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

Інститут телекомунікаційних систем

Кафедра Інформаційно-телекомунікаційних мереж

«На правах рукопису»

УДК _____

«До захисту допущено»

В.о. завідувача кафедри

_____ Лариса ГЛОБА

«___»_____2021 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-науковою програмою «Інформаційно-комунікаційні технології»
зі спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Система інформаційної підтримки транспортного потоку для проекту розумного міста»

Виконала:
студентка VI курсу, групи ТІ-91мн
Миніч Марина Адамівна _____

Керівник:
Доцент кафедри ІТМ, к.т.н., доцент,
Могильний Сергій Борисович _____

Рецензент:
Доцент кафедри РОС РТФ, к.т.н., доцент
Мосійчук Віталій Сергійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2021 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
Кафедра Інформаційно-телекомунікаційних мереж

Рівень вищої освіти – другий (магістерський)

Спеціальність – 172 Телекомунікації та радіотехніка

Освітньо-наукова програма «Інформаційно-комунікаційні технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Лариса ГЛОБА
«___»_____2021 р.

ЗАВДАННЯ
на магістерську дисертацію студентці
Миніч Марині Адамівні

1. Тема дисертації «Система інформаційної підтримки транспортного потоку для проекту розумного міста», науковий керівник роботи Могильний Сергій Борисович, доцент кафедри інформаційно-телекомунікаційних мереж ІТС, доцент, к.т.н., затверджені наказом по університету від «15» березня 2021 р. №817-с.

2. Термін подання студентом дисертації 11.05.2021 р.

3. Об’єкт дослідження — інтелектуальна система аналізу автомобільного трафіку у місті.

4. Предмет дослідження — система для створення та аналізу відеопослідовності на граничних пристроях системи.

5. Перелік завдань, які потрібно розробити:

- Аналіз проблем розумного міста.
- Дослідити способи детектування та відстеження об’єктів.
- Запропонувати систему для аналізу трафіку транспортних засобів на граничних пристроях.
- Розробити програмне забезпечення для обчислення інтенсивності трафіку.

6. Орієнтовний перелік ілюстративного матеріалу
   1) Тема, актуальність, мета, задачі.
   2) Огляд проблем розумного міста та недоліки хмарних обчислень.
   3) Огляд алгоритмів детектування та трекінгу об’єктів.
   4) Реалізація апаратної частини граничного пристрою.
   5) Створення алгоритму підрахунку транспортних засобів та експериментальне дослідження
   6) Визначення меж використання запропонованої системи інформаційної підтримки.
   7) Загальні висновки.
7. Орієнтовний перелік публікацій:
   - Могильний С.Б., Миніч М.А. Дослідження ефективності методів моніторингу рухомих об’єктів в системах розумного будинку. Матеріали XIV Міжнародної науково-технічної конференції «Перспективи телекомунікацій» (ПТ-20) К: НТУУ «КПІ ім. І. Сікорського», 2020, с. 240-242
   - Могильний С.Б., Миніч М.А. Система аналізу інтенсивності транспортних потоків на Raspberry Pi 4. Priority directions of science and technology development. Abstracts of the 9th International scientific and practical conference. SPC “Sci-conf.com.ua”. Kyiv, Ukraine. 2021.

9. Дата видачі завдання 10 вересня 2019 року.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Проведення аналізу проблем розумного міста.	10.01.2021 – 15.02.2021	виконано
2	Дослідження способів детектування та відстеження об’єктів.	18.01.2021 – 20.03.2021	виконано
3	Розробка системи аналізу трафіку транспортних засобів.	28.03.2021 - 11.04.2021	виконано
4	Розробка програмного забезпечення для обчислення інтенсивності трафіку.	15.04.2020 - 01.05.2020	виконано

Студентка Марина МИНІЧ

Науковий керівник Сергій МОГИЛЬНИЙ

РЕФЕРАТ

Робота містить 70 сторінок, 27 рисунків, 4 формули та 5 таблиць. Використано 36 джерел.

Мета роботи: зменшення кількості трафіку при передачі даних, отриманих за допомогою камер відеоспостереження, до центрального серверу, шляхом попереднього оброблення відеоданих на граничних пристроях - мікрокомп'ютерах.

У роботі проведено аналіз існуючих детекторів, алгоритмів стеження за об’єктами, а також проблем, які можна вирішити шляхом введення автоматизованих систем стеження за транспортом. Запропоновано систему аналізу та обробки відеопослідовності на граничному пристрої. Програмно реалізовано алгоритм визначення інтенсивності трафіку. Зібрано експериментальну установку з мікрокомп’ютера, камери та додаткового модуля глибокого навчання Intel Movidius Neural Compute Stick2. Зроблені висновки та рекомендації за отриманими результатами.

Ключові слова: машинне навчання, комп’ютерний зір, розумне місто, аналіз трафіку.

ABSTRACT

The work contains 70 pages, 27 figures, 4 formulas and 5 tables. 36 sources used.

Purpose: reducing the amount of traffic when transmitting data obtained by video surveillance cameras to a central server, by pre-processing video data on edge devices - microcomputers.

The paper analyzes the existing detectors, object tracking algorithms, as well as problems that can be solved by introducing automated vehicles tracking systems. The system of analysis and processing of video sequence on the boundary device is offered. Software algorithm for determining traffic intensity is implemented. An experimental setup was assembled from a microcomputer, a camera and an Intel Movidius Neural Compute Stick2. Conclusions and recommendations based on the obtained results are done.

Key words: machine learning, computer vision, smart city, traffic analysis.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IoT	Internet of Things
ІКТ	Інформаційно-комунікаційні технології
CNN	Convolutional neural network
R-CNN	Region convolutional neural network
GDPR	General Data Protection Regulation
IDC	International Data Corporation
USB	Universal Serial Bus
YOLO	You Look Only Once
SSD	Single-shot Multibox Detector
mAP	mean Average Precision
FPS	Frames per Second
MS COCO	Microsoft Common Objects in Context
R-FCN	Region-based Fully Convolutional Networks
IoU	Intersection over Union
OpenCV	Open Source Computer Vision

ВСТУП

Актуальність. Концепція «розумного міста» є предметом обговорення протягом багатьох років і багато міст в усьому світі все активніше впроваджують її окремі елементи для управління «розумним містом». «Розумне місто» - це, перш за все, місто, в якому традиційні системи працюють більш ефективно за рахунок використання інформаційно-комунікаційних технологій (ІКТ).

Одним із основних компонентів розумного міста є автоматизоване управління дорожнім рухом. Це технологічне рішення, яке сьогодні можна інтегрувати для швидкого і економічного покращення трафіку транспорту на міських вулицях.

Проблеми, з якими стикаються великі міста, здебільшого добре відомі. Так, збільшення кількості транспортних засобів у них призводить до уповільнення руху, а, отже, до виснажливих заторів, що посилюють забруднення повітря, а також викликають аварії, чим ще більше збільшують затори. Екстрені служби уповільнюють реакцію на виклики, що потенційно загрожує життю людей. Все це разом знижує загальну якість життя в містах.

Система розумного міста функціонує за рахунок безперервної обробки та поновлення даних, що надходять з інформаційних каналів, що уособлюють в собі електронні пристрої (різноманітні датчики, вимірювачі, камери спостереження тощо). Отримані дані можуть використовуватись для моніторингу та автоматичного спрямування трафіку та зменшення завантаженості. Проте існуючі системи виконують обробку даних вже після їх отримання на центральному сервері, що потребує збільшення пропускної здатності каналів, або ж максимального використання їх використання, хоча більшість кадрів не несуть корисної інформації. Тому, для того, щоб

зменшити кількість переданої інформації на центральний сервер, тим самим зменшуючи трафік, пропонується система аналізу транспортних потоків з обробкою інформації на граничних пристроях. Граничним пристроєм в даному випадку є мікрокомп’ютер з під’єднаною камерою та додатковим модулем штучного інтелекту. Мікрокомп’ютер розпізнає транспортний засіб, після чого отримані дані передає на трекер і підраховує кількість транспортних засобів. Отже, передаються на сервер лише дані про інтенсивність трафіку, а не весь відеопотік. Такий підхід дозволяє різко зменшити обсяги переданої інформації, що вже не вимагає надзвичайно високої пропускної здатності каналу зв’язку.

Мета роботи: зменшення кількості трафіку при передачі даних, отриманих за допомогою камер відеоспостереження, до центрального серверу, шляхом попереднього оброблення відеоданих на граничних пристроях - мікрокомп'ютерах.

Для досягнення мети дослідження було поставлено та вирішено такі **основні задачі**:

1. Провести аналіз проблем розумного міста.
2. Дослідити способи детектування та відстеження об’єктів.
3. Запропонувати власну систему аналізу трафіку транспортних засобів.
4. Розробити програмне забезпечення для обчислення інтенсивності трафіку.

Теоретичний результат дослідження:

1. Визначено переваги впровадження запропонованої системи відстеження кількості транспортних засобів на дорогах.
2. Запропоновано апаратну реалізацію даної системи.
3. Визначено межі використання запропонованого підходу до відстеження транспортних засобів.

Практичний результат:

1. Розроблено алгоритм роботи системи відстеження та підрахунку транспортних засобів.
2. Реалізовано запропонований алгоритм та досліджено його роботу.

РОЗДІЛ 1.

АНАЛІЗ ПРОБЛЕМ РОЗУМНОГО МІСТА ТА ПІДХОДИ ЇХ ВИРІШЕННЯ

1.1. Розумне місто та його функції

Великі міста завжди були й залишаються центрами розвитку цивілізації. Перед сучасними містами постає багато викликів і серед них чи не найскладніший - поєднати комфорт та соціальну привабливість для громадян з розвиненою інфраструктурою, екологічною безпекою та швидким розвитком технологій.

Сучасне місто – це рушійна сила економіки країни, осередок культури й освіти, майданчик для реалізації технологічних та соціальних інновацій. З огляду на це, нині значно посилюється конкуренція міст, адже розвиток комфортної інфраструктури безпосередньо впливає на економічні показники міста, визначає його привабливість для кваліфікованих спеціалістів та інвесторів. Створення розумного міста передбачає комплексні соціальні та технологічні трансформації, що стають можливими шляхом розвитку сучасних інформаційно-комунікаційних технологій, розробленням нових стандартів енергоефективності та появи нової якості відносин між громадою та місцевою владою. Жителі сучасного міста перестають бути виключно користувачами, перетворюючись на постачальників міського сервісу.

ІКТ дозволяють використовувати менше енергетичних ресурсів, задовольняючи незмінний обсяг потреб, та зменшувати масштаби парникової емісії, що означає впровадження розумної системи міського транспорту, оновленої системи водопостачання та утилізації відходів, а також створення ефективніших систем опалення та охолодження будинків.

При цьому, всі системи між собою мають бути взаємопов’язані та працювати як єдиний злагоджений механізм. До ІКТ додається людський та соціальній капітал, який відповідає за підвищення безпеки громадських місць та створення зручностей для жителів. Таким чином, концепція «розумного міста» спрямована на надання реальних переваг для життя населення та функціонування бізнесу відповідно до принципів сталого розвитку.

Сучасні інформаційні технології виконують в "розумному місті" три важливі завдання:

- забезпечують швидкі комунікаційні канали передачі інформації,
- здійснюють збір та передачу необхідних даних службам управління міським господарством,
- виконують роль засобу зворотного зв'язку між адміністрацією міста та його жителями.

Пристрої інтернету речей, такі як датчики, камери активно збирають інформацію про стан міських комунікацій та інфраструктури, їх доповнює інформація, яка отримується безпосередньо від жителів міста. Уся отримана інформація безперервно обробляється та оновлюється, що дає змогу системі працювати у реальному часі. Для передачі зібраних даних використовуються швидкісні комунікаційні мережеві канали, за допомогою яких інформація максимально швидко передається на наступний рівень – в центр обробки даних (ЦОД). Після комп'ютерної обробки зібраних даних, відбувається аналіз отриманих результатів, та передача інформації на вищий рівень управління та аналізу – в служби міської адміністрації, яка проводить аудит даних та вибирає шляхи оптимізації та поліпшення ефективності роботи

міського господарства. Проте дуже важливо, що за допомогою кінцевих пристроїв IoT, об'єднаних сучасними комунікаційними технологіями, можна збирати дані практично в режимі реального часу. Це дозволяє максимально оперативно і ефективно вирішувати завдання міського благоустрою.

Розумне місто має шість основних складових, які полягають у наступному[1]:

- «розумна економіка» (smart economy) – цифровізація бізнесу та торгівлі, зростання продуктивності, інноваційно-технологічне виробництво товарів та доставка послуг тощо;
- «розумне переміщення» (smart mobility) – транспортні та логістичні системи, створені на інформаційно-комунікаційних технологіях, що дозволяли б використання декількох видів транспорту для переміщення у будь-яку точку міста;
- «розумні люди» (smart people) – розвиток електронних навичок у людей, підвищення рівня освіченості, підвищення кваліфікації, розвиток креативності та стимуляція інноваційних проривів;
- «розумне життя» (smart living) – запровадження способу життя, поведінки та моделі споживання за використанням інформаційно-комунікаційних технологій, покращення здоров’я та культурний розвиток;
- «розумне врядування» (smart governance) – інтерактивне місцеве правління, яке забезпечує ефективне всеохоплююче функціонування міста.
- «розумне довкілля» (smart environment) – запровадження принципів енергоефективності та зменшення викидів парникових газів за рахунок

застосування замкнутих енергетичних мереж, систем контролю та моніторингу рівня забруднення.

Деякі міста вже давно розвиваються в даному напрямку, серед них Азія та Європа. Для прикладу, у Дубаї та Сінгапурі в проектування інфраструктури та будівельні проекти включають пристрої IoT. Нижче наведено декілька міст, які визнані як провідні у розвитку розумних міст [2].

- Сінгапур: зіткнувшись з проблемою старіння населення, уряд підвищує рівень життя за допомогою датчиків по всьому місту для збору даних про рух та діяльність пішоходів, інформація з яких потім направляється до відомств для подальшого аналізу та дій щодо надання послуг. Також, розумні технології, що інтегровані в житло, надають можливості інженерам аналізувати потік вітру, проникнення сонячного світла та затінені ділянки для кращого проектування та розміщення нових будівель.
- Дубай: оцифрування 90 різних установ, серед яких електроенергія, транспорт, зв'язок, інфраструктура, економічні послуги та містобудування, доступ до яких можливий через додаток. Система моніторингу зі штучним інтелектом за водіями автобусів значно зменшила дорожньо-транспортні пригоди, спричинені втомою. У місті вже є три автономних відділення поліції, де люди платять штрафи, або повідомляють про випадки, не контактуючи з людьми.
- Осло: бореться зі зміною клімату за допомогою управління освітленням, опаленням та охолодженням. Метою міста є скорочення викидів на 95% до 2030 року, створюючи можливості для розвитку електромобілів (заохочують безкоштовними паркінгами та звільнення

від оплати податків), інтелектуальної мережі та технологій зарядки електромобілів.

- Копенгаген: рухається до розумного розвитку, створюючи систему з контролем якості повітря, поводження з відходами, використання енергії, поєднуючи з системами паркування, світлофорами, лічильниками та електромобілями. Оскільки багато жителів пересуваються на велосипедах, було створено додаток, що використовує усі дані з системи і веде по міських вулицях, повідомляючи як швидко потрібно крутити педалі, щоб загорілось зелене світло на світлофорі та інші.

Отже, великі міста потребують розумного управління системами міського благоустрою задля раціонального використання ресурсів і задоволення потреб жителів, і, очевидно з вищенаведених прикладів, вже в багатьох містах світу частково створені та впроваджені такі системи, що підвищує вірогідність на застосування таких систем і в Україні.

1.2. Розумні системи управління дорожнім трафіком та існуючі проблеми

Автоматичне управління дорожнім рухом є важливих компонентом системи розумного міста. Таке технічне рішення необхідне для покращення економічних показників, безпеки руху транспорту, екологічного контролю забрудненості повітря, а також аналізу дорожнього руху та виявлення вузьких місць міста, що потребують удосконалень.

Правильне технологічне рішення можна масштабувати до будь-якого розміру та безболісно модернізувати в будь-який час. Одночасно ці технологічні рішення готують розумні міста до майбутніх еволюцій

технологій, включаючи взаємопов’язані транспортні засоби (connected vehicles) та впровадження 5G [3].

Однією з найбільших проблем, що сприяє появі усіх інших, є затори. Через велику кількість транспортних засобів, що перевищують пропускну здатність доріг, часто можна спостерігати скупчення машин у центрі міста.

За даними рейтингу аналітичної компанії TomTom у 2020 році [4], Київ посів 7 місце серед 416 міст світу за інтенсивністю заторів на дорогах. Також у тридцятку увійшли ще 3 українських міста, що свідчить про існування проблеми та її не вирішення.

Дані скупчення провокують появу інших проблем, таких як посилення забруднення у місті, затримки доставок, запізнення на роботу, затримка екстрених служб. Усі ці проблеми підвищують рівень незадоволення жителів та знижують якість життя у місті.

Зіткнувшись із цими проблемами, інноваційні міста - "Розумні міста" - використовують узгоджений масив апаратних, програмних та хмарних рішень для збільшення потоку трафіку та підвищення безпеки. Розумні системи управління дорожнім рухом - це автоматизовані системи, що включають новітні досягнення технології Internet of Things (IoT). Ці системи можуть оптимізувати рух транспорту та підвищити безпеку за допомогою датчиків, камер, маршрутизаторів та стільникових технологій для динамічного регулювання таких механізмів управління, як світлофори, лічильники автостради, швидкісні смуги руху автобусів, дошки оголошень на шосе і навіть обмеження швидкості.

Деякі з ключових функціональних можливостей, які міста досягають за допомогою цих систем, включають наступне:

- Виявлення заторів: за допомогою камер та датчиків, які постійно контролюють перехрестя, спеціалісти можуть контролювати все місто з міського центру управління дорожнім рухом.
- Адаптивне управління: виявлення заторів також дозволяє адаптивне управління, яке викликає динамічні налаштування систем, включаючи світлофори, сигналізацію на пандусі та смуги швидкого транзитного руху автобусів.
- Виявлення місць з підвищеним рівнем забрудненості повітря, спричинених скупченням автомобілів.
- Динамічні зміни ціни за паркувальні місця в залежності від попиту, що зменшує кількість охочих надовго затримуватись у центрі міста.
- Визначення популярних напрямків у місті для додавання нових маршрутів громадського транспорту.
- Екстрена маршрутизація: Найважливішим застосуванням інтелектуальної системи управління дорожнім рухом є можливість надання пріоритетного доступу до поліції, пожежних служб та служб швидкої допомоги.

Саме через наведені вище причини впровадження інтелектуальних систем управління дорожнім трафіком є одним з головних завдань при проектуванні розумного міста.

В Україні, а саме в Києві, на даний час існує система виявлення порушників швидкісного режиму ЦОДР [5]. Дана система автоматично виявляє автомобіль, що рухається по проїзній частині з перевищенням швидкості, фотографує та надсилає на центральний пункт поліції. Також на офіційному сайті вказано, що деякі перехрестя Києва оснащені відеодетекторами з розробленими планами координації транспорту.

Оскільки, ці плани запрограмовані вручну, це означає, що не враховується поточна ситуація. Тому, дані системи ще потребують удосконалень.

1.3. Переваги граничних обчислень над хмарними

Інтернет речей швидко вливається у повсякденне життя, як нова технологія останніх кількох років. Як парадигма, Інтернет речей передбачає, що більшість фізичних пристроїв, таких як смартфони, транспортні засоби, датчики, виконавчі механізми та будь-які інші вбудовані пристрої, будуть з'єднані один з одним та будуть обмінюватися даними з центрами обробки даних. Пристосувавшись до різноманітних популярних систем, таких як інтелектуальний транспорт, розумне місто, інтелектуальна мережа та інтелектуальна система охорони здоров'я, люди відчуватимуть дискомфорт, якщо Інтернет речей не заповнить їх дім та роботу [6]. Таким чином, Інтернет речей надзвичайно впливає на повсякденне життя користувачів і є запорукою майбутнього. Проте неопрацьовані дані, створені пристроями IoT, в цілому, не приносять користі. Такі надзвичайно великі масиви даних вимагають значних потужностей для обробки та вилучення інформації, щоб надати потенційно корисні факти. Пристрої IoT спрямовані на реалізацію цього завдання: перетворення зібраних даних у корисні дані.

У наш час різні типи камер широко використовуються в містах та майже на кожному транспортному засобі. Вони містять багато корисних даних, які можуть бути використані для багатьох цілей, таких як пошук зниклої дитини або як доказ поліцейського розслідування. Однак дані з цих камер зазвичай не завантажуються в хмару через проблеми конфіденційності та велику кількість трафіку, що дорого обходиться. Навіть

якщо дані доступні в хмарі, завантаження та пошук серед величезної кількості даних може зайняти багато часу.

Обчислювальні інфраструктури Cloud, Fog та Edge сьогодні використовуються в різних системах, які працюють з даними. Ці парадигми надають організаціям можливість використовувати різні обчислювальні послуги та зберігають дані залежно від організаційних вимог. Хоч вищенаведені обчислювальні архітектури здаються схожими, але досить сильно різняться з точки зору своїх характеристик. Це дозволяє їм задовольняти різні вимоги, необхідні для реальних програм [7].

У контексті Інтернету речей традиційні хмарні обчислення мають кілька недоліків [8].

1) Затримка: нові програми мають високі вимоги в режимі реального часу. У традиційній моделі хмарних обчислень програми надсилають дані до центру обробки даних і отримують відповідь, що збільшує затримку системи. Наприклад, високошвидкісні автономні транспортні засоби вимагають мілісекунди часу відгуку. Серйозні проблеми виникають, коли затримка системи перевищує очікування через неполадки з мережею.

2) Пропускна здатність: передача великих обсягів даних, що генеруються граничними пристроями, до хмари в режимі реального часу призведе до великого тиску на пропускну здатність мережі. Наприклад, Boeing 787 генерує більше 5 ГБ / с даних, але пропускна здатність між літаком і супутниками недостатня для підтримки передачі в реальному часі [9].

3) Доступність: оскільки все більше і більше служб Інтернету розгортається у хмарі, доступність цих послуг стала невід'ємною частиною

повсякденного життя. Наприклад, користувачі смартфонів, які звикнуть до голосових послуг, наприклад, Siri, почуватимуться роздратовано, якщо послуга недоступна протягом короткого періоду часу. Тому для хмарних провайдерів це виклик надавати послуги безперервно і без збоїв, забезпечуючи відмовостійкість.

4) Енергія: центри обробки даних споживають багато енергії. Зі збільшенням обсягу обчислень та передачі, споживання енергії стане вузьким місцем, що обмежить розвиток хмарних обчислювальних центрів.

5) Безпека та конфіденційність: дані про взаємозв'язок тисяч домогосподарств - тісно пов'язані з життям користувачів. Наприклад, внутрішні камери, що передають відеодані від будинку до хмари, збільшать ризик витоку приватної інформації користувачів. Завдяки застосуванню Загального регламенту захисту даних ЄС (GDPR) [18] питання безпеки даних та конфіденційності стали більш важливими для компанії з хмарних обчислень.

Ці виклики підштовхнули розвиток граничних обчислень, що вимагають обробки даних на межі мережі. Дана парадигма швидко розвивається з 2014 року, надаючи можливість зменшити затримки та оплату за ширину смуги, усунути обмеження обчислювальних можливостей центру хмарних даних та збільшити доступність, а також захистити конфіденційність даних та безпеку.

Враховуючи вищенаведені недоліки щодо хмарних обчислень, було прийнято рішення про застосування граничних обчислень, які можуть бути реалізовані за допомогою мікрокомп’ютерів.

Граничні обчислення трансформують спосіб обробки та доставки даних із мільйонів пристроїв по всьому світу. В останні роки [10], з розповсюдженням Інтернету речей та широким впровадженням бездротових мереж, кількість граничних пристроїв та даних, що генеруються на межі, швидко зростають. Згідно з прогнозами Міжнародної корпорації даних (IDC) [11], глобальні дані досягнуть 180 цетабайт (ZB), і 70% даних, сформованих пристроями Інтернету речей, будуть оброблені на границях мережі до 2025 року. IDC також прогнозує, що до 2025 року в усьому світі буде підключено понад 150 мільярдів пристроїв.

Більш швидкі мережеві технології, такі як бездротова мережа 5G, дозволяють граничним обчислювальним системам пришвидшити створення або підтримку додатків у режимі реального часу, наприклад, обробка відео та аналітика, автономні автомобілі, штучний інтелект та робототехніка. Незважаючи на те, що метою граничних обчислень було покриття витрат на пропускну здатність даних, що переміщуються на великі відстані, через зростання кількості даних, що генеруються IoT, зростання кількості додатків у реальному часі, які потребують обробки на границях, рухають технологію вперед.

Граничні обчислення - це частина розподіленої обчислювальної топології, в якій обробка інформації розташована близько до кінцевих пристроїв, де речі та люди виробляють або споживають цю інформацію.

Порівняльна характеристика технологій наведена в Таблиці 1.1 [12].

Таблиця 1.1

Порівняння технологій граничних та хмарних обчислень

Порівняльні	Граничні обчислення	Хмарні обчислення

характеристики		
Кількість збережених даних	Мала	Величезна
Кількість оброблених даних	Мала	Величезна
Обчислювальна потужність	Менш потужні	Більш потужні

Продовження таблиці 1.1

Час обробки і відповіді	Швидко	Повільно
Безпека даних	Безпечно	Не безпечно
Стандартизація	Не стандартизовано	Не стандартизовано
Витрати на рік	Менш затратно	Більш затратно

Також необхідно враховувати, що обсяг даних, що зберігаються на граничних пристроях – обмежений, і можливість збільшити його при необхідності потребує часу та ресурсів для налаштування.

Якщо брати до уваги пристрої, які контролюють виробниче обладнання, або підключену до Інтернету відеокамеру, яка надсилає прямі кадри з віддаленого офісу, то хоч цей пристрій може передавати їх по мережі досить легко, виникають проблеми, коли кількість пристроїв, що передають дані одночасно, зростає. Замість того, щоб одна відеокамера передавала живі кадри, необхідно враховувати, що їх може бути сотні чи тисячі. Через затримку не тільки страждає якість, але й можуть бути величезні витрати на пропускну здатність.

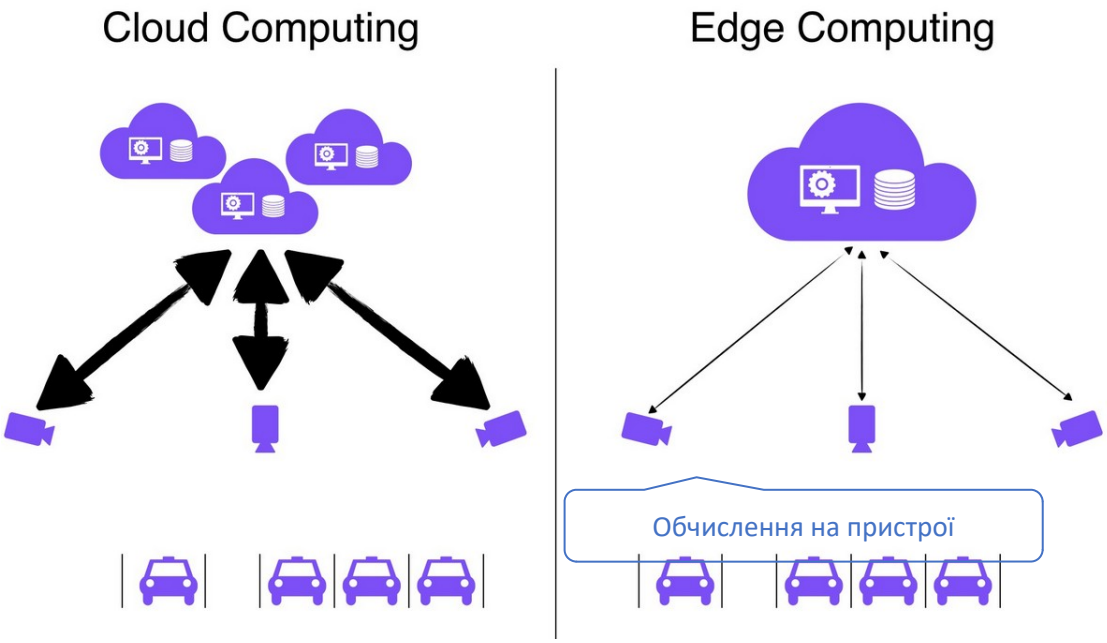

Рис. 1.1. Два підходи: хмарні обчислення та граничні обчислення. У випадку хмарних обчислень – усі зібрані дані передаються напряму до хмари, в той час як використовуючи підхід з граничним обчисленням, усі обчислення виконуються на камерах і кількість даних для передачі на хмару – значно менша.

Граничне обладнання допомагає вирішити цю проблему, будучи локальним джерелом обробки та зберігання даних для багатьох з цих систем. Наприклад, граничний шлюз може обробляти дані з граничного пристрою, а потім надсилати назад лише відповідні дані через хмару, зменшуючи потреби в пропускній здатності. Або він може надсилати дані назад на граничний пристрій у разі потреби в реальному часі. Ці граничні пристрої можуть бути різними, для прикладу, датчик IoT, ноутбук працівника, смартфон, камери безпеки у розумному місті.

Однак найчастіше найбільшою перевагою граничних обчислень є можливість швидшої обробки та зберігання даних, що забезпечує ефективність програм в режимі реального часу, які є критично важливими.

Висновки

1. Автоматизовані системи управління транспортом є однією з найважливіших галузей для розвитку розумного міста.

2. Існуючі системи в Україні не аналізують трафік, а лише фіксують порушення швидкості.

3. Кількість корисної інформації, що передається на центральні сервери для подальшої обробки отриманих даних, порівняно з загальним об’ємом дуже мала. Тому, необхідно переносити обробку і аналіз на граничні пристрої.

РОЗДІЛ 2.

АРХІТЕКТУРА СИСТЕМИ АНАЛІЗУ ТРАНСПОРТНИХ ПОТОКІВ

2.1. Запропонована архітектура системи аналізу транспорту

Уявно запропоновану систему можна розділити на серверну та граничну частини, що об’єднані за допомогою або оптоволокна, або 5G. Проте, у даній роботі фокус зосереджений на граничний пристрій, та його функції, хоча певні відомості щодо топології, переваг та недоліків хмарних обчислень наведені далі.

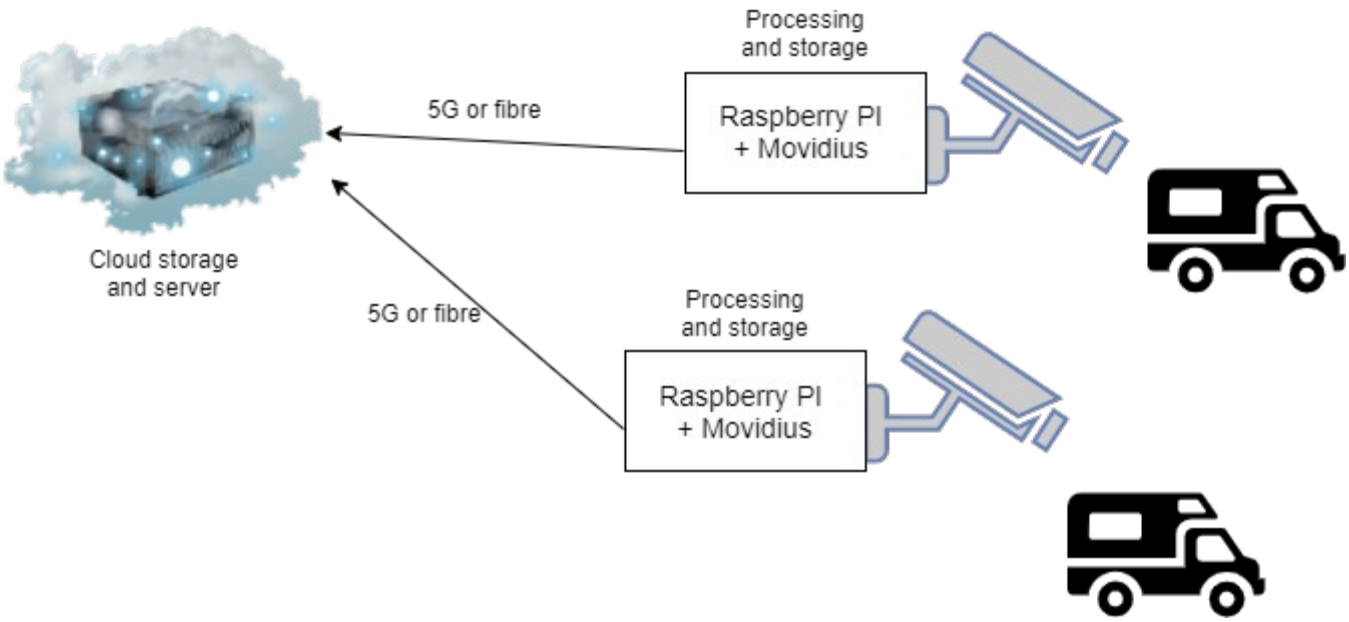

Рис. 2.1. Схема системи аналізу транспорту в системі розумного міста

На Рис.2.1. представлена схема запропонованої системи, що складається з серверу та сховища у хмарі та кінцевого обладнання у вигляді мікрокомп’ютера з локальним сховищем та під’єднаної відеокамери, що встановлені по місту.

2.2. Причини перенесення обробки на граничні пристрої

З огляду на проаналізовану інформацію щодо хмарних обчислень, можна виділити три основні проблеми, через які варто переносити обробку на граничні пристрої.

Перша проблема – це швидке збільшення об’єму даних на серверах, що потребує достатньо ресурсів для його обробки та зберігання. Звичайно, хмарні технології дозволяють миттєво додавати обчислювальні ресурси та збільшувати об’єм сховища за необхідності, проте вартість послуг збільшується, зі збільшенням потреб, що в загальному означає постійний ріст оплати за використання сервісу.

Друга проблема полягає в кількості переданого трафіку. При неперервній передачі відео від кінцевого пристрою до хмари, пропускна здатність каналу досягатиме свого максимуму, що також впливає на економічну складову з боку вартості передачі трафіку та технічну – збільшує час передачі інформації.

Ще одна проблема стосується часу обробки інформації. Якщо розглядати варіант реалізації з відправкою даних на сервер, то час обробки відео залежить від затримки та швидкості опрацювання даних на сервері, а значить, від кількості серверів.

Зважаючи на вищенаведені недоліки, в запропонованій системі раціонально обробляти відеопотік на граничному обладнанні.

2.3. Варіанти обробки даних

Важливим компонентом системи є дані, оскільки на їх основі проводиться подальший аналіз результатів щодо стану дорожньої ситуації у місті. Обробка даних у реальному часі надає переваги для вирішення

завдань, окрім аналізу транспортного потоку, по регуляції світлофорів, пошуку порушників чи злочинців та інші.

У запропонованій системі збереження інформації про транспортний потік – тимчасове, хоч і числові дані не потребують великих обсягів сховища. Час зберігання даних можна регулювати за потреби, проте не варто їх зберігати довго, так як кінцевий пристрій може вийти з ладу, що спровокує втрату даних. Саме тому, необхідно відправляти зібрані дані на центральний сервер, що задовольняє усі вимоги по відмовостійкості, реплікації та доступності, приблизно раз на годину.

2.4. Вибір топології системи

Стосовно топології для побудови системи, розглянемо дві найбільш раціональні: зірка та ієрархія, оскільки, система має один центральний елемент – сервер.

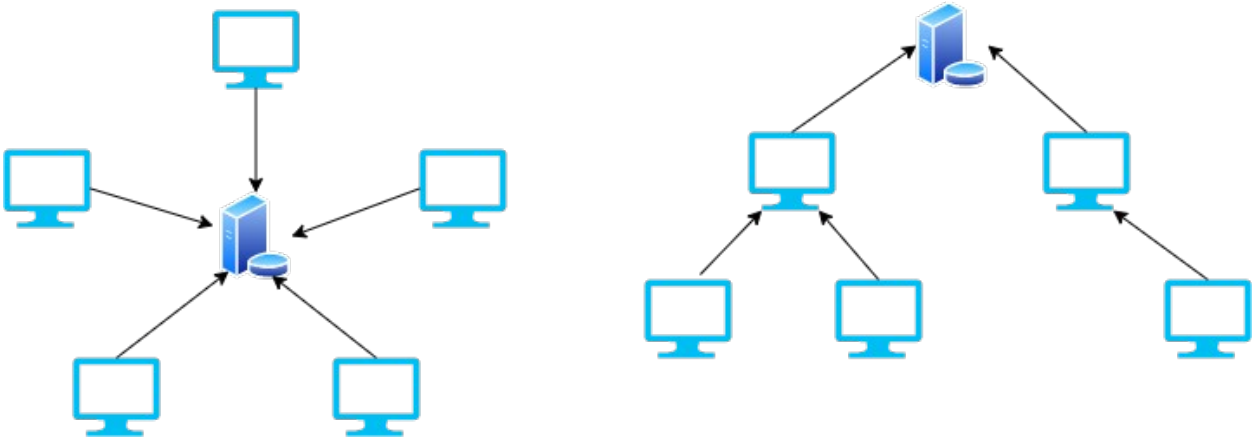

Рис. 2.2. Топології побудови системи: зірка (ліворуч) та ієрархія (праворуч)

Так як кожен з граничних пристроїв є самостійним і не потребує ніяких додаткових дій з іншими пристроями, а напряму звертається до серверу, тому найбільш оптимальною топологією «мережі» елементів системи є зірка. Хоча, ієрархію також можна застосовувати після певного

порогового значення кількості кінцевих пристроїв, щоб зменшити навантаження на єдиний сервер і в подальшому вже консолідовані дані передавати до серверу.

2.5. Системні вимоги

2.5.1. Функціональні вимоги до системи

В цьому розділі описані функціональні вимоги до запропонованої системи, що визначають її поведінку.

Основні функції:

- Розпізнавання транспортних засобів на дорозі;
- Трекінг руху транспортних засобів;
- Надання кожному транспортному засобу власного ідентифікатору;
- Підрахунок кількості транспортних засобів в реальному часі кожну хвилину;
- Зберігання зібраних даних в локальному сховищі;
- Передача даних підрахунку до центрального серверу кожну годину;

2.5.2. Нефункціональні вимоги

- Якість відеозйомки не менш ніж 320px
- Швидкість зміни кадрів більше ніж 4FPS
- Віддалений доступ до кінцевих пристроїв системи
- Доступність системи вище ніж 80%
- Точність розпізнавання об’єктів не менше 30 mAP

2.6. Складові апаратної частини системи

Розглядаючи систему з боку реалізації, вона складається з двох частин, необхідних для її повноцінного функціонування: апаратної та програмної.

В якості апаратного забезпечення кінцевого пристрою було обрано мікрокомп’ютер Raspberry Pi 4 з додатковим модулем Movidius від Intel та під’єднаною Pi Camera.

- **Raspberry Pi 4**

Рис. 2.3. Мікрокомп’ютер Raspberry Pi 4

Нижче наведені характеристики мікрокомп’ютера Raspberry Pi 4 [13]:

- 1.5GHz 64-bit quad-core ARM Cortex-A72 CPU (ARM v8, BCM2837)
- 1GB, 2GB, or 4GB RAM (LPDDR4)
- On-board wireless LAN (dual-band 802.11 b/g/n/ac)
- On-board Bluetooth 5.0, low-energy (BLE)
- 2x USB 3.0 ports
- 2x USB 2.0 ports
- Gigabit Ethernet

- Power-over-Ethernet (this will require a PoE HAT)
- 40-pin GPIO header
- 2× micro-HDMI ports (up to 4Kp60 supported)
- H.265 (4Kp60 decode)
- H.264 (1080p60 decode, 1080p30 encode)
- OpenGL ES, 3.0 graphics
- DSI display port
- CSI camera port
- Combined 3.5mm analog audio and composite video jack
- Micro-SD card slot
- USB-C power

Перевагами даної моделі є збільшена тактова частота процесора, збільшений обсяг оперативної пам'яті, поява роз'єму USB Type-C для підключення блоку живлення, а також гігабітний Ethernet. У якості локального сховища можна використовувати або Micro-SD карту, або ж жорсткий диск.

- **Movidius Neural Compute Stick 2**

Movidius Neural Compute Stick 2 – це візуальний процесор для ефективного глибокого навчання нейронних мереж, який можна застосовувати на граничних пристроях системи. Він достатньо невеликий за розмірами та зручний у використанні: підключення через USB порт.

Пристрій призначений для багатьох завдань, а до переваг входять:

- створення і масштабування з винятковою продуктивністю;

- швидка підготовка до розробки в середовищі Windows 10, Ubuntu або macOS;

- розробка з використанням типових інфраструктур і готових додатків;

- можливості роботи без залежності від хмарної середовища;

- створення прототипів з використанням недорогих кінцевих пристроїв, таких як Raspberry Pi 3 і вище, або інших кінцевих систем ARM.

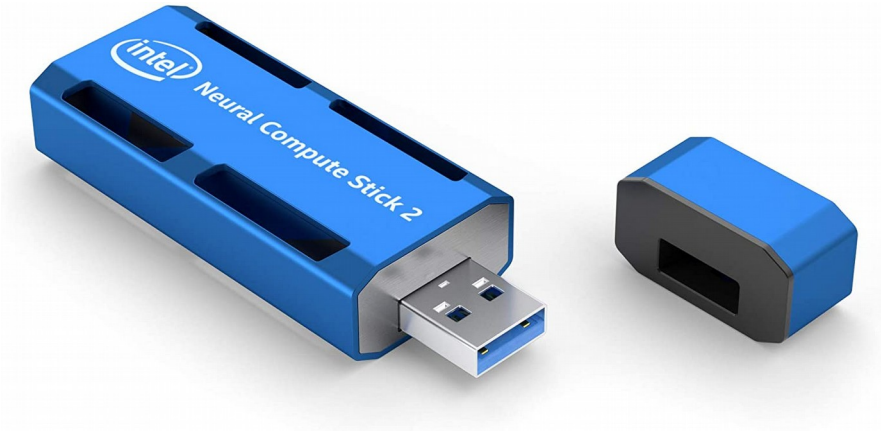

Рис. 2.4. Модуль Intel Movidius Neural Compute Stick 2

В основі модуля візуальний процесор Intel® Movidius™ Myriad™ X, який має 16 векторних 128 бітних LIW-ядра SHAVE (Streaming Hybrid Architecture Vector Engine). Myriad X має апаратну підтримку кодування 4k-відео зі швидкістю до 60 кадрів за секунду. Максимальна теоретична продуктивність Movidius Myriad X становить 4 Терафлопси, при цьому енергоспоживання SoC не більше 1 Вт.

Підтримуються інфраструктури: TensorFlow і Caffe. Сумісні операційні системи: Ubuntu 16.04.3 LTS (64-розрядна), CentOS 7.4 (64-розрядна) і Windows 10 (64-розрядна). Також Intel Neural Compute Stick 2 підтримує Open Visual Inference & Neural Network Optimization

(OpenVINO). Важливо, що можна об’єднати кілька «флешок» Intel® NCS 2 водну платформу для масштабування продуктивності [14].

- **Камера Raspberry Pi**

Для захоплення відео було обрано камеру Raspberry Pi.

Характеристики:

- Фокус: фіксований
- Відео режими: 1080p30, 720p60, 640x480p60/90
- Розширення фото: 8 Мп (3280 x 2464 пікселів)
- Контролер: Sony IMX219
- Довжина шлейфа: 15 см
- Розміри плати: 25 x 20 x 9 мм

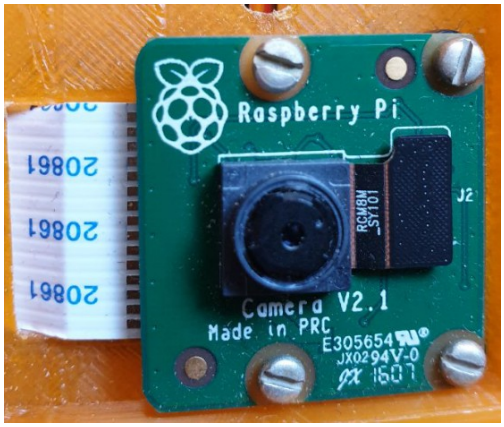

Рис. 2.5. Камера Raspberry Pi

- **Локальне сховище:**

Рис. 2.6. SD-карта

Для локального зберігання зібраних даних щодо кількості транспортних засобів, можна обрати будь-яку SD-карту, наприклад розміром 16GB. Тому що обсяг отриманої інформації – числовий, а як відомо, такі дані не потребують великих обсягів пам’яті.

Зважаючи на всі обмеження з боку апаратної частини необхідно обрати такі алгоритми для розпізнавання та стеження транспортних засобів, що будуть задовольняти поставлені умови. Дані алгоритми будуть розглянуті в наступних розділах.

- **Апаратна установка граничного пристрою**

На рисунку 8 представлена зібрана нами експериментальна установка у складі мікрокомп’ютера Raspberry Pi 4 з додатковим модулем Intel Movidius Compute Stick та Pi Camera. Ця комплектація не є кінцевим продуктом, а використовується для перевірки запропонованих алгоритмів. Очевидно, що промисловий варіант потребує вдосконалень дизайну та захисту.

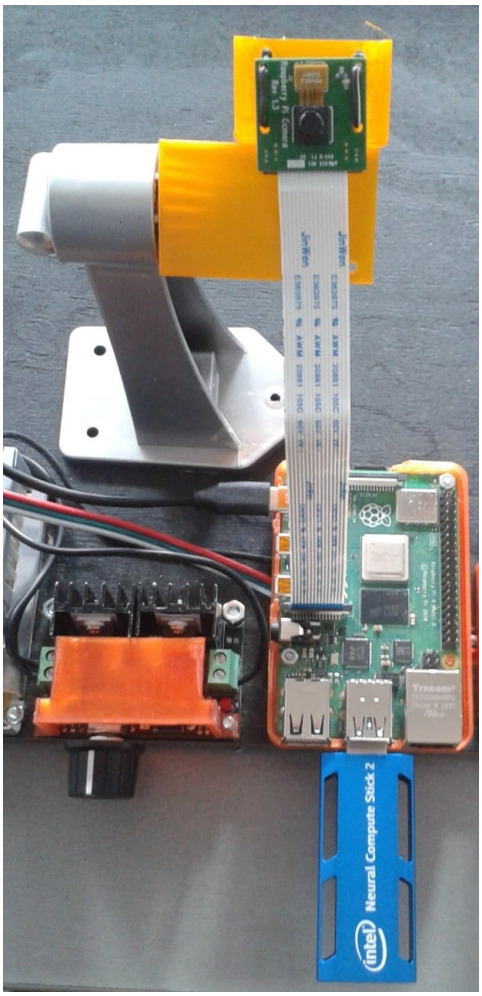

Рис. 2.7. Експериментальна установка

Висновки

1. Запропоновано систему аналізу інтенсивності транспорту у місті.

2. Вказано варіанти обробки інформації та обґрунтовано вибір топології системи.

3. Описано функціональні та нефункціональні вимоги до системи.

4. Обрано апаратне забезпечення та зібрано експериментальну установку.

РОЗДІЛ 3.

РОЗПІЗНАВАННЯ ТА ТРЕКІНГ ТРАНСПОРТНИХ ЗАСОБІВ

3.1. Підходи до детектування об’єктів

Автоматичне відстеження транспортних засобів має вирішальне значення для аналізу відеоданих дорожнього руху у режимі реального часу. Протягом багатьох років було здійснено численні спроби вирішити цю проблему за допомогою широкого спектру алгоритмів, заснованих як на машинному навчанні, так і на глибокому навчанні [15].

3.1.1. На основі машинного навчання

Віднімання кадру

У цьому відносно простому методі розраховується різниця між двома послідовними кадрами для виявлення рухомого об’єкта за допомогою OpenCV та Python. Спочатку відео перетворюють у відтінки сірого зі згладжуванням, змінюють інтенсивність відносно порогового значення та виділяють контур. Рухомі контури, визначені алгоритмами, позначають обмежувальними рамками [16].

Рис. 3.1. Визначення об’єкту на основі послідовної різниці кадрів

Як видно з зображень, обмежувальні рамки навколо рухомого об’єкта визначені не точно і виділено багато зайвих областей, що не містять

рухомих об’єктів. Також, на даний алгоритм впливає якість відео, умови освітлення, наявність тіней тощо.

Підхід на основі ознак

Каскади Хаара - це алгоритм виявлення об’єктів машинного навчання, який базується на концепції виявлення особливостей, запропонованій Віолою та Джонсом, також відомою як Алгоритм Віоли-Джонса [17]. Алгоритм Віоли-Джонса використовує зсувне вікно для пошуку корисних ознак, які потім використовуються для підготовки моделі, а надалі використовують їх як класифікатор.

Каскади Хаара використовують алгоритм Віоли-Джонса для пошуку певного типу особливості, відомого як Хароподібні ознаки. Їх можна класифікувати на три основні типи: граничні, лінійні та чотирикутні. Алгоритм шукає Хароподібні ознаки в зображенні, як показано на Рис. 3.2., і цей процес оптимізований за допомогою таких методів, як обчислення інтергрального зображення, масштабування, адаптивне підсилення та каскадування. Адаптивний Boosting або Adaboost - це техніка машинного навчання, де слабші додаткові функції поєднуються з більш сильними, щоб підвищити точність моделі.

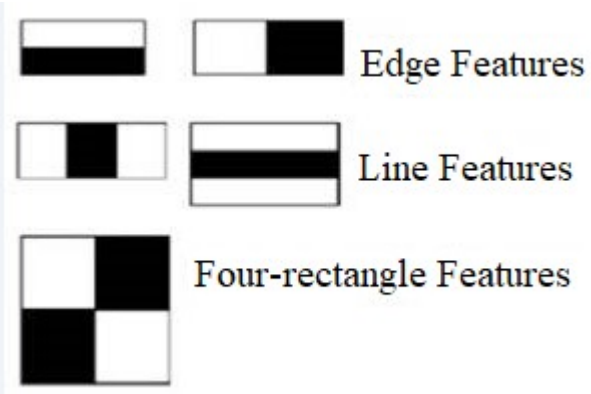

Рис. 3.2. Хароподібні ознаки

Однак, незважаючи на інший підхід, точність виявлення не наближена до ідеальної, проблема тіней, якості відео та освітленості також впливають на роботу алгоритму.

3.1.2 Підхід на основі глибокого навчання (Deep Learning)

Останнім часом нейронні згорткові мережі (CNN) значно поліпшили класифікацію зображень і точність виявлення об'єктів. У порівнянні з класифікацією зображень, детектування об'єктів – завдання непросте, для вирішення якого потрібні складніші методи. Складність виникає через те, що розпізнавання вимагає точної локалізації об'єктів, що створює дві основні проблеми. По-перше, необхідно обробити безліч можливих місць розташування об'єктів (їх називають «кандидатами»). По-друге, ці кандидати забезпечують тільки грубу локалізацію, яку необхідно уточнити, щоб домогтися точної локалізації. Для вирішення цих проблем часто йдуть на компроміс зі швидкістю, точністю та простотою.

Для того аби застосувати алгоритми глибокого навчання для комп’ютерного зору необхідно декілька елементів (Рис.3.3.) [36]: вхідне зображення, попередньо-натренована нейронна мережа або власна нейронна мережа, основна частина орієнтована на задачу і вихідне зображення.

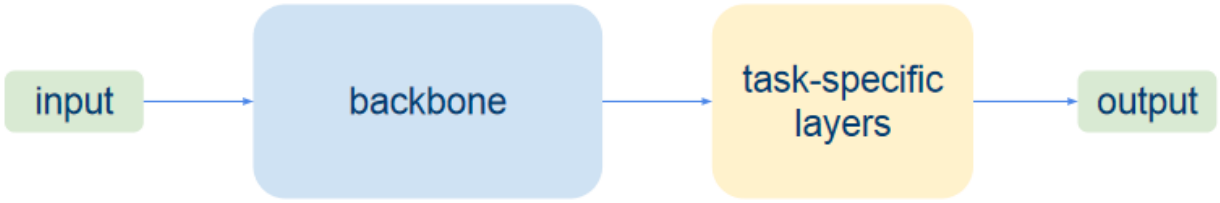

Рис. 3.3. Компоненти алгоритму глибокого навчання для комп’ютерного зору

Попередньо натренована нейронна мережа

Така мережа складається з певної кількості згорткових шарів, кожен з яких відповідає за певний набір ознак, що можна знайти на вхідному зображенні. Попередньо натреновані нейронні мережі тренуються на великих наборах даних, таких як ImageNet, MSCOCO та інші. Перевагою цих мереж з відкритим кодом є можливість їх використовувати будь-кому і досліджувати. Немає необхідності вигадувати з нуля і витрачати купу часу для їх тренування. Ось деякі основні представниками – AlexNet, ResNet, Inception, MobileNet, VGG, NAS. Усі вони відрізняються один від одного за часом виконання, розміром моделі та іншими показниками.

Для використання у реальному часі і на вбудованих пристроях, таких як мікрокомп'ютери чи мобільні телефони, необхідно, щоб розмір моделі був мінімальний, а швидкість задовільно високою, щоб застосовувати у реальному часі. У статті [35] наведена порівняльна характеристика (Рис.3.4.) найбільш популярних сімейств нейронних мереж.

Top-1 помилка свідчить про те, що прогнозований з найбільшою ймовірністю клас, не відповідає правильно розпізнаному.

На Рис. 3.4. зображені бульбашки, розмір яких відповідає розміру моделі. А чим менші значення по осі x та y, тим краще.

Так як, у даній роботі важливо, щоб розмір моделі був якнайменшим, при цьому швидкість обробки також прагне до мінімуму, то обраною попередньо натренованою нейронною мережею було обрано MobileNet v2.

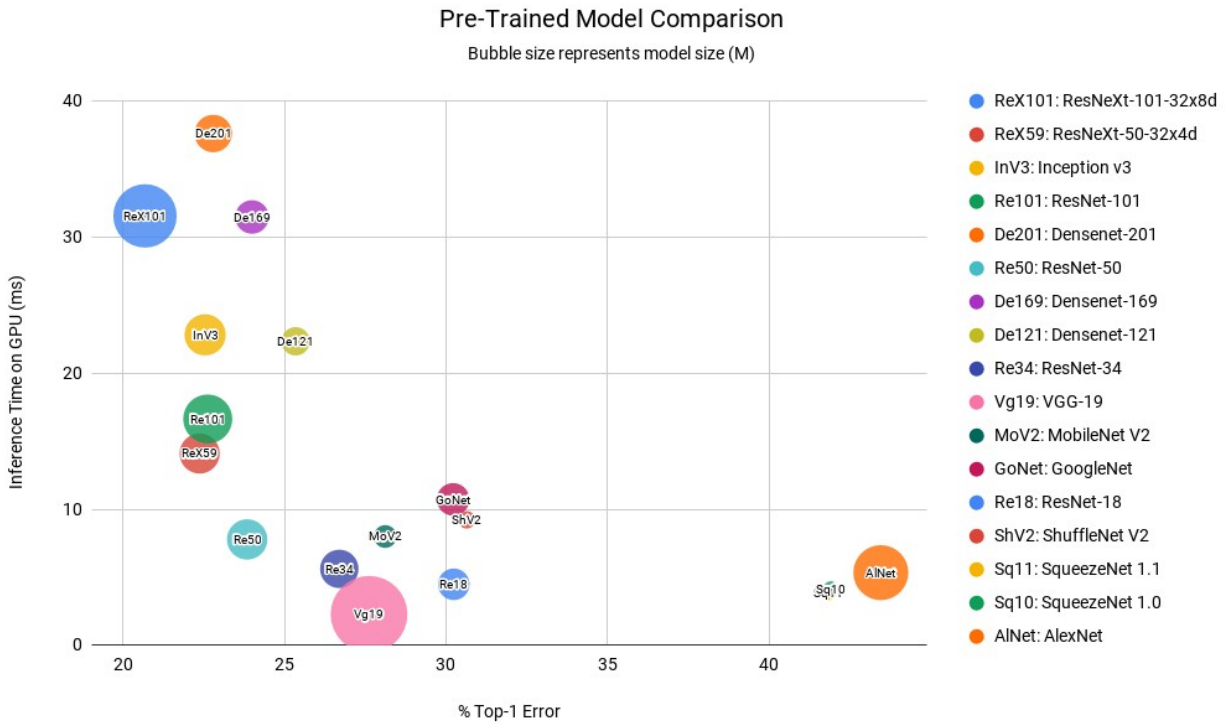

Рис. 3.4. Залежність часу відповіді від точності

Детектування об’єктів

В останні роки активно проводяться дослідження з розпізнавання об’єктів, засновані на глибокому навчанні, і завдяки цьому з’явилося багато моделей з покращеними характеристикам, а також розроблено технології розпізнавання декількох об’єктів на зображеннях [18]. Існує кілька репрезентативних технологій розпізнавання об’єктів на основі глибокого навчання, включаючи R-CNN [19], Faster R-CNN [20], Fast R-CNN [21], YOLO [22], SSD [23], R-FCN [24] тощо.

Існує два типи детекторів об’єктів: двоступеневі детектори, такі як Faster R-CNN або Mask R-CNN та одноступінчасті, такі як YOLO та SSD. Двоступеневі моделі досягають найвищих показників точності, але, як правило, повільніші. Проблема R-CNN полягає в тому, що для навчання

мережі потребує багато часу, оскільки вона повинна класифікувати 2000 регіональних пропозицій для кожного зображення, а отже, реалізація в реальному часі неможлива [26]. Одноступінчасті моделі досягають нижчих показників точності, але вони значно швидші, ніж двоступеневі детектори об’єктів.

R-CNN

Детектор R-CNN складається з двох етапів, регіональної пропозиції та класифікації. Спочатку за допомогою вибіркового пошуку (Selective Search) детектор знаходить певну кількість рамок (близько 2000), які показують область цільового об’єкта. Вибірковий пошук використовує місцеві ознаки, такі як текстура, інтенсивність, колір, тощо, щоб генерувати всі можливі місця розташування об’єкта. Потім класифікація проводиться шляхом застосування CNN до всіх обмежувальних рамок. Проте, R-CNN як алгоритм потребує величезну кількість обчислювальних ресурсів і швидкість обробки занадто повільна. Тому, щоб компенсувати ці недоліки, Faster R-CNN виконує детектування об'єкта один раз на вихідній карті об'єктів після проходження через CNN.

Faster R-CNN

Faster R-CNN використовує регіональну мережу пропозицій (Region Proposal Network) для рішення проблеми, спричиненої селективним алгоритмом пошуку. Порівнюючи швидкість обробки, Faster R-CNN у 200 разів швидше, ніж R-CNN. Метод розпізнавання об'єктів серії R-CNN має недолік низької швидкості обробки, тому не придатний для додатків у режимі реального часу.

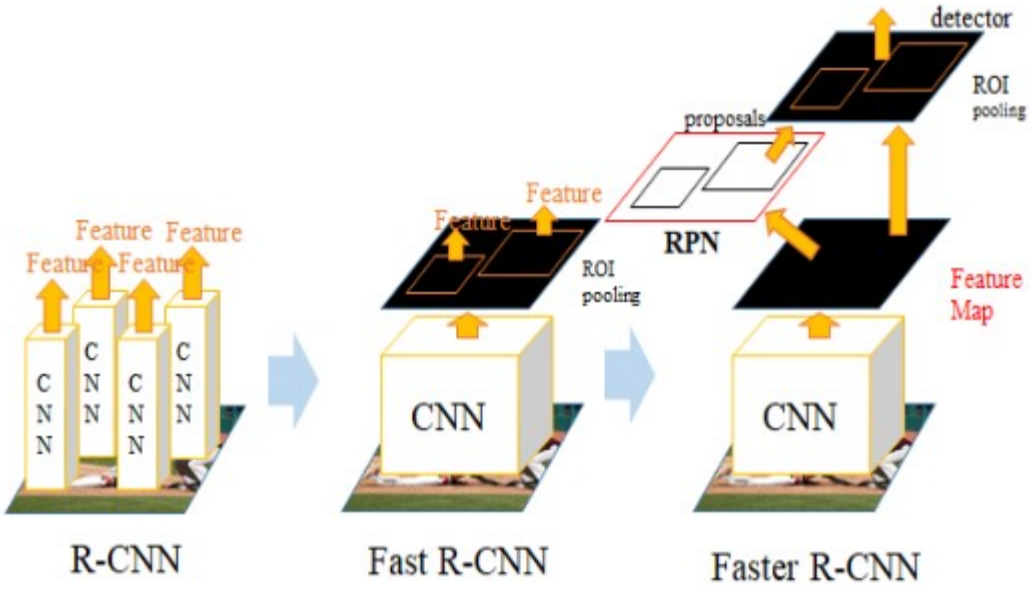

Рис.3.5. R-CNN структура

Таблиця 3.1.

Швидкісне порівняння R-CNN сімейства

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Час на одне зображення	50 сек	2 сек	0.2 сек
Прискорення	1x	25x	250x
mAP (mean average precision)	66.0	66.9	66.9

У таблиці 3.1. наведено порівняння швидкодії та точності у сімействі R-CNN, з покращенням показників у процесі розвитку.

YOLO

YOLO (You Look Only Once) - популярний і швидкий алгоритм, який широко використовується як для виявлення транспортних засобів, так і для їх класифікації. Являє собою одноступінчастий детектор, тому що регіональна пропозиція (Regional proposal) та класифікація виконуються

одночасно. Зовнішня частина мережевої структури YOLO - це модифікована структура GoogleNet. Загалом, чим глибша нейронна згорткова мережа, тим краща продуктивність зі збільшенням кількісті шарів. Однак із поглибленням мережі кількість параметрів, що вивчаються, збільшується. Проблема операції глибокої згортки полягає у величезному обчисленні та величезній кількості значень параметрів, які необхідно встановити. YOLO ділить зображення на області сітки S * S і передбачає обмежувальні рамки кожної області сітки. Разом з обмежувальними рамками, YOLO також передбачає ймовірність приналежності до певного класу. Після цього координати рамки та вірогідність приналежності до класу поєднуються, і на основі цих даних об'єкт знаходиться за допомогою придушення немаксимумів.

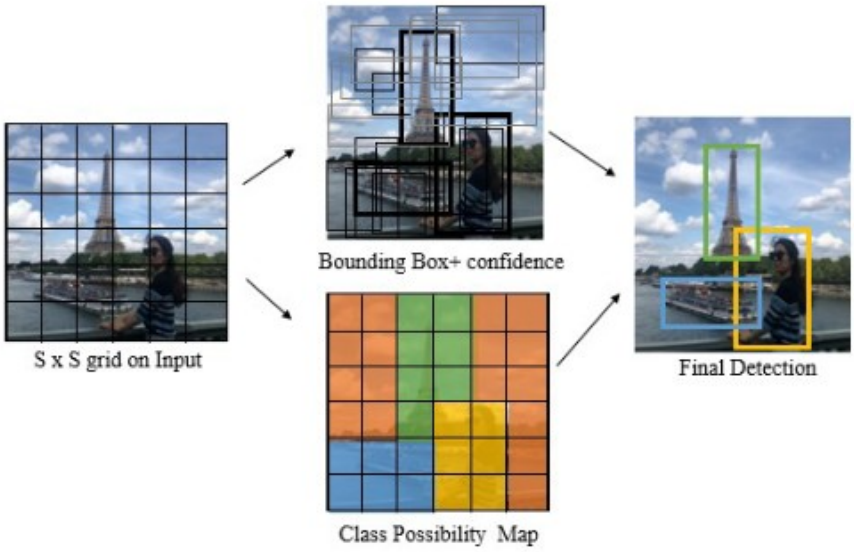

Рис. 3.6. Послідовність детектування об’єкта за допомогою YOLO

Доведено, що YOLO працює надзвичайно швидко, з частотою кадрів до 45 кадрів в секунду. Проте, оскільки YOLO, YOLOv2 та YOLOv3 вимагають великих обчислювальних потужностей для виявлення, було

розроблено алгоритм TinyYOLO [27], що оптимізований для використання у вбудованих системах та мобільних пристроях. Мережі Tiny YOLO та удосконалена Yolo-Lite [28] поступаються мережам YOLO з точки зору mAP, але працюють із значно вищим FPS.

Рис. 3.7. Застосування YOLOv3

З огляду на зображення, можна помітити, що алгоритм добре справляється з поставленою задачею виявлення транспортного засобу, як вдень, так і вночі, та незважаючи на тіні і якість зображення.

Хоч YOLO достатньо швидкісний алгоритм, проте має деякі обмеження щодо точності та виявлення дрібних предметів. Для подолання цих недоліків було створено Single-Shot Detector (SSD).

SSD (Single-shot Multibox Detector)

Алгоритм SSD виявляє об’єкти за допомогою розподілення зображення на численні поля, що одночасно проходять через єдину згорткову нейронну мережу. В подальшому координати та значення класу, розраховані за допомогою моделі, порівнюються з рамкою за замовчуванням, і на основі цього порівняння створюється остаточна обмежувальна рамка.

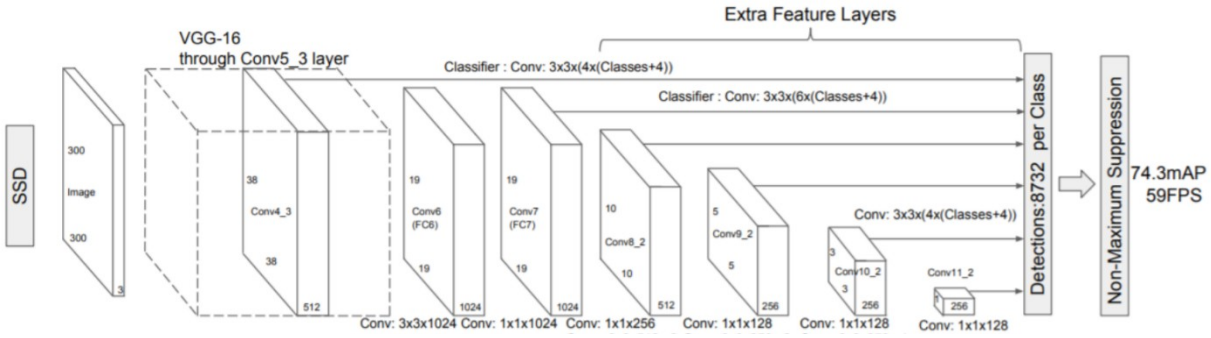

Рис. 3.8. Архітектура SSD

Той факт, що SSD використовує різні карти ознак для комбінування передбачень, призводить до збільшення кількості виявлень кожного класу та зображення, а різна роздільна здатність на цих картах ознак призводить до збільшення можливості виявлення об'єктів різного розміру.

Рис. 3.9. Результати роботи алгоритму вдень та вночі

Аналізуючи зображення, можна помітити, що тіні та якість зображення не впливають на продуктивність алгоритму, проте точність в підвищеній освітленості значно вища, ніж вночі.

Інші підходи глибокого навчання

Окрім SSD та YOLO, існують також й інші алгоритми, для прикладу Mask R-CNN, R-FCN, Retinanet тощо.

Порівняння підходів глибокого навчання до детектування

У таблиці нижче наведена порівняльна характеристика алгоритмів виявлення за найбільш критичними вимогами, а саме mAP (mean Average Precision) та FPS (Frames per Second). Представлені лише одноступінчасті алгоритми, так як двоступеневі алгоритми занадто повільні для детектування в реальному часі. Дані результати отримані на основі датасету MS COCO [29] і свідчать про те, що точність виявлення об'єктів в середньому однакова у всіх, окрім Tiny YOLO. Проте швидкодія значно відрізняється. Tiny YOLO навпаки лідер, саме тому він зазвичай застосовується у додатках, що потребують опрацювання даних в реальному часі.

Таблиця 3.2.

Порівняльна характеристика одноступінчастих детекторів

Модель	**mAP (mean average precision)**	**FPS (швидкість зміни кадрів за секунду)**
SSD500	46.5	19
YOLOv2	48.1	40
Tiny YOLO	23.7	244
R-FCN	51.9	12
Retinanet-101-500	53.1	11
YOLOv3-608	57.9	20
YOLOv3-416	55.3	35

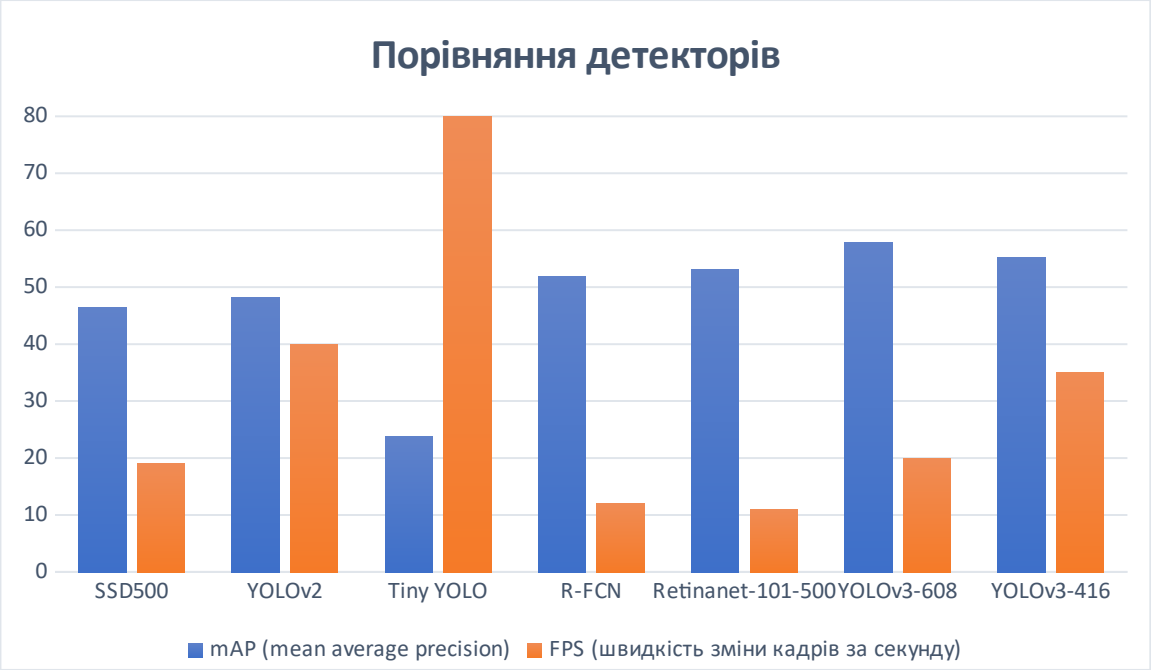

Рис. 3.10. Діаграма порівняння детекторів

Дані з таблиці 3.2 візуально представлені на діаграмі, що спрощує сприйняття інформації і дає змогу оцінити переваги та недоліки кожного.

Нижче в таблиці 3.3 представлені переваги та недоліки деяких алгоритмів на основі аналізу літературних джерел [30].

Таблиця 3.3.

Переваги та недоліки детекторів

Метод	Переваги	Недоліки
Fast R-CNN	Розрахунок характеристик CNN проводиться за одну ітерацію, досягаючи того, що виявлення об'єктів відбувається в	Використання зовнішнього генератора регіону-кандидата створює вузьке місце у процесі виявлення.

Продовження Таблиці 3.3.

	25 разів швидше, ніж метод R-CNN (для аналізу зображення потрібно в середньому 20 секунд).	
Faster R-CNN	Метод дозволяє виявляти об'єкти майже в режимі реального часу, приблизно 0,12 секунди на зображення.	Незважаючи на ефективність алгоритму, він недостатньо швидкий для використання в реальному часі.
R-FCN	Час тестування R-FCN набагато швидший, ніж R-CNN.	R-FCN має конкурентне значення mAP, але нижче, ніж Faster R-CNN.
Mask R-CNN	Локалізація об’єктів є більш точною, коли здійснюється сегментація об’єктів на зображеннях.	Час виконання швидший, порівнюючи з Faster R-CNN, проте не набагато, тому не може бути застосований у реальному часі.

Продовження Таблиці 3.3

YOLO	Локалізація об’єктів	Метод має труднощі з

	достатньо ефективна, що дозволяє використовувати його в реальному часі.	правильним виявленням дрібних предметів.
SSD	Локалізація об’єктів швидка, порівняно з Faster R-CNN.	Точність виявлення об'єктів нижча порівняно з методами Fast R-CNN та Faster R-CNN.

Для вибору детектора необхідно врахувати його можливість працювати з достатньо високим FPS, що даватиме змогу використовувати в реальному часі. У таблиці 3.4. [14] представлено порівняння ефективності різних моделей нейронних мереж на мікрокомп’ютері Raspberry Pi без та разом з Intel Neural Compute Stick 2.

Таблиця 3.4.

Порівняння ефективності різних моделей нейронних мереж на мікрокомп’ютері Raspberry Pi

Модель	**Raspberry Pi (use TensorFlow-Lite)**	**Raspberry Pi + Intel Neural Compute Stick 2**
MobileNet-v2	4.4 FPS (Pi 3) 8 FPS (Pi 4)	30 FPS (Pi 3)
MobileNet-v2 SSD	2.6 FPS (Pi 3) 4.7 FPS (Pi 4)	11 FPS (Pi 3) 41 FPS (Pi 4)

Продовження Таблиці 3.4.

ResNet-50	2.4 FPS (Pi 3) 4.3 FPS (Pi 4)	16 FPS (Pi 3) 60 FPS (Pi 4)

EfficientNet-80	14.6 FPS (Pi 3) 25.8 FPS (Pi 4)	95 FPS (Pi 3) 180 FPS (Pi 4)

З огляду на наведену інформацію щодо попередньо натренованої нейронної мережі та детекторів, було обрано MobileNet-v2 SSD у якості елементів детектування.

3.2. Класифікація

Класифікація транспортних засобів виникла як ще одне важливе застосування комп'ютерного зору та методів штучного інтелекту. Вона знайшла широке застосування в таких сферах, як спостереження, система безпеки, затори, запобігання аваріям тощо. Хоч і розроблено багато різноманітних алгоритмів класифікації, проте у даній роботі розглянуто лише два з них, SSD та YOLO.

Кращий спосіб оцінити ефективність класифікатора – переглянути матрицю плутанини [14]

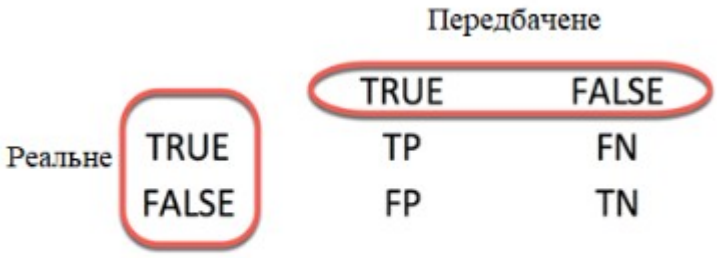

Рис. 3.11. Матриця плутанини

Матриця плутанини візуалізує точність класифікатора, порівнюючи фактичні і передбачувані класи. Двійкова матриця плутанини складається з квадратів:

TP: істинне позитивне – прогнозовані значення правильно прогнозуються як фактичні позитивні;

FP: хибне позитивне – прогнозовані значення неправильно передбачили фактичний позитив, а отже, негативні значення прогнозуються як позитивні;

FN: хибне негативне –позитивні значення прогнозуються як негативні;

TN: істинне негативне значення – прогнозовані правильні значення прогнозуються як фактичні негативні.

З матриці плутанини легко порівняти фактичний клас і передбачуваний клас. Матриця плутанини дає змогу добре зрозуміти істинне позитивне й хибне позитивне. У деяких випадках бажано мати більш стислу метрику.

Точність

Метрика точності показує точність позитивного класу. Вона вимірює ймовірність правильного прогнозу позитивного класу:

$$Точність = \frac{TP}{TP + FP} \quad (3.1)$$

Максимальний бал – 1, якщо класифікатор ідеально класифікує всі позитивні значення. Точність сама по собі не дуже корисна, оскільки вона ігнорує негативний клас. Метрику, як правило, поєднують з метрикою нагадування. Нагадування також називають чутливістю або справжньою позитивною швидкістю.

Чутливість

Чутливість обчислює відношення правильно визначених позитивних класів. Ця метрика засвідчує наскільки добре модель розпізнає позитивний клас:

$$Чутливість = \frac{TP}{TP + FN} \quad (3.2)$$

За даними метриками знаходять спочатку AP (Average Precision) усереднену точність, а потім і значення mAP.

3.3. Відстеження об’єктів (трекінг)

Відстеження об'єкта визначається як процес затримки рухомого об'єкту і можливість визначити, чи є об'єкт таким самим як у попередньому кадрі. Тобто, відстеження – це локалізація об’єкта в послідовних кадрах відео. У комп’ютерному зорі та машинному навчанні відстеження є достатньо широким терміном, який охоплює концептуально подібні, але технічно різні ідеї. Використання відстеження відео використовують для додатків з доповненою реальністю, систем спостереження та безпеки, редагування відео, контролю дорожнього руху та медичної візуалізації.

Метою відстеження є об'єднання цільових об'єктів у послідовних кадрах відео. Це об’єднання складно застосовувати, коли об’єкти рухаються швидко по відношенню до частоти кадрів відео. Все стає ще складніше, коли відстежувані об’єкти з часом змінюють свою орієнтацію. У цьому випадку системи відеоспостереження зазвичай використовують модель руху, яка детально описує, як об’єкт може виглядати в певних орієнтаціях.

Існує багато різних трекерів для задоволення різних потреб розробників. Деякі з них розглянуто нижче, такі як Перетин над об’єднанням (IoU) та Трекінг центроїда. Було обрано саме ці трекери, так як

обидва не використовують візуальні дані, а лише дані, отримані від детектора, що призводить до високої швидкодії.

IoU (Intersection over Union)

Автор цього алгоритму [31], в основному використовував IOU для порівняння обмежуючих рамок у двох послідовних кадрах для зіставлення об’єктів. Було зроблено припущення, що детектор здатний добре працювати в кожному кадрі. Саме тому припустили, що дві обмежувальні рамки, що належать одному об’єкту в двох послідовних кадрах, матимуть високу вірогідність перетину, а значить високий IOU. Враховуючи певний поріг IOU σ під час відстеження, трекер обчислює IOU між двома обмежувальними рамками – одна з нового кадру, а друга - з попереднього кадру - і визначає об’єкт з найкращим збігом IOU відповідно до порогового значення. Значення IOU обчислюється як:

$$IOU(a,b)=\frac{(a)\cap(b)}{(a)\cup(b)} \quad (3.3)$$

Оскільки автори не враховують жодної інформації про зображення в своєму алгоритмі, метод здатний перевершити інші своєю простотою, що призводить до надзвичайно високих швидкостей відстеження. Основна ідея даного підходу полягає в тому, що поточні детектори об’єктів є достатньо надійними, саме тому помилки детектування, спричинені хибнопозитивними/негативними виявленнями, можна ігнорувати. Виходячи з цього припущення, завдання відстеження кількох об’єктів, що слідує за парадигмою відстеження шляхом виявлення, стає тривіальним. Трекер IOU з’єднує виявлення наступних кадрів виключно шляхом їх просторового накладання на треки. У випадку, якщо будь-який виявлений об’єкт не був зіставлений з існуючим об’єктом на попередньому кадрі, тоді цей об’єкт

розпізнається як новий об’єкт відстеження. Проте проблема полягає в тому, що будь-яка існуюча послідовність відстежених кадрів для певного об’єкта закінчується, якщо на новому кадрі не було знайдено об’єкта. Хоч і детектори вважаються надійними, проте не ідеальні, що буде призводити до збільшеної кількості хибних обчислень щодо кількості транспортних засобів.

Трекінг центроїда

Трекінг центроїда [32] полягає у розрахунку Евклідової відстані між об’єктами, що знайдені на двох послідовних кадрах відео. Даний алгоритм складається з декількох кроків. Нижче розгорнуто наведені кожен з них.

Крок 1. Отримання набору обмежувальних рамок з координатами (x, y) для кожного об’єкта від детектора на кожному кадрі. Необхідно, щоб обмежувальні рамки обчислювались для кожного кадру у відео, або можна сказати, для кожного об’єкта, виявленого з відеокамери.

Крок 2. Обчислення саме центроїда (центра) об’єкта завдяки обмежувальним рамкам. Так як це початковий набір координат обмежувальних рамок, тому необхідно присвоїти унікальний ідентифікатор кожному знайденому об’єкту.

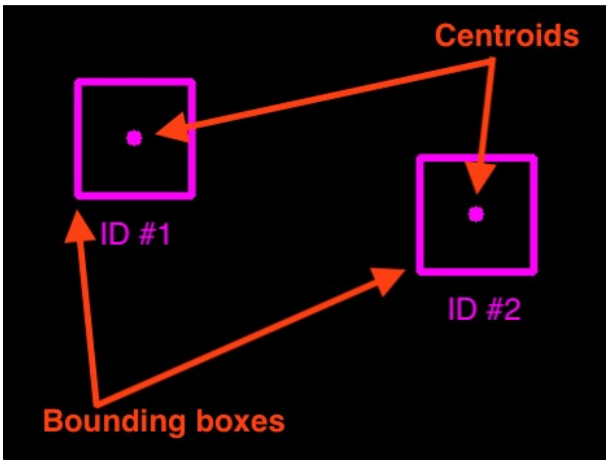

Рис. 3.12. Центроїди та обмежувальні рамки

Крок 3. Для кожного наступного кадру з обчисленими обмежувальними рамками, так само вираховується центроїд для кожного об’єкта. Після чого, обчислюється Евклідова відстань за формулою

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \quad (3.4)$$

від отриманого центроїда до всіх центроїдів, що були обраховані на попередньому кадрі.

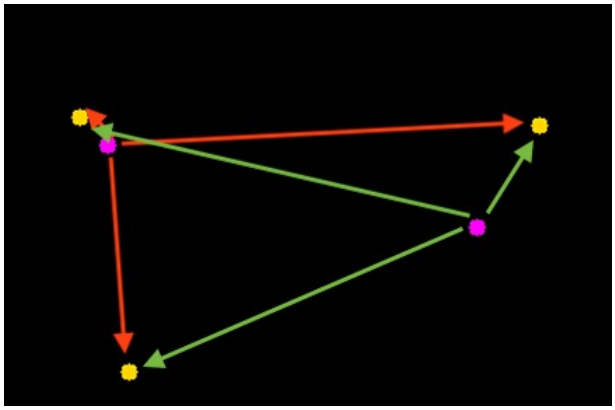

Рис. 3.13. Розрахунок евклідової відстані

Крок 4. Порівняння отриманих відстаней. Найменша відстань і є зсувом об’єкта від кадру до кадра, тому новий ідентифікатор не присвоюється, а лишається попередній і оновлюються координати обмежувальних рамок. Всі об’єкти, що залишились, маркуються ідентифікаторами.

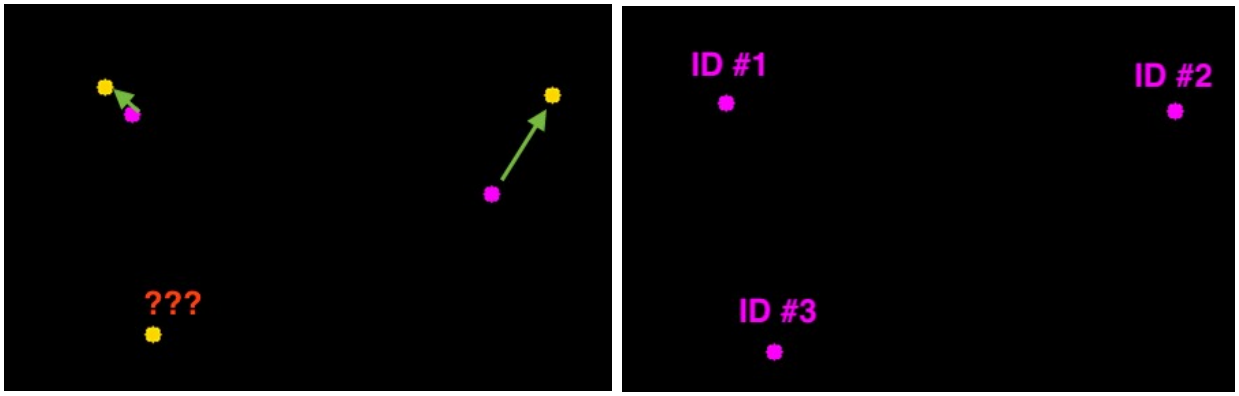

Рис. 3.14. Співставлення поточного центроїда з попереднім (ліворуч); присвоєння ідентифікаторів (праворуч)

Крок 5. У випадку зникнення об’єкту, необхідно видаляти ідентифікатор об’єкту через певну кількість послідовних кадрів.

Дані алгоритми можна впровадити за допомогою бібліотеки комп’ютерного зору OpenCV.

Крім того, відомі алгоритми бібліотеки OpenCV, такі як BOOSTING, MIL, KCF, MedianFlow, MOSSE, CSRT, можуть також застосовуватись для трекінгу, проте для кожного об’єкту створюється новий екземпляр трекеру, і тому при збільшенні об’єктів для трекінгу, сповільнюється час обробки. Через наведені причини було вирішено використовувати трекер центроїда.

Висновки:

1. Серед розглянутих алгоритмів складно виділити єдиний, що працюватиме у будь-якій ситуації, тому що кожен має як переваги, так і недоліки. Наприклад, алгоритми сімейства R-CNN і всі інші двоступеневі детектори не можуть бути застосовані у реальному часі, хоча точність детектування у них найбільша. А такі як SSD, YOLO та інші одноступеневі детектори поступаються точністю двоступеневим, проте швидкість обробки

– набагато більша. Саме тому, необхідно розглядати кожен випадок окремо і зважати на особливості поставлених задач.

2. З огляду на порівняння детекторів та попередньо натренованих нейронних мереж, у якості компонента детектування було обрано MobileNet-v2 SSD.

3. Розглянуто трекери, що не використовують інформацію про зображення, а лише ті, що базуються на інформації з детектора (координатах обмежувальних рамок), що зменшує використання обчислювальних ресурсів, оскільки вони дуже обмежені. Наведено аргументи проти використання вбудованих трекерів бібліотеки OpenCV.

4. Обрано трекер центроїда, так як є можливість налаштувати кількість кадрів, після яких трекер видаляє дані про об’єкт стеження, що дає змогу при неякісному детектуванні зберегти інформацію про рух об’єкта.

РОЗДІЛ 4.

ПРОГРАМНА ЧАСТИНА СИСТЕМИ АНАЛІЗУ ТРАНСПОРТНОГО ПОТОКУ

4.1. Блок-схема алгоритму детектування, трекінгу та підрахунку транспортних засобів

На блок-схемі (Рис. 4.1.) представлений алгоритм роботи програми і полягає він в наступному: відеопотік, що захоплюється камерою розкладається на кадри. У випадку, якщо кадр не вдалось виділити відбувається повторне зчитування, аби виконання програми не зупинялось. Після чого кожен кадр проходить через алгоритм детекції та трекінгу. Тобто, на одному кадрі запускається розпізнавання зображених об’єктів відповідних класів (транспортні засоби). Якщо місцеположення об’єкта не виявлено, тоді програма повертається на крок з детектуванням. Потім координати отриманих обмежувальних рамок об’єктів передаються на вхід трекера, завдяки чому обчислюється центроїд об’єкта в поточному кадрі і прораховується евклідова відстань. На основі отриманих даних, проводиться співставлення двох об’єктів на різних кадрах. Якщо ж, не вдалось співставити, отже, було виявлено новий об’єкт і йому присвоюється новий ідентифікатор. Далі координати отриманого центроїда аналізуються по значенню у, аби розпізнати, чи пересік об’єкт уявну лінію для підрахунку, і коли координата належить певному проміжку – означає, що об’єкт можна додавати до загальної кількості усіх транспортних засобів і зупиняти трекінг цього об’єкта, а інакше – проводити ті ж самі операції надалі.

Необхідно зауважити, що хоча і описано підрахунок одного транспортного засобу, проте розглядається масив об’єктів.

Також, для початку програми, очевидно, що попереднього кадру не існує, тому усім виявленим об’єктам присвоюється унікальний ідентифікатор.

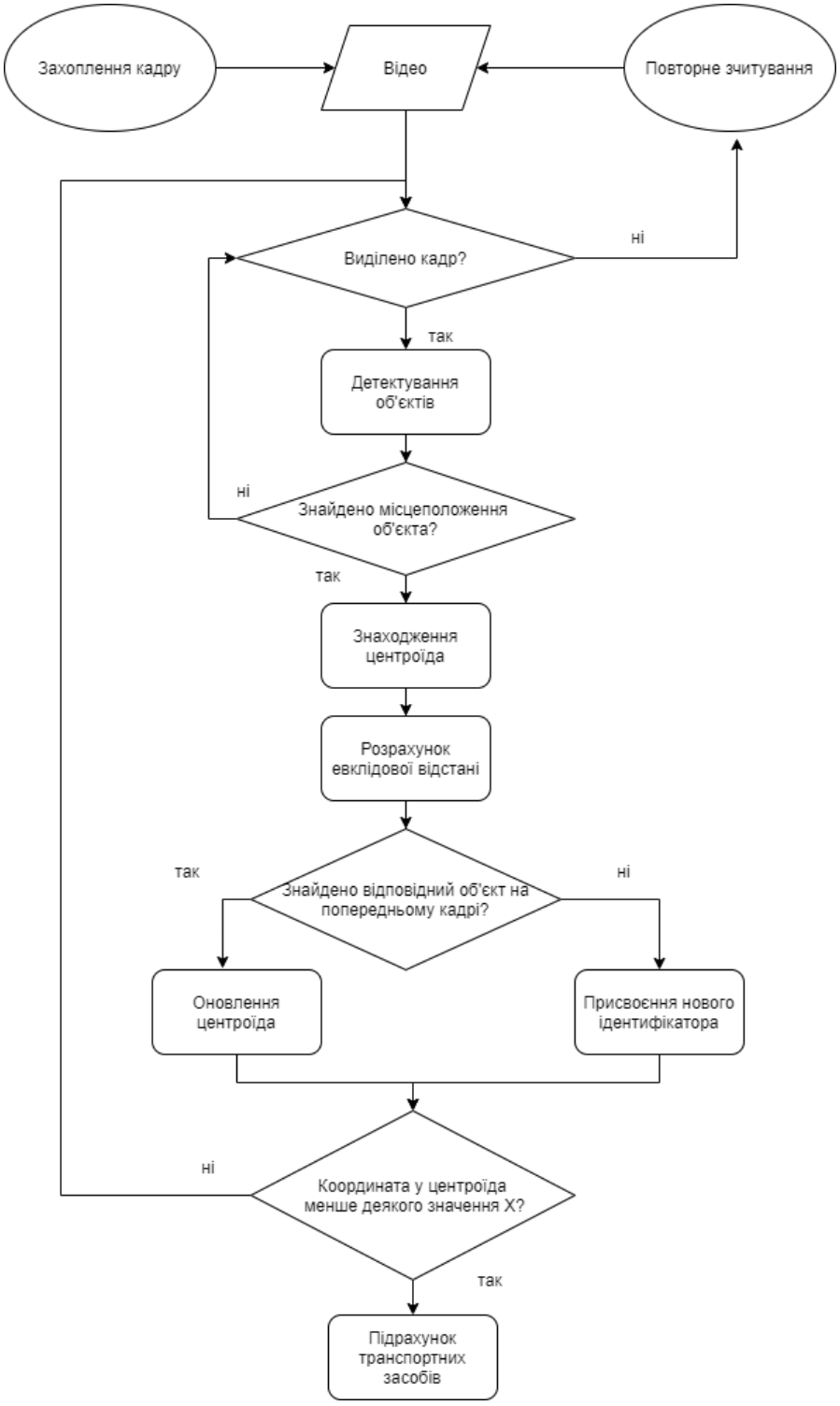

Рис. 4.1. Блок-схема алгоритму підрахунку транспортних засобів

4.2. Результати роботи програми

Для всіх детекторів використовуються стандартні моделі, попередньо навчені на шаблоні COCO [33], де класи включають автомобіль, вантажівку та автобус. Офлайн-методи у відстеженні за допомогою однієї камери зазвичай призводять до кращої продуктивності, оскільки всі агреговані дані можуть бути використані для асоціації даних. Онлайн підходи часто використовують неточні особливості зовнішнього вигляду, щоб компенсувати відсутність інформації про майбутнє.

Нижче представлені результати роботи програми, що використовує у якості детектора SSD з попередньо натренованою мережею MobileNet [34] та трекінгом центроїда.

На рисунках можна побачити порівняння результатів при різних умовах освітлення та оклюзії. Видно, що в сонячну погоду і з мінімальною кількістю транспортних засобів на проїзній частині справляється добре (Рис. 4.3.), проте точність і стеження, і детектування знижується при збільшенні транспортних засобів, що рухаються один за одним, так як вони перекриваються і трекер губиться (Рис. 4.4.).

Рис. 4.2. Вхідні кадри

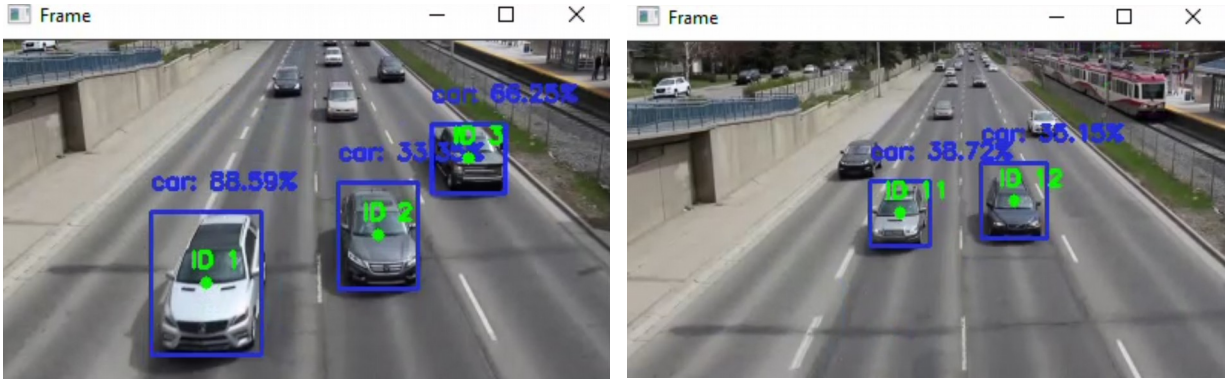

Рис. 4.3. Вихідні кадри

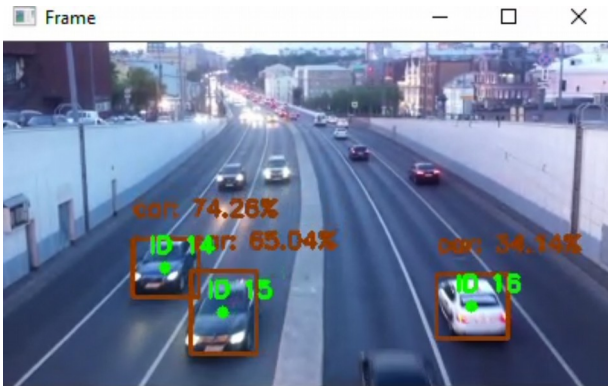

Рис. 4.4. Кадри роботи програми

4.3. Підхід до підрахунку кількості транспортних засобів

У даній роботі було припущено, що камера стежить лише за проїзною частиною, по якій транспортні засоби можуть рухатись в одному напрямку. Для того, щоб обчислити кількість транспортних засобів за певний проміжок часу, необхідно визначити уявну лінію на відеокадрах (необхідна калібровка камери при встановленні), при перетині якої лічильник буде фіксувати кількість автомобілів. Саме таким підходом, можна зменшити хибні спрацювання трекера, так як частина відеокадру буде використана лише для детектування і звичайного трекінгу, без впливу на кінцеву інформацію.

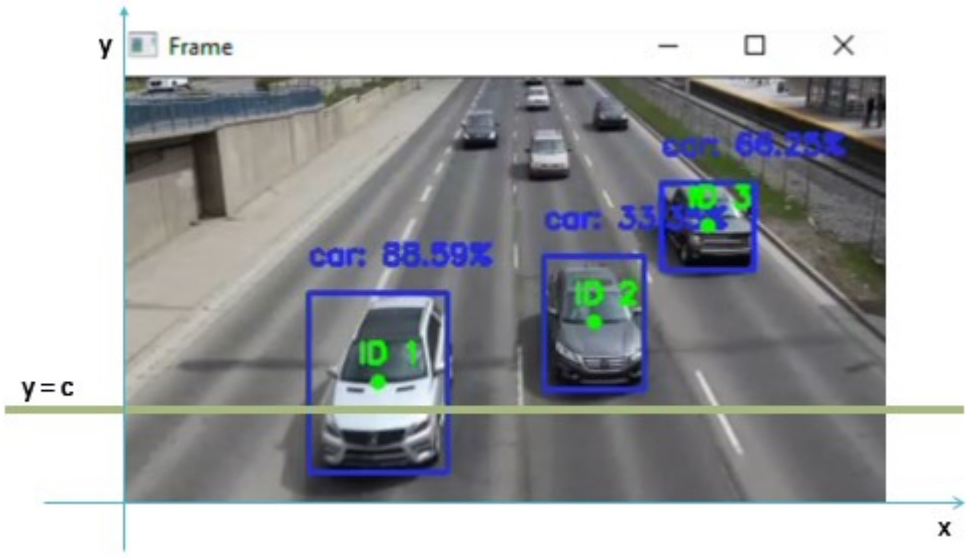

Рис.4.5. Підхід до підрахунку кількості транспортних засобів

Проте необхідно все ж таки враховувати, що точність підрахунку повністю залежить від трекеру, а трекер від детектора.

На вхідній послідовності початково зафіксовано 28 транспортних засобів. Запропонована програма підрахувала 21 автомобіль, що складає 75% точності. Так як алгоритм детектування одноступеневий, а, отже, заздалегідь було відомо зниження точності.

Також, якщо необхідно підраховувати кількість транспортних засобів в дві сторони, тоді потрібно додати ще одну функцію, яка буде порівнювати поточну координату центроїда y' з попередньою y. У випадку, якщо $y' > y$, отже, транспортний засіб рухається у прямому напрямку, а якщо $y' < y$, то у зворотному.

4.4. Межі використання запропонованої системи

Очевидно, що дана версія запропонованої системи не комерційна, і потребує вдосконалення. Проте отримані результати можна використовувати для подальших досліджень щодо здатності граничного пристрою працювати з більш складними, але більш точними нейронними мережами, а також трекерами, що використовують історію кадрів, аби передбачувати рух об'єктів. А також, дослідити інші пристрої апаратного забезпечення.

Отримані дані (кількість транспортних засобів) з граничного пристрою можна застосовувати для аналітики транспортних потоків, зберігаючи отримані дані в таємниці.

Висновки

1. Наведено блок-схему роботи алгоритму детектування, відстеження та підрахунку транспортних засобів.

2. Розглянутий підхід до обрахунку кількості транспортних засобів достатньо простий і зрозумілий, а також не потребує великої кількості обчислювальних ресурсів.

3. Результати роботи програми показують, що запропонований алгоритм справляється з поставленою задачею.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1. Розглянуто переваги та недоліки хмарних обчислень та обгрунтовано вибір на користь граничних обчислень.

2. У роботі проведено огляд та аналіз певних існуючих детекторів, на машинному навчанні та глибокому, та трекерів, найпростіших, що використовують лише інформацію, отриману від детектора.

3. Запропоновано систему в масштабованому варіанту, в якій обробка та аналіз даних проводиться на граничних пристроях.

4. Реалізована програмна частина та експериментально досліджено її результати.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A three-tier development model for digital transformation of citizen services [Електронний ресурс]: за даними PwC: Creating the smart cities of the future. – Режим доступу до публікації: https://www.pwc.com /us/en/industries/capital-projects-infrastructure/library/assets/pwc-future-smart-cities.pdf
2. Top 10 Growing Smart Cities [Електронний ресурс]: за даними The American Society of Mechanical Engineers / John Kosowatz / 03.02.2020 – Режим доступу до статті: https://www.asme.org/topics-resources/content/top-10-growing-smart-cities
3. Smart Traffic Management: Optimizing Your City's Infrastructure Spend [Електронний ресурс]: за даними Digi International / Nathan Wade / 10.04.2020 – Режим доступу до статті: https://www.digi.com /blog/post/smart-traffic-management-optimizing-spend
4. Traffic Index 2020 [Електронний ресурс]: за даними TomTom – Режим доступу до статті: https://www.tomtom.com/en_gb/traffic-index/ranking/
5. Офіційний сайт КП «Центр організації дорожнього руху» https://dorservice.kiev.ua/
6. Wei Yu. A Survey on the Edge Computing for the Internet of Things / Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, And Xinyu Yang. // Special section on mobile edge computing – 09.03.2018 – Режим доступу: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8123913
7. Cloud, Fog, and Edge Computing: A Software Engineering Perspective: 2018 International Conference on Computer and Applications (ICCA) / Amal Al-Qamash, Iten Soliman, Rawan Abulibdeh, Moutaz Saleh.

8. Edge Computing: Proceedings of the IEEE, vol. 107, no. 8, august 2019 / Weisong Shi, George Pallis, Zhiwei Xu.
9. M. Finnegan. Boeing 787s to create half a terabyte of data per flight, says Virgin Atlantic – Computerworld UK / 06.03.2013
10. What is edge computing and why it matters [Електронний ресурс]: за даними NetworkWorld / Keith Shaw / 13.11.2019 Режим доступу: https:// www.networkworld.com/article/3224893/what-is-edge-computing-and-how-it-s-changing-the-network.html
11. M. Zwolenski. The digital universe: Rich data and the increasing value of theInternet of Things / M. Zwolenski, L. Weatherill – Austral. J. Telecommun. Digit. Econ., vol. 2, no. 3, p. 47, 2014
12. Bajic Bojana. Edge Computing vs. Cloud Computing: Challenges and Opportunities in Industry 4.0 / Bajic Bojana. Cosic Ilija, Katalinic Branko, Moraca Slobodan, Lazarevic Milovan, Rikalovic Aleksandar / Proceedings of the 30th DAAAM International Symposium, 2019.
13. Why Everyone Should Try the Raspberry Pi 4: New Features and Impressive Specs [Електронний ресурс]: за даними MakeUseOf / Christian Cawley / 27.06.2019. Режим доступу: https://www.makeuseof.com/tag/raspberry-pi-4-overview/
14. Могильний С.Б. Машинне навчання з використанням мікрокомп'ютерів: навч.-метод. посіб. /за ред.О.В.Лісового та ін. –К., 2019. – 226с.
15. Tamrakar Yunik. Empirical Evaluation of Vehicle Detection, Tracking And Recognition Algorithms Operating On Real Time Video Feeds / Honors Theses – 09.05.2020

16. Nishu Singla. Motion Detection Based on Frame Difference Method / International Journal of Information & Computation Technology – ISSN 0974-2239 Volume 4, Number 15 (2014), pp. 1559-1565
17. Adam Schmidt. The Performance of the Haar Cascade Classifiers Applied to the Face and Eyes Detection / Adam Schmidt, Andrzej Kasinski / Computer Recognition Systems 2 – pp.816-823.
18. Jeong-ah Kim. Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition / Jeong-ah Kim, Ju-Yeong Sung, Se-ho Park / 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), 2020, pp. 1-4
19. Ross Girshick. Rich feature hierarchies for accurate object detection and semantic segmentation. / Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik / Proceedings of the IEEE conference on computer vision and pattern recognition, 2014 – pp. 580-587.
20. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks / Advances in neural information processing systems, 2015 – pp. 91-99.
21. Ross Girshick. Fast R-CNN / Proceedings of the IEEE International Conference on Computer Vision, 2015 – pp.1440-1448.
22. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection / Proceedings of the IEEE conference on computer vision and pattern recognition, 2016 – pp. 779-788.
23. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector / European conference on computer vision, 2016.

24. Jifeng Dai, Yi Li, Kaiming He, Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks / Neural Information Processing Systems (NIPS), 2016.
25. Dr. S. V. Viraktamath, Pratiksha Navalgi, Ambika Neelopant. Comparison of YOLOv3 and SSD Algorithms / International Journal of Engineering Research & Technology (IJERT), Volume 10, Issue 02, 2021.
26. Kanishk Wadhwa, Jay Kumar Behera, Accurate Real-Time Object Detection using SSD / International Research Journal of Engineering and Technology (IRJET), Vol. 7, Issue 05,2020
27. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection / Proceedings of the IEEE conference on computer vision and pattern recognition, 2016 – pp. 779-788
28. Jonathan Pedoeem, Rachel Huang. YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers / 2018 IEEE International Conference on Big Data (Big Data), 2018 – pp. 2503-2510.
29. YOLO: Real-Time Object Detection [Електронний ресурс]. Режим доступу: https://pjreddie.com/darknet/yolo/
30. S A Sanchez. A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework / IOP Conf. Ser.: Mater. Sci. Eng., 2020.
31. E. Bochinski, V. Eiselein, and T. Sikora. High-speed tracking-by-detection without using image information / 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) – pp. 1–6, 2017.
32. J. C. Nascimento, A. J. Abrantes and J. S. Marques. An algorithm for centroid-based tracking of moving objects / IEEE International Conference

on Acoustics, Speech, and Signal Processing. Proceedings, 1999, vol.6 – pp. 3305-3308.

33. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, C. Lawrence Zitnick. Microsoft COCO: Common objects in context / Proc. ECCV, pp 740–755, 2014.
34. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications / arXiv:1704.04861, 17.04.2017.
35. PyTorch for Beginners: Image Classification using Pre-trained models [Електронний ресурс]: за даними LearnOpenCV / Vishwesh Shrimali – 03.06.2019. Режим доступу: https://learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/
36. Introduction to Deep Learning [Електронний ресурс]: за даними Intel / Anna Petrovicheva – Режим доступу до презентації: https://delta-course.org/docs/delta7/ AIWorkshop/Introduction_to_Deep_Learning.pdf