р. № 1344-с

2. Термін подання студентом роботи 8.06.2021.

3. Вихідні дані до роботи взяті з мережі інтернет та з приватних відеокамер водіїв з їхнього дозволу

4. Зміст роботи: основні поняття, задача автоматичного розпізнавання автомобільного номеру, теоретичні відомості, побудова інформаційної системи, функціонально-вартісний аналіз програмного продукту

5. Перелік ілюстративного матеріалу: слайди інформаційна система, згорткові нейронні мережі, результати

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В, канд.економ.наук, доцент		

7. Дата видачі завдання 11 лютого 2021 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	25.04.2021	
2.	Підготовка першого розділу.	30.04.2021	
3.	Підготовка другого розділу.	05.05.2021	
4.	Розробка програмного продукту.	24.05.2021	
5.	Підготовка третього розділу	25.05.2021	
6.	Підготовка економічної частини	28.05.2021	
7.	Оформлення розділів відповідно до нормоконтролю.		
8.	Підготовка презентації доповіді.	30.05.2021	
9.	Оформлення дипломної роботи.		

Студент Владислав Нікітін

Керівник Володимир Савастьянов

РЕФЕРАТ

Дипломна робота: 101 ст., 45 рис, 7 табл., 2 дод., 18 джерел.

ІДЕНТИФІКАЦІЯ ТРАНСПОРТНОГО ЗАСОБУ. ШТУЧНИЙ ІНТЕЛЕКТ. МАШИННЕ НАВЧАННЯ. АНАЛІЗ ТРАФІКУ

Об’єкт дослідження – автомобільний трафік, зафіксований за допомогою камери із відеореєстратора встановленого на транспортному засобі.

Предмет дослідження – методи штучного інтелекту, нейронні мережі та моделі з комбінацій нейронних мереж, які використовуються для ідентифікації фізичних об’єктів, зокрема транспортних засобів, що можуть однозначно ідентифікуватися за допомогою номерного знаку, та нейронні мережі та методи, що можуть бути використані для якомога більш точного визначення тексту номерного знаку.

Методи дослідження – методи штучного інтелекту, методи класифікації та регресії машинного навчання, нейронні мережі.

Актуальність – задача побудови моделі, що може потенціально мати широке використання для покращення багатьох сервісів пов’язаних з дорожнім рухом.

Результати роботи – створена модель, що однозначно ідентифікує транспортний номерний знак по кадру, якщо розширення зображення це дозволяє зробити людині.

Шляхи подальшого розвитку – збір більш різноманітних даних для покращення точності моделі, створення зручного для використання різними сервісами інтерфейсу, зменшення об’єму пам’яті, що займає модель.

ABSTRACT

Thesis: 101 pages, 44 figures, 7 tables, 2 appendices, 18 sources.

VEHICLE IDENTIFICATION. ARTIFICIAL INTELLIGENCE. MACHINE LEARNING. TRAFFIC ANALYSIS

The object of the study is car traffic recorded by a camera from a video recorder installed on the vehicle.

The subjects of research are artificial intelligence methods, neural networks and models of combinations of neural networks used to identify physical objects, including vehicles that can be uniquely identified by a license plate, neural networks and methods that can be used to more accurately define license plate text.

Research methods are methods of artificial intelligence, methods of classification and regression of machine learning, neural networks.

Relevance is the challenge of building a model that has the potential to be widely used to improve many traffic services

The results of the work - a model is created that uniquely identifies the transport license plate in the frame, if the expansion of the image allows a person to do so.

Ways of further development - collecting more diverse data to improve the accuracy of the model, creating a user-friendly interface with different services, reducing the amount of memory occupied by the model.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ANPR – Automatic Number Plate Recognition

mAP – mean Average Precision

FPS – Simultaneous localization and mapping

CNN – Convolutional Neural Network

R-CNN – Region Based Convolutional Neural Network

YOLO – You Only Look Once

IDE – Integrated Development Environment

ANN – Artificial Neural Network

ВСТУП

Проблема ідентифікації трафіку є досить поширеним викликом з яким стикається сьогоднішня система транспорту. Якби задачу неможливо було автоматизувати, дорожнім службам по всьому світі довелося би зупиняти водіїв та переглядати неймовірні об’єми відеозаписів для отримання необхідної інформації про кожен транспортний засіб. Це значно гальмувало б процес пошуку автомобілю з потрібним номером, а іноді робило б цю задачу неможливою для виконання.

Якщо казати про українську транспортну систему, то існує дуже невелика кількість готових рішень, що могли бути впроваджені. В ході цієї роботи було створено програмний продукт (як модель машинного навчання) як вирішення даної проблеми саме для України.

Перший етап вирішення завдання - отримання фронтальних зображень для автомобілів, що наближаються. Важливо, щоб якість зображення дозволяла однозначно ідентифікувати номер транспортного засобу. Потім необхідно визначити положення номерного знаку на зображенні автомобіля. Це задача поділяється на підзадачу класифікації та підзадачу регресії. Підзадача регрессії полягає у визначенні положення контурів у яких лежить об'єкт. Задача класифікації полягає у визначенні класу об'єкту в контурі. Після виявлення та сегментації зображення та виділення окремих номерних знаків задача зводиться до визначення тексту на зображенні номерного знаку.

Для цього потрібно провести деяку обробку вхідного зображення: прибрати елементи, що можуть бути ідентифіковані як шум, форматувати зображення до необхідного розширення, визначити положення кожного символу на номерному знаку та розпізнати їх, кожен окремо.

Також, за ціль було поставлено створити програмний продукт, що може обробляти відео в реальному часі, бо відео з відеореєстратора може бути досить

довгим та обробка його в реальному часі значно б підвищила ефективність використання моделі.

Отже, задачі даної дипломної роботи:

- дослідити моделі нейронних мереж, що можуть бути використані для швидкого пошуку номерного знаку транспортного засобу на зображенні роздільної здатності відповідної роздільній здатності відео з автомобільного відеореєстратору;
- навчити обрану модель на датасеті, де обов'язково мають бути присутні зображення з українських доріг для збільшення орієнтованості моделі на українські автомобільні номери порівняно з аналогічними моделями інших програмних продуктів загального використання;
- вирішити задачу визначення символів номеру транспортного засобу по зображенню і координатам контурів номеру;
- побудувати програмний продукт, що здатний зібрати статистику та провести аналіз опрацьованого відео

РОЗДІЛ 1 ОСНОВНІ ПОНЯТТЯ, ЗАДАЧА АВТОМАТИЧНОГО РОЗПІЗНАВАННЯ АВТОМОБІЛЬНОГО НОМЕРУ

1.1 Зображення та нейронні мережі. Основні поняття

Розглянемо основні поняття до таких належать:

ANPR (англ. automatic number-plate recognition) — автоматичне розпізнавання автомобільного номеру [1].

Faster R-CNN, R-FCN, SSD, FPN, RetinaNet та YOLO — нейронні мережі та суміжні методи та технології, що використовується для визначення фіч зображення.[2-7]

Фічі зображення – певні особливості зображення, які можна виділити з картинки.

Екстрактор – система, що автоматично виділяє фічі з картинки.

mAP (англ. mean Average Precision) — AP забезпечує міру якості на всіх рівнях для класифікації одного класу, його можна розглядати як площа під кривою "precision-recall". Тоді mAP - це середнє значення AP в багатоклассовій класифікації [8].

FPS (англ. Frames Per Second) — кількість кадрів, що встигають обробитися за 1 секунду.

ANN (англ. Artificial Neural Network) — комп’ютерна система обробки, що створена на основі біологічної нейронної мережі[13]

CNN (англ. Convolutional Neural Networks) — згорткові нейронні мережі, аналогічні традиційним ANN тим, що складаються з нейронів, які самооптімізіруются шляхом навчання. Різниця полягає в тому, що при обробці зображень, згорткова нейронна мережа здатна виділити специфічні фічі

зображення дякуючи її архітектурі, що робить згорткову нейронну мережу більш ефективною при роботі з картинками[13].

1.2 Актуальність та постановка задачі

Реєстраційний номер транспортного засобу або номерний знак закріплюється за транспортним засобом з метою офіційної ідентифікації. Ідентифікатор, часто цифровий або буквено-цифровий, може бути використаний для однозначної ідентифікації транспортного засобу в базі даних про транспортні засоби певного регіону. Існує безліч причин, чому окремим особам або організаціям може бути необхідно ідентифікувати транспортний засіб і, відповідно, його власника.

Прикладами можуть бути правоохоронні органи, контроль за дорожнім рухом та доступ до заборонених зон, електронний збір плати за дорожній рух або перевірка дозволів на паркування.

В ході цієї роботи має бути створена інформаційна система, що може використовуватись для визначення частоти зустрічі транспортних засобів на дорозі в ході подорожі. Ця система може бути використана як для простого спостереження за навколишнім світом транспортного засобу, так і з міркувань безпеки, адже існує безліч причин, коли потрібно визначити, чи не зустрічався той чи інший транспортний засіб, чи не переслідувався транспортний засіб, що використовує створену інформаційну систему, іншим транспортним засобом.

Ця задача є досить актуальною та перспективною, не існує такої системи, що може бути встановлена на автомобіль в Україні. Тобто для українського ринку IT задача актуальна.

У повсякденному житті на дорогах величезний рух, за цим сценарієм програма повинна виконувати дуже швидку обробку. В іншому випадку порушники та злочинці можуть врятуватися. Виявлення одного номерного знака та надійне розпізнавання його символів є непростим завданням, оскільки воно покладається на алгоритми, що вимагають складних обчислень. Для цього були

розроблені спеціальні системи, що забезпечують необхідну обчислювальну потужність. Після того, як номерний знак транспортного засобу був визначений, його потрібно подати до, часто віддаленої IT-системи, щоб мати змогу здійснити його пошук в базі даних, та остаточно ідентифікувати транспортний засіб і, можливо, його водія. Тоді цю інформацію можна обробити та використовувати для спеціальних цілей.

Системи ідентифікації транспортних засобів так чи інакше використовують багато країн. В Австралії кілька державних поліцейських служб та Міністерство юстиції (штат Вікторія) використовують як стаціонарні, так і мобільні системи ANPR[9]. Місто Мехелен, Бельгія використовує систему ANPR з вересня 2011 року для сканування всіх автомобілів, які перетинають межі міста (в'їзні та виїзні)[10]. В Канаді федеральні, провінційні та муніципальні служби поліції використовують програмне забезпечення автоматичного розпізнавання номерних знаків. Також, системи автоматичної ідентифікації транспортного засобу використовують такі країни як Данія, Німеччина, Франція, Угорщина, Саудівська Аравія, Швеція, Туреччина, Україна, Великобританія та Сполучені Штати[.

Окрім успішного використання існуючих систем, є ряд областей, де вони не є практичними, занадто важкими, недостатньо мобільними або надто дорогими для використання. Це можуть бути приватні та громадські місця для паркування на постійних або тимчасових місцях паркування. У цій дипломній роботі ми хочемо досліджується можливість розробки високомобільної системи розпізнавання номерних знаків, яка могла б використовуватися для доповнення або заміни деяких існуючих систем та була гарно зорієнтована саме на українські дороги. Тому проблема, яку потрібно вирішити ця робота, полягає в наступному: «Впровадити систему виявлення та розпізнавання в режимі реального часу що дозволяє швидко і точно обробляти інформацію про номерний знак".

1.3 Особливості задачі

Існує досить багато способів вирішити поставлену задачу. Вибраний підхід має бути досить простим з точку зору споживання обчислювальних потужностей, аби бюджет для придбання техніки для використання програмного продукту не перевищував можливості бізнесу.

З іншого боку, програмний продукт має працювати досить швидко, аби мати змогу обробляти відео в режимі реального часу та, в той же час, досить точно, бо ціна помилки за неправильний номерний знак може бути досить висока, наприклад, втрата злочинця, що пересувається за допомогою автомобіля.

Історія ANPR значно довша, ніж усвідомлює більшість людей. Через його плідне використання в останні роки для широкого спектру застосувань, таких як дослідження дорожнього руху, контроль доступу та паркування, багато людей, якщо їх запитають, здогадуються, що це винахід, що відноситься до цього тисячоліття.

1.4 Історія розвитку систем ANPR

На подив більшості людей, історія ANPR йде корінням в минуле сторіччя, оскільки вона була винайдена в 1976 році у Великобританії в відомому в той час своїми науковими розробками відділенні поліції (PSDB) (зараз це відділення наукових розробок Міністерства внутрішніх справ), а ранні системи були розроблені для використання в 1979 році. Ранні пробні системи були розгорнуті в Великобританії на дорозі A1 і на перетині тунелю Дартфорд на автостраді M25, а перший арешт, який був приписаний виявлення викраденого автомобіля за допомогою ANPR, стався лише в 1981 році. З моменту своєї появи технологія ANPR розвивалася і адаптувалася відповідно до вимог часу, знаходячи нові сфери застосування і виходячи за рамки тільки поліцейської діяльності і безпеки[11].

В 1993 ANPR вперше розгорнуто в рамках мережі камер "Сталеве кільце" навколо лондонського Сіті. Цей проект був найбільшою операцією такого типу на

той момент. Він був реалізований лондонською поліцією в спробі покласти край низці терористичних вибухів у фінансовому районі, скоєних IRA[11].

В 1997 році - створено Національний центр даних ANPR поліції (NADC) в якості доповнення до Національної комп'ютерної службі поліції. До створення NADC дані, зібрані за допомогою систем ANPR, локалізувалися в межах відповідного поліцейського підрозділу, яке експлуатувало камери, але Національний центр даних ANPR дозволив проводити аналіз за межами поліцейських підрозділів, централізувавши всі дані ANPR поліції з усієї Великобританії[11].

В 2003 році введена схема стягнення плати за пробки в Лондоні, яка по ряду причин була спрямована на зниження інтенсивності руху в центрі Лондона. Зона зняття плати, що охоплює 20 квадратних кілометрів столиці, управляється організацією Transport for London і являє собою кільце з майже 700 камер ANPR, які встановлені на кожній дорозі в зоні стягнення плати та за її межами. Це перше великомасштабне використання ANPR в цивільних цілях[11].

В 2005 році утворена компанія ANPR International Limited. Компанія народилася з переконання, що технологія ANPR може зробити набагато більше, ніж вона вже зробила на тодішній день. Першою розробкою компанії стало створення офісної платформи eyeTRAFFIC, яка призначена для збору даних ANPR з ряду різних джерел мережі і обробки даних за допомогою ряду модульних програм[11].

В 2006 році ANPR International розгорнула свою першу систему статичних камер для управління парковками - bayGAURDIAN, яка стала першим модулем, створеним для офісної системи eyeTRAFFIC. Система дозволила операторам парковок відстежувати в'їжджаючі і виїжджаючі автомобілі, розраховуючи кількість часу, проведеного автомобілем на парковці, і автоматично визначаючи ті автомобілі, які були авторизовані або залишалися занадто довго[11].

В 2007 році модуль bayGUARDIAN розширено за рахунок інтеграції в обладнання Charge Parking. Система повністю інтегрується з рядом лічильників

Pay & Display, дозволяючи системам бек-офісу автоматично виявляти автомобіль, який не заплатив за паркування або заплатив занадто мало[11].

В 2009 році ANPR International розробляє свій перший мобільний продукт ANPR - streetSWEEPER, загального призначення, включаючи дослідження дорожнього руху, мобільне спостереження і контроль неподаткових транспортних засобів для Агентства з ліцензування водіїв і транспортних засобів (DVLA). У перший же день роботи в Центральному Лондоні streetSWEEPER виявив більше 27 неподаткових автомобілів всього за кілька годин[11].

В 2011 році розроблена і впроваджена система RoadGUARDIAN для боротьби з проблемою поїздок автомобілів по забороненим дорогах, щоб уникнути заторів "Rat-running". Система розгорнута спільно з Радою графства, а виявлення контролюється поліцією, яка виписує фіксовані штрафні повідомлення будь-якого порушника[11].

В 2012 році - розроблено перша система реєстрації пошкоджень автомобіля (DRS) для реєстрації стану автомобілів, что прибувають на парковку в аеропорту, з метою запобігання шахрайських страхових претензій до персоналу аеропорту з приводу пошкодження автомобіля[11].

В 2013 році - ANPR International отримує нагороду за технологічні досягнення за мобільне дослідження дорожнього руху, проведене для Ради графства Глостершир. У дослідженні використовувався парк автомобілів, обладнаних системою StreetSWEEPER, для збору даних про профіль руху і звичках парковки в ряді міст і селищ, щоб допомогти Раді краще зрозуміти транспортні потоки і використання транспортних засобів[11].

В 2014 році - speedSENTINEL розроблений для вирішення проблеми перевищення швидкості на транспортних розв'язках. Sentinel був розроблений як мобільна система камер контролю швидкості, яку можна буксирувати до транспортних розв'язок і розгортати в довільному порядку, щоб переконатися, що оператори автобусів дотримуються строгі обмеження швидкості, встановлені для безпеки пішоходів на жвавих станціях[11].

В 2016 році - Система реєстрації пошкоджень транспортних засобів (DRS) розширена і тепер включає в себе двосмугову систему, що дозволяє інтегрувати кілька смуг руху і до 14 камер з даними ANPR і програмним забезпеченням попереднього бронювання транспортних засобів для прискорення обслуговування клієнтів і підвищення зручності. Перша система з двома смугами руху запущена в аеропорту Донкастер Шеффілд і з’ясувала, що середній час, що витрачається відвідувачами на висадку з автомобіля, становить 32 секунди[11].

В 2017 році ANPR International запускає gateGUARDIAN як рішення для захисту бар'єрів з метою вирішення проблеми безпеки, пов'язаної з заїздом автомобілів "хвостом" на контрольовану бар'єром парковку або виїздом з неї. Перша система запущена в експлуатацію в одному з аеропортів Великобританії для вирішення проблеми високого рівня інцидентів з автомобілями, які намагаються уникнути оплати за парковку в зоні преміум-класу для зустрічі і висадки пасажирів[11].

1.5 Загальноприйняті етапи вирішення задачі

Традиційно, вирішення задачі ANPR складається з таких етапів: квантування зображення - перетворення зображення RGB у чорно-біле, фільтрування шуму, брудних деталей та фіксація не точних координат, розділення зображень цифр
використання нейронної мережі для розпізнавання однозначної цифри

1.5.1 Квантування

Початкове зображення виглядає наступним чином (рис. 1.1)[12]

Рисунок 1.1 - Зображення до квантування

Після квантування (переходу до чорно-білого формату), зображення стає таким (рис. 1.2) (рис. 1.3) (рис. 1.4)[12]

Рисунок 1.2 - Зображення після квантування

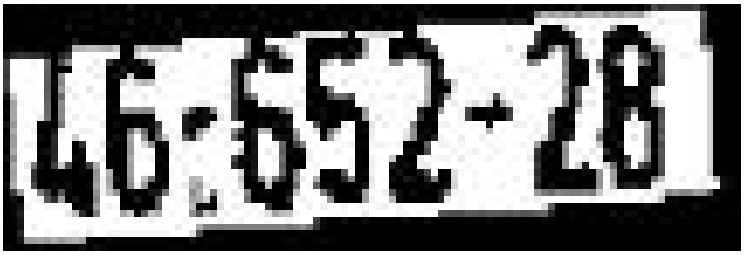


Рисунок 1.3 - Зображення після квантування

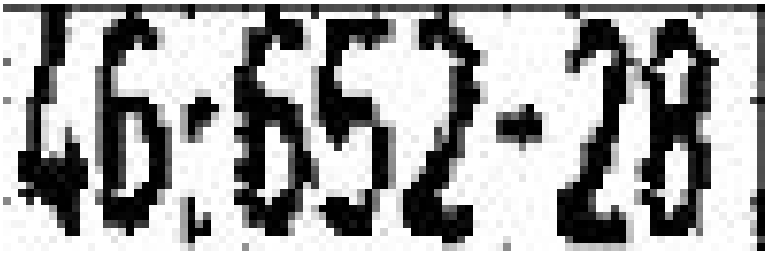


Рисунок 1.4 - Деформація зображення до прямокутника

1.5.2 Фільтрація

Потім, прибираються всі зайві шуми з зображення (рис. 1.5)[12] (інформація, що не несе змісту для вирішення нашої задачі).

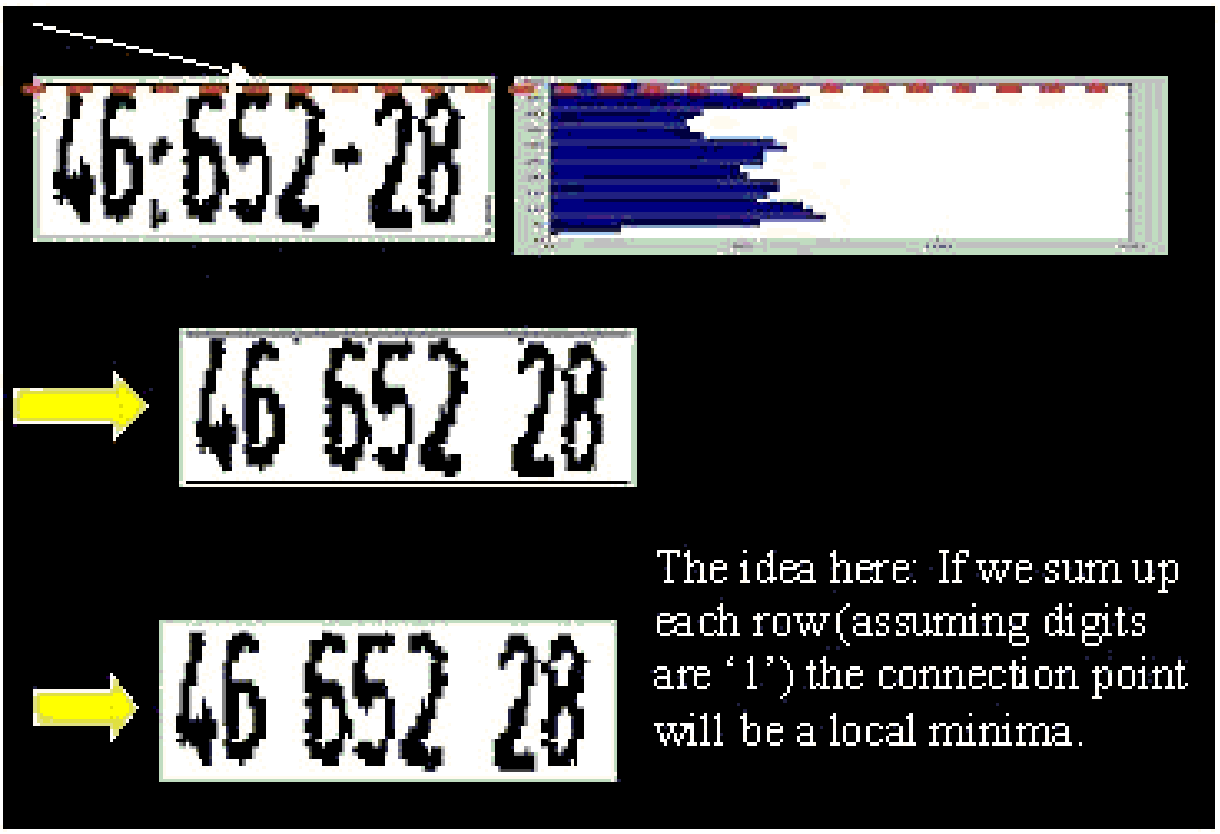


Рисунок 1.5 - Виділення та усунення шумів з зображення

1.5.3 Розпізнавання цифр

Далі, починається робота нейронної мережі, щодо розпізнавання тексту на картинці (рис. 1.6)[12].

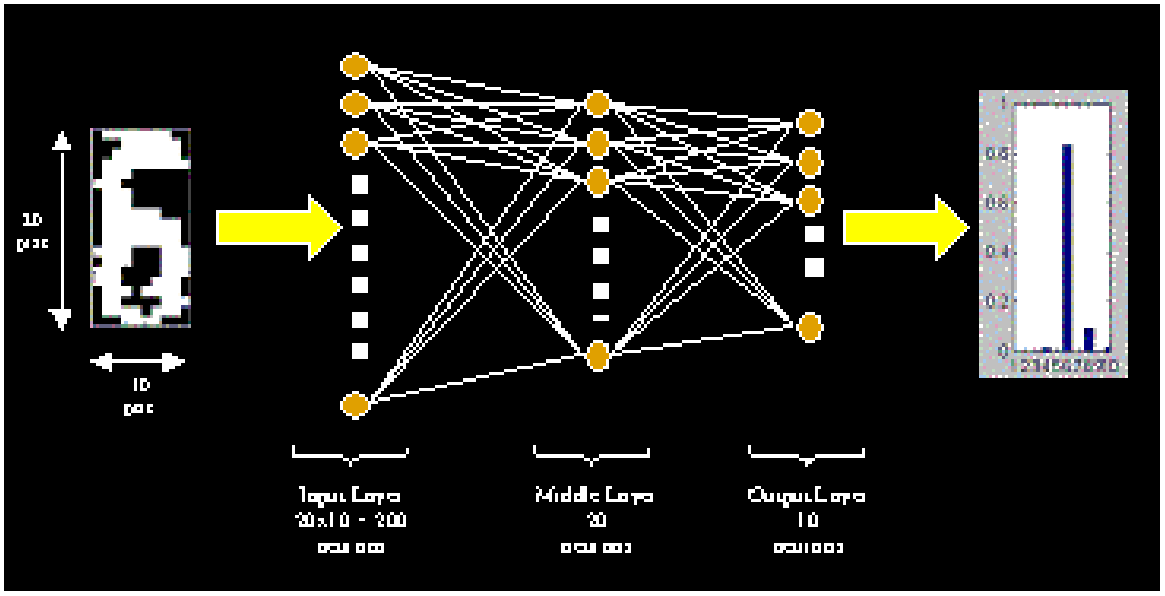


Рисунок 1.6 - Розпізнавання цифри із зображення за допомогою нейронної мереж

Навожу загальну схему такого варіанту вирішення задачі (рис. 1.7)[12]:

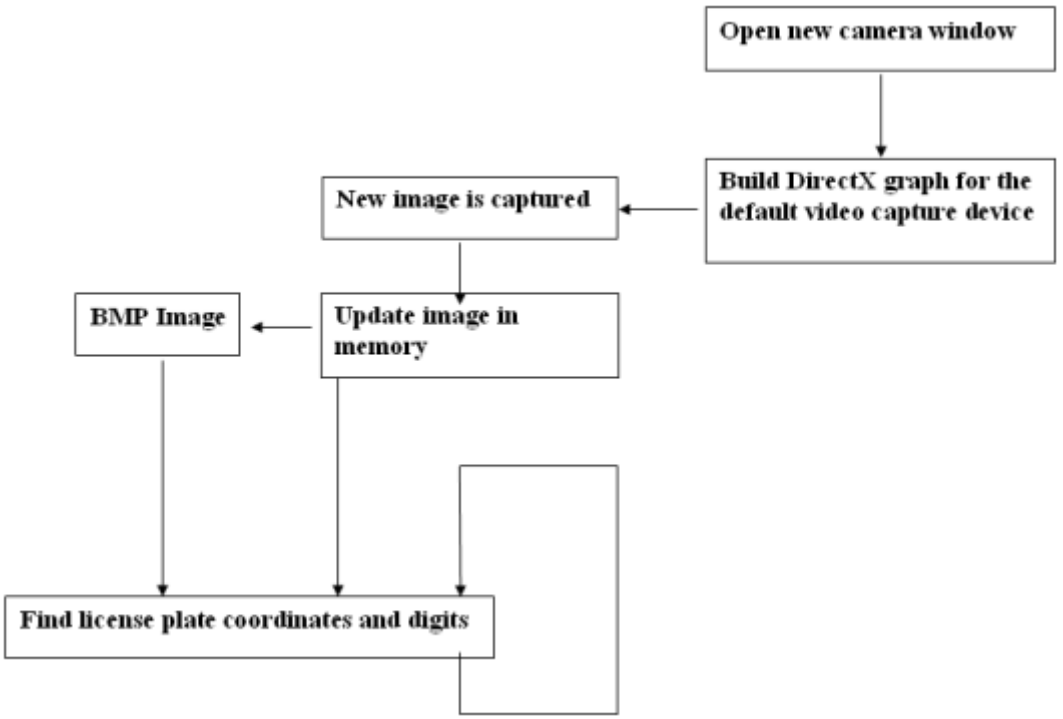


Рисунок 1.7 – Загальна схема-приклад роботи програми ANPR

1.6 Висновки до розділу 1

Спочатку, в розділі було розглянуто основні поняття, що будуть надалі використовуватись в цій роботі. Далі була визначена актуальність даної задачі та зроблена детальна її постановка. Було наголошено на тому, що задача є актуальною через відсутність аналогічного програмного продукту, що пристосований для українського формату номерів, тобто є наразі можуть використовуватись лише програми загального призначення. Потім, було розглянуто особливості даної задачі, - те на що потрібно буде звернути увагу при створенні інформаційної системи. Також, було розглянуто історію систем ANPR та загальноприйняті етапи вирішення задачі ANPR.

Наразі, сучасний рівень розвитку нейронних мереж та техніки дозволяє витрачати набагато менше зусиль для створення ANPR. Основна проблема полягала в тому, що система мала працювати в реальному часі, або вона буде банально не встигати обробити кількість даних, що поступає. В такому випадку, важлива інформація може бути втрачена, та до неточностей результатів роботи нейронної мережі додадуться неточності в статистиці обробленого відео, що виникають через те, що інформація була опрацьована лише частково.

Метод цієї роботи буде заснований на виделенні фіч зображення[21] за допомогою сучасних нейронних мереж. Їх порівняння та архітектури будуть розглянуті в наступних розділах цієї роботи. Серед цих мереж є ті, що однозначно можуть обробляти відео з FPS 30-60[20], тобто в режимі реального часу при цьому не потребуючи значних для сучасних комп'ютерів обчислювальних потужностей [18-19].

РОЗДІЛ 2 ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Згорткові нейронні мережі

Згорткові нейронні мережі (CNN) аналогічні традиційним ANN за своєю архітектурою в плані того, що вони складаються з послідовних шарів нейронів (рис. 2.1)[13][23].

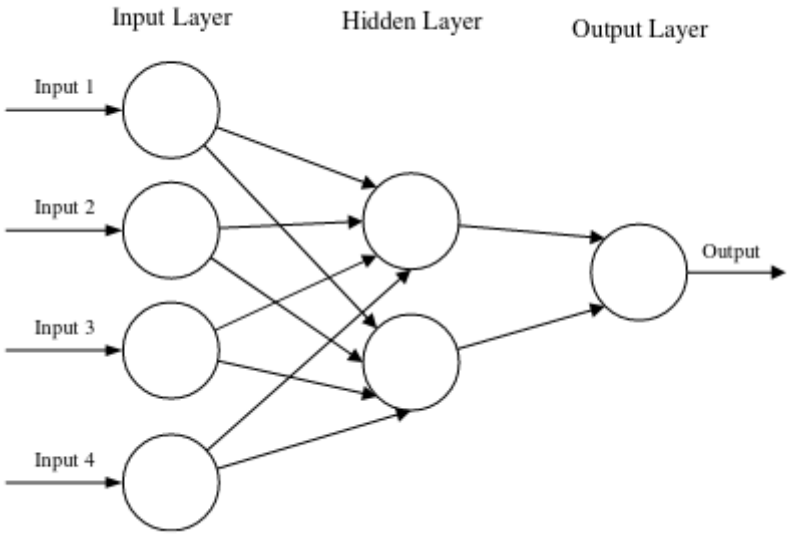


Рисунок 2.1 - Проста нейронна мережа прямого зв'язку (FNN), що складається з вхідного шару, прихованого шару та вихідного шару.

Єдина значна відмінність між CNN і традиційними ANN полягає у тому, що CNN в основному використовується для вирішення задачі розпізнавання зображень. CNN дозволяють побудувати архітектуру конкретної мережі базуючись на специфіці тої чи іншої множини зображень, зробити мережу більше орієнтованою під конкретну задачу, і в цей час значно скоротити кількість параметрів, необхідних для налаштування моделі [23].

Однією з ключових відмінностей полягає в тому, що нейрони, з яких складаються шари в CNN, організовані в трьох вимірах: просторова розмірність входу (висота і ширина) і глибина. Глибина відноситься не до загальної кількості шарів в ANN, а до третього виміру обсягу активації. На відміну від стандартних ANN, нейрони в межах будь-якого даного шару з'єднуються тільки з невеликою

областю попереднього шару [21-25].

На практиці це означає, що якщо розмірність даних буде 64 × 64 × 3 (висота, ширина і глибина), то в результаті кінцевий вихідний шар буде мати розмірність 1 × 1 × n (де n представляє собою можливу кількість класів), так як ми згорнули повну розмірність вхідного шару в меншу за допомогою збільшення глибини (рис. 2.2) (рис. 2.3)[14][26-30]

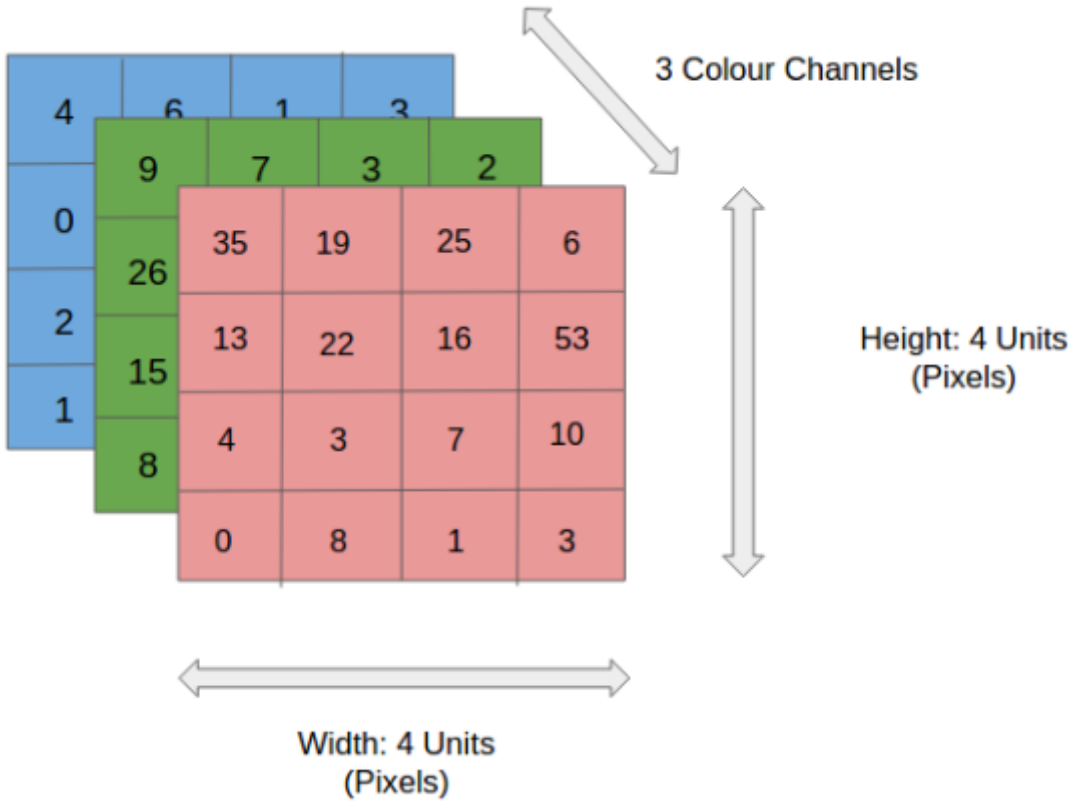


Рисунок 2.2 — Приклад розмірності даних 4 × 4 × 3 для CNN

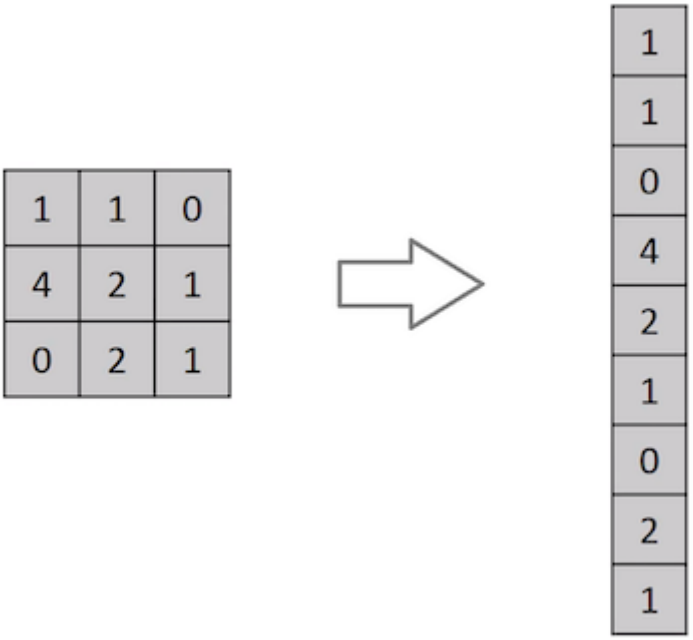


Рисунок 2.4 — Приклад зниження розмірності даних в CNN.

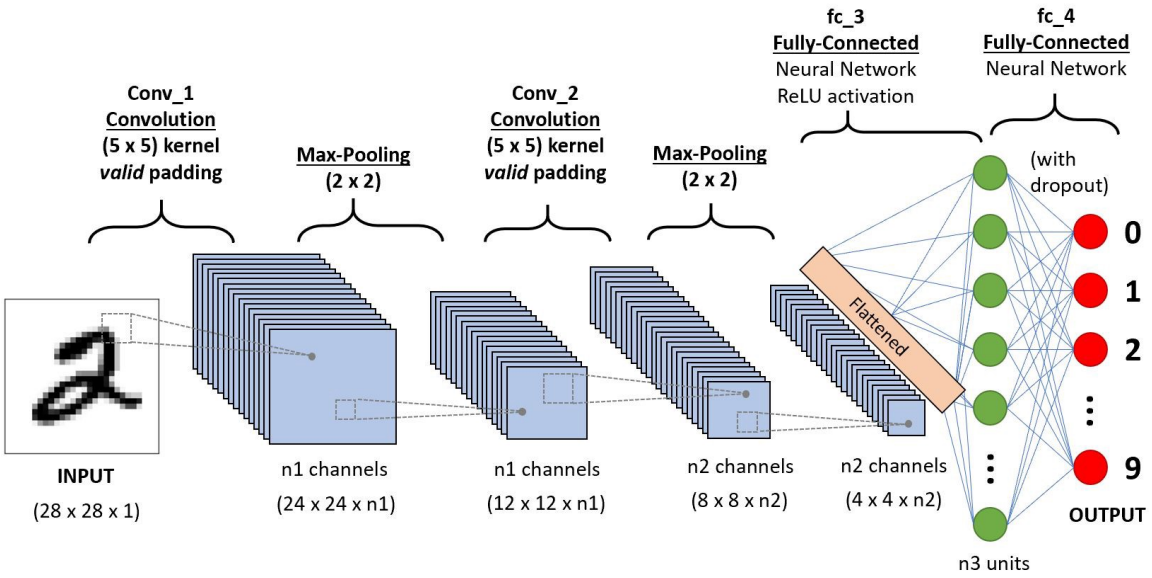


Рисунок 2.3 — Архітектура CNN для класифікації рукописних цифр

CNN складається з 3 типів шарів. Це згорткові шари, агрегувальні шари та повноз’єднані шари. Їх поєднання формує архітектуру CNN (рис. 2.4)[13].

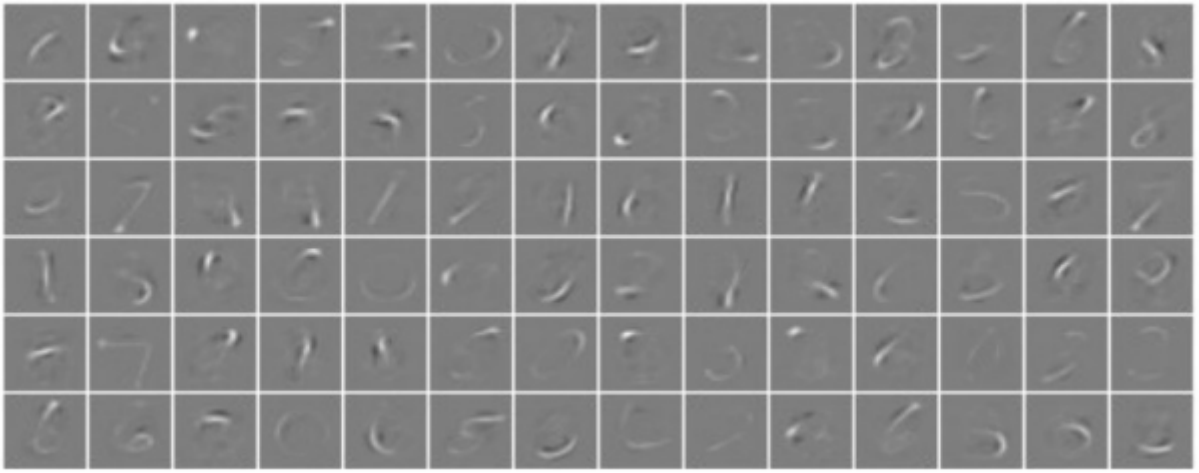


Рисунок 2.4 — Результат оброки першим згортковим шаром згорткової нейронної мережі для задачі розпізнавання рукописних цифр.

2.1.1 Згортковий шар

Очевидно, що згортковий шар грає важливу роль в роботі CNN. Параметри шару зосереджені навколо використання ядер, що навчаються[24][36-38].

Коли дані потрапляють до згорткового шару, шар робить згортку кожного фільтру по просторовій розмірності, створюючи двовимірну матрицю активації [24]. Як можна побачити на рис. 2.5, скалярний добуток обраховується для кожного значення в цьому ядрі. На основі цього, нейронна мережа навчає ядро, що активується, коли зустрічає деяку конкретну фічу у вхідних даних. Такі ядра, як правило, називаються ядрами активації (рис. 2.5)[13].

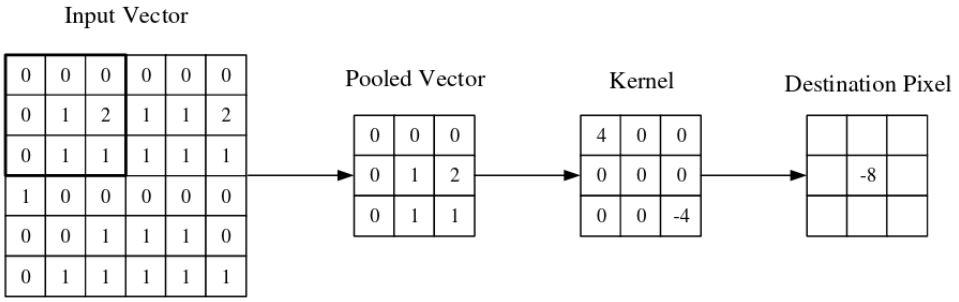


Рисунок 2.5 — Візуальна демонстрація роботи згорткового шару. Центральний елемент ядра. Розміщуються над вхідним вектором, котрий обчислюється та замінюється зваженою сумою себе та всіх пікселів, що лежать поруч

Кожне ядро буде мати відповідну матрицю активації, що будуть скомпоновані для формування глибини вихідної розмірності згорткового шару.

Згорткові шари також можуть значно спростити складність моделі за рахунок оптимізації вхідних даних. Дані можуть бути оптимізовані за допомогою трьох гіперпараметрів: глибини, кроку та заповнення нульовими значеннями.

Глибина вихідного об’єму, що створюється згортковими мережами може бути задана вручну через кількість нейронів в шарі, що підключені до однієї й тої ж ділянки вхідного сигналу. Це можна також спостерігати в інших формах ANN, де всі нейрони в скритому шарі напряму з’єднані з кожним окремим нейроном. Зменшення цього гіперпараметра може значно мінімізувати загальне число нейронів мережі, але при цьому значно знизити можливості моделі, що до розпізнавання образів.

Ми також можемо визначити крок в якому ми встановлюємо глибину просторової розмірності вхідного сигналу для розміщення рецептивного поля. Наприклад, якщо ми задамо крок 1, тоді ми отримаємо сильно перекрите рецептивне поле, що викликатиме значні активації. Як альтернатива, якщо встановити крок на велике число, то кількість перекриття зменшиться та на виході

ми отримаємо сигнал меншої просторової розмірності.

Заповнення нульовими значеннями — це простий процес заповнення границі вхідних даних нульовими значеннями. Це Ефективний метод, що дозволяє додатково контролювати розмірність вхідних об’ємів.

Поділ параметрів працює на припущенні, що якщо фіча одного регіону корисна в одній області, то вона може бути корисна і в іншій. Якщо ми обмежимо кожну окрему карту активації в межах вихідного обсягу однаковими вагами та зміщенням ми побачимо значне зменшення кількості параметрів, які отримуємо в результаті обробки даних згортковим шаром.

В результаті цього на етапі зворотного розповсюдження, вихід кожного окремого нейрона буде представляти собою загальний градієнт, що може бути підсумований по всій глибині. Таким чином, поновлюється тільки один набір вагів, а не кожен конкретний.

2.2.2 Агрегувальний шар

Агрегувальні шари мають на поступове зниження розмірності результату і, таким чином, наступне зниження кількості параметрів моделі та її обчислювальної складності[32-34].

Агрегувальний шар працює над кожною матрицею активації на вході та масштабує її розмірність за допомогою функції max. В більшості згорткових нейронних мереж вони представлені у вигляді шарів з максимальним з'єднанням з ядрами розмірності 2, що використовуються з кроком 2 впродовж вхідного сигналу. В результаті, матриція активації зменшується до 25% від початкового розміру, при цьому глибина залишається стандартного розміру [38]

2.2.2 Повноз’єднальний шар

Повноз’єднальний шар містить нейрони які безпосередньо з’єднані з нейронами в двох сусідніх шарах, не пов’язані з будь якими шарами всередині них. Це повний аналог архітектури стандартної ANN (рис. 2.5)[14].

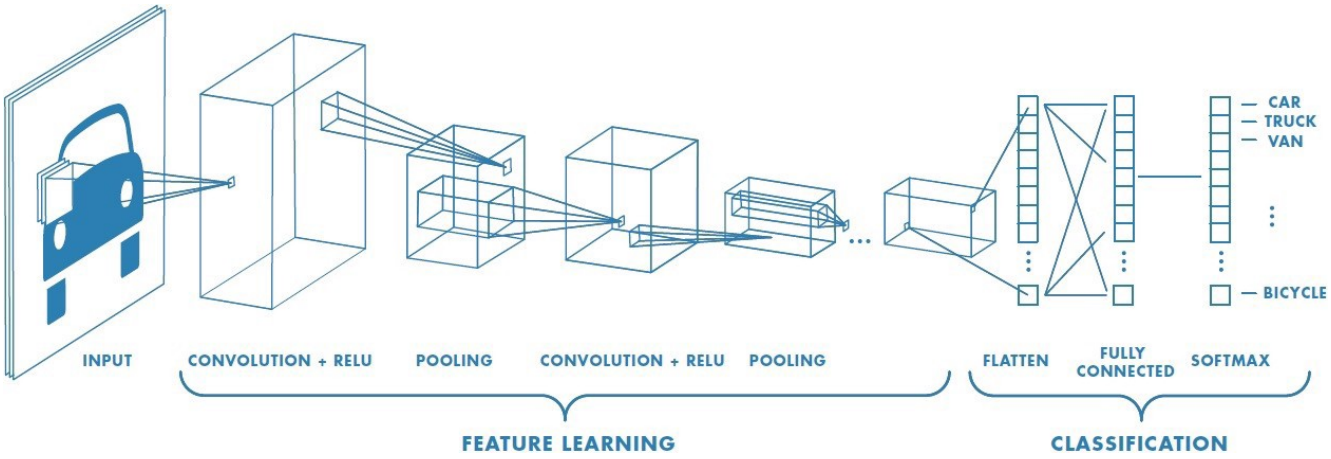


Рисунок 2.6 — Архітектура CNN для розпізнавання автомобілів.

2.3 Порівняльна характеристика згорткових мереж та технологій, що можуть бути використані для вирішення задачі

Для результату, представленого нижче, модель навчена за допомогою PASCAL VOC 2007[39], так і за 2012 рік. mAP (середня точність) вимірюється за допомогою тестового датасету PASCAL VOC 2012. Що стосується SSD, на діаграмі відображаються результати для вхідних зображень 300 × 300 та 512 × 512. Для YOLO було отримано результати для зображень 288 × 288, 416 × 461 та 544 × 544. Зображення з більшою роздільною здатністю для тієї ж моделі мають кращий показник mAP, але повільніше обробляються (рис. 2.7)[15].

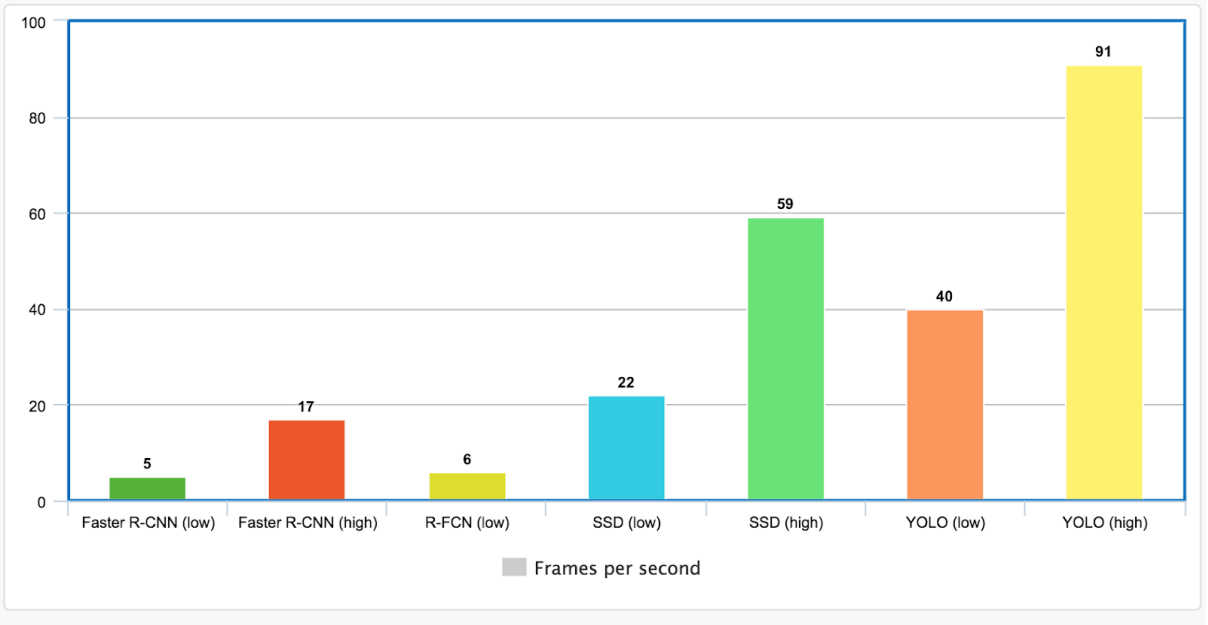


Рисунок 2.7 — Порівняльна точність різних нейронних мереж.

Нижче наведено швидкість виділення фіч та обробки зображень. Представлені найвищий і найнижчий показник частоти кадрів, взятий із відповідних досліджень. Тим не менше, наведений нижче результат може бути дуже упередженим (рис. 2.8)[15].

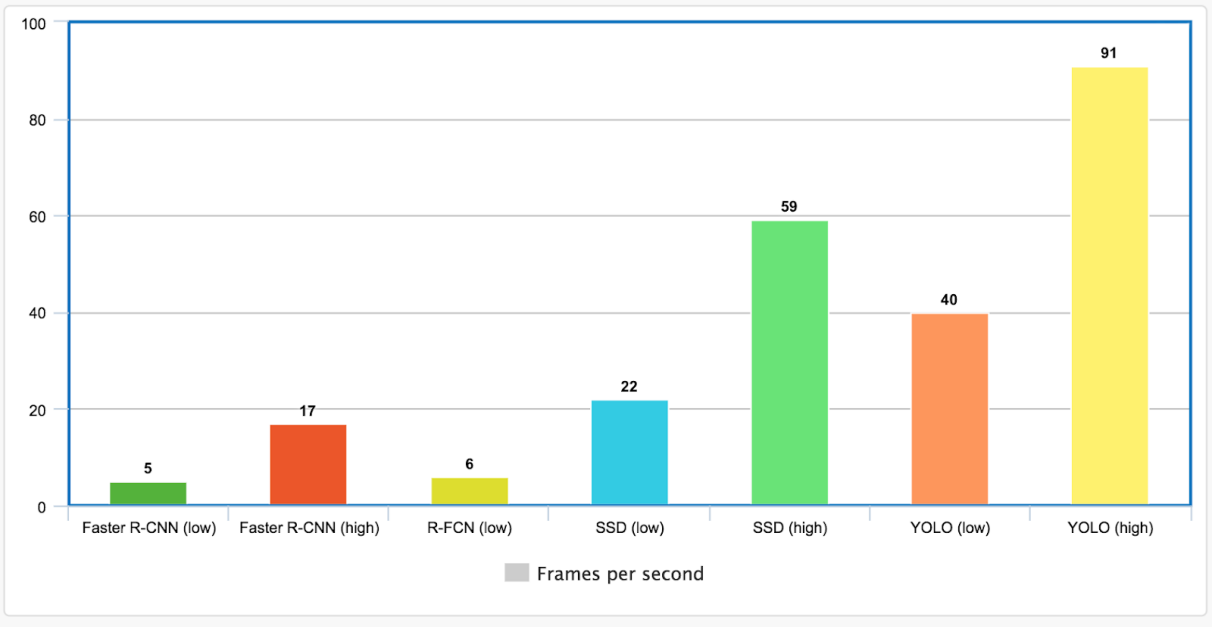


Рисунок 2.8 — FPS різних мереж.

Як бачимо, з точки зору точності, всі мережі мають досить наближені одне до одного показники, але YOLO має найвищий FPS рейтинг.

Насправді, найважливіше питання не в тому, який детектор краще. На нього неможливо відповісти. Реальне питання полягає в тому, який детектор при якій конфігурації параметрів дасть нам найкращий баланс швидкості і точності, який необхідний для вирішення конкретної проблеми. Нижче наведено порівняння компромісу між точністю і швидкістю (час вимірюється в мілісекундах).

На зображенні нижче наведено точність в залежності від часу, де маркери вказують на мета-архітектуру, а кольори - на екстрактор ознак. Кожна пара (мета-архітектура, екстрактор ознак) може використовуватися декількома точками на цьому графіку через зміни розміру вхідних даних, швидкості і т.д. (рис. 2.9)[15]

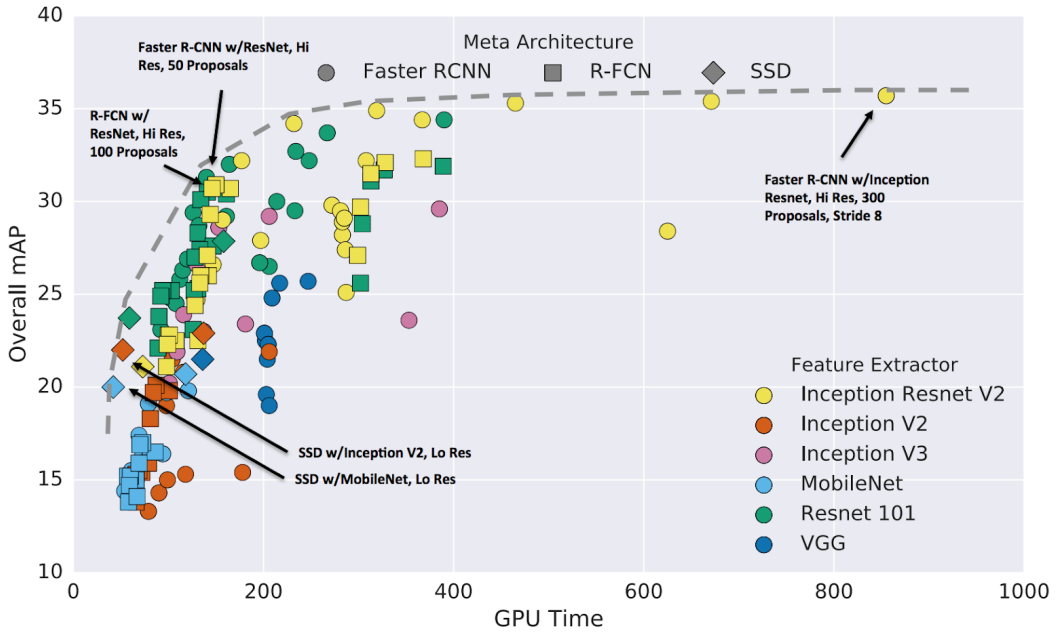


Рисунок 2.9 — mAP до часу GPU для різних мереж

В цілому, Faster R-CNN є більш точним, в той час як R-FCN і SSD швидше. Faster R-CNN, яка використовує Inception Resnet з 300 регіноами (proposals), дає найвищу точність при 1 FPS для всіх протестованих випадків. SSD на MobileNet має найвищий mAP серед моделей, призначених для обробки в реальному часі.

Цей графік також допомагає нам знайти оптимальні точки, де можна проміняти точність на хорошу швидкість. Моделі R-FCN з використанням Residual Network забезпечують хороший баланс між точністю і швидкістю, Більш швидка R-CNN з Resnet може досягти аналогічної продуктивності, якщо ми обмежуємо кількість регіонів до 50 (рис. 2.10)[15].

Model summary	minival mAP	test-dev mAP
(Fastest) SSD w/MobileNet (Low Resolution)	19.3	18.8
(Fastest) SSD w/Inception V2 (Low Resolution)	22	21.6
(Sweet Spot) Faster R-CNN w/Resnet 101, 100 Proposals	32	31.9
(Sweet Spot) R-FCN w/Resnet 101, 300 Proposals	30.4	30.3
(Most Accurate) Faster R-CNN w/Inception Resnet V2, 300 Proposals	35.7	35.6

Test-dev performance of the "critical" points along our optimality frontier.

Рисунок 2.10 — результати порівняння моделей

Більш faster R-CNN і R-FCN кращі у виділенні фіч, але у випадку SSD це не є досить важливим (рис. 2.11)[15].

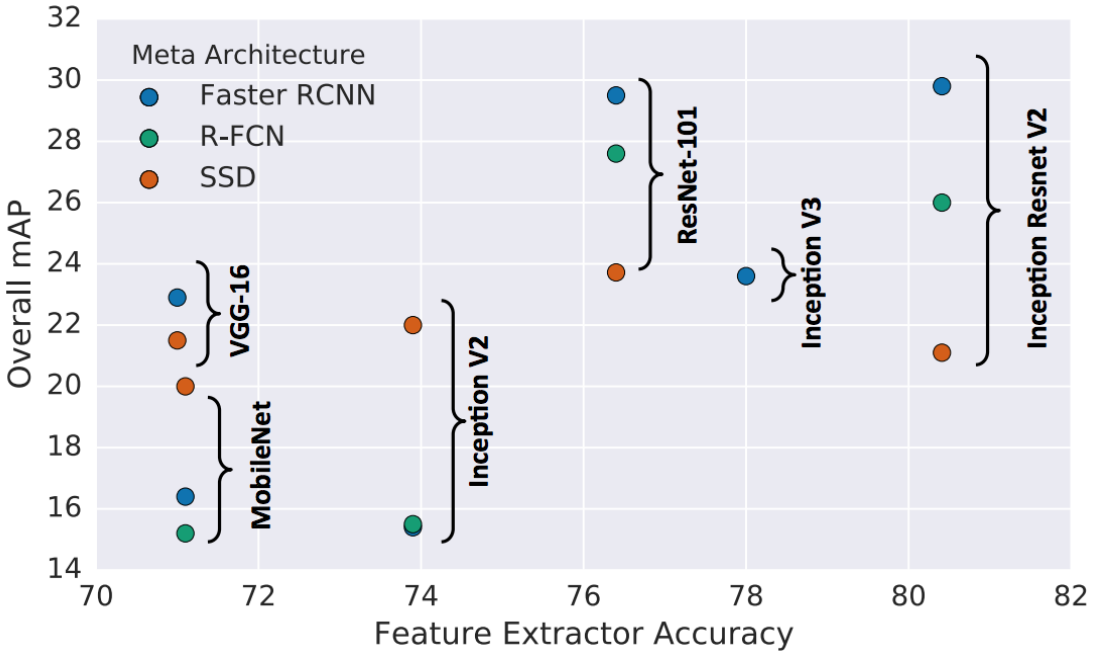


Рисунок 2.11 —Відношення точності виділення фіч до mAP

Для великих об'єктів SSD працює досить добре навіть з простим екстрактором. Більше за це, SSD навіть може зрівнятися з точністю інших детекторів при використанні кращого екстрактора. Але, на жаль на дрібних об'єктах SSD працює набагато гірше в порівнянні з іншими методами.

Наприклад, у SSD є проблеми з виявленням пляшок в середині таблиці нижче, в той час як інші методи можуть це зробити (рис. 2.12)[15].

(a) SSD+Mobilenet, lowres

(b) SSD+InceptionV2, lowres

(c) FRCNN+Resnet101, 100 proposals

(d) RFCN+Resnet10, 300 proposals

Рисунок 2.12 — Приклад проблеми SSD.

Більш висока роздільна здатність зображень значно покращує виявлення дрібних об'єктів і одночасно допомагає виявленню великих об'єктів. При зменшенні роздільної здатності в два рази в обох вимірах точність знижується в середньому на 15,88%, але час виведення також зменшується в середньому на 27,4% (рис. 2.13) (рис. 2.14)[15].

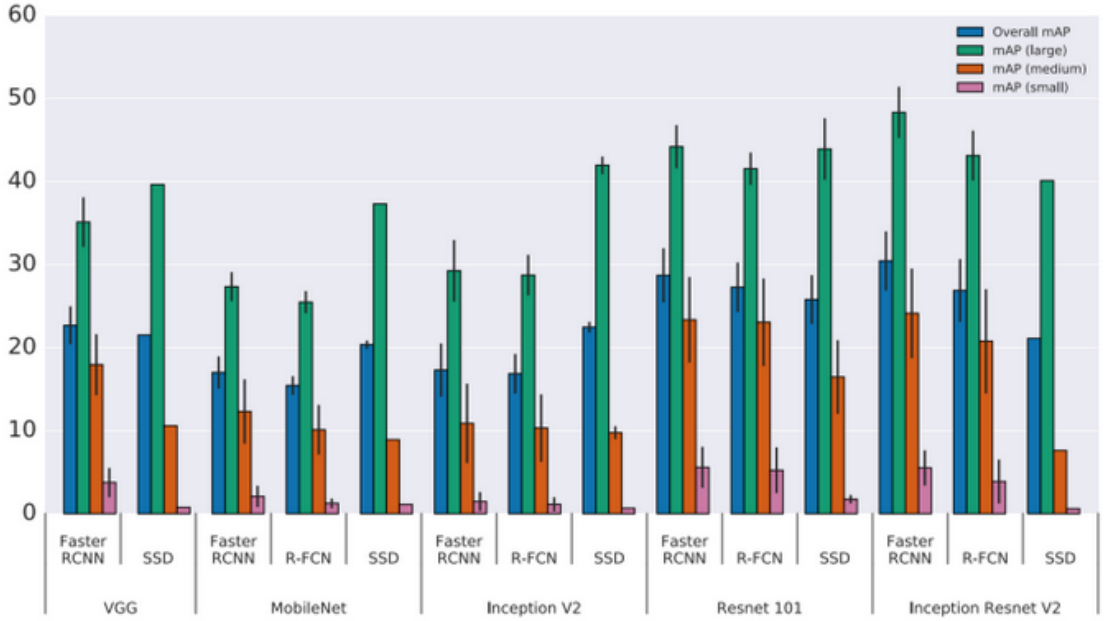


Рисунок 2.13 — Порівняльні характеристики різних мереж

Рисунок 2.14 — Вплив роздільної здатності зображення

Кількість регіонів (proposals) може істотно вплинути на Faster R-CNN (FRCNN) без істотного зниження точності. Наприклад, з Inception Resnet, Faster R-CNN може підвищити швидкість в 3 рази при використанні 50 регіонів

(proposals) замість 300. При цьому падіння точності становить всього 4%. Оскільки R-FCN має набагато менше роботи на ROI, поліпшення швидкості набагато менш значне (рис. 2.15)[15].

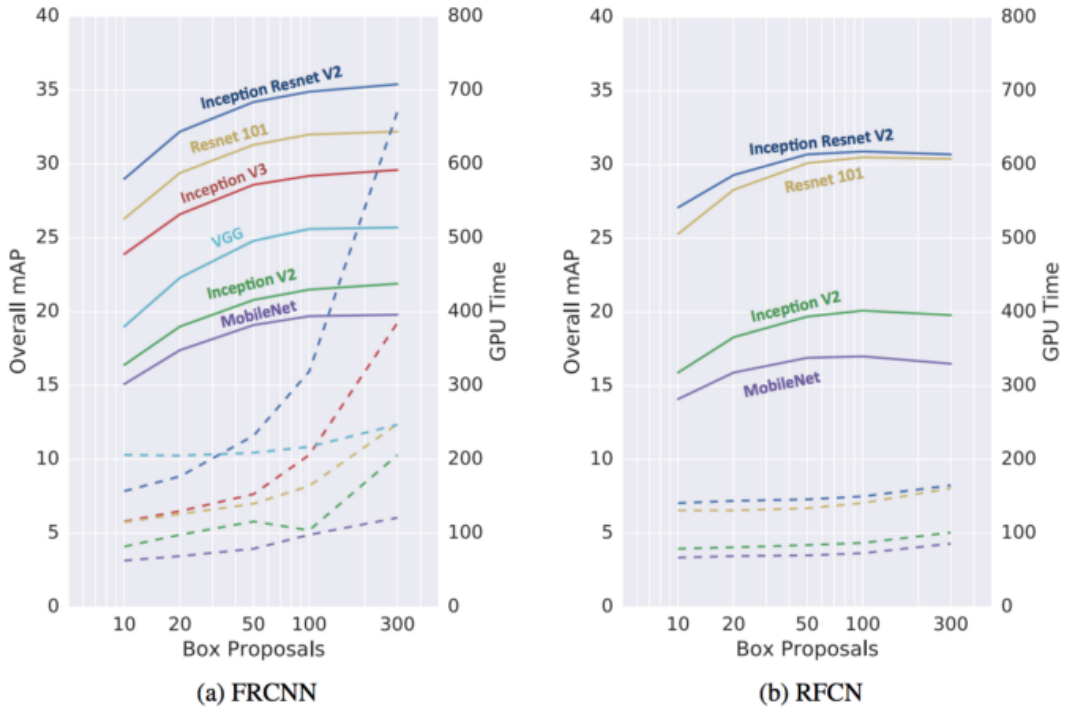


Рисунок 2.15 — Вплив збільшення кількості регіонів на mAP

Нижче наведено час роботи GPU для різних екстракторів фіч (рис. 2.16)[15].

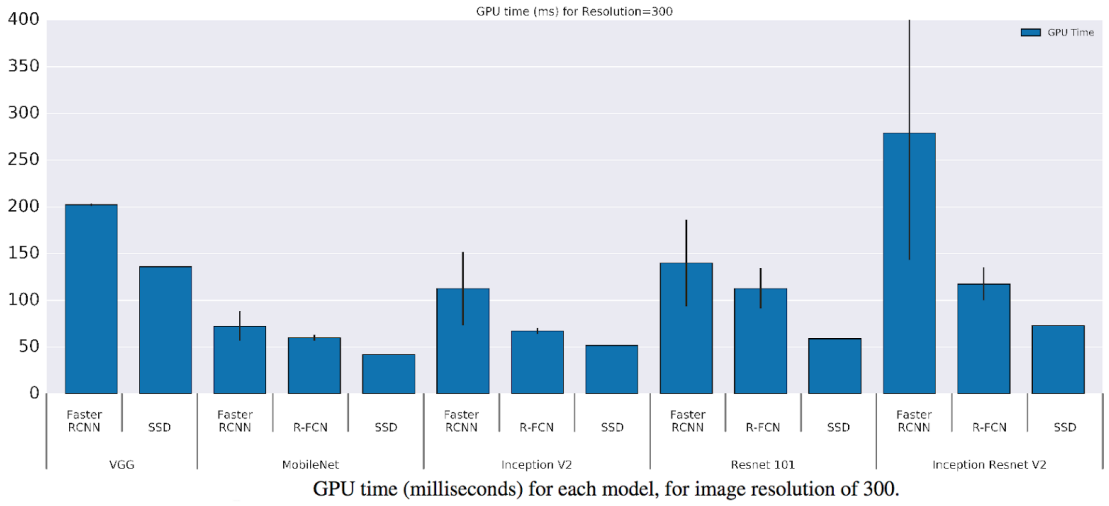


Рисунок 2.16 — Швидкість GPU для різних мереж

Хоча в багатьох роботах для вимірювання складності використовується FLOPS (кількість операцій з плаваючою комою), це не обов'язково відображає точну швидкість. Щільність моделі (розріджена модель в порівнянні з щільною) впливає на те, як довго вона буде працювати. Як не дивно, менш щільна модель зазвичай займає в середньому більше часу для завершення кожної операції з плаваючою комою. На діаграмі нижче нахил (FLOPS і співвідношення GPU) для більшості щільних моделей більше або дорівнює 1, в той час як для легкої моделі він менше одиниці. Тобто менш щільні моделі менш ефективні, навіть незважаючи на меншу загальний час виконання. Однак причина цього до кінця не вивчена (рис. 2.17)[15].

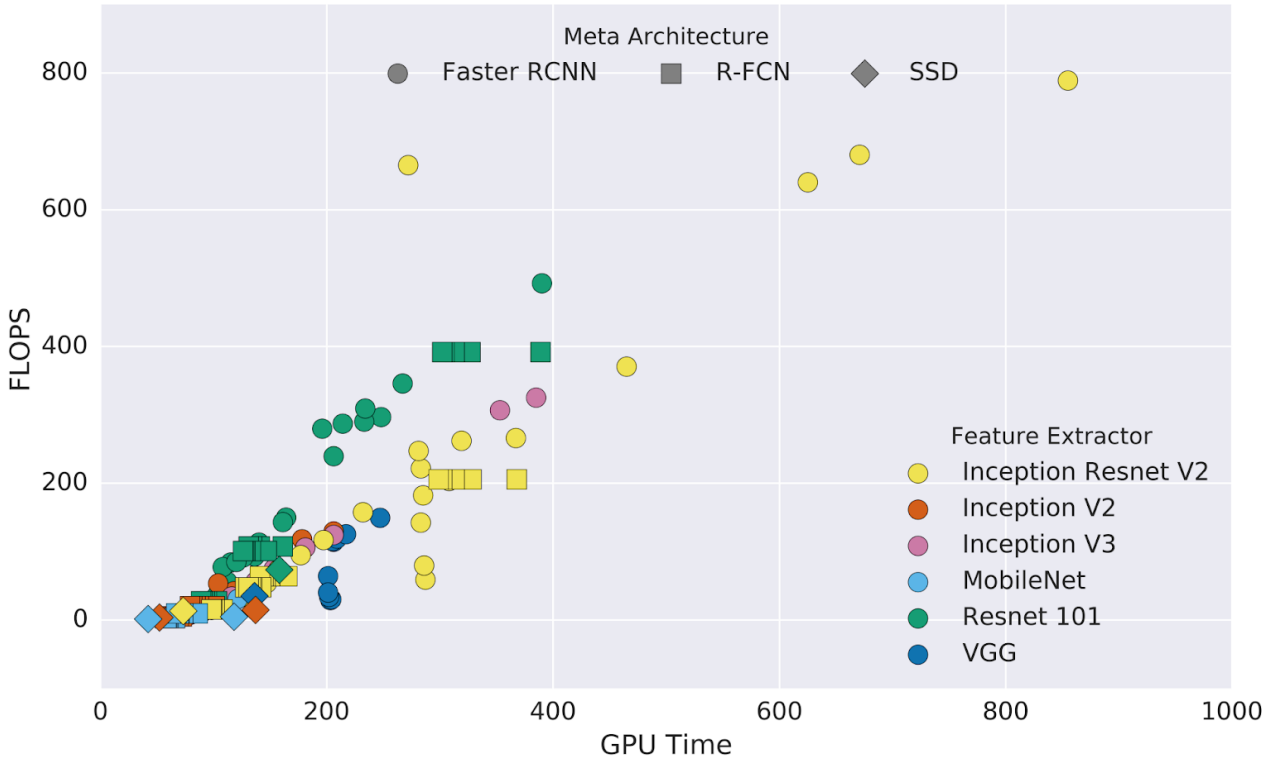


Рисунок 2.17 — Відношення FLOPS до часу роботи GPU

MobileNet займає менше місця. Він вимагає менше 1 Гб (загалом) пам'яті (рис. 2.18)[16].

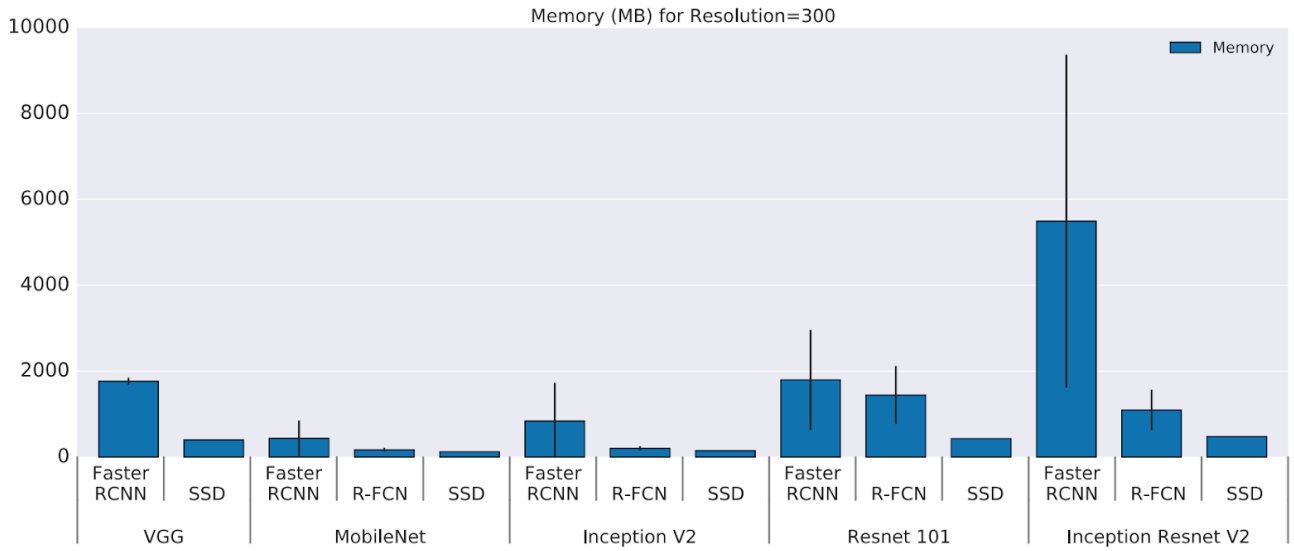


Рисунок 2.18 — Кількість пам’яті, що займають різні мережі.

2.4 Висновки до розділу 2

Перш за все в розділі було розглянуто згорткові нейронні мережі. Наведено їх особливості, якими вони відрізняються від звичайних нейронних мереж. Також, був описаний математичний апарат, який стоїть за використанням згорткових нейронних мереж.

Було детально розглянуто загальну архітектуру згорткової мережі. Зазвичай, вони складаються з трьох видів шарів. Також було наведено детальний опис принципу роботи кожного з видів шарів. Згортковий шар відповідає за згортку, - зміну розміру вхідного вектора зображення. Згортковий шар можна натренувати так, щоб зберігати дані про конкретні фічі та особливості, що зустрілися у вхідних даних для подальшого поширення мережею. Агрегувальний шар відповідає за зміну розмірності даних, що обробляються мережею. Повноз’єднальний шар виконує ті ж функції, що і в звичайній нейронній мережі, тобто певне перетворення даних. Як результат, після обробки згортковою нейронною мережею, дані змінюють розмірність та якійсь фічі стають більш вираженими.

Також було проведено детальне порівняння та аналіз різних згорткових нейронних мереж для вирішення задачі пошуку автомобільних номерів. Особлива

увага приділялася для важливих для задачі метрик таких як швидкість роботи моделей (важливо, бо в результаті цієї роботи має утворитись модель, що працює в режимі реального часу), точність роботи моделі і т.д. За допомогою цих знань можна полегшити прийняття рішення, щодо вибору конкретної моделі для реалізації задачі. В результаті була вибрана модель RetinaNet про яку мова піде в наступному розділі.

РОЗДІЛ 3 ПОБУДОВА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вибір моделі

Базуючись на проведених дослідженнях та переглянутих теоретичних відомостях, була вибрана RetinaNet[16]. Далі буде більш детально розглянуто архітектура RetinaNet.

3.2 RetinaNet. Теоретичні відомості

Одноетапна архітектура нейронної мережі RetinaNet базується на комбінації Feature Pyramid Network (FPN) зверху над архітектурою ResNet[40] для створення різноманітної, багатомасштабної згорткової піраміди фіч. До цієї основи RetinaNet додає дві підмережі, одну для класифікації якірних рамок та іншу для регресії якірних рамок (визначення їх положення). Дизайн мережі спеціально досить нескладний, що дозволяє сфокусуватись на функції втрат Focal Loss, що зводить нанівець розрив у точності між цим одноетапним детектором та сучасними двохетапними детекторами, як Faster R-CNN с FPN, тим не менш досягаючи більшої швидкості (рис. 3.1) [16].

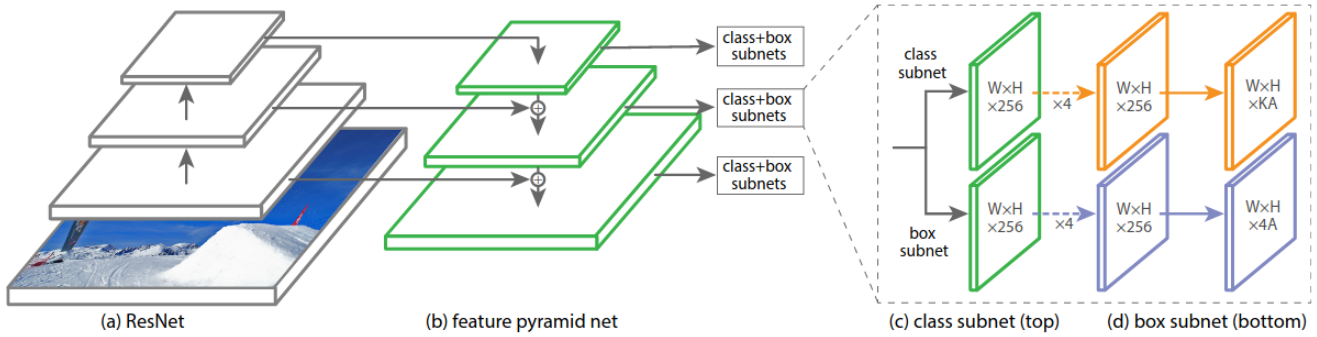


Рисунок 3.1 - Архітектура RetinaNet

3.2.1 Feature Pyramid Network

Мережа Feature Pyramid Network приймає на вхід одно масштабне зображення будь якого розміру та на вихід віддає пропорційні по розміру матриці фіч. Після цього матриця поєднується з відповідною нижньою матрицею (що пройшла ConvNet[41] згортковий шар 1 × 1 для зменшення розмірності) шляхом поелементного додавання. Цей процес повторюється до тих пір, поки не буде згенерована матриця найбільшої роздільною здатністю (рис. 3.2)[17].

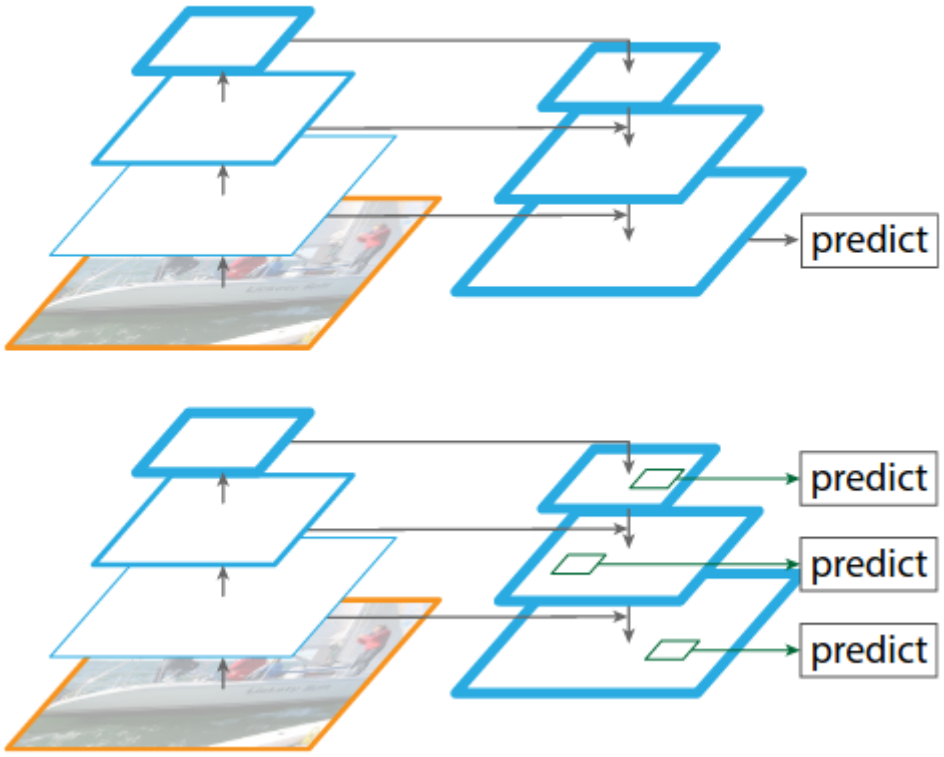


Рисунок 3.2 - Пірамідальна архітектура з пропуском з’єднань, де результат визначається на найкращому рівні.

Шлях “знизу вверх” - це обчислення “вперед-назад” в опорній мережі , що визначає ієрархію фіч, що складається з матриць фіч в семи масштабах з шагом масштабування 2. Часто існує багато шарів, що створюють вхідні карти однакового розміру, та ми говоримо, що ці шари знаходяться на одному етапі

мережі. Для нашої піраміди фіч ми визначаємо один рівень піраміди для кожного етапу. Ми вибираємо вихід останнього шару у якості референсного набора матриць фіч, які ми будемо доповнювати для створення піраміди[17].

Шлях “зверху вниз” виділяє фічі більшої роздільної здатності шляхом вибірки просторово більш грубих, але семантично більш сильних матриць фіч з більш високих рівнів піраміди. Ці фічі потім посилюються фічами з шляху “знизу вверх” через бокові зв’язки. Матриця фіч “знизу вверх” має семантику більш низького рівня, але її активація більш точно локалізована (рис. 3.3)[17].

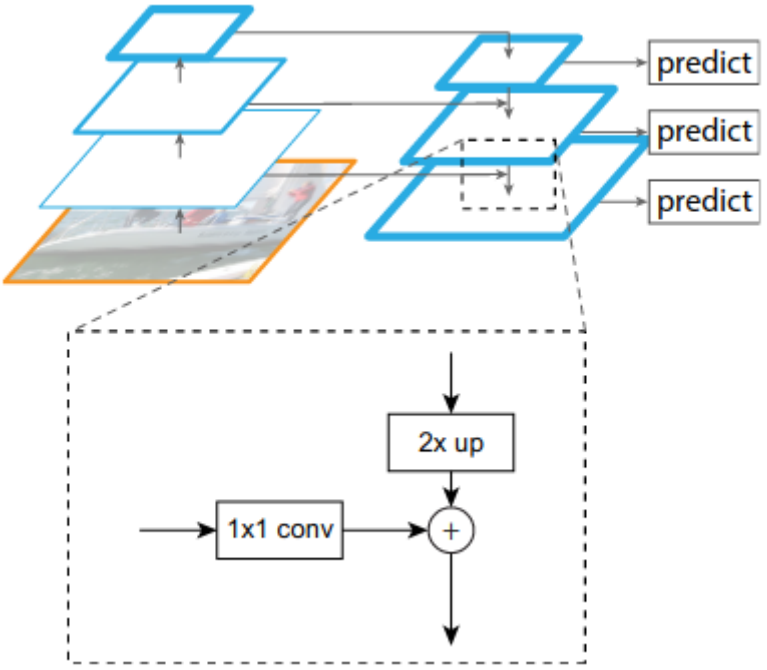


Рисунок 3.3 - блок, що показує латеральний зв’язок та шлях вниз за допомогою об’єднання додаванням.

3.3 Focal Loss

Focal Loss - це функція втрат ентропії з фокусуючим параметром γ (в якості показника)[42]. Або формально:

$$FL(p_t) = -(1 - p_t)^{\gamma} log(p_t) \quad (1)$$

Фокусирующий параметр плавно регулює швидкість, з якою легкі приклади отримують менші ваги. Коли $\gamma = 0$, FL еквівалентно функції втрат ентропії, і зі збільшенням γ ефект модулюючого фактора також збільшується.

3.4 Результати автоматичного визначення автомобільних номерів

Наводжу приклади результатів класифікації зображень з номерами за допомогою моделі навченох на 200 зображеннях мережею RetinaNet.

Рисунок 3.5 - результат визначення автомобільного номеру на зображенні.

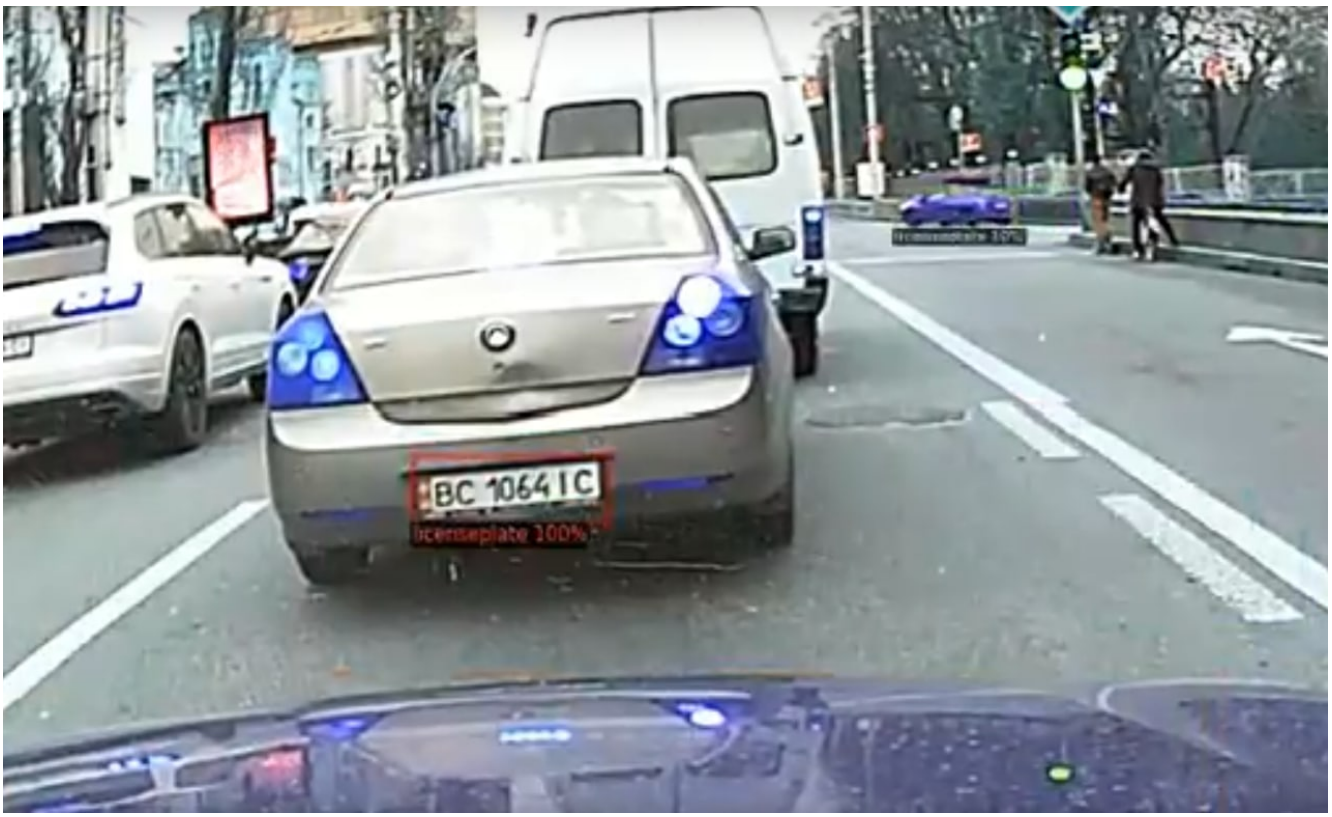


Рисунок 3.6 - результат визначення автомобільного номеру на зображенні.

Рисунок 3.7 - результат визначення автомобільного номеру на зображенні.

3.5 Розпізнавання тексту

Для визначення тексту зображення, що знаходиться в на частині картинки, що була визначена як номер транспортного засобу, перш за все потрібно зробити деякі перетворення.

Перш за все, потрібно привести зображення, до чорно-білого. Після цього ми маємо дещо розмити зображення. Це потрібно для усунення дрібних деталей. Потім ми, власне усуваємо дрібні деталі за допомогою перетворень зображення та виконуємо інверсію по кольору, щоб чорний текст знаходився на білому фоні. Так текст буде простіше розпізнати.

Рисунок 3.8 - оригінальний номер транспортного засобу.

Рисунок 3.9 - чорно-білий номер транспортного засобу.

Рисунок 3.10 - розмитий номер транспортного засобу.

Рисунок 3.11 - номер транспортного засобу після перетворень для усунення шуму.

Рисунок 3.12 - інвертований номер транспортного засобу.

Потім, ми можемо виділити кожен об’єкт букви окремо на картинці.

Рисунок 3.13 - окремі символи номеру транспортного засобу.

Та нарешті розпізнати їх

Рисунок 3.14 - визначений номер транспортного засобу.

3.6 Інформаційна система

В результаті роботи була створена інформаційна система, що здатна визначити за відеозаписом з відеорегістратору, частоту появи тих чи інших транспортних засобів.

Так ми можемо автоматизувати задачу визначення чи не переслідується транспортний засіб, що веде відеозапис іншими транспортними засобами. У випадку, якщо якийсь із транспортних засобів зустрічався занадто часто, ми можемо ідентифікувати його, зберегти та повідомити користувачів інформаційної системи.

3.7 Приклад роботи інформаційної системи

Після обробки відео, ми отримуємо файл у форматі json де для кожного кадру відео вказані: номер кадру, час кадру, номери, що були в кадрі помічені.


```
{'frame_number': 980,
 'frame_time': 24.5,
 'licenseplates': [{'p': 0.5517650383383335, 'text': 'AA6166DR'},
  {'p': 0.8828249172485868, 'text': 'AA8902QT'},
  {'p': 0.967393493617422, 'text': 'AA1702ZN'},
  {'p': 0.9726594365488883, 'text': 'AA5096VK'},
  {'p': 0.6261261983444717, 'text': 'AA9644OZ'}]},
{'frame_number': 981,
 'frame_time': 24.525,
 'licenseplates': [{'p': 0.5713767644745678, 'text': 'AA6166DR'},
  {'p': 0.8828249172485868, 'text': 'AA8902QT'},
  {'p': 0.967393493617422, 'text': 'AA1702ZN'},
  {'p': 0.9726594365488883, 'text': 'AA5096VK'},
  {'p': 0.7261261983444717, 'text': 'AA9644OZ'}]},
{'frame_number': 982,
 'frame_time': 24.55,
 'licenseplates': [{'p': 0.5713767644745678, 'text': 'AA6166DR'},
  {'p': 0.9828249172485868, 'text': 'AA8902QT'},
  {'p': 0.7009440997144949, 'text': 'AA1702ZN'},
  {'p': 0.9726594365488883, 'text': 'AA5096VK'},
  {'p': 0.8261261983444717, 'text': 'AA9644OZ'}]},
```


Рисунок 3.15 - приклад формату в якому ми отримуємо результат.

Потім, ми можемо підбити статистику та дізнатись скільки загалом часу ми могли спостерігати тей чи інший транспортний засіб

	number	frequency	time_total
0	AA5096VK	1316	32.886292
1	AA6166DR	1698	42.432313
2	AA9644OZ	1141	28.513115
3	AA8902QT	1432	35.785083
4	AA1702ZN	1741	43.506865
5	AI5154IH	444	11.095375
6	AA4135AO	695	17.367760

Рисунок 3.16 - час спостереження тих чи інших номерів.

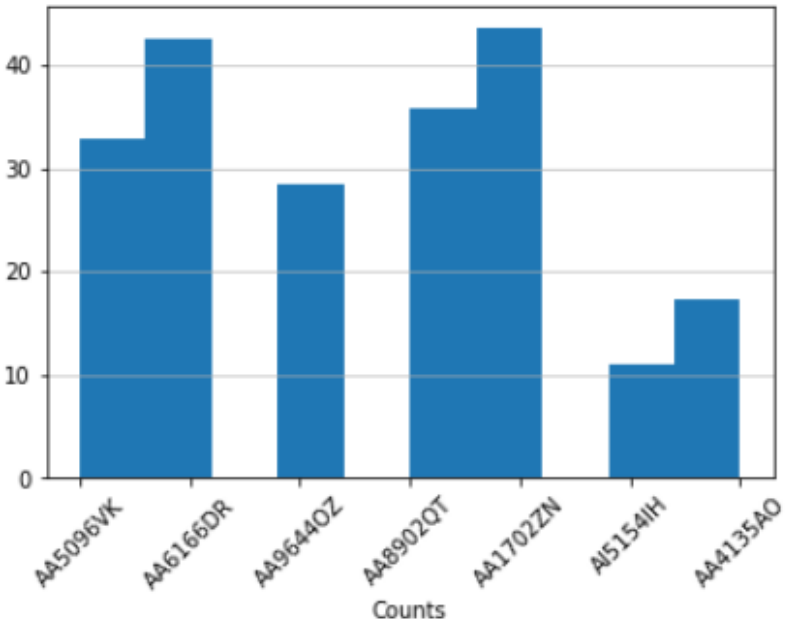


Рисунок 3.17 - гістограма часу спостереження тих чи інших номерів.

3.8 Висновки до розділу 3

В розділі був підсумований вибір моделі RetinaNet для вирішення задачі пошуку автомобільного номеру. Була детально розглянута архітектура мережі RetinaNet. Був розглянутий екстрактор фіч Feature Pyramyd Network, його принцип роботи та особливості, математичний апарат, що забезпечує його роботу. Була

розглянута функція Focal Loss у якості функції втрат для мережі.

Після розгляду всіх теоретичних відомостей, що стосуються безпосередньо RetinaNet, були наведені результати роботи RetinaNet для поставленої задачі пошуку ідентифікаційного номеру транспортного засобу у вигляді контурів номеру на зображенні відеореєстратору автомобілю та ймовірність з якою мережа ідентифікує об'єкт в контурі як автомобільний номер

Далі був детально описаний процес перетворення зображення автомобільного номеру для отримання чіткого зображення символів номеру. Цей процес включає декілька етапів, що наведені у якості малюнків.

В ході роботи була створена інформаційна система, що може визначати з відеозапису визначити, які транспортні засоби були присутні на кожному кадрі відеозапису та агрегувати ці дані. В подальшому на даних, що збережені в json форматі можливо побудувати будь які потрібні статистики.

Інформаційна система може бути дуже корисною для персональної безпеки. Можливо також у реальному часі відстежувати та зберігати дані про автомобілі, що зустрічаються на дорозі та робити висновки про факт переслідування тим чи іншим автомобілем.

В роботі був наведений приклад роботи інформаційної системи на конкретному відео. Було наведено формат в якому зберігаються оброблені дані та приклад гістограми, що була побудована за цими даними.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, розробленого для вирішення задачі “Аналіз та ідентифікація автомобільного трафіку відносно транспортного засобу” за допомогою згорткових нейронних мереж.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для вирішення задачі “Аналіз та ідентифікація автомобільного трафіку відносно транспортного засобу”. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому

фактичний аналіз представляє собою аналіз функцій програмного продукту, для вирішення задачі “Аналіз та ідентифікація автомобільного трафіку відносно транспортного засобу”.

Технічні вимоги до програмного продукту є наступні:

- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту для вирішення задачі “Аналіз та ідентифікація автомобільного трафіку відносно транспортного засобу”. Беручи за основу цю функцію, можна виділити наступні:

- F_1 – вибір мови програмування;
- F_2 – вибір фреймворку машинного навчання;
- F_3 – вибір середовища розробки.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Python

б) C++

Функція F_2:

а) Tensorflow;

б) Caffe

Функція F_3:

а) Jupyter Notebook;

б) Visual Studio.

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

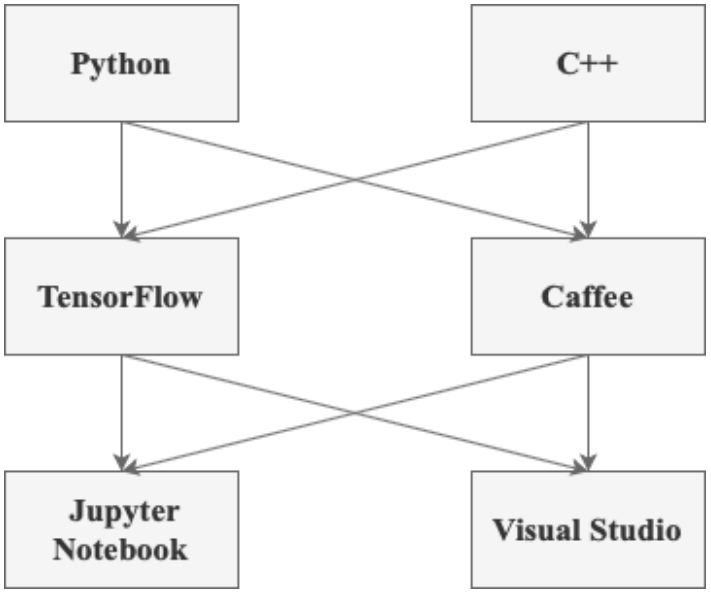


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіанти основних функцій.

Таблиця 4.2 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	A	Швидка розробка програми, доступність бібліотек, кросплатформеність	Низька швидкість роботи, особливо, якщо потрібно обробляти велику кількість даних

	Б	Код швидко виконується	Іде багато часу на розробку програми
F_2	*А*	Надійно працює з складними проектами	Не підтримується багатьма мовами
	Б	Надійність	Додатковий час на інсталяцію та вивчення
F_3	*А*	Підтримується багатьма мовами програмування, легко запускається на будь-якому сервері	Складність розгортання проекту на сервері
	Б	Багато інструментів, безпечна	Підтримує одночасно лише одну мову програмування

На основі цієї карти будуємо позитивно-негативну матрицю варіантів основних функцій (Таб.4.2). Робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1:

Перевагу віддаємо швидкості вивчення, простоті використання та наявності стандартних бібліотек для обчислення. Для спрощення роботи по написанню коду варіант Б має бути відкинутий.

Функція F_2:

Обидва варіанти можна використовувати в розробці.

Функція F_3:

Віддаємо перевагу варіанту А в разі вибору мови програмування Python.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а – F_2а – F_3а

F_1а – F_2б – F_3а

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

4.3 Обґрунтування системи параметрів ПП

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

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

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

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у таблиці 5.4.

Таблиця 4.4 - Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	10000	14000	19000
Об’єм пам’яті	X2	Мб	420	128	64

Час попередньої обробки даних	X3	мс	4	3	2
Потенційний об’єм програмного коду	X4	кількість рядків коду	4000	2500	1000

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

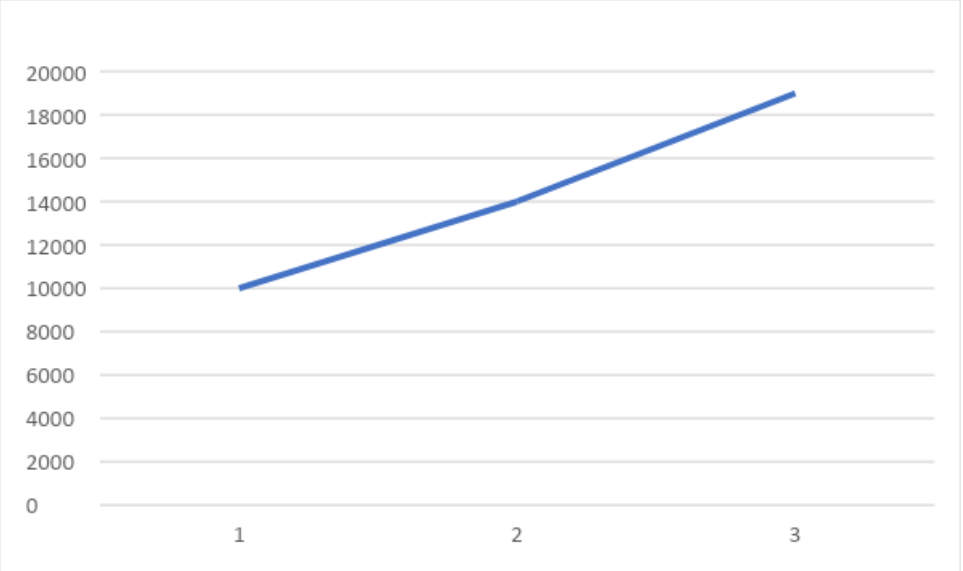


Рисунок 4.2 – X1, швидкодія мови програмування

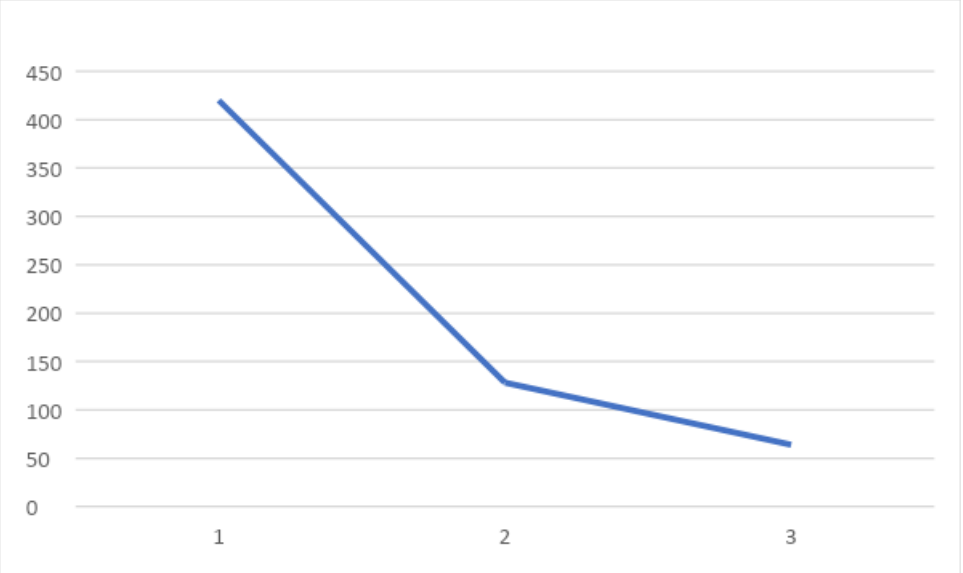


Рисунок 4.3 – X2, об’єм пам’яті

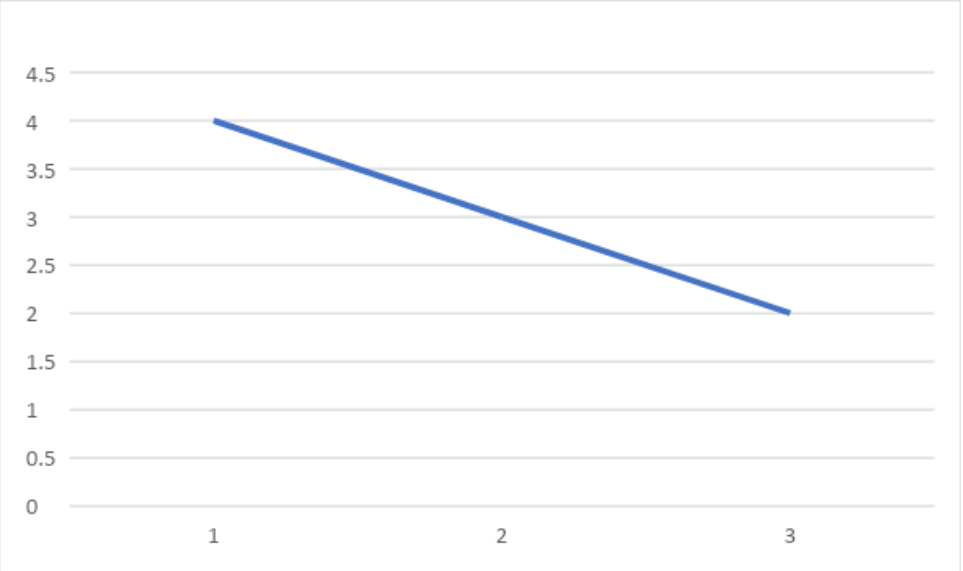


Рисунок 4.4 – X3, час попередньої обробки даних

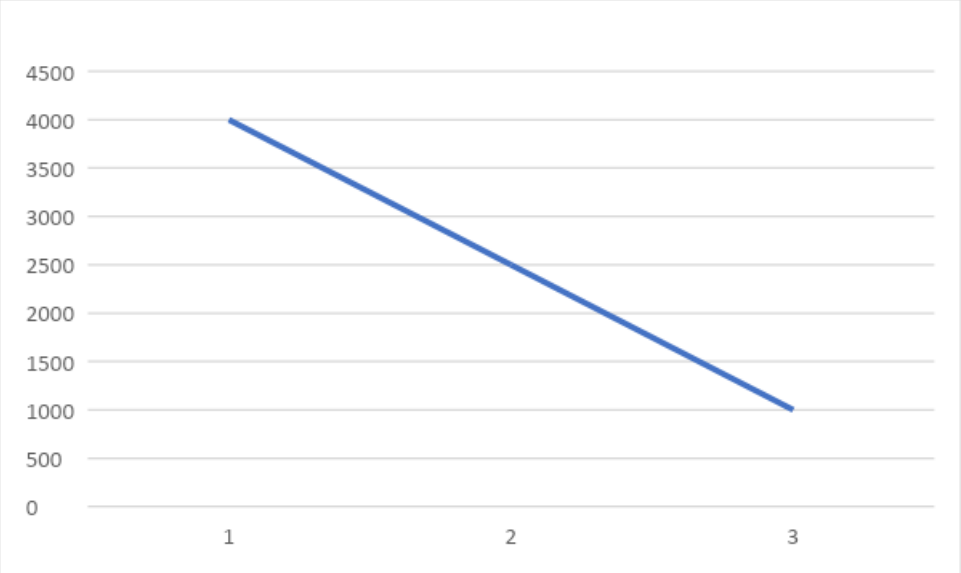


Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

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

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

Результати експертного ранжування наведені у таблиці 5.4.

Таблиця 5.4 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	5	2	5	3	4	5	28	3,5	12,25
X2	Об'єм пам'яті	Mb	2	1	3	1	2	1	2	12	-12,5	156,25
X3	Час попередньої обробки даних	мс	5	3	5	5	4	5	3	30	5,5	30,25
X4	Потенційний об'єм програмного коду	Кількість рядків коду	3	5	4	3	5	4	4	28	3,5	12,25
	Разом		14	14	14	14	14	14	14	98	0	211

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

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

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 98, \qquad (4.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 24,5 \qquad (4.2)$$

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

$$\Delta_i = R_i - T. \qquad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 211. \qquad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 211}{7^2(4^3-4)} = 0,86 > W_k = 0,67. \qquad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.5.

Таблиця 4.5 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	<	<	>	>	>	>	1,5

X1 і X3	<	>	<	=	<	<	>	<	0,5
X1 і X4	>	>	<	=	<	=	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	<	<	<	<	<	<	<	<	0,5
X3 і X4	>	<	>	>	<	>	<	>	1,5

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

$$a_{ij} = \{1.5 \text{ при } X_i > X_j \; 1.0 \text{ при } X_i = X_j \; 0.5 \text{ при } X_i < X_j . \#(4.6)$$

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

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

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \qquad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \qquad (4.8)$$

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

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \qquad (4.9)$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \qquad (4.10)$$

.Як видно з таблиці 4.6, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.6 - Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	1,5	0,5	1,5	4,5	0,36	17,75	0,21	73,38	0,21
X2	0,5	1,0	1,5	0,5	3,5	0,28	15,75	0,26	65,63	0,27
X3	1,5	1,5	1,0	1,5	5,5	0,44	22,75	0,2	93,6	0,2
X4	0,5	1,5	0,5	1,0	3,5	0,28	13,75	0,33	57,63	0,32
Всього:					12,5	1	70	1	290,25	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X2 (Об’єм пам’яті), X3 (час попередньої обробки даних) та X4 (потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X1 (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 5.7):

$$(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j} \qquad (4.11)$$

де n – кількість параметрів;

$K_{\text{в}i}$ – коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 5.7 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	10000	6	0,21	1,1
F2	А	X2	64	5	0,27	1,55
	Б	X2	128	2	0,2	0,24
F3	А	X3	1000	8	0,32	1,8

За даними з таблиці 4.7 за формулою:

$$K_K = K_{\text{ТУ}}\left[F_{1k}\right] + K_{\text{ТУ}}\left[F_{2k}\right] + \ldots + K_{\text{ТУ}}\left[F_{zk}\right], \qquad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1{,}1 + 1{,}55 + 1{,}8 = 4{,}45,$$

$$K_{K2} = 1{,}1 + 0{,}24 + 1{,}8 = 3{,}14.$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як

$$T_О = T_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \qquad (5.13)$$

де $T_Р$ – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_Р$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_Р$ =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

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

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

$$T_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328{,}64 \text{ людино-годин.}$$

$$T_{II} = (122.4 + 19.44 + 6.91 + 19.44) \cdot 8 = 1345.52 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 20000 грн., один аналітик в області даних з окладом 22500. Визначимо середню зарплату за годину за формулою:

$$СЧ = \frac{М}{T_m \cdot t} \text{грн.,} \quad \#(5.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$СЧ = \frac{20000+20000+22500}{3 \cdot 21 \cdot 8} = 124,00 \text{ грн.} \quad \#(5.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$СЗП = С_ч \cdot Т_i \cdot КД, \#(5.16)$$

де $С_ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 124{,}00 \cdot 1328.64 \cdot 1.2 = 197606{,}40$ грн.

II. $С_{ЗП} = 124{,}00 \cdot 1345.52 \cdot 1.2 = 200213{,}37$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 197606{,}40 \cdot 0.22 = 43473{,}40$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 200213{,}37 \cdot 0.22 = 44046{,}94$ грн.

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

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

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 20000 \cdot 0{,}2 = 48000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 48000 \cdot (1 + 0.2) = 57600 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД}= С_{ЗП} \cdot 0.22 = 57600 \cdot 0,22 =12672 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕО М – 20000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 20000 = 5750 \text{ грн.},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 20000 \cdot 0.05 = 1150 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} =(Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.9 =$$

$$= 1677.6 \text{ годин},,$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1677,6 \cdot 0,3 \cdot 0,9 \cdot 3,51 = 1592,34 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 20000 \cdot 0,67 = 13400 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \qquad (5.17)$$

$С_{ЕКС} = 57600 + 12672 + 10589,76 + 5750 + 1150 + 1592,34 + 13400 = 102754,10$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 57932,27 / 1677.6 = 34,53 \text{ грн/год.}$$

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

$$C_М = C_{М-Г} \cdot T, \#(5.18)$$

I. $C_М = 34{,}53 \cdot 1328{,}64 = 45881{,}69$ грн.

II. $C_М = 34{,}53 \cdot 1345.52 = 46464{,}61$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0,67, \#(5.19)$$

I. $C_Н = 102811{,}43 \cdot 0{,}67 = 68883{,}66$ грн.

II. $C_Н = 104117{,}62 \cdot 0{,}67 = 69758{,}80$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н, \#(5.20)$$

I. $C_{ПП} = 102811{,}43 + 22618{,}51 + 45881{,}69 + 68883{,}66 = 240195{,}29$ грн.

II. $C_{ПП} = 104117{,}62 + 22905{,}88 + 46464{,}61 + 69758{,}80 = 243246{,}91$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР_j} = К_{К_j}/С_{Ф_j}, \#(5.21)$$

$$К_{ТЕР1} = 4{,}25/\ 240195{,}29 = 1{,}77 \cdot 10^{-5},$$

$$К_{ТЕР2} = 3{,}74/\ 243246{,}91 = 1{,}53 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 1{,}77 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості

$К_{ТЕР} = 1{,}77 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Python;
- Використання моделей з великою ємністю
- Використання стандартного інтерфейсу візуалізації, швидкість розробки

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

4.8 Висновки до розділу 4

Отже враховуючи всі дослідження, що описані вище, можна сказати, що перший варіант реалізації є найбільш оптимальним зі сторони якісноекономічної оцінки. Його коефіцієнт техніко-економічного рівня складає $1{,}77 \cdot 10^{-5}$. Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Python;
- використання готових бібліотек;
- використання фреймворку detectron2.

Цей варіант виконання програмного продукту дозволяє побудувати ефективну й надійну систему із зручним інтерфейсом, широким функціоналом та швидкодією.

Після детального обговорення й аналізу було оцінено ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Проведено повний функціонально-вартісний аналіз програмного продукту. Визначено та проведено оцінку основних функцій програмного продукту. Визначено параметри, які характеризують програмний продукт. Проведено експертне оцінювання параметрів та аналіз якості варіантів реалізації функцій.

Проведено економічний аналіз варіантів розробки – трудомісткість, витрати на заробітну плату та інші витрати.

На основі аналізу вибрано остаточний варіант реалізації програмного продукту.

ВИСНОВКИ

У даній роботі була створена інформаційна система, що може бути корисна для систем персональної безпеки. Система здатна виявляти та ідентифікувати

транспортні засоби, що були присутніми на відеозапису з відорегістратору автомобілю. Для цього була використана мова програмування Python та фреймворк detectron2. Для відображення даних були використані бібліотеки pandas та matplotlib.

У першому були розглянуті такі теми як актуальність створеного програмного програмного продукту на сучасному ринку, задача ідентифікації автомобільного номеру, історичні відомості про неї, етапи та варіанти її вирішення.

В другому розділі були проаналізовані згорткові нейронні мережі як метод вирішення даної задачі. Були наведені відомості як про згорткові мережі в загальному, так і про поширені та визнані ефективними конкретні мережі. Була розглянута архітектура згорткових нейронних мереж. Також, було проведено порівняння різних згорткових нейронних мереж та вибраний конкретний варіант для вирішення поставленої задачі.

У третьому розділі була більш детально розглянута архітектура та математичний апарат вибраної мережі. Було наведено як результати роботи RetinaNet для задачі ідентифікації автомобільного номеру, так і результати результати інформаційної системи.

Побудована система є повноцінним програмним продуктом, що може бути використаний як для персонального використання так і для бізнесу.

Шляхами до до розвитку можна вважати створення більш гнучкого інтерфейсу для побудови статистик по агрегованим даним та покращення роботи моделі.

ЛІТЕРАТУРА

1. An Introduction to ANPR. [Електронний ресурс] – Режим доступу: https://www.cctv-information.co.uk/article/an-introduction-to-anpr/

2. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. [Електронний ресурс] – Режим доступу: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

3. R-FCN: Object Detection viaRegion-based Fully Convolutional Networks. [Електронний ресурс] – Режим доступу: https://papers.nips.cc/paper/2016/file/577ef1154f3240ad5b9b413aa7346a1e-Paper.pdf

4. SSD: Single Shot MultiBox Detector. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/1512.02325.pdf

5. Feature Pyramid Networks for Object Detection. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1612.03144

6. Focal Loss for Dense Object Detection. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1708.02002

7. You Only Look Once: Unified, Real-Time Object Detection. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1506.02640

8. Evaluating Object Detection Models Using Mean Average Precision . [Електронний ресурс] – Режим доступу:

https://blog.paperspace.com/mean-average-precision/

9. Privacy Tech-Know Blog: Who's Watching Where You're Driving [Електронний ресурс] – Режим доступу:https://www.priv.gc.ca/en/blog/20170613/

10. Denmark: Targeted ANPR data retention turned into mass surveillance. [Електронний ресурс] – Режим доступу:https://edri.org/denmark-targeted-anpr-data-retention-turned-into-mass-surveillance/

11. The History of ANPR. [Електронний ресурс] – Режим доступу:http://www.anpr-international.com/history-of-anpr/

12. Car License Plate Recognition. [Електронний ресурс] – Режим доступу: http://visl.technion.ac.il/projects/2002w02/

13. An Introduction to Convolutional Neural Networks. [Електронний ресурс] – Режим доступу:https://www.politie.be/5906/

14. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

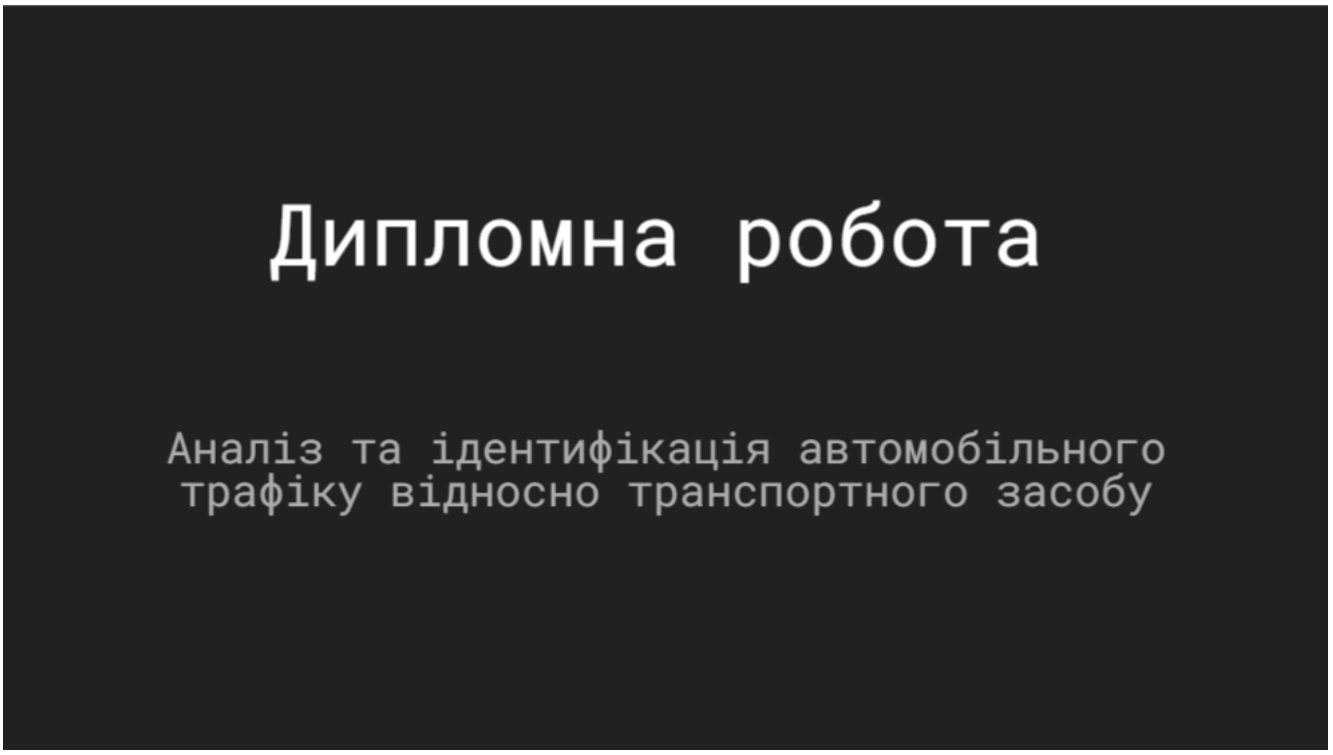
15. Speed/accuracy trade-offs for modern convolutional object detectors. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/1611.10012.pdf

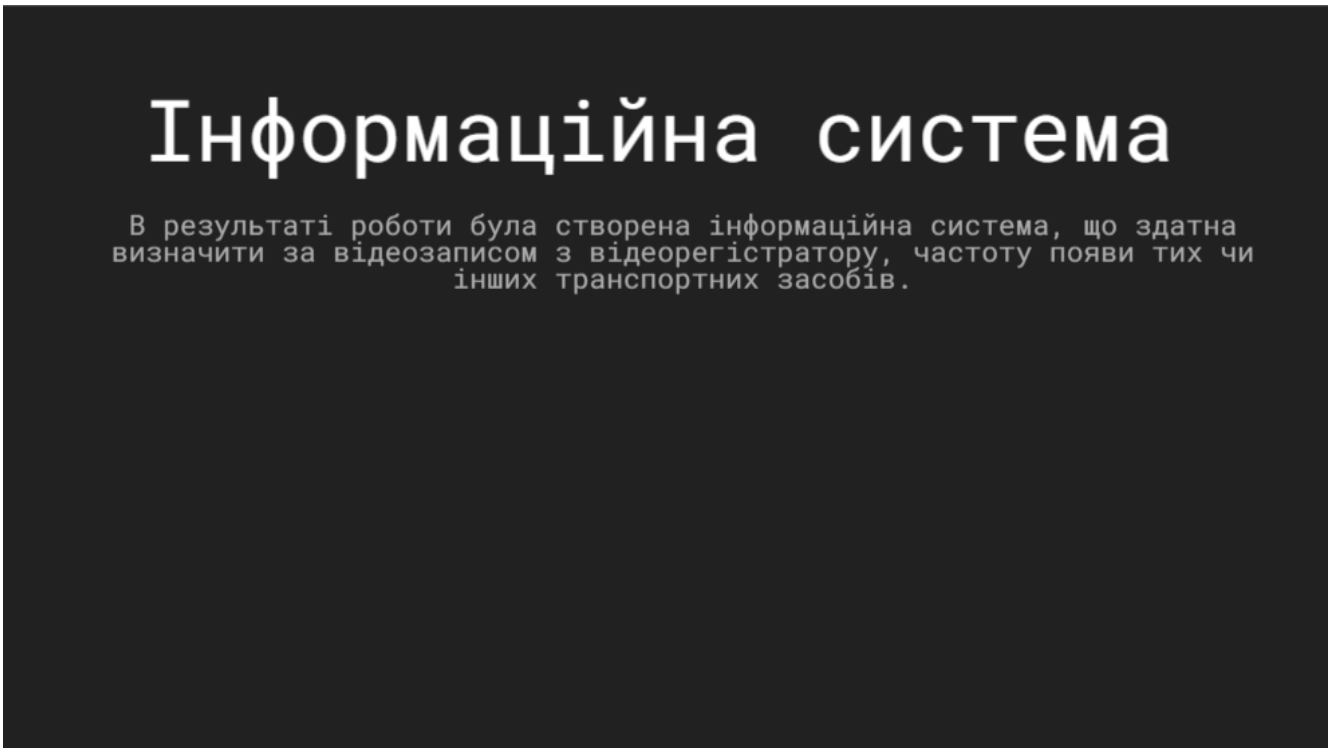
16. Focal Loss for Dense Object Detection. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1708.02002
17. Feature Pyramid Networks for Object Detection. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1612.03144
18. Using the CPU and GPU for Real-Time Video Enhancement on Mobile Computer [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/224202671_Using_the_CPU_and_GPU_for_real-time_video_enhancement_on_a_mobile_computer
19. High-speed image acquisition with GPU processing. [Електронний ресурс] – Режим доступу: https://www.activesilicon.com/resources/high-speed-image-acquisition-with-gpu-processing/
20. Deep Learning-based Real-time Video Processing. [Електронний ресурс] – Режим доступу: https://www.kdnuggets.com/2021/02/deep-learning-based-real-time-video-processing.html
21. Feature Selection and Feature Extraction in Pattern Analysis: A Literature Review [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/1905.02845.pdf
22. Recent Advances in Convolutional Neural Networks. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/1512.07108.pdf%C3%A3%E2%82%AC%E2%80%9A
23. Convolutional neural networks: an overview and application in radiology. [Електронний ресурс] – Режим доступу: https://insightsimaging.springeropen.com/track/pdf/10.1007/s13244-018-0639-9.pdf
24. Convolutional Layer. [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/topics/mathematics/convolutional-layer

25. Convolutional networks and applications in vision. [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/abstract/document/5537907
26. What is the best multi-stage architecture for object recognition? [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/abstract/document/5459469
27. Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair. [Електронний ресурс] – Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.165.6419
28. ImageNet Classification with Deep Convolutional Neural Networks. [Електронний ресурс] – Режим доступу: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html
29. Going Deeper With Convolutions. [Електронний ресурс] – Режим доступу: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html
30. Deep Sparse Rectifier Neural Networks. [Електронний ресурс] – Режим доступу: http://proceedings.mlr.press/v15/glorot11a.html
31. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1502.01852
32. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. [Електронний ресурс] – Режим доступу: https://link.springer.com/chapter/10.1007/978-3-642-15825-4_10
33. Fractional Max-Pooling. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1412.6071
34. Striving for Simplicity: The All Convolutional Net. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1412.6806
35. Pooling. [Електронний ресурс] – Режим доступу: http://ufldl.stanford.edu/tutorial/supervised/Pooling/

36. Convolutional Neural Networks (CNNs / ConvNets). [Електронний ресурс] – Режим доступу: https://cs231n.github.io/convolutional-networks/#fc
37. One by One [1 x 1] Convolution - counter-intuitively useful. [Електронний ресурс] – Режим доступу: https://iamaaditya.github.io/2016/03/one-by-one-convolution/
38. Convolutional Neural Networks cheatsheet. [Електронний ресурс] – Режим доступу: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks
39. PASCAL VOC 2007. [Електронний ресурс] – Режим доступу: https://paperswithcode.com/dataset/pascal-voc-2007
40. Deep Residual Learning for Image Recognition. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1512.03385
41. ImageNet Classification with Deep Convolutional Neural Networks. [Електронний ресурс] – Режим доступу: https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
42. Focal Loss. [Електронний ресурс] – Режим доступу: https://paperswithcode.com/method/focal-loss

ДОДАТОК А ІЛЮСТРОВАНИЙ МАТЕРІАЛ

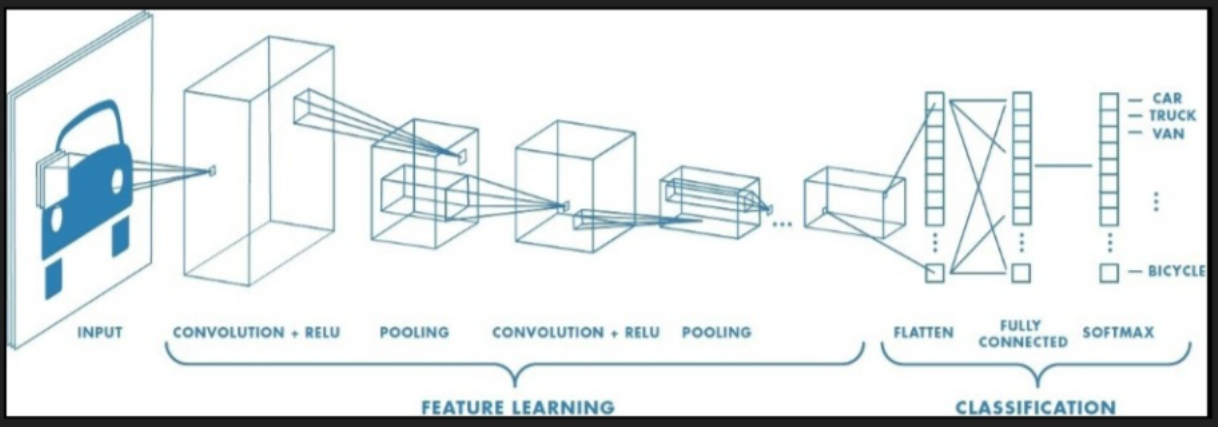




Початкова задача

Першочергово потрібно визначити положення номерних знаків

Згорткові нейронні мережі



RetinaNet

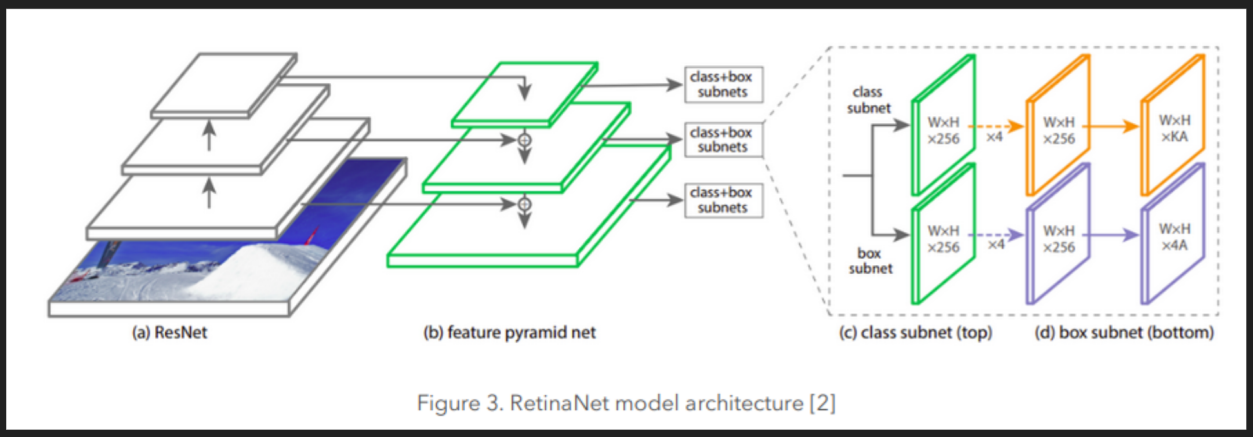


Figure 3. RetinaNet model architecture [2]

Результати пошуку номерних знаків

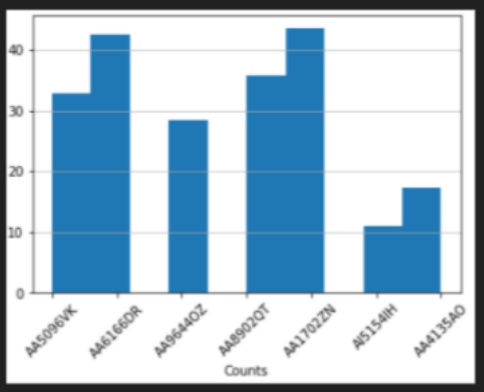
Результати пошуку номерних знаків

Tesseract

Text recognition result

Інформаційна система

	number	frequency	time_total
0	AA5096VK	1316	32.886292
1	AA6166DR	1698	42.432313
2	AA9644OZ	1141	28.513115
3	AA8902QT	1432	35.785083
4	AA1702ZN	1741	43.506865
5	AI5154IH	444	11.095375
6	AA4135AO	695	17.367760



Дякую за увагу

ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ

```
from detectron2 import model_zoo
from detectron2.config import get_cfg
from detectron2.engine import DefaultTrainer
from fvcore.common.file_io import PathManager
import os
import numpy as np
import xml.etree.ElementTree as ET

import pytesseract
from detectron2.structures import BoxMode
from detectron2.data import DatasetCatalog, MetadataCatalog
from detectron2.utils.visualizer import ColorMode
from google.colab.patches import cv2_imshow
from detectron2.engine import DefaultPredictor
from detectron2.utils.visualizer import Visualizer
import cv2
def load_voc_instances(dirname: str, split: str):
    """
    Load licenseplates VOC detection annotations to Detectron2 format.
    Args:
    dirname: Contain "annotations", "images"
    split (str): one of "train", "test"
    """
```

```
with PathManager.open(os.path.join(dirname, split + ".txt")) as f:
fileids = np.loadtxt(f, dtype=np.str)

dicts = []
for fileid in fileids:
anno_file = os.path.join(dirname, "annotations", fileid + ".xml")
jpeg_file = os.path.join(dirname, "images", fileid + ".jpg")

tree = ET.parse(anno_file)

r = {
"file_name": jpeg_file,
"image_id": fileid,
"height": int(tree.findall("./size/height")[0].text),
"width": int(tree.findall("./size/width")[0].text),
}
instances = []

for obj in tree.findall("object"):
cls = obj.find("name").text
bbox = obj.find("bndbox")
bbox = [float(bbox.find(x).text) for x in ["xmin", "ymin", "xmax", "ymax"]]
instances.append(
        {"category_id": CLASS_NAMES.index(cls), "bbox": bbox, "bbox_mode":
BoxMode.XYXY_ABS}
```

```
        )
        r["annotations"] = instances
        dicts.append(r)
        return dicts
def register_licenseplates_voc(name1, dirname, split):
        DatasetCatalog.register(name1,
                        lambda: load_voc_instances(dirname, split))
        return MetadataCatalog.get(name1).set(thing_classes=CLASS_NAMES,
                        dirname=dirname,
                        split=split)

name = "licenseplates_train"
test_name = "licenseplates_test"
drive_path = "drive/MyDrive/Diploma"
output_dir = f"{drive_path}/models/model_retina_china"
model_retina1_path = f"{output_dir}/model_final.pth"
video_path = f"{drive_path}/videos/video1.avi"
result_json_path =  f"{drive_path}/videos"
meta = register_licenseplates_voc(name, "drive/MyDrive/Diploma/licenseplates2",
"train")
meta_test = register_licenseplates_voc(test_name,
 "drive/MyDrive/Diploma/licenseplates2", "test")


trainer = Trainer(cfg)
trainer.resume_or_load(resume=False)
```

```
trainer.train()


cfg.MODEL.WEIGHTS = model_retina1_path
cfg.MODEL.RETINANET.SCORE_THRESH_TEST = 0.55   # set a custom testing
threshold
predictor = DefaultPredictor(cfg)
def process_video(path):
  cap = cv2.VideoCapture(path)
  fourcc = cv2.VideoWriter_fourcc(*'DIVX')
  fps = 30
  w, h = 540, 380
      stream   =   cv2.VideoWriter("/content/drive/MyDrive/Diploma/videos/result1.avi",
fourcc, fps, (w, h))
  # Loop untill the end of the video
  i = 0
  bp = 0
  while (cap.isOpened()):
        try:
        # Capture frame-by-frame
        ret, frame = cap.read()
        frame = cv2.resize(frame, (w, h), fx = 0, fy = 0,
                           interpolation = cv2.INTER_CUBIC)


        # using cv2.Gaussianblur() method to blur the video
```

```
            # (5, 5) is the kernel size for blurring.
            predicted = visualize(frame)
            stream.write(predicted)

            # define q as the exit button
            if cv2.waitKey(25) & 0xFF == ord('q'):
            break

            i += 1
            print(i)
            except Exception as e:
            print(bp/i, e)
            bp += 1
            if bp/i > 0.1:
            break
    # release the video capture object
    cap.release()
    stream.release()
    cv2.destroyAllWindows()
    # Closes all the windows currently opened.
    cv2.destroyAllWindows()
def process_video_lc_pls(path):
    cap = cv2.VideoCapture(path)

    fps = cap.get(cv2.CAP_PROP_FPS)
```

```
  w, h = 540, 380
  # Loop untill the end of the video
  i = 0
  bp = 0
  result = []
  while cap.isOpened():
        try:
        frame_exists, curr_frame = cap.read()
        if frame_exists:
        result.append({
        'frame_number': i,
        'frame_time': round(i/fps, 10),
        'licenseplates': get_licenceplates(curr_frame)
        })
        print(f'frame {i}')
        i+=1
        else:
        break
        except Exception as e:
        print(f'exception {e}')
  cap.release()
  return result
def format_image(image):
  # cv2_imshow(image)
  grayscale_resize_test_license_plate = cv2.cvtColor(
```

```
        image, cv2.COLOR_BGR2GRAY)
# cv2_imshow(grayscale_resize_test_license_plate)
gaussian_blur_license_plate = cv2.GaussianBlur(
        grayscale_resize_test_license_plate, (5, 5), 0)
# cv2_imshow(gaussian_blur_license_plate)
ret, thresh = cv2.threshold(gaussian_blur_license_plate,
                            0, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY_INV)
# cv2_imshow(thresh)
rect_kern = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
# dilation = cv2.dilate(thresh, rect_kern, iterations=1)
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)
sorted_contours = sorted(contours, key=lambda ctr: cv2.boundingRect(ctr)[0])
symbols = []

# factored = cv2.bitwise_not(thresh)
# factored = cv2.medianBlur(factored, 5)
# cv2_imshow(factored)
print(f'{len(contours)} got')
for cnt in sorted_contours:
        x,y,w,h = cv2.boundingRect(cnt)
        height, width = image.shape[:2]
        roi = thresh[y-5:y+h+5, x-5:x+w+5]
        roi = cv2.bitwise_not(roi)
        roi = cv2.medianBlur(roi, 5)
```

```
        if roi is None: continue
        cv2_imshow(roi)
        area = h * w
        # if area < 100:
        #   print(f'skipped by area {area}')
        #   continue
        area_ratio = area/(height*width)
        if not (0.02 < area_ratio < 0.4):
        # print(f'skipped by area_ratio {area_ratio}')
        continue

        sides_ratio = w/h
        if not (0.25 < sides_ratio < 4):
        # print(f'skipped by sides_ratio {sides_ratio}')
        continue
        # print(f'passed {round(area_ratio, 4)} {round(sides_ratio, 3)}')
        symbols.append(roi)

  # print(f'{len(symbols)} passed\n '+ '-' * 30 +'\n')
  return symbols
  # return gaussian_blur_license_plate
def get_text(img, bb):
  bb = [int(i) for i in bb]
  plate_box = img[bb[1]:bb[3], bb[0]:bb[2]]
  symbols = format_image(plate_box)
```

```
  result = ""
  for s in symbols:
        result += pytesseract.image_to_string(s, lang ='eng',
        config ='--oem 3 --psm 6 -c tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789')
  result = result.replace("\n", "").replace(" ", "").upper()
  # cv2_imshow(plate_box)
  return result
def get_licenceplates(res):
  grey_scaled = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
  gray_rgb = cv2.cvtColor(grey_scaled,cv2.COLOR_GRAY2RGB)
  outputs = predictor(gray_rgb)
  instances = outputs['instances'].get_fields()
  boxes = instances['pred_boxes'].tensor.tolist()
  scores = instances['scores'].tolist()
  texts = [{'text': get_text(res, b), 'p': s} for b, s in zip(boxes, scores)]
  return texts
def visualize(res ,show=False):
  grey_scaled = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
  gray_rgb = cv2.cvtColor(grey_scaled,cv2.COLOR_GRAY2RGB)
  outputs = predictor(gray_rgb)  # format is documented at
  boxes = outputs['instances'].get_fields()['pred_boxes'].tensor.tolist()
  for b in boxes:
        get_text(res, b)
  # v = Visualizer(res[:, :, ::-1],
```

```
  #                 scale=1,
  #                 instance_mode=ColorMode.IMAGE_BW,
  #                 metadata=meta
  # )
  # out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
  # if not show:
  #   cv2_imshow(out.get_image()[:, :, ::-1])
  return out.get_image()
def predict_all_by_path(path):
  files = os.listdir(path)
  for file in files:
          im = cv2.imread(f'{path}/{file}')
          visualize(im, True)
 res = process_video_lc_pls(f'{result_json_path}/sample.mp4')

import json
with open(f'{result_json_path}/sample.json', 'w') as f:
  json.dump(res, f)
predict_all_by_path(f'{drive_path}/example_images')
import logging
import numpy as np
import os
import tempfile
import xml.etree.ElementTree as ET
from collections import OrderedDict, defaultdict
```

```
from functools import lru_cache
import torch

from detectron2.data import MetadataCatalog
from detectron2.utils import comm

from detectron2.evaluation.evaluator import DatasetEvaluator

class Trainer(DefaultTrainer):
        @classmethod
        def build_evaluator(cls, cfg, dataset_name):
        return VOCDetectionEvaluator(dataset_name)

class VOCDetectionEvaluator(DatasetEvaluator):

        def __init__(self, dataset_name):
        """
        Args:
        dataset_name (str): name of the dataset, e.g., "dataset_test"
        """
        self._dataset_name = dataset_name
        meta = MetadataCatalog.get(dataset_name)
        self._anno_file_template = os.path.join(meta.dirname, "annotations", "{}.xml")
        self._image_set_path = os.path.join(meta.dirname, meta.split + ".txt")
        self._class_names = meta.thing_classes
```

```
self._cpu_device = torch.device("cpu")
self._logger = logging.getLogger(__name__)

def reset(self):
self._predictions = defaultdict(list)  # class name -> list of prediction strings

def process(self, inputs, outputs):
for input, output in zip(inputs, outputs):
image_id = input["image_id"]
instances = output["instances"].to(self._cpu_device)
boxes = instances.pred_boxes.tensor.numpy()
scores = instances.scores.tolist()
classes = instances.pred_classes.tolist()
for box, score, cls in zip(boxes, scores, classes):
        xmin, ymin, xmax, ymax = box
        self._predictions[cls].append(
        f"{image_id} {score:.3f} {xmin:.1f} {ymin:.1f} {xmax:.1f} {ymax:.1f}"
        )

def evaluate(self):
"""
Returns:
dict: has a key "segm", whose value is a dict of "AP", "AP50", and "AP75".
"""
all_predictions = comm.gather(self._predictions, dst=0)
```

```
        if not comm.is_main_process():
        return
        predictions = defaultdict(list)
        for predictions_per_rank in all_predictions:
        for clsid, lines in predictions_per_rank.items():
                predictions[clsid].extend(lines)
        del all_predictions


        self._logger.info("Evaluating {}. ".format(self._dataset_name))


        with tempfile.TemporaryDirectory(prefix=f"{self._dataset_name}_voc_eval_") as
dirname:
        res_file_template = os.path.join(dirname, "{}.txt")


        aps = defaultdict(list)  # iou -> ap per class
        for cls_id, cls_name in enumerate(self._class_names):
                lines = predictions.get(cls_id, [""])


                with open(res_file_template.format(cls_name), "w") as f:
                f.write("\n".join(lines))


                for thresh in range(50, 100, 5):
                rec, prec, ap = voc_eval(
                res_file_template,
                self._anno_file_template,
```

```
                self._image_set_path,
                cls_name,
                ovthresh=thresh / 100.0
                )
                aps[thresh].append(ap * 100)

        ret = OrderedDict()
        mAP = {iou: np.mean(x) for iou, x in aps.items()}
        ret["bbox"] = {"AP": np.mean(list(mAP.values())), "AP50": mAP[50], "AP75": mAP[75]}
        return ret


##############################################################################
########
#
# Below code is modified from
# https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/datasets/voc_eval.py
# --------------------------------------------------------
# Fast/er R-CNN
# Licensed under The MIT License [see LICENSE for details]
# Written by Bharath Hariharan
# --------------------------------------------------------

"""Python implementation of the PASCAL VOC devkit's AP evaluation code."""
```

```
@lru_cache(maxsize=None)
def parse_rec(filename):
        """Parse a PASCAL VOC xml file."""
        tree = ET.parse(filename)
        objects = []
        for obj in tree.findall("object"):
        obj_struct = {}
        obj_struct["name"] = obj.find("name").text
        obj_struct["pose"] = obj.find("pose").text
        obj_struct["truncated"] = int(obj.find("truncated").text)
        obj_struct["difficult"] = int(obj.find("difficult").text)
        bbox = obj.find("bndbox")
        obj_struct["bbox"] = [
        float(bbox.find("xmin").text),
        float(bbox.find("ymin").text),
        float(bbox.find("xmax").text),
        float(bbox.find("ymax").text),
        ]
        objects.append(obj_struct)

        return objects


def voc_ap(rec, prec):
```

```
        """Compute VOC AP given precision and recall."""

        # correct AP calculation
        # first append sentinel values at the end
        mrec = np.concatenate(([0.0], rec, [1.0]))
        mpre = np.concatenate(([0.0], prec, [0.0]))

        # compute the precision envelope
        for i in range(mpre.size - 1, 0, -1):
        mpre[i - 1] = np.maximum(mpre[i - 1], mpre[i])

        # to calculate area under PR curve, look for points
        # where X axis (recall) changes value
        i = np.where(mrec[1:] != mrec[:-1])[0]

        # and sum (\Delta recall) * prec
        ap = np.sum((mrec[i + 1] - mrec[i]) * mpre[i + 1])

        return ap


def voc_eval(detpath, annopath, imagesetfile, classname, ovthresh=0.5):
        """rec, prec, ap = voc_eval(detpath,
                                annopath,
                                imagesetfile,
```

```
        classname,
        [ovthresh])
Top level function that does the PASCAL VOC evaluation.
detpath: Path to detections
detpath.format(classname) should produce the detection results file.
annopath: Path to annotations
annopath.format(imagename) should be the xml annotations file.
imagesetfile: Text file containing the list of images, one image per line.
classname: Category name (duh)
[ovthresh]: Overlap threshold (default = 0.5)
"""
# assumes detections are in detpath.format(classname)
# assumes annotations are in annopath.format(imagename)
# assumes imagesetfile is a text file with each line an image name

# first load gt
# read list of images
with open(imagesetfile, "r") as f:
lines = f.readlines()
imagenames = [x.strip() for x in lines]

# load annots
recs = {}
for imagename in imagenames:
recs[imagename] = parse_rec(annopath.format(imagename))
```

```
        # extract gt objects for this class
        class_recs = {}
        npos = 0
        for imagename in imagenames:
        R = [obj for obj in recs[imagename] if obj["name"] == classname]
        bbox = np.array([x["bbox"] for x in R])
        difficult = np.array([x["difficult"] for x in R]).astype(np.bool)
        # difficult = np.array([False for x in R]).astype(np.bool)  # treat all "difficult" as
GT
        det = [False] * len(R)
        npos = npos + sum(~difficult)
        class_recs[imagename] = {"bbox": bbox, "difficult": difficult, "det": det}

        # read dets
        detfile = detpath.format(classname)
        with open(detfile, "r") as f:
        lines = f.readlines()

        splitlines = [x.strip().split(" ") for x in lines]
        image_ids = [x[0] for x in splitlines]
        confidence = np.array([float(x[1]) for x in splitlines])
        BB = np.array([[float(z) for z in x[2:]] for x in splitlines]).reshape(-1, 4)

        # sort by confidence
```

```
sorted_ind = np.argsort(-confidence)
BB = BB[sorted_ind, :]
image_ids = [image_ids[x] for x in sorted_ind]

# go down dets and mark TPs and FPs
nd = len(image_ids)
tp = np.zeros(nd)
fp = np.zeros(nd)
for d in range(nd):
R = class_recs[image_ids[d]]
bb = BB[d, :].astype(float)
ovmax = -np.inf
BBGT = R["bbox"].astype(float)

if BBGT.size > 0:
# compute overlaps
# intersection
ixmin = np.maximum(BBGT[:, 0], bb[0])
iymin = np.maximum(BBGT[:, 1], bb[1])
ixmax = np.minimum(BBGT[:, 2], bb[2])
iymax = np.minimum(BBGT[:, 3], bb[3])
iw = np.maximum(ixmax - ixmin + 1.0, 0.0)
ih = np.maximum(iymax - iymin + 1.0, 0.0)
inters = iw * ih
```

```
# union
uni = (
        (bb[2] - bb[0] + 1.0) * (bb[3] - bb[1] + 1.0)
        + (BBGT[:, 2] - BBGT[:, 0] + 1.0) * (BBGT[:, 3] - BBGT[:, 1] + 1.0)
        - inters
)

overlaps = inters / uni
ovmax = np.max(overlaps)
jmax = np.argmax(overlaps)

if ovmax > ovthresh:
if not R["difficult"][jmax]:
        if not R["det"][jmax]:
        tp[d] = 1.0
        R["det"][jmax] = 1
        else:
        fp[d] = 1.0
else:
fp[d] = 1.0

# compute precision recall
fp = np.cumsum(fp)
tp = np.cumsum(tp)
rec = tp / float(npos)
```

```
# avoid divide by zero in case the first detection matches a difficult
# ground truth
prec = tp / np.maximum(tp + fp, np.finfo(np.float64).eps)
ap = voc_ap(rec, prec)

return rec, prec, ap
```