Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розпізнавання автомобілів в реальному часі
за допомогою комп'ютерного зору і машинного навчання
(тема)

Виконав:
студент _2_ курсу, групи _САУм-18-1
Столяренко К.С.
(прізвище, ініціали)
Спеціальність
124 Системний аналіз
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Системний аналіз і управління
(повна назва освітньої програми)
Керівник проф. Кіріченко Л.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Тевяшев А.Д.
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _124 Системний аналіз
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Системний аналіз і управління
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
"____" _____ 2019 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Столяренко Костянтину Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи _Розпізнавання автомобілів в реальному часі
за допомогою комп'ютерного зору і машинного навчання

затверджена наказом по університету від 31 жовтня 2019 р. № 1601 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 9 грудня 2019 р.

3. Вихідні дані до роботи _цифрові зображення

4. Перелік питань, що потрібно опрацювати в роботі
1. Системний аналіз проблеми розпізнавання рухомих об'єктів
у реальному часі за допомогою технологій комп'ютерного зору
і машинного навчання
2. Вибір і обґрунтування методу розв'язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Системний аналіз проблеми
4. Мета і задачі
5. Технології аналіза та ідентифікації рухомих об'єктів
6. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	вересень 2019 р.	виконано
2	Вибір та обґрунтування методу	жовтень – листопад 2019 р.	виконано
3	Розробка алгоритму і програми	листопад – грудень 2019 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	листопад – грудень 2019 р.	виконано
5	Робота над текстом пояснювальної записки	грудень 2019 р.	виконано
6	Представлення роботи на рецензію в ЕК	грудень 2019 р.	виконано

Дата видачі завдання _2_ вересня 2019 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Кіріченко Л.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 96 с., 19 рис., 7 табл., 1 додатки, 17 джерел.

МАШИННЕ НАВЧАННЯ, КОМП’ЮТЕРНИЙ ЗІР, НЕЙРОННА МЕРЕЖА, VISION, CORE ML, РЕАЛЬНИЙ ЧАС, РУХОМІ ОБ'ЄКТИ, АВТОМОБІЛІ, ПІШОХОДИ.

Об’єкт дослідження – технології комп'ютерного зору і машинного навчання.

Мета роботи – реалізація програми ідентифікації рухомих об'єктів у реальному часі, а так само застосування технологій машинного розпізнавання і комп'ютерного зору.

Метод дослідження – технологія Vision та Core ML.

У роботі проведений системний аналіз проблеми розпізнавання рухомих об'єктів у реальному часі за допомогою технологій комп'ютерного зору і машинного навчання. Так само були зібрані великі обсяги даних для створення і тренування моделі розпізнавання, що ідентифікує об'єкти класів людина і автомобіль.

За допомогою мови програмування Swift був створений програмний продукт для мобільного пристрою, який в реальному часі виявляє рухомі об'єкти такі як автомобілі і люди, та проведений порівняльний аналіз відповідних методів.

В результаті на основі проведених випробувань, була обрана оптимальна технологія, а так само поліпшена швидкість і точність розпізнавання фінальної програми.

ABSTRACT

Introductory note: 96 pages, 7 tables, 19 figures, 1 appendixes, 17 sources.

MACHINE LEARNING, COMPUTER VISION, NEURAL NETWORK, VISION, CORE ML, REAL TIME, MOVEMENTS, VEHICLES, PEDESTRIANS.

Object of research – technology of computer vision and machine learning.

Purpose of work – implementation of the program of identification of moving objects in real time, as well as the use of machine recognition technologies and computer vision.

Methods of research – Vision and Core ML technologies.

System analysis of the problem of recognition of moving objects in real time was implemented by using computer vision and machine learning technologies. Large amounts of data have also been collected to create and train a identification model that recognize objects of the classes of man and car.

Using the Swift programming language was created software product for mobile device that detects moving objects such as cars and people in real time, and benchmarked the methods.

As a result, based on the tests, was chosen optimum technology as well as improved speed and accuracy of recognition of the final program.

ЗМІСТ

ВСТУП

З недавніх пір в інформаційних технологіях почав набирати популярність вже існуючий досить давно в науці напрямок машинного навчання.

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

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

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

Будучи великим підрозділом штучного інтелекту, машинне навчання використовує в своїй основі моделі нейронних мереж. Машинне навчання виходить зі сфери математиків і алгоритмістів, і все глибше проникає в багато сфер сучасного світу.

Так про машинне навчання говорять уже в IT-сфері – розпізнавання зву-

ків і зображень; рекламні компанії – передбачення втрати клієнтів, маркетингові дослідження; медична діагностика – аналізуючи історії хвороб пацієнтів, можна виявляти непомітні для людини зв'язку і встановлювати невідомі раніше симптоми небезпечних захворювань; технічна діагностика та біоінформатика. Прогнозування покупок користувачів цілком покладається на нейронні мережі та дерева рішень з використанням даних з інтернет-магазину [2].

Найважливіше автомобільна індустрія – автопілоти та додаткові програми які бачать усіх учасників дорожнього руху та допомогають водієві.

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

У сучасному світі все частіше і актуальніше стає питання розробки і застосування технологій машинного інтелекту для вирішення численних завдань. Саме тому виникає потреба в створенні автоматизованого механізму допомоги людині у всіх сферах її життєдіяльності. Саме тому в даній роботі буде розглянуто один з найперспективніших напрямків інтелектуального аналізу – машинне навчання і комп’ютерний зір.

Мета даної атестаційної роботи полягає в розробці програмного забезпечення, використовуючи технологію машинного навчання та засоби копьютерного зору, для ідентифікації рухомих об’єктів у реальному часі, таких як автомобілі і пішоходи які є головними учасниками дорожнього руху.

1 СИСТЕМНИЙ АНАЛІЗ ПРОБЛЕМИ РОЗПІЗНАВАННЯ РУХОМИХ ОБ'ЄКТІВ У РЕАЛЬНОМУ ЧАСІ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЙ КОМП'ЮТЕРНОГО ЗОРУ І МАШИННОГО НАВЧАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Системний аналіз проблеми розпізнавання рухомих об'єктів у реальному часі за допомогою технологій комп'ютерного зору і машинного навчання

1.1.1 Вербальна модель системи

Об’єктом аналізу є технології комп'ютерного зору і машинного навчання які застосовуються для ідентифікації рухомих об'єктів у реальному часі. За допомогою цих технологій можна розширити функціонал вже існуючої програми або розробити цілком нову яка буде допомагати людині у багатьох сферах її життєдіяльності.

У сучасному світі все частіше і актуальніше стає питання розробки і застосування технологій машинного інтелекту для вирішення численних завдань, тому в даній роботі буде розглянуто один з найперспективніших напрямків інтелектуального аналізу – машинне навчання і комп`ютерної зір.

Машинне навчання є одним з видів штучного інтелекту, де комп'ютери «вчаться самі», будучи ще не запрограмованими на дії, які вони вміють, після свого «навчання».

На основі машинного навчання дещо пізніше з'явилося поняття комп'ютерного зору. Комп'ютерний зір застосовується для розпізнавання об'єктів та для інтелектуальної обробки всього, що можна побачити людським оком.

Для навчання використовують тренувальні моделі. Дані для створення моделі можуть бути отримані декількома способами: з якого-небудь електронного ресурсу за допомогою спеціальних програмних інструментів або викорис-

танням даних деяких дослідницьких фірм у відкритому доступі. У даній роботі дані будуть отримані саме першим способом.

Дослідження в цій області вимагають уважності та правильного аналізу отриманих результатів.

Технології комп'ютерного зору і машинного навчання широко використовуються у ряді програм та дисциплін.

Так, наприклад, машинне навчання використовують для маркетингових досліджень, медичної діагностики людини, а також розпізнавання звуків і зображень. Більш детально зображеннями займаються технології комп'ютерного зору, наприклад для створення автомобільних автопілотів які "бачать" розмітку дороги і пішоходів, а також інші автомобілі і не дозволяють відбутися аварії на дорозі.

Сфера застосування даних технологій вкрай велика і продовжує збільшуватися з кожним днем, на хвилях технологічного прогресу.

1.1.2 Морфологічний опис системи

Модель «чорний ящик» – модель досліджуваної системи, що зосереджена на досліджені реакції системи, як цілого, на зміни зовнішнього середовища (рис. 1.1). Система є максимально простою і відображає входи та виходи досліджуваної системи. Цей метод дослідження системи найбільш підходить при виявленні реакції системи на входи, що надходять до неї. І хоча цей ящик є відокремленим, але він не є повністю ізольованим. Система пов’язана з навколишнім середовищем та за допомогою цих зв’язків впливає на середовище. Зв’язки, що направлені від системи до середовища називають виходами системи, а навпаки, із середовища до системи – входами. Тому для побудови такої моделі необхідно вказати лише входи та виходи [3].

Назва «чорний ящик» повністю підкреслює повну відсутність інформації про внутрішню складову системи. Така модель, що на перший погляд здається

достатньо простою, часто є дуже корисною, але ж і тривіальною задачею її назвати складно, тому що не завжди зрозуміло у якій кількості та які саме параметри необхідно включати в модель.

Дослідження за допомогою метода «чорного ящика» полягає в тому, що відбувається попереднє дослідження за взаємодією системи з навколишнім середовищем та встановлюються усі вхідні та вихідні параметри, серед яких виділяють найбільш суттєві джерела впливу. Потім відбувається вибір входів та виходів для дослідження з врахуванням необхідних засобів впливу на систему та засобів контролю та нагляду за поведінкою системи.

На наступному етапі проводяться впливи на вхід системи та регістрація її виходів. У процесі вивчення досліджувач та «чорний ящик» утворюють систему зі зворотним зв’язком, а первинні результати дослідження – множина пар станів входу й виходу, аналіз яких дозволяє встановити між ними причинно-наслідковий зв'язок.

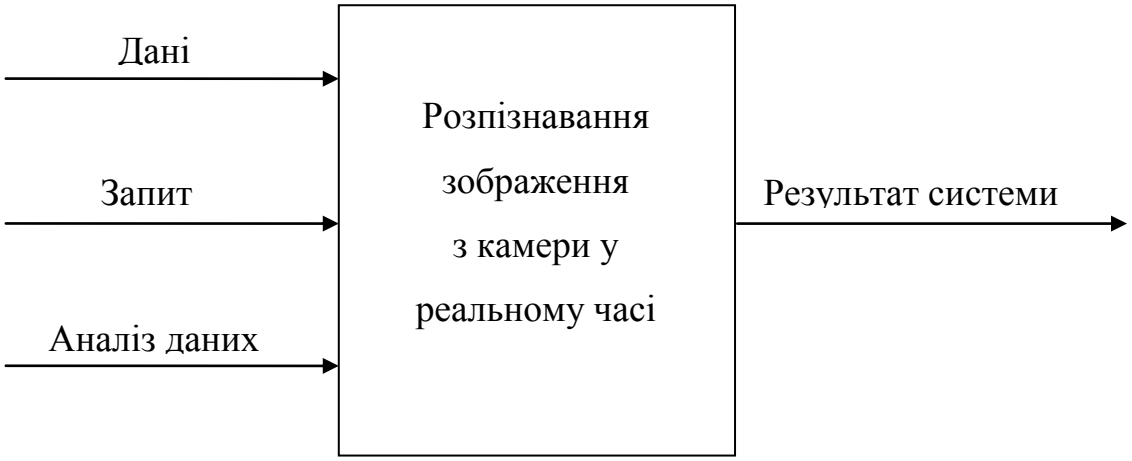

Рисунок 1.1 – Модель «чорний ящик»

1.1.3 Функціональна модель системи

Функціональна модель системи може бути представлена графічно за допомогою контекстної діаграми IDEF0. У такому випадку система постає у вигляді взаємодіючих функцій, або, інакше кажучи, функціональних блоків. Спочатку проводиться загальний опис системи, після чого відбувається функціона-

льна декомпозиція, тобто розбиття її на більш детальніші діаграми, які в свою чергу теж можуть бути декомпоновані. В результаті застосування IDEF0 до поточної системи отримаємо модель цієї системи, що складається з ієрархічно впорядкованої множини діаграм. Подібне дроблення проводиться до тих пір, допоки не буде досягнутий потрібний рівень детального опису. На рисунку 1.2 зображена загальна функціональна модель системи із зовнішніми зв’язками.

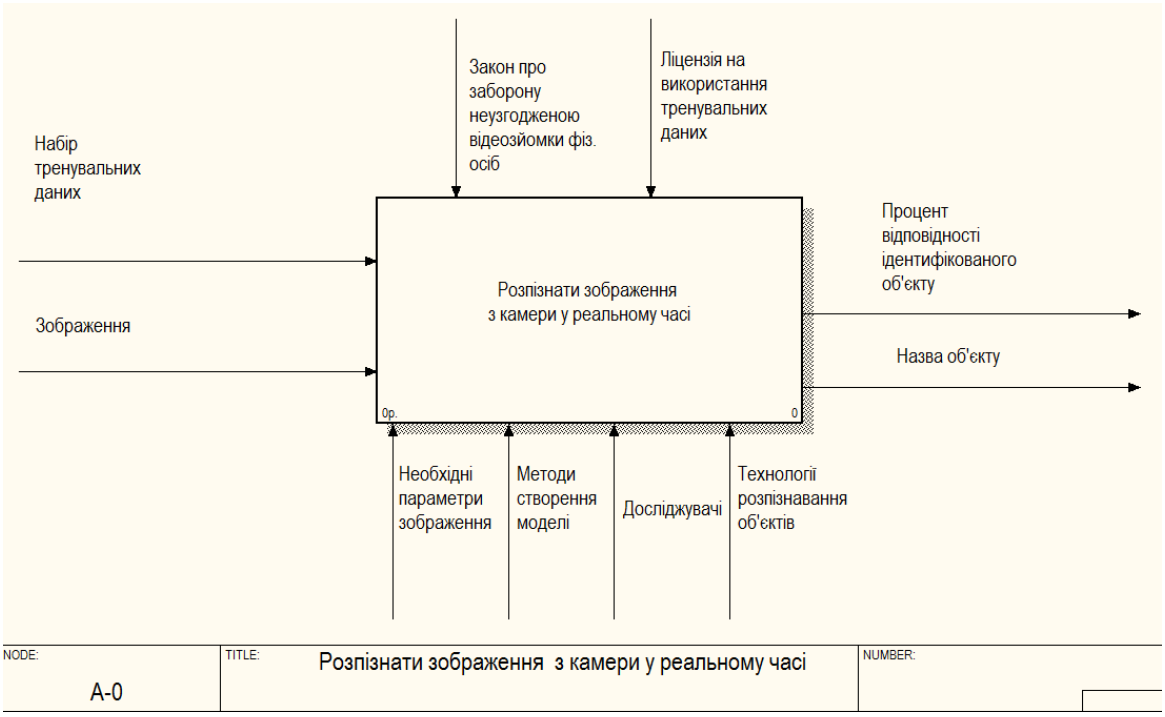

Рисунок 1.2 – Контекстна IDEF0 діаграма

Рисунок 1.3 подає декомпозицію минулої діаграми на 3 функціональні блоки, між якими послідовні прямі зв’язки.

1.1.4 Інформаційна модель системи

Інформаційна модель системи також може бути представлена графічно за допомогою діаграми потоків даних DFD (рис. 1.4). Особливістю даного типу діаграм є відображення «відправників» та «отримувачів» даних, визначення відношень між процесами, що зв’язують у потоки декілька функціональних бло-

ків між собою. Аналогічно побудові функціональної моделі, після реалізації контекстної діаграми при необхідності відбувається детальніша декомпозиція окремих елементів для кращого розглядання компонентів системи.

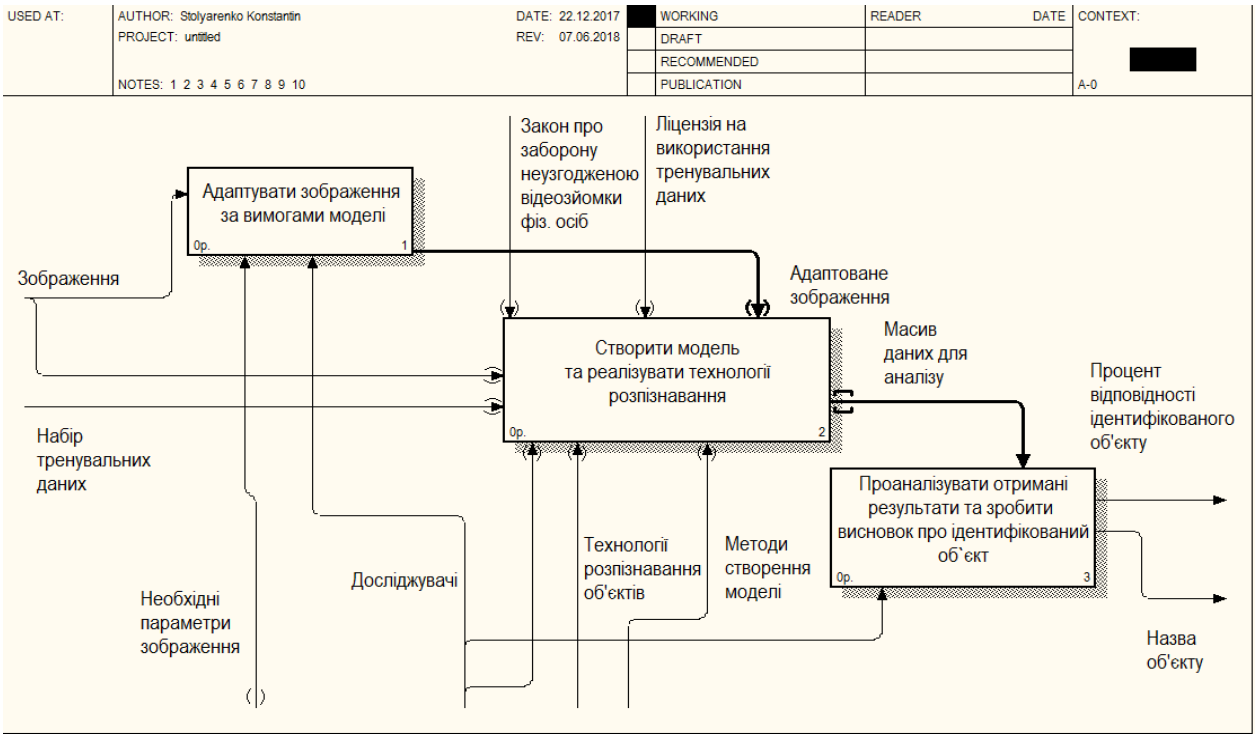

Рисунок 1.3 – Декомпозиція першого рівня функціонального блоку «Розпізнавання зображення з камери у реальному часі»

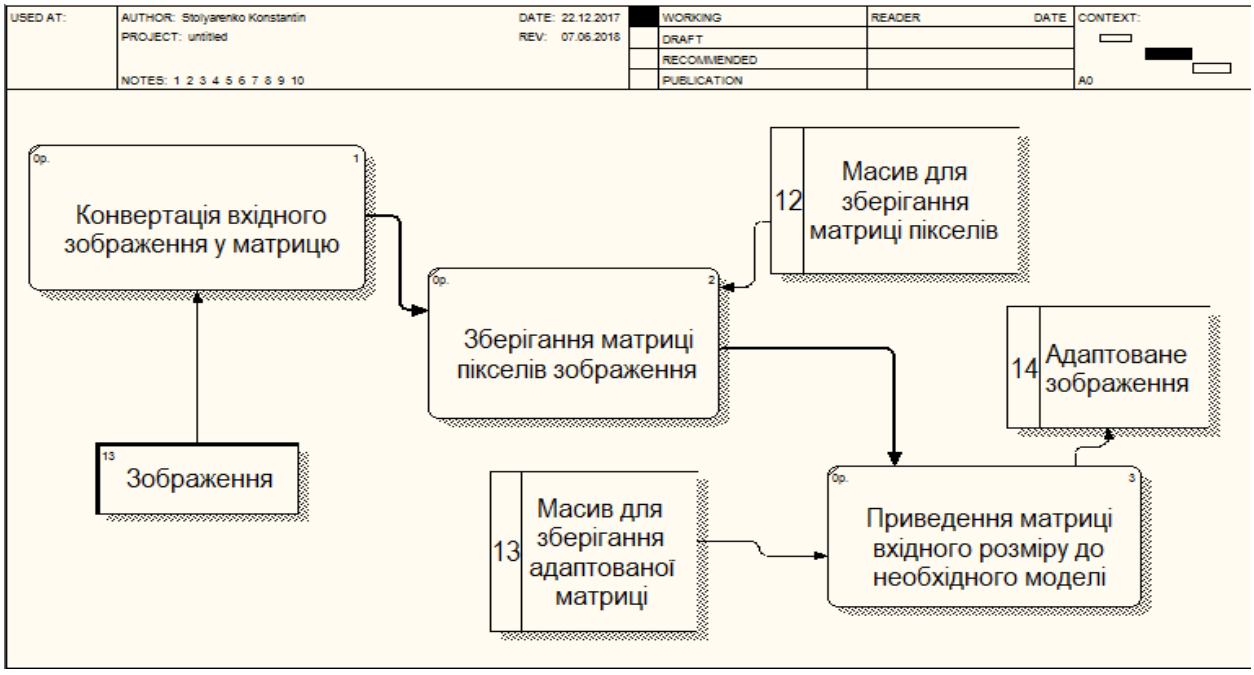

Рисунок 1.4 – Контекстна DFD діаграма

Наявність в діаграмах DFD елементів для опису джерел, приймачів і сховищ даних дозволяє більш ефективно і наглядно описати процес документообігу [4]. Однак для опису логіки взаємодії інформаційних потоків більш підходить стандарт опису процесів IDEF3, званий також workflow diagramming – методологією моделювання, що використовує графічний опис інформаційних потоків, взаємин між процесами обробки інформації та об'єктів, що є частиною цих процесів. Діаграми workflow можуть бути використані в моделюванні бізнес-процесів для аналізу завершеності процедур обробки інформації. За допомогою них можна описати сценарії дій співробітників організації. Кожен сценарій супроводжується описом процесу і може бути використаний для документування кожної функції.

IDEF3 – це метод, основна мета якого описати ситуацію, коли процеси виконуються в певній послідовності, а також описати об'єкти, які беруть участь спільно в одному процесі.

За для більш детального моделювання опишемо послідовність операцій у невизначених процесах, скориставшись ще однією методологією графічного структурного аналізу – діаграмою IDEF3 (рис. 1.5).

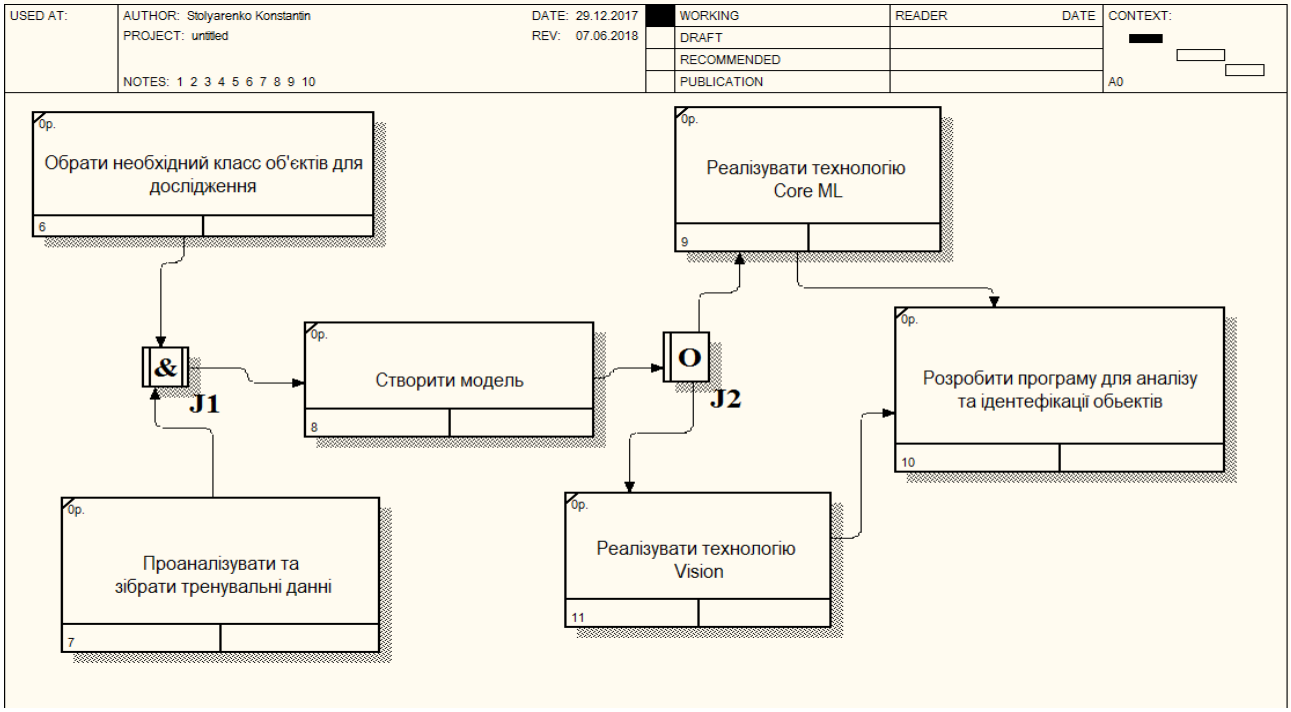

Рисунок 1.5 – IDEF3 діаграма

1.2 Аналіз технологій вирішення проблеми розпізнавання рухомих об'єктів у реальному часі за допомогою технологій комп'ютерного зору і машинного навчання

1.2.1 Модель аналізу проблеми

Об’єкт дослідження – модель порівняльного аналізу технологій розпізнавання рухомих об'єктів у реальному часі.

Мета дослідження – виявити оптимальний варіант реалізації та отримання якісного результату.

Математична модель має бути вирішена при застосуванні різних алгоритмів з подальшим їх порівнянням з точки зору якості технологій розпізнавання об'єктів та особливостей програмної реалізації. В результаті очікується визначення найбільш оптимального алгоритму, який може бути застосований для вирішення даної задачі.

Виділимо критерії, які на наш погляд здійснюють найбільший вплив на очікуваний результат. Такими критеріями є:

– складність реалізації (К1);

– сучасність (К2);

– здатність до змін (К3);

– функціонал розпізнавання (К4).

Розглянемо вищевказані критерії більш детально та конкретизуємо що саме буде порівнюватися у запропонованих далі альтернативах. В свою чергу, альтернативами будуть обрані: технологія комп’ютерного зору Vision та технологія машинного навчання Core ML.

Порівнюючи складність реалізації альтернатив, маємо на увазі час, рівень знань у даній області та необхідний досвід програмування який буде необхідній для успішної реалізації програми без технічних помилок.

Порівнюючи альтернативи на сучасність, нас цікавить частота оновлення технологій компанією розробником та передовий підхід до реалізації функціоналу.

Здатність до змін полягає в зручній і простій можливості розробників доповнювати і покращувати вже існуючий продукт.

При порівнянні альтернативи на функціонал розпізнавання, перевага віддаватиметься алгоритму, який має найбільшу кількість реалізованих функцій і можливостей для розпізнавання об`єктів.

Будуть розглянуті такі альтернативи:

– технологія Vision (A1);

– технологія Core ML (A2);

– технологія OpenCV (A3).

Відомо, що технологія Vision – це набір інструментів, які допомагають налаштувати весь функціонал обробки зображень. Серед цих інструментів – CoreML який взаємодіє з натренованою моделлю. Vision допомагає препроцесувати, змінювати розмір, обрізати зображення, виявляти прямокутники, штрих-коди, особи і багато іншого. Крім роботи, що виконується безпосередньо на зображеннях, вона також допомагає виконувати запити до моделі, що дуже важливо, якщо в програмі складна послідовність запитів для обробки.

З чистим CoreML розробнику доведеться реалізувати всі ці функції самостійно, тому що завдання CoreML тільки в налаштуванні моделі і надання повного доступу для взаємодії з нею.

Vision не є просто обгорткою навколо CoreML, тому що це ця технологія набагато ширше, ніж запит на виконання і ініціалізацію моделей, але вона використовує CoreML для деяких своїх функціональних можливостей.

Технологія OpenCV включає в себе алгоритми комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Реалізована на мові C/C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов. Є вкрай непростою для реалізації і не властива для мобільних пристроїв в силу вимог до обчислювальної техніки.

На рисунку 1.6 зображена ієрархічна модель процесу аналізу незадоволеностей.

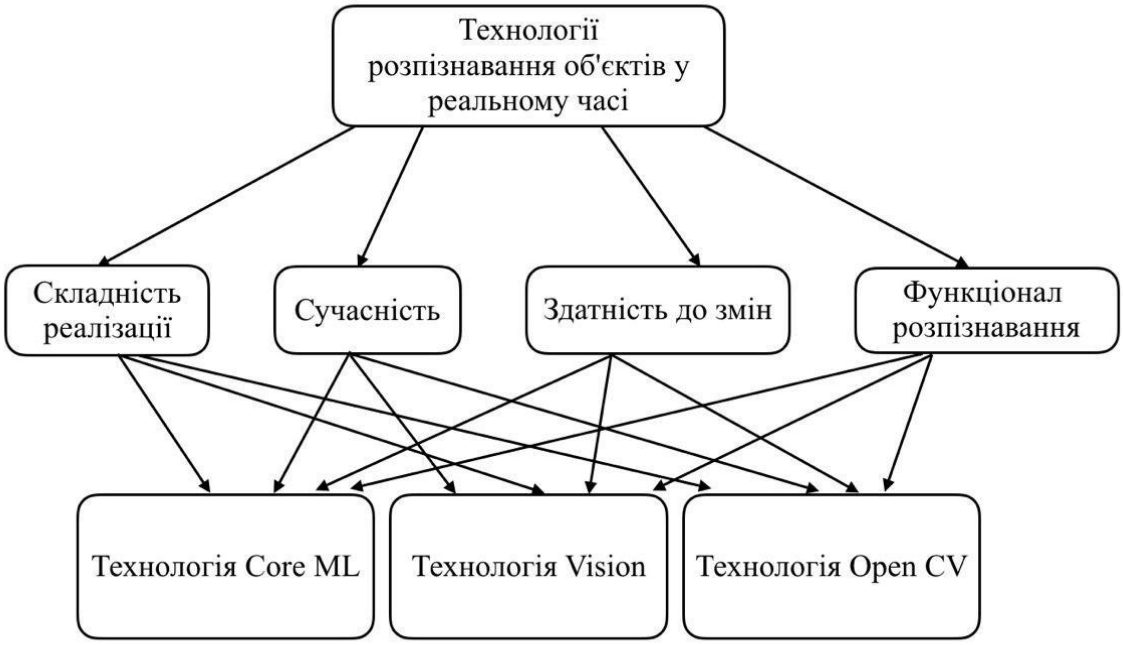

Рисунок 1.6 – Ієрархічна модель процесу аналізу незадоволеностей

1.2.2 Оцінювання вектора пріоритетів незадоволеностей методом аналізу ієрархій

За допомогою методу попарних порівнянь побудуємо модель процесу аналізу незадоволеностей. Аналіз незадоволеностей включає в себе такі рівні:

– нульовий рівень – компоненти проблеми;

– перший рівень – класифікація незадоволеностей;

– другий рівень – характеристики компонентів, які впливають на результат поставленої задачі.

На рисунку 1.6 представлений нульовий рівень проблеми.

На першому рівні аналізу проблеми побудуємо матрицю попарних порівнянь критеріїв з метою оцінки впливу кожної незадоволеності на поставлену проблему. Результати наведені в таблиці 1.1. для встановлення відносної важливості критеріїв використана шкала відношень Сааті.

Таблиця 1.1 – Матриця попарних порівнянь критеріїв

	К1	К2	К3	К4	Вектор пріоритетів
К1	1	5	7	8	0,65246
К2	1/5	1	3	4	0,19853
К3	1/7	1/3	1	3	0,09806
К4	1/8	1/4	1/3	1	0,05096

Для знаходження індексу узгодженості знаходимо суми елементів матриці за стовбцями:

$$y_1 = 1 + 1/5 + 1/7 + 1/8 = 1,4679,$$

$$y_2 = 5 + 1 + 1/3 + 1/4 = 6,58333,$$

$$y_3 = 7 + 3 + 1 + 1/3 = 11,333,$$

$$y_4 = 8 + 4 + 3 + 1 = 16.$$

Тоді $\lambda_{\max} \approx 4,19131$ та індекс узгодженості $CI^k = 0,063$.

Оскільки матриця попарних порівнянь критеріїв – це матриця четвертого порядку, то відношення узгодженості:

$$CR^k = \frac{CI^k}{0,9} = 0,0708.$$

Оскільки відношення узгодженості є близьким до 0,1, то вважатимемо, що матриця попарних порівнянь критеріїв побудована правильно.

Далі формуємо матриці попарних альтернатив за кожним критерієм з метою порівняння методів між собою за кожним критерієм окремо.

Результати наведені в таблицях 1.2. – 1.5.

Таблиця 1.2 – Матриця порівнянь за критерієм К1

К1	А1	А2	А3	Вектор пріоритетів
А1	1	5	9	0,72826
А2	1/5	1	6	0,21757
А3	1/9	1/6	1	0,05417

Для знаходження індексу узгодженості знаходимо суми елементів матриці за стовбцями:

$$y_1 = 1 + 1/5 + 1/9 = 1,3111,$$

$$y_2 = 5 + 1 + 1/6 = 6,11667,$$

$$y_3 = 9 + 6 + 1 = 16.$$

Тоді $\lambda_{\max} \approx 3,16323$ та індекс узгодженості $CI^A_{K1} = 0,0816$.

Оскільки матриця попарних порівнянь альтернатив – це матриця третього порядку, то відношення узгодженості:

$$CR^A_{K1} = \frac{CI^k}{0,58} = 0,140719.$$

Таблиця 1.3 – Матриця порівнянь за критерієм К2

К2	А1	А2	А3	Вектор пріоритетів
А1	1	3	7	0,64919
А2	1/3	1	5	0,27896
А3	1/7	1/5	1	0,07193

Для знаходження індексу узгодженості знаходимо суми елементів матри-

ці за стовбцями:

$$y_1 = 1 + 1 / 3 + 1 / 7 = 1,47619,$$

$$y_2 = 3 + 1 + 1 / 5 = 4,2,$$

$$y_3 = 7 + 5 + 1 = 13.$$

Тоді $\lambda_{\max} \approx 3,06489$ та індекс узгодженості $CI^A_{K2} = 0,032444$.

Відношення узгодженості:

$$CR^A_{K2} = \frac{CI^k}{0,58} = 0,055938.$$

Таблиця 1.4 – Матриця порівнянь за критерієм К3

К3	А1	А2	А3	Вектор пріоритетів
А1	1	1/4	1/6	0,07911
А2	4	1	1/5	0,21184
А3	6	5	1	0,70905

Для знаходження індексу узгодженості знаходимо суми елементів матриці за стовбцями:

$$y_1 = 1 + 4 + 6 = 11,$$

$$y_2 = 1 / 4 + 1 + 5 = 6,25,$$

$$y_3 = 1 / 6 + 1 / 5 + 1 = 1,36667.$$

Тоді $\lambda_{\max} \approx 3,16323$ та індекс узгодженості $CI^A_{K3} = 0,0816$.

Відношення узгодженості:

$$CR_{K3}^{A} = \frac{CI^{k}}{0,58} = 0,14072.$$

Таблиця 1.5 – Матриця порівнянь за критерієм К4

К4	А1	А2	А3	Вектор пріоритетів
А1	1	7	5	0,72229
А2	1/7	1	1/4	0,07272
А3	1/5	4	1	0,20498

Для знаходження індексу узгодженості знаходимо суми елементів матриці за стовбцями:

$$y_1 = 1 + 1/7 + 1/5 = 1,34286,$$

$$y_2 = 7 + 1 + 4 = 12,$$

$$y_3 = 5 + 1/4 + 1 = 6,25.$$

Тоді $\lambda_{\max} \approx 3,12371$ та індекс узгодженості $CI_{K4}^{A} = 0,0618$.

Відношення узгодженості:

$$CR_{K4}^{A} = \frac{CI^{k}}{0,58} = 0,106649.$$

Розрахуємо вектор глобальних пріоритетів альтернатив

$$\vec{p} = \begin{bmatrix} 0,64861 \\ 0,25517 \\ 0,12960 \end{bmatrix}.$$

Розрахуємо індекс узгодженості та відношення узгодженості для всієї ієрархії:

$$CI = 0{,}13462,$$
$$RI = 0{,}90 + 0{,}58 = 1{,}48,$$
$$CR = \frac{CI}{RI} = 0{,}09096,$$

що теж можна вважати доброю узгодженістю.

Найбільша компонента вектора локальних пріоритетів критеріїв відповідає першому критерію. Отже, маємо наступні пріоритети за критеріями порівняння: складність реалізації, сучасність, здатність до змін, функціонал розпізнавання.

Порівнюючи альтернативи за обраними критеріями, отримали вектор глобальних пріоритетів, найбільша компонента якого відповідає першій альтернативі – технології Vision.

1.3 Змістовна та формальна постановка задачі

1.3.1 Змістовна постановка задачі

Комп'ютерний зір (КЗ) – це технологія, за допомогою якої машини можуть знаходити, відстежувати, класифікувати та ідентифікувати об'єкти, витягуючи дані з зображень і аналізуючи отриману інформацію. КЗ основанно на нейронних мережах і використовує методи машинного навчання для того що б поліпшити якість життя користувача в різних сферах життя, таких як: автомобільна індустрія, робототехніка, медицина, розваги і весь IT сектор.

У сучасному світі актуальність застосування машинного інтелекту, а саме комп'ютерного зору просто неоціненна. Сучасні заводи по виробництву різної

продукції, засоби пересування, асистенти і роботи помічники просто не обходяться без технологій КЗ.

З кожним роком розробники з усього світу впроваджують нові технології та алгоритми, для того що б навчити машини бачити краще і якісніше. Кожен етап поліпшення КЗ наближає можливість застосування цієї технології все в нових і нових сферах, знаходячи найкращі рішення складних задач і їх подальше вдосконалення.

До недавнього часу КЗ цілком використовувалося для потужної обчислювальної техніки, в силу вимог до заліза, але останнім часом ця технологія використовується і в мобільній електроніці.

Розглянемо КЗ на мобільному пристрої для розпізнавання рухомих об'єктів реального світу. Об'єктами в даному завданні є автомобілі та люди.

Використовуючи камеру мобільного пристрою, аналізуючи отримані зображення необхідно визначити до якого класу і з якою ймовірністю належать видимі камерою об'єкти, використовуючи при цьому технології машинного навчання і КЗ для рухомих об'єктів.

Реалізуємо механізм машинного навчання, який є підмодулем КЗ і допомагає з вирішенням завдань щодо ідентифікації розпізнаних об'єктів. Камера мобільного пристрою фіксуючи в кожен момент часу видимий кадр, передає його у вигляді матриці пікселів яку необхідно привести до необхідних стандартів для проведення розпізнавання.

Для розпізнавання необхідно зібрати великі обсяги даних для створення і тренування моделі розпізнавання.

Технології КЗ покликані допомогти з розпізнаванням об'єкта і взаємодією з раніше створеної натренованої моделлю для проведення порівняльного аналізу раніше підготовлених аналогів з новопоступовою інформацією.

Сформовані збіги після аналізу кожного кадру необхідно надати в зрозумілій користувачеві формі для кінцевої мети поставленого завдання, від видимих камерою пікселів до змістовного виду ідентифікованого об'єкта.

1.3.2 Формальна постановка задачі

Нехай маємо на вхід зображення I, яке є матрицею розмірності $n \times m$, де кожен елемент матриці p_{nm} – це піксель зображення.

$$I = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ p_{m1} & p_{m1} & \cdots & p_{mn} \end{pmatrix},$$

де елементи матриці p_{nm} – пікселі зображення,

n – ширина зображення в пікселях,

m – висота зображення в пікселях.

У кінцевій програмі ми маємо модель (ML Model) M, яка приймає на вхід зображення розмірності $299 \times 299px$.

Технології Core ML та Vision будемо позначати відповідно CML та V, які будуть використані для аналізу зображення I та роботи безпосередньо з моделлю M.

Для попередньої обробки даних введемо поняття адаптера (Adapter) A, який буде відповідати за вхідні розмірності зображення I та необхідну для моделі M. Наступний рисунок 1.7 ілюструє взаємодію всіх компонентів завдання які будуть використані у фінальній програмі.

Аналізуючи схему роботи програми формалізуємо усі основні процеси необхідні для її функціонування.

Маємо основне завдання програми розпізнавання $MainTask$ – розпізнати рухомі об’єкти та надати відповідну інформацію.

$$MainTask := In \rightarrow Out,$$

де In – зображення,

Out – вихідний вектор з інформацією про об’єкт.

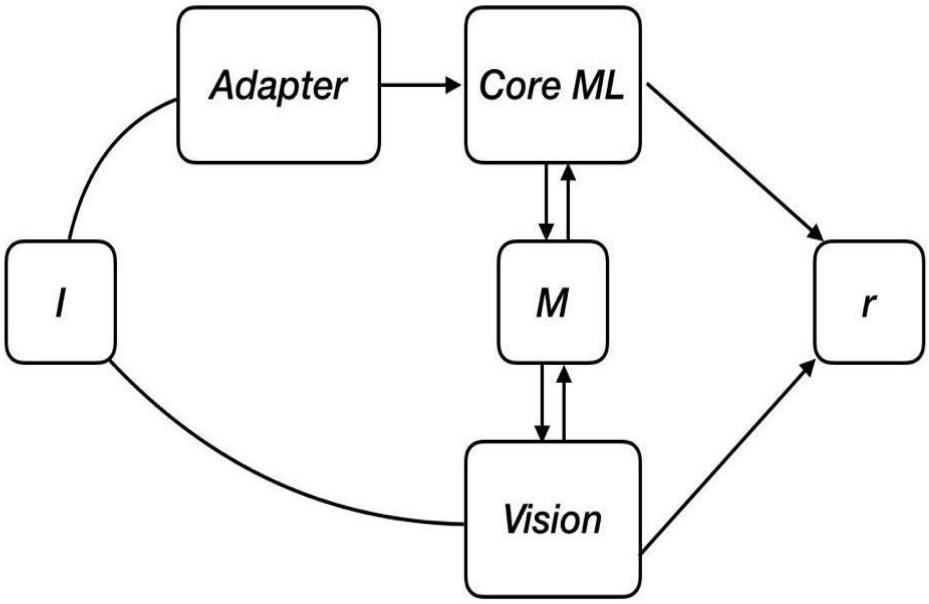

Рисунок 1.7 – Схематична модель процесу роботи програми розпізнавання

Розглянемо підпроцеси головного завдання. Формально на вхід маємо зображення I, яке в залежності від вибору технології передається в адаптер A або ж безпосередньо в V.

$$Task_1 := if(\text{Tech} = \text{CML}) : \{I_{n\times m} \to A\} \text{else}\{I_{n\times m} \to V\},$$

де Tech – технологія CML або V;

I – зображення;

A – адаптер.

Дана задача розбивається на 2 підзадачі

$$Task_1 : \{Task_2, Task_3\},$$

де $Task_2$ – розпізнавання за допомогою технології CML;

$Task_3$ – розпізнавання за допомогою технології V .

У загальному вигляді формалізована задача процесів розпізнавання в даній програмі буде виглядати так:

$$MainTask := In \to \{Task_1 \ldots Task_l\} \to Out;$$

$$In \coloneqq I_{n\times m};$$

$$Out \coloneqq \{Out_1, Out_2\};$$

$$Out_1 \coloneqq \{\vec{r_1} = (\mu, \varepsilon)\};$$

$$Out_2 \coloneqq \{\vec{r_2} = (\mu, \varepsilon)\};$$

$$Task_1 := \{if(\text{Tech} = \text{CML}) \to Task_2, if(\text{Tech} = V) \to Task_3\};$$

$$Task_2 := \{Task_4, Task_5, Task_6\};$$

$$Task_4 := A : \{I_{n\times m} \to I_{p\times p}\};$$

$$Task_5 := \{I_{p\times p} \to CML\};$$

$$Task_6 := \{Task_5 \to \text{M} \to CML\} \to Out_1;$$

$$Task_3 := \{Task_7, Task_8\};$$

$$Task_7 := \{I_{n\times m} \to V\};$$

$$Task_8 := \{Task_7 \to \text{M} \to V\} \to Out_2,$$

де CML та V – технології розпізнавання;

$Task_1 \ldots Task_l$ – процеси основної задачі $MainTask$;

$l = 8$ – кількість під задач;

A – адаптер;

$Task_1$ – в залежності від вибору технології ділиться на $Task_2$ та $Task_3$;

$Task_2$ – складається із задач $Task_4, Task_5, Task_6$;

$Task_3$ – складається із задач $Task_7, Task_8$;

$Task_4$ – перетворює зображення в адаптері $I_{n\times m} \to I_{p\times p}$;

$n \times m$ – початковий розмір зображення;

$p \times p$ – необхідний моделі;

$Task_5$ – передає підготовлене зображення в технологію CML;

$Task_6$ – включаючи $Task_5$ забезпечує взаємодію даних між технологією CML та натренованою моделлю M по завершенні одержуючи $Out_1 := \{\vec{r_1} = (\mu, \varepsilon)\}$;

$Task_7$ – передає підготовлене зображення в технологію V;

$Task_8$ – включаючи $Task_7$ забезпечує взаємодію даних між технологією та натренованою моделлю M по завершенні одержуючи $Out_2 := \{\vec{r_2} = (\mu, \varepsilon)\}$;

$\vec{r}_{1,2} = (\mu, \varepsilon)$ – вектор;

μ – мітка класу;

ε – ймовірність приналежності до неї аналізованого об'єкта I.

Adapter перетворює зображення довільного розміру $n \times m$ в необхідний для моделі M розмір $299 \times 299 px$. CML та V використовуються безпосередньо для роботи з моделлю і проведення аналізу вхідного зображення і зіставлення його з показниками натренованої моделі M.

На виході після взаємодії між компонентами програми ми отримуємо вектор $\vec{r} = (\mu, \varepsilon)$.

Варто зазначити, що для технології V не потрібна попередня обробка даних, V використовує вбудовані методи для аналізу зображень та адаптації їх під вимоги моделі розпізнавання M.

1.4 Постановка задач дослідження

Метою даної роботи є розробка моделі програми розпізнавання рухомих об'єктів в реальному часі і подальша реалізація програмного продукту.

Виходячи з проведеного системного аналізу системи «Аналіз технологій вирішення проблеми розпізнавання рухомих об'єктів у реальному часі за допомогою технологій комп'ютерного зору і машинного навчання» сформулюємо задачі для дослідження в рамках даної атестаційної роботи:

– вдосконалити знання технологій машинного навчання і комп'ютерного зору і визначити які саме можуть бути використані для створення програми з розпізнаванням рухомих об'єктів;

– вибрати інструменти програмної реалізації, платформу, розробити архітектуру програми в якій будуть застосовані методи машинного навчання і комп'ютерного зору;

– зібрати велику кількість тренувальних даних і створити модель для розпізнавання;

– на основі отриманої програми провести ряд експериментів, а саме: розпізнавання за допомогою технології CoreML та Vision різних об'єктів, а так само працездатність програми для цільового користувача;

– на основі отриманих результатів після тестування перевірити роботу програми на множині об'єктів в реальному часі з різними параметрами, відстанню камери до об'єкта, використанням технології, а так само визначити і виправити знайдені недоліки;

– на основі отриманих даних зробити висновки по виконаній роботі.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ’ЯЗАННЯ

2.1 Огляд технологій аналізу і розпізнавання об'єктів

2.1.1 Машинне навчання

На сьогоднішній день в тому чи іншому вигляді за аналіз і розпізнавання об'єктів зокрема відповідає машинне навчання. Машинне навчання оточує вас всюди, хоча, може бути, ви про це і не підозрюєте. Саме завдяки машинному навчанню пошукова машина розуміє, які результати показувати у відповідь на ваш запит. Коли ви переглядаєте пошту, велика частина спаму проходить повз вас, тому що він був відфільтрований за допомогою машинного навчання. Якщо ви вирішили що-небудь купити на Amazon.com або заглянули на Netflix подивитися фільм, система машинного навчання послужливо запропонує варіанти, які можуть припасти вам до смаку. За допомогою машинного навчання Facebook вирішує, які новини вам показувати, а Twitter підбирає відповідні твіти. Коли б ви не користувалися комп'ютером, дуже ймовірно, що десь задіяне машинне навчання [5].

Єдиним способом змусити комп'ютер щось робити – від складання двох чисел до управління літаком – було впорядкування певного алгоритму, скрупульозно пояснює машині, що саме від неї вимагається. Однак алгоритми машинного навчання – зовсім інша справа: вони вгадують всі самі, роблячи висновки на основі даних, і чим більше даних, тим краще у них виходить. Це означає, що комп'ютери не треба програмувати: вони програмують себе самі.

Машинне навчання (Machine Learning) – великий підрозділ штучного інтелекту, що вивчає методи побудови алгоритмів, здатних навчатися. Розрізняють два типи навчання. Навчання по прецедентах, або індуктивне навчання, засноване на виявленні загальних закономірностей по приватним емпіричним даним. Дедуктивне навчання передбачає формалізацію знань експертів і їх перенесення в комп'ютер у вигляді бази знань. Дедуктивне навчання прийнято від-

носити до області експертних систем, тому терміни машинне навчання і навчання по прецедентах можна вважати синонімами.

Машинне навчання знаходиться на стику математичної статистики, методів оптимізації та класичних математичних дисциплін, але має також і власну специфіку, пов'язану з проблемами обчислювальної ефективності та перенавчання. Багато методів індуктивного навчання розроблялися як альтернатива класичним статистичним підходам. Багато методи тісно пов'язані з витяганням інформації та інтелектуальним аналізом даних (Data Mining). Найбільш теоретичні розділи машинного навчання об'єднані в окремий напрям, теорію обчислювального навчання (Computational Learning Theory, COLT).

Машинне навчання – не тільки математична, а й практична, інженерна дисципліна. Чиста теорія, як правило, не призводить відразу до методів і алгоритмів, які можуть застосовуватися на практиці. Щоб змусити їх добре працювати, доводиться винаходити додаткові евристики, що компенсують невідповідність зроблених в теорії припущень умов реальних завдань. Практично жодне дослідження в машинному навчанні не обходиться без експерименту на модельних або реальних даних, що підтверджує практичну працездатність методу.

Машинне навчання є великою областю в розробці штучного інтелекту. Існує два типи штучного інтелекту (ІІ): слабкий (вузькоспрямований) і сильний (загальний). Слабкий ІІ призначений для виконання вузького списку завдань. Такими є голосові помічники Siri і Google Assistant і багато інших. Сильний ІІ, в свою чергу, здатний виконати будь-яку людську завдання. На даний момент реалізація сильного ІІ неможлива, він є утопічною ідеєю. Нейронна мережа – один із способів реалізації штучного інтелекту (ІІ).

2.1.2 Нейронні мережі

Розділ машинного навчання, з одного боку, утворився в результаті поділу науки про нейронних мережах на методи навчання мереж і види топологій їх

архітектури, з іншого боку – увібрав в себе методи математичної статистики. Говорячи про прикладних математичних науках а так само про програмну реалізації під нейронною мережею ми розуміємо Штучні нейронну мережу.

Штучна нейронна мережа (ШНМ) – математична модель, а також її програмне або апаратне втілення, побудована за принципом організації та функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Це поняття виникло при вивченні процесів, що протікають в мозку, і при спробі змоделювати ці процеси [6]. Першою такою спробою були нейронні мережі У. Маккалок і У. Питтса. Після розробки алгоритмів навчання одержувані моделі стали використовувати в практичних цілях: в задачах прогнозування, для розпізнавання образів, в задачах управління та ін.

ШНМ є системою з'єднаних і взаємодіючих між собою простих процесорів (штучних нейронів). Такі процесори зазвичай досить прості (особливо в порівнянні з процесорами, використовуваними в персональних комп'ютерах). Кожен процесор подібної мережі має справу тільки з сигналами, які він періодично отримує, і сигналами, які він періодично посилає іншим процесорам. І, тим не менше, будучи з'єднаними в досить велику мережу з керованим взаємодією, такі окремо прості процесори разом здатні виконувати досить складні завдання.

З точки зору машинного навчання, нейронна мережа являє собою окремий випадок методів розпізнавання образів, дискримінантного аналізу, методів кластеризації і т. п. З математичної точки зору, навчання нейронних мереж – це багатопараметрична завдання нелінійної оптимізації.

Нейронні мережі не програмуються в звичному сенсі цього слова, вони навчаються. Можливість навчання – одне з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами.

В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що в разі успішного навчання мережа зможе повернути вірний результат на підставі даних, які були відсутні в навчальній вибірці, а також неповних і (або)

«зашумлених», частково спотворених даних.

Базові види нейромереж, такі як перцептрон і багатошаровий перцептрон (а також їх модифікації), можуть навчатися як з учителем, так і без вчителя, з підкріпленням і самоорганізацією [7]. Деякі нейромережі і більшість статистичних методів можна віднести тільки до одного зі способів навчання. Тому, якщо потрібно класифікувати методи машинного навчання залежно від способу навчання, буде некоректним відносити нейромережі до певного виду, правильніше було б типізувати алгоритми навчання нейронних мереж.

Нейромережа моделює роботу людської нервової системи, особливістю якої є здатність до самонавчання з урахуванням попереднього досвіду. Таким чином, з кожним разом система здійснює все менше помилок. Як і наша нервова система, нейромережа складається з окремих обчислювальних елементів – нейронів, розташованих на декількох шарах.

Дані, що надходять на вхід нейромережі, проходять послідовну обробку на кожному шарі мережі. При цьому кожен нейрон має певні параметри, які можуть змінюватися в залежності від отриманих результатів – в цьому і полягає навчання мережі.

Припустимо, що задача нейромережі – відрізняти кішок від собак. Для набудівництва нейронної мережі подається великий масив підписаних зображень кішок і собак. Нейромережа аналізує ознаки (в тому числі лінії, форми, їх розмір і колір) на цих картинках і будує таку розпізнавальну модель, яка мінімізує відсоток помилок щодо еталонних результатів.

На рисунку 2.1 нижче представлений процес роботи нейромережі, завдання якої – розпізнати цифру поштового індексу, написану від руки.

До 2010 року просто не існувало бази даних, досить великий для того, щоб якісно навчити нейромережі вирішувати певні завдання, в основному пов'язані з розпізнаванням і класифікацією зображень. Тому нейромережі досить часто помилялися: плутали кішку з собакою, або, що ще гірше, знімок здорового органу зі знімком органу, ураженого пухлиною.

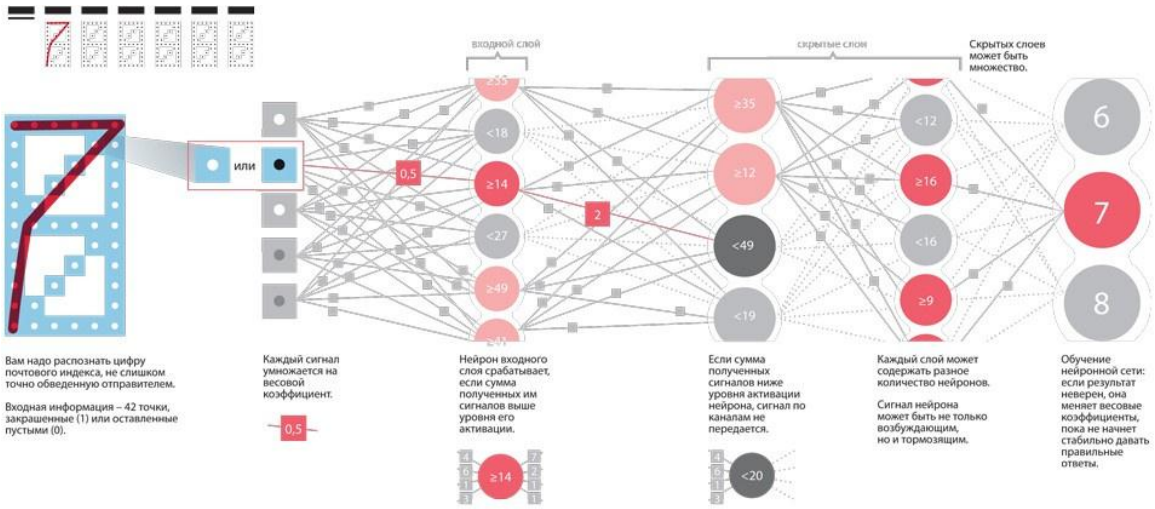

Рисунок 2.1 – Процес роботи нейронної мережі, розпізнає цифру поштового індексу

В 2010 році з'явилася база ImageNet, що містить 15 мільйонів зображень в 22 тисячах категорій. ImageNet багаторазово перевищувала обсяг існуючих баз даних зображень і була доступна для будь-якого дослідника. З такими обсягами даних нейромережі можна було вчити приймати практично безпомилкові рішення.

До цього на шляху розвитку нейромереж стояла інша, не менш істотна, проблема: традиційний метод навчання був неефективний. Незважаючи на те що важливу роль відіграє число шарів в нейронної мережі, важливий також і метод навчання мережі. Використовувався раніше метод зворотного шифрування міг ефективно навчати тільки останні шари мережі. Процес навчання опинявся занадто тривалим для практичного застосування, а приховані шари глибинних нейромереж не функціонували належним чином.

2.2 Машинне навчання та комп'ютерний зір у світі Apple

З кожним роком зростає потреба у вивченні великих даних як для компаній, так і для активних ентузіастів. У таких великих компаніях, як Apple або

Google, все частіше використовуються такі інструменти аналізу даних, як технології Core ML, Vision а так само різні бібліотеки для мови програмування Python. Відповідно до Закону Мура, кількість транзисторів на інтегральній схемі подвоюється кожні 24 місяці. Це означає, що з кожним роком продуктивність наших комп'ютерів зростає, а значить і раніше недоступні межі пізнання знову «зміщуються вправо» – відкривається простір для вивчення великих даних. Деякий час тому комп'ютерна програма Google DeepMind AlphaGo обіграла професіонала Лі Седола з Південної Кореї в азіатську стратегічну настільну гру. Ааналогічні програми, так само обходили професійних гравців в шахи. Для цього вони аналізували всі можливі ходи і вибирали найбільш вдалі. Роботи подібного софта стала можливою завдяки штучному інтелекту, машинного і глибокому навчанню. Друге поняття в даному випадку – частина першого, а третє – частина другого.

Компанія Apple використовує машинне навчання та нейронні мережі для різних завдань. Застосування цих технологій ми бачимо в різних функціях сучасних пристроїв. Наприклад технологія розпізнавання осіб в Face ID для розблокування телефону саме його власником базується саме на цьому принципі. Звичайно, технології прямо з Kinect, який колись розробляла Microsoft теж в темі, але вони не всі. Якби Apple не навчила сканер особи відрізняти справжніх людей від фото і макетів, гріш ціна була б всієї затії відмовлятися від звичного нам відбитку пальців Touch ID технології яка використовувалася до 2018 року з тією ж метою ідентифікації власника телефону.

Аналогічна ситуація зі змінами нашої зовнішності в часі – то борода відросте на все обличчя, то окуляри перед розблокуванням не знімеш або чубок з чола не прибереш. Всі ці зміни штучний інтелект обробляє на підставі нейронних мереж, який продовжують вчити, надаючи йому мільйони фотографій.

Основними бібліотеками що реалізують технології машинного навчання і комп'ютерного зору є Core ML і Vision.

За допомогою Core ML можна взяти, наприклад, фотографію кота і відправити за рівнями мережі. Потім повторити те ж саме з ще однією і так далі.

Аналогічний принцип для тесту власного нейронного механізму використовувала компанія Google, якій вдалося натренувати системи відбирати відео з кішками в YouTube. Кожен нейрон в мережі в даному випадку аналізує отримані дані і порівнює з завченими аналогами, які були скормлю йому раніше. Чим більше схожість, тим вище ймовірність того, що в кадрі дійсно присутній кіт, а в сумі нейронна мережа може дати відповідь на це питання з великою точністю.

Яскравим прикладом Vision є реалізована технологія Animoji, яка повторює людські емоції переносячи їх на обраний смайл. Принцип роботи технології продемонстрували на рисунку 2.2.

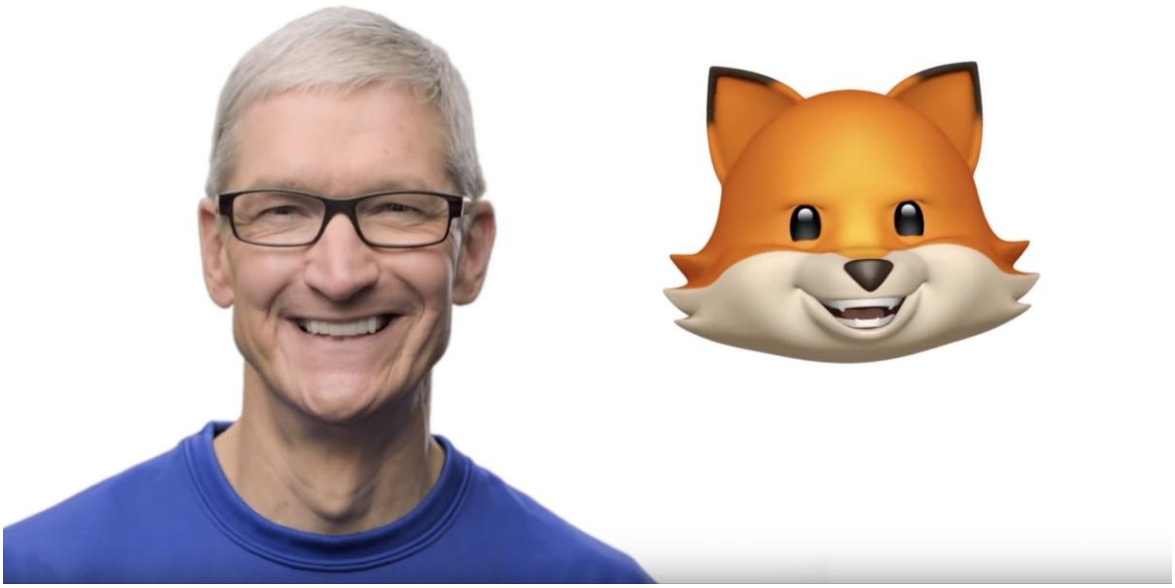

Рисунок 2.2 – Імітація смайла людині на базі штучного інтелекту

До відносно недавнього часу нейтронні мережі просто ігнорувалися. Вони були цілком реальними ще в минулому столітті, але користі від них було було не так багато. Це обумовлено тим, що ще кілька років тому продуктивні системи, які могли б працювати в нейронними мережами, були вкрай дорогими, а дані в необхідному числі недоступними. Потім вчені змогли розширити нейронні мережі до неймовірних розмірів, зробили прийнятною швидкість роботи і згодували їм тонни контенту. Так з'явилося глибоке навчання.

Повертаючись до компанії Apple основи Core ML і Vision запустили роз-

робників в сміливий новий світ машинного навчання з безліччю захоплюючих можливостей. Vision дозволяє виявляти і відстежувати особи, а сторінка «Машинознавство» на Apple надає готові до використання моделі, які виявляють об'єкти і сцени, а також NSLinguisticTagger для обробки природної мови. Якщо ви хочете створити свою власну модель, спробуйте нову версію Turi Create від Apple, щоб продовжити одну зі своїх попередньо підготовлених моделей з вашими даними.

Потім прийшов час зануритися в машинне навчання (ML), використовуючи одну з численних платформ від Google, Microsoft, Amazon або Berkeley. І щоб зробити життя ще більш захоплюючою, вам потрібно буде вибрати нову мову програмування і новий набір інструментів для розробки.

За допомогою мови Python можна навчати глибоку наукову згортальну модель нейронної мережі, перетворити її в Core ML і інтегрувати її в програму iOS.

2.2.1 Технологія Core ML в iOS

З 1950-х років дослідження в світі штучного інтелекту розробило безліч підходів комп'ютерного навчання. Ядро Core ML від Apple підтримує нейронні мережі, дерева рішень (tree ensembles – деревовидний алгоритм прийняття рішень), метод опорних векторів (support vector machines), узагальнені лінійні моделі (generalized linear models), генерація ознак (feature engineering) і пайплайн моделі (pipeline models) [8]. Варто зауважити, що за допомогою нейронних мереж вже реалізовано багато, наприклад з останніх гучних подій Google вже спонсорує конкурс, завдання якого визначити 5000 видів рослин і тварин. Такі голосові помічники, як Siri і Alexa, також з'явилися на світло, завдяки існуванню нейронних мереж.

Нейронні мережі, представляючи собою шари вузлів, які по-різному з'єднані між собою вимагають великого збільшення обчислювальної потужності,

наприклад модель розпізнавання об'єктів Inception v3, має 48 шарів і близько 20 мільйонів параметрів. Розрахунки – це в основному множення матриць, з якими GPU (графічний процесор) справляється дуже ефективно. Останнім часом, на тлі зниження вартості GPU (графічний процесор) дозволяє дослідникам створювати багатошарову глибоку нейронну мережу, приклад якої можна побачити на рисунку 2.3.

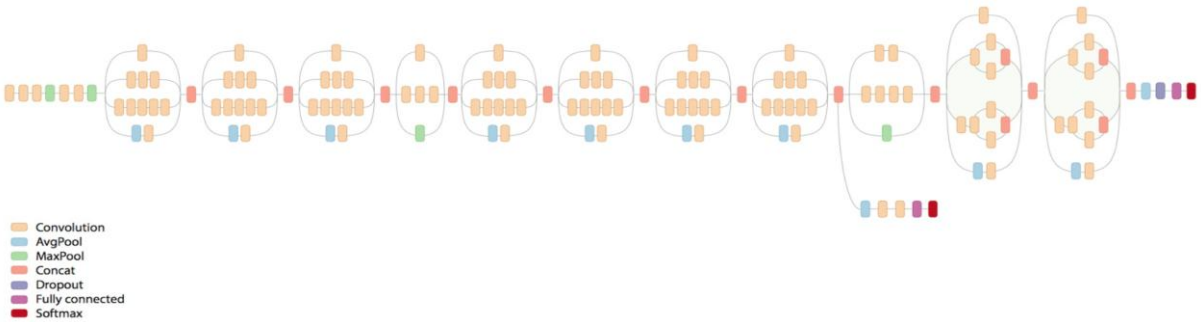

Рисунок 2.3 – Багатошарова нейронна мережа

Нейронним мережам необхідно надавати велику кількість вхідних даних, бажано, щоб ці дані надавали весь можливий спектр даних. Якраз саме поява машинного навчання і вплинула на великий ріст призначених для користувача даних. Навчання моделі означає, що ви надаєте дані для навчання нейронних мереж, а далі вони самі обчислюють необхідну формулу для комбінування вхідних параметрів і отримання вже вихідних даних. Навчання відбувається в автономному режимі, як правило, на комп'ютерах, які містять велику кількість GPU (графічних процесорів).

Для того, щоб використовувати цю модель, вам необхідно дати їй нові вхідні дані, і вона вирахує вихідні дані (результат): це називається логічним висновком. Висновок також виробляє велику кількість обчислень, для обчислення результату від нових вхідних даних. Виконання цих обчислень на портативних пристроях стало можливим завдяки таким технологіям, як Metal.

На WWDC 2017 Apple представила нову технологію для роботи з технологіями машинного навчання Core ML. На основі нього в iOS реалізовані власні

продукти Apple: Siri, Camera і QuickType. Core ML дозволяє спростити інтеграцію машинного навчання в додатки і створювати різні інтелектуальні додатки а так само впроваджувати «розумні» функції для існуючий додатків. На рисунку 2.4 представлена спрощена схема дій з використанням моделі Core ML model в технології Core ML з подальшою інтеграцією в кінцеву програму.

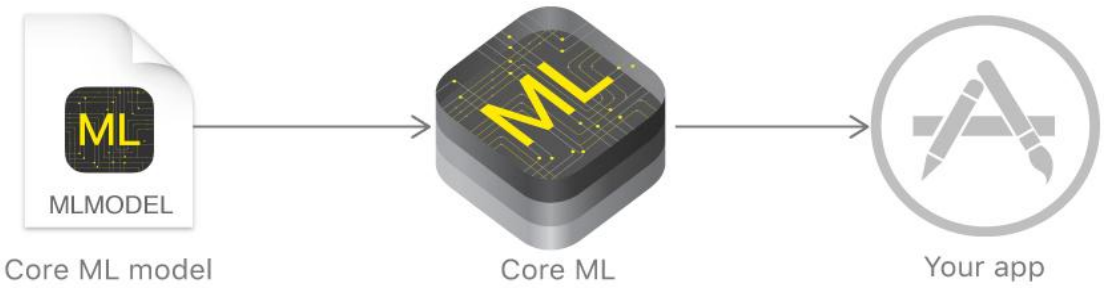

Рисунок 2.4 – Схема інтеграції моделі в кінцеву програму

За допомогою Core ML в додатку можна реалізувати наступні функції:

розпізнавання зображень в реальному часі;

– інтелектуальне введення тексту;

– розпізнавання образів;

– аналіз тональності;

– розпізнавання рукописного тексту;

– ранжування пошуку;

– стилізація зображень;

– розпізнавання осіб;

– ідентифікація голосу;

– визначення музики і багато іншого.

Існують різні типи навчальних моделей, які вирішують одну і ту ж проблему (наприклад, розпізнавання об'єктів), але з різними алгоритмами.

Core ML дозволяє легко імпортувати в вашу програму різні алгоритми машинного навчання, такі як: tree ensembles, SVMs і generalized linear models. Він використовує низькорівневі технології, такі як Metal, Accelerate і BNNS. Результати обчислень відбуваються майже миттєво [9].

На рисунку 2.5 продемонстрували вкладення технологій в кінцеву програму від нейронних мереж в чистому вигляді до сучасної технології розпізнавання Vision.

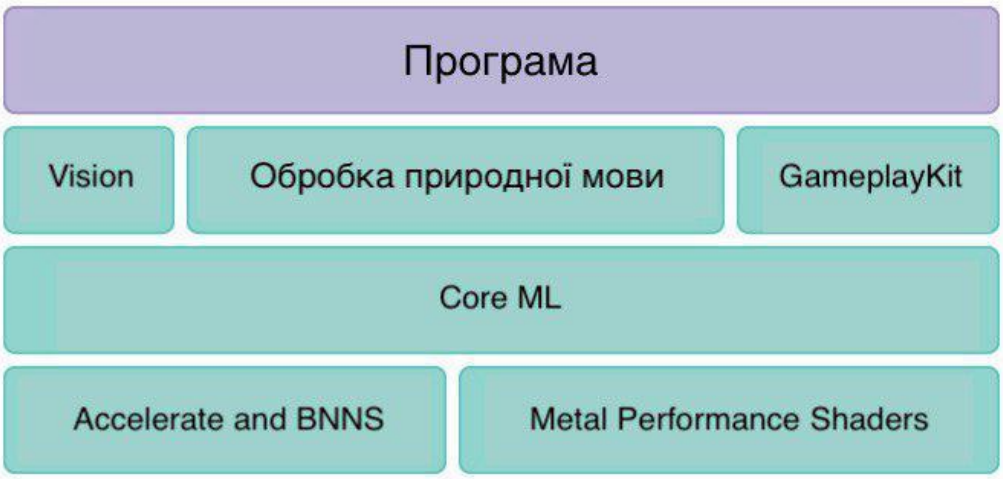

Рисунок 2.5 – Схема вкладення технологій в кінцеву програму

Навчена модель є результатом застосування алгоритму машинного навчання до набору даних навчання. Модель робить прогнози на основі нових вхідних даних. Наприклад, модель, що пройшла навчання по історичних цінностей регіону, може бути в змозі передбачити ціну будинку, враховуючи кількість спалень і ванних кімнат. Core ML оптимізований для роботи на пристрої, що зводить до мінімуму обсяг пам'яті і енергоспоживання. Робота строго на пристрої забезпечує конфіденційність даних користувача і гарантує, що ваша програма залишиться функціональним і чуйним, коли підключення до мережі недоступно.

У машинному навчанні все починається з моделі, системи, яка робить прогнози або ідентифікаційні дані. Викладання комп'ютера для навчання включає в себе алгоритм машинного навчання з даними навчання, з якими можна вчитися.

Результат, отриманий при навчанні, зазвичай відомий як моделі машинного навчання. Таблиця 2.1 показує моделі і сторонні інструменти, підтримувані технологією Core ML.

Таблиця 2.1 – Моделі і сторонні інструменти, підтримувані технологією Core ML

Тип моделі	Підтримувані моделі	Підтримувані інструменти
Нейронні мережі	Випереджаюча, згортальна, повторювана	Caffe, Keras 1.2.2+
Ансамбль дерев	Випадкові ліси, посилені ліси, дерева рішень	Scikit-learn 0.18, XGBoost 0.6
Допоміжні векторні механізми	Скалярна регресія, багато класова класифікація	Scikit-learn 0.18, LIBSVM 3.22
Узагальнені лінійні моделі	Лінійна регресія, логістична регресія	Scikit-learn 0.18
Інженерна техніка	Розряджена векторизація, щільна векторизация, категоріальна обробка	Scikit-learn 0.18
Модель трубопроводу	Послідовні ланцюгові моделі	Scikit-learn 0.18

Щоб перетворити моделі даних в формат Core ML, не обходимо використовувати програмне забезпечення під назвою Core ML Tools. Так як постановка задачі розпізнавання мільйонів об'єктів є досить великою, ми будемо використовувати вибірки даних які надаються моделлю Inception v3 яка включає в себе величезну кількість прикладів об'єктів існуючого світу. На основі них за допомогою Core ML ми будемо аналізувати зображення з камери мобільного телефону і визначати, що саме знаходиться на знімках. Надалі застосовуючи технології постійного захоплення зображень буде розроблений механізм моментального розпізнавання, а саме розпізнавання об'єктів в реальному часі.

2.2.2 Технологія Vision в iOS

Технологія Vision працює на основі Core ML і допомагає з відстеженням і розпізнаванням осіб, тексту, об'єктів, штрих-кодів. Також є визначення горизонту і отримання матриці для вирівнювання зображення.

Core ML підтримує Vision для аналізу зображень, Foundation для обробки природної мови (наприклад, клас NSLinguisticTagger) і GameplayKit для оцінки вчинених дерев рішень. Сам Core ML грунтується на примітивах низького рівня, таких як Accelerate і BNNS, а також на шиях Metal Performance Shaders [10]. Якщо порівнювати між собою 2 технології розпізнавання то в загальному плані отримаємо, що Core ML – спрощує використання учнів моделей в ваших додатках, а Vision – надає легкий доступ до моделей від Apple для розпізнання осіб, орієнтирів, тексту, прямокутників, штрих-кодів і об'єктів.

Розробник також може обернути будь-яку модель аналізу зображень Core ML в моделі Vision, що до речі і було зроблено при розробці програмного продукту. Оскільки ці дві структури побудовані на Metal, вони дозволяють ефективно працювати на пристрої і розробнику не буде потрібно відправляти дані користувачів на сервер. Натреновану модель можна створити за допомогою підтримуваних інструментів для машинного навчання, таких як Caffe, Keras або scikit-learn. Перетворення учнів моделей в Core ML (Converting Trained Models to Core ML) – дає можливість конвертувати його в формат Core ML який використовується в розробляється програмі.

Так як Vision працює з використанням методів Core ML основними і приємними відмінностями є: простота реалізації, додатковий функціонал розпізнавання зображень, відсутність необхідності попередньої обробки даних, а так само постійні оновлення та підтримка технології компанією розробником.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Особливості програмної реалізації задач на мові Swift

Пристрої компанії Apple є визнаним лідером у сфері інформаційних технологій. За останніми даними частка iOS серед інших мобільних операційних систем коливається в районі 35-40%, а частка Mac OS X серед настільних систем становить за різними даними 15-20%. Подібне широке поширення пристроїв від компанії Apple народжує потребу в програмному забезпеченні для цих пристроїв.

Традиційно основною мовою програмування під iOS і MacOS був Objective-C, проте 2 червня 2014 року на конференції розробників Apple WWDC 2014 був представлений новий і більш зручний мову програмування – Swift.

Swift поєднує в собі все краще від мов C і Objective-C раніше і до цього дня використовуються для розробки продуктів в екосистемі Apple. В Swift використовуються шаблони безпечного програмування і додані сучасні функції, що перетворюють створення додатка в простий, більш гнучкий і інтелектуальний процес.

Swift розроблявся Apple кілька років. Основою нової мови программірованія послужили існуючі компілятор, відладчик і бібліотеки. Apple спростили процес управління пам'яттю за допомогою механізму автоматичного підрахунку посилань – Automatic Reference Counting (ARC) [11]. Бібліотеки також зазнали серйозної модернізації. Objective-C почав підтримувати блоки, літерали і модулі – все це створило сприятливі умови для впровадження сучасних технологій. Саме ця підготовча робота послужила фундаментом для нової мови програмування, який застосовується зараз для розробки програмних продуктів під Apple.

Розробникам Objective-C Swift здасться знайомим, він поєднує в собі читабельність іменованих параметрів і міць динамічної об'єктної моделі Objective-C. Він відкриває доступ до вже існуючих бібліотек Cocoa і сумісний з кодом, написаним на Objective-C. Побудований на цій загальній основі мову пропонує безліч нових можливостей і уніфікує процедурні та об'єктно-орієнтовані аспек-

ти мови програмування. Swift не відлякає і початківців програмістів. Це перший потужний мову програмування, такий же зрозумілий і захоплюючий, як скриптова мова. Він підтримує так звані playground – ігрові майданчики, які дозволяють програмістам новачкам експериментувати з кодом, бачачи результат в режимі реального часу без необхідності компілювати і запускати програму.

У порівнянні з Objective-C Swift володіє наступними особливостями:

– Swift є чистим об'єктно-орієнтованою мовою програмування;

– простота, ясний і чіткий синтаксис;

– сувора типізація. Кожна змінна має певний тип;

– автоматичне управління пам'яттю.

Однак при цьому Swift повністю сумісний з раніше написаними прикладними інтерфейсами Cocoa API, для яких використовувалися C і Objective-C.

Swift є компільовані мовою програмування. Тобто розробник пише вихідний код і потім, використовуючи компілятор, компілює цей код в керуючу програму. Потім цей файл програми можна завантажити в AppStore і поширювати серед інших користувачів.

Swift увібрав в себе все краще від сучасних мов і розроблений з урахуванням великого досвіду компанії Apple.

3.2 Структура та опис програми

Програма виконана в середовищі XCode на мові об’єктно орієнтованого програмування Swift.

Програма складається з 4 логічних блоків:

– модель DriverAssist.mlmodel;

– адаптер камери VideoCameraAdapter;

– основні класи аналізу зображень та забезпечення логіки роботи програми ViewControllers, Manager, Struct;

– інтерфейс програми Main.storyboard і LaunchScreen.storyboard.

Модель являє собою одиничний розділ в якому знаходиться натренована модель яка здатна виявляти домінуючі об'єкти, присутні в зображенні, з набору з 2 категорій, таких як транспортні засоби та людина.

Адаптер камери включає в себе 5 класів: VideoCameraType.swift, AVCaptureDevice+Extension.swift, UIImage+CVPixelBuffer.swift, CVPixelBuffer+Helpers.swift, DVideoCapture.swift.

Клас VideoCameraType.swift дозволяє визначити нам з якої камери ми захоплюємо зображення – з основної або фронтальної.

У класі AVCaptureDevice+Extension.swift реалізовані методи приведення отриманих зображень в необхідні для розпізнавання формати які розширюють функціонал базового класу AVCaptureDevice.

Класи UIImage+CVPixelBuffer.swift та CVPixelBuffer+Helpers.swift потрібні нам для розширення функціоналу класу UIImage новими методами рабти з пікселями CVPixelBuffer та розширеня самого класу CVPixelBuffer допоміжними функціями Helpers.

У класі DVideoCapture.swift приходять основні процеси роботи безпосередньо з камерою, відеопотоком, захопленням зображень і конвертації видимого камерою світу в зручні для роботи з комп'ютером пікселі.

В основних класах криється весь механізм роботи програми.

У StartViewController.swift відбувається ініціалізація інтерфейсу, графіки і надання можливості вибору користувачеві використовувати один з двох технологій Vision і Core ML.

Класс BoundingBox.swift потрібен нам для відтворення різних прямокутників які будуть показувати користувачеві розпізнані об'єкти.

В класс PredictionManager.swift ми винесли функціонал пов'язаний з ідентифікацією різних рухомих об'єктів на відео потоці що складається із зображень.

У класі MainViewController.swift ми проводили експерименти з технологгіями, часом, параметрами.

Головний клас DriverAssistViewController.swift криє в собі взаємодію між

усіма раніше створеними і описаними класами та реалізує основну логіку програми. У ньому відбувається отримання даних з камери, перетворення їх з допомогою адаптерів в необхідні для моделі форми, безпосередньо розпізнавання і отримання результатів. Відображення відео з камери, виділення різних об'єктів прямокутниками із зазначенням точності розпізнавання, назви класу та переслідування видимих об'єктів. Оновлення призначеного для користувача інтересу відбувається в реальному часі за принципом циклу і припиняється тільки після повернення користувача на екран StartViewController.swift.

Безпосередньо в класах Main.storyboard і LaunchScreen.storyboard знаходиться макет дизайну і зовнішнього вигляду програми. Відведені місця для тексту, кнопок, вікон відеопотоку, різних кнопок управління програмою а так само обрані кольори, шрифти і розміри всього призначеного для користувача інтерфейсу.

3.3 Особливості ідентифікації рухомих об'єктів в кадрі

Детектування рухомих об'єктів стало важливою частиною вирішення безлічі прикладних задач в різних сферах. Існує кілька підходів для детектування об'єктів: методи, що виключають фон, методи спостереження за особливими точками, методи формування фону, моделі з автоматичним переміщенням.

Алгоритми, що використовують стаціонарні камери для зйомки, використовують підхід відділення динамічних об'єктів на кадрі від фону кадру. На основі статичних точок, які не змінюють свого положення на зображенні, можна визначити рухомі контури.

Адаптивні фонові моделі застосовуються через їх можливості розпізнавати зміни на кадрах, які викликані зміною освітлення в сценах поза приміщенням або зміною фону через рух камери. Однак ці методи неефективні при швидкій зміні сцени і зазвичай працюють некоректно.

В рамках даної роботи ми використовуємо технологію Vision, яка містить

в собі реалізацію безлічі різних алгоритмів, що істотно спрощує можливість реалізації даної програми через те, що реалізація і вибір оптимального алгоритму є досить об'ємною і витратною за часом завданням і не обмежується рамками даної роботи.

Наприклад, один з алгоритмів – алгоритм Rodriguez-Canosa G. R., Thomas S., Cerro J. який дозволяє визначити рухомі об'єкти на відеозображенні в реальному режимі часу [12].

Основними етапами алгоритму є: обчислення дійсного оптичного потоку, обчислення штучного оптичного потоку, ідентифікація динамічних точок і їх фільтрація. Оптичний потік – векторне поле, що характеризує рух об'єктів, яке виникає через рух камери, щодо сцени. Вектори даного поля показують як змінилося становище точок в поточному кадрі щодо попереднього. Визначення дійсного оптичного потоку реалізується алгоритмом Лукаса-Канаді [13]. Цей метод є диференціальним локальним алгоритмом знаходження оптичного потоку. Неоднозначність рівнянь вирішується за рахунок інформації про сусідніх пікселях кожної точки. Вводиться припущення про те, що оптичний потік однаковий в околиці кожної точки.

Штучний потік представляється становищем всіх точок, отриманих для дійсного оптичного потоку, математично спроектованих на наступний кадр з урахуванням руху камери. Іншими словами, обчисливши штучний оптичний потік, можна визначити вектор напрямку руху камери. Рух камери представляється поворотом і зміщенням, які отримують в результаті роботи алгоритму PTAM [14]. Для отримання штучного потоку використовується гомографіческая проекція. Гомографія проектує положення точки з площини однієї камери в площину іншої.

Алгоритм Лукаса-Канаді поєднує в собі визначення рухомих об'єктів, засноване на виявленні особливостей статичних точок, і порівняння оптичного потоку для виявлення точок, що належать динамічним об'єктам.

Швидкість роботи алгоритму залежить від розміру оброблюваних кадрів: чим більше кадр, тим довше він обробляється. З іншого боку, чим якісніше кадр і чим більше його розмір, тим точніше буде визначено оптичний потік. Отже, в

залежності від конкретного завдання і наявних обчислювальних потужностей потрібно вибирати, з якою якістю і швидкістю будуть проведені обчислення.

Володіючи даними знаннями, ми можемо підійти до реалізації програми з допомогою технології Vision більш детально, з повним розумінням необхідних умов, параметрів і результатів які ми можемо отримати.

Для роботи з Vision в рамках даного завдання, нам потрібно реалізувати динамічний вибір параметра fps (frames per second) з англійської – частота кадрів в секнуду. FPS – частота (швидкість), з якої пристрій формування зображення відображає послідовні зображення, звані кадрами. Термін застосовується до кіно і відео камер, комп'ютерної графіки і систем захоплення руху.

Дана необхідність обумовлена як раз наявними обчислювальним потужностями, яких досить мало на мобільному пристрої. Для того що б отримати оптимальний показник, ми повинні розуміти скільки кадрів в секунду може обробляти наш процесор. Якщо значення буде занадто великим ми отримаємо непрацюючу програму, якщо значення буде занадто маленьким ми будемо бачити ефект зависання і несвоєчасного поновлення користувацького інтерфейсу.

Так само, для створення моделі ми повинні зібрати велику кількість вихідних даних – фотографій машин і людей в одному кадрі в природному середовищі.

Природне середовище, або кінцева середовище в якій планується застосовувати даний продукт – це дороги на яких пересуваються автомобілі і які можуть перетинати пішоходи.

Як ми знаємо, чим більше, різноманітніше і якісніше зібрано вихідні дані – тим точніше буде результат розпізнавання за допомогою створеної моделі.

Для того, що б зібрати таку велику кількість різноманітних даних, ми скористалися безліччю відеозаписів на просторах інтернету, які були зроблені за допомогою відеореєстратора встановленого на автомобілі і викладені людьми в загальний доступ.

Ми зробили розкадрування даних відео роликів, прибрали нечіткі кадри, і отримали велику кількість зображень які ми використовували для створення тренувальної та тестової вибірок для навчання моделі DriverAssist.

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ

Метою розробки програми було створити інструмент який в реальному часі міг би при встановленні мобільного пристрою з камерою на автомобілі розпізнавати з певною точністю рухомих учасників дорожньоо руху, таких яких автомобілі та пішохіди.

У програмі використовуються дві технології (бібліотеки) створені компанією Apple для розробників.

Кожен з технологій є інструментом який працює з однією і тією ж натренованої моделлю DriverAssist.mlmodel яка дозволяє віднести вхідні дані до якогось класу і зробити прогноз про розпізнаний об'єкт.

Надамо хронологію дій у вигляді послідовного алгоритму проведення кроків.

Крок 1. Для роботи з бібліотеками необхідно не тільки імплементувати їх в програму, а й реалізувати необхідний функціонал для попередньої підготовки даних.

Далі розглянемо докладно роботу кожної з технологій.

Core ML дозволяє легко імпортувати в програму різні алгоритми машинного навчання, такі як: tree ensembles, SVMs і generalized linear models. Він використовує низькорівневі технології, такі як Metal, Accelerate і BNNS. Результати обчислень відбуваються майже миттєво.

Технологія Vision працює на основі Core ML і допомагає з відстеженням і розпізнаванням осіб, тексту, об'єктів, штрих-кодів. Також є визначення горизонту і отримання матриці для вирівнювання зображення.

Крок 2. Створюючи проект, попередньо необхідно створити Core ML модель, використовуючи об'ємні вибірки тренувальних даних, яка буде використовуватися для обох технологій. Для поточного завдання треба зібрати багато відеоматеріалів з автомобільних відеореєстраторів, прибрати з відеозаписів фрагменти поганої якості і з поганим освітленням, і зробити нарізку кадрів для отримання великого датасета який складається із зображень реальних учасників

руху таких як автомобілі і пішоходи.

Крок 3. Далі потрібно створити інтерфейс програми і визначити основні класи програми.

Крок 4. Нам потрібно вивести зображення з камери в реальному часі, для цього створимо необхідні адаптери камери і визначимо вимоги моделі до вхідних даних.

Крок 5. Наша модель яку ми будемо використовувати для аналізу зображення вимагає на вхід кольорове зображення розміром 299 на 299 пікселів. Для цього в адаптерах камери прописуємо обробку що надходять з камери зображень в необхідні моделі формати і забезпечуємо грамотну і безвідмовну роботу механізму.

Крок 6. Так як камера захоплює величезну кількість візуальної інформації в секунду, треба обмежити кількість кадрів що передаються та обробляються щомиті тому що ми можемо зламати механізм ідентифікації великим потоком даних.

Ми можемо вказати фіксоване значення кадрів в секунду рівне 5-10, так як ми вже знаємо що потужності телефону вистачить для обробки такого обсягу даних. Але для задачі розпізнавання і переслідування рухомих об'єктів нам потрібна більша кількість кадрів. Для цього ми зробимо динамічне обчислення оптимальної кількості кадрів в секунду на основі потужності і навантаження процесора у пристрої.

Візуальний ефект моментальності розпізнавання не буде порушений, і моделі буде простіше обробляти ту кількість кадрів в секунду яку може обробити процесор і видавати нам відповідні якісні передбачення.

Крок 7. Для технології Core ML нам самим необхідно подбати про те, щоб змінити розмір зображень до 299 на 299 пікселів, на відміну від Vision, який буде робити це автоматично. Підхід Vision для роботи з натренованої моделлю чистіший і простий в реалізації. Це обумовлено тим, що Vision представили дещо пізніше ніж механізм машинного навчання і аналізу даних Core ML.

У нашій програмі ми використовуємо обидві технології для того, що б

ознайомитися з перевагами в роботі з кожною з них, конкретно для задачі розпізнавання рухомих об'єктів. Це обумовлено тим, що потік даних і частота оновлення призначеного для користувача інтерфейсу повинна бути досить великою, і нам потрібно вибрати оптимальний варіант який ми зафіксуємо для фінальної програми.

Для цього створимо головний екран додатка на котрому ми будемо вибирати з ким із технології ми будемо працювати та робити висновки, а також опцію для фінальної програми. Головний екран можна побачити на рисунку 4.1.

Рисунок 4.1 – Стартовий екран програми з вибором технології

Після вибору необхідної технології натисканням кнопок Vision або Core ML ми потрапляємо на тестовий екран додатка який являє собою велике вікно в яке виводиться інформація з камери, а так само форму внизу для виведення інформації про видимі об'єкти. При натисканні опції Driver Assist ми потрапимо на екран фінальної програми, з обраної після експерименту технологією і всіма налаштуваннями для подальшого використання та демонстрації.

Почнемо з CoreML, поступово наводячи камеру на різні нерухомі та рухомі об'єкти. Ми бачимо, що з камери надходять зображення, обробляються нами і проходять аналіз в моделі, після чого ми отримуємо різні передбачення, точність розпізнавання, клас приналежності об'єкта і кількість кадрів в секунду.

На рисунках 4.2–4.3 ми бачимо різні нерухомі та рухомі об'єкти видимі камерою смартфона та передбачення моделі і механізма CoreML щодо інформації на них.

Як ми можемо бачити при одному фіксованому об'єкті в кадрі ми отримуємо досить гарне оновлення кадрів в секунду, але при великій кількості, вже рухомих об'єктів, ми бачимо низьку частоту оновлення кадрів в секунду, що свідчить про довгий час розпізнавання та призводить до поганого оновлення призначеного для користувача інтерфейсу.

Далі розглянемо результати роботи Vision, який не вимагає попередньої обробки даних і додаткового функціоналу по роботі з об'єктами які знаходяться в русі.

На рисунках 4.4–4.5 ми бачимо різні нерухомі та рухомі об'єкти видимі камерою смартфона та передбачення моделі і механізма Vision щодо інформації на них.

Як ми бачимо технологія Vision показує ті ж значення на одному фіксованому об'єкті, але краще справляється з великою кількістю об'єктів в кадрі, бачить більше об'єктів і демонструє кращі показники за кількістю кадрів в секунду.

У правильному варіанті проектування додатків існує принцип Reusability, що в перекладі з англійської означає повторне використання чогось. Дотримуючись стандартів правильного проектування будемо використовувати 1 екран з розпізнаванням для обох технологій, завдяки чому зовні на рисунках ви будете бачити аналогічний дизайн. Для фінальної програми, після проведення експериментів, ми реалізуємо одну найкращу технологію, а так само покращимо призначений для користувача інтерфейс довівши його до демонстраційного зразка.

На рисунках 4.6–4.7 ми бачимо результати фінальної програми з різними нерухомі та рухомі об'єктами за допомогою Vision.

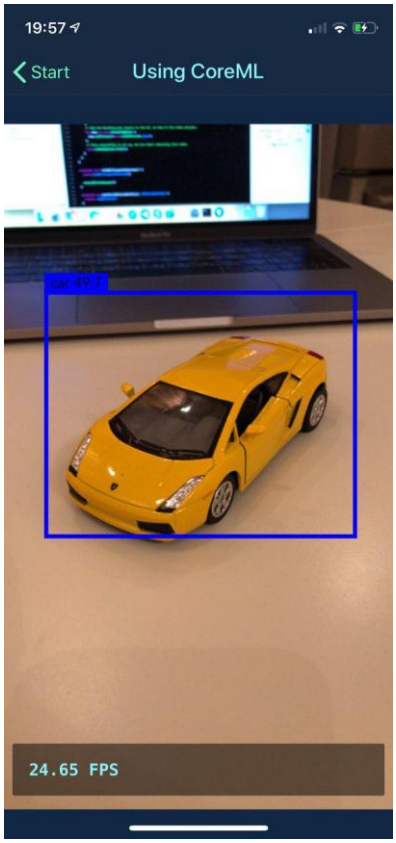

Рисунок 4.2 – Програма з CoreML достатньо добре розпізнає нерухомий об'єкт

Рисунок 4.3 – Програма з CoreML погано розпізнає рухомі об'єкти

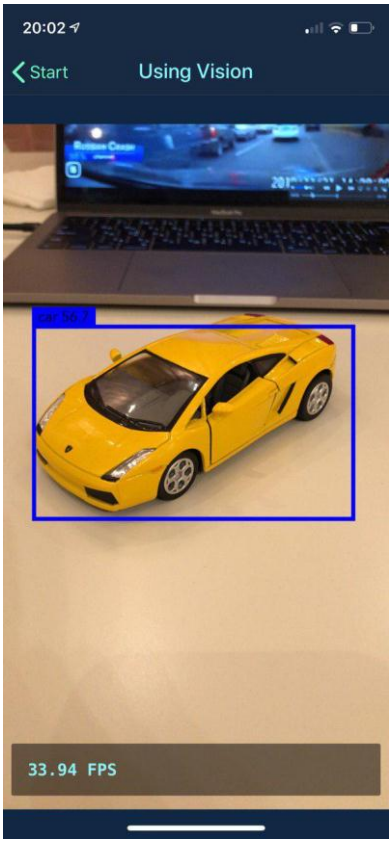

Рисунок 4.4 – Програма з Vision добре розпізнає нерухомий об'єкт

Рисунок 4.5 – Програма з Vision добре розпізнає рухомі об'єкти

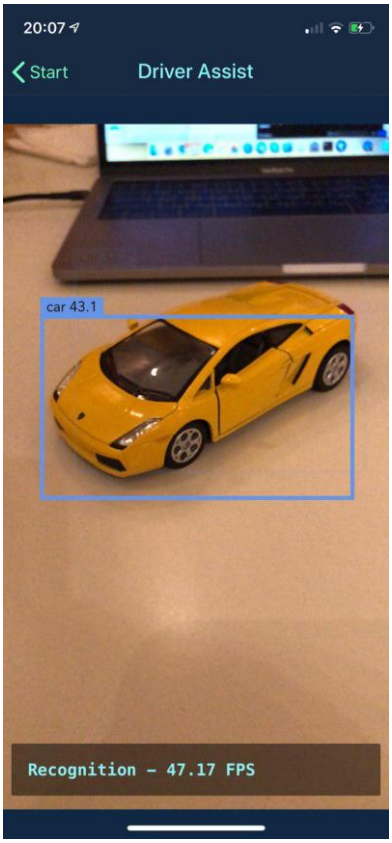

Рисунок 4.6 – Фінальна програма з Vision відмінно розпізнає нерухомий об'єкт

Рисунок 4.6 – Фінальна програма з Vision відмінно розпізнає рухомі об'єкти

Провівши порівняльний аналіз технологій розпізнавання і їх кінцевого застосування у фінальній програмі можна скласти порівняльну таблицю 4.1 технологій Vision та Core ML.

Таблиця 4.1 – Порівняння технологій

	Технологія Vision	Технологія Core ML
Складність реалізації	Простіше	Складніше
Можливість розширення функціоналу	Немає	Є
Необхідність попередньої підготовки даних	Не потребує попередньої підготовки даних	Необхідна повна відповідність даних вимогам моделі
Сучасність	Підтримує найсвіжіші оновлення і постійно допрацьовується	Використовується у задачах більш складних і фундаментальних, оновлюється несвоєчасно
Швидкість роботи	Швидше	Повільніше
Сфера використання	Має більш широкий спектр галузей для застосування технології	Застосовується у всіх сферах, але має менший функціонал
Робота з рухомими об'єктами	Закладена спочатку, краще справляється із завданням	Може бути реалізована додатково, вимагає більшого часу під час обробки зображень
Кількість кадрів в секунду одержувана на мобільному процесорі	Чимала, через меншого часу на обробку	Недостатнє, через необхідність попередньої підготовки і обробки зображень

5 АНАЛІЗ МОЖЛИВИХ ЗАСТОСУВАНЬ

Результати даної атестаційної роботи можуть бути застосовані при розробці розумних автопілотів, додатків помічників водієві, розробці систем контролю дорожнього руху, а також робототехніки для орієнтації ідентифікації та переслідування роботами необхідних в рамках завдання об'єктів.

Робототехніка є традиційною сферою застосування машинного зору. Однак основна частка парку роботів довгий час припадала на промисловість, де оживлення роботів не було б зайвим, але завдяки добре контрольованим умовам (низької недетермінованості середовища) можливими виявлялися вузькоспеціалізовані рішення, в тому числі і для задач машинного зору. Крім того, промислові додатки допускали використання дорогого устаткування, що включає оптичні та обчислювальні системи. У зв'язку з цим показовим є те, що частка парку роботів, яка припадає на промислових роботів, стала менш 50% лише на початку 2000-х років [15]. Стала розвиватися робототехніка, призначена для масового споживача. Для побутових роботів, на відміну від промислових, критичною є вартість, а також час автономної роботи, що має на увазі використання мобільних та вбудованих процесорних систем. При цьому такі роботи повинні функціонувати в недетермінованих середовищах. Наприклад, в промисловості довгий час використовувалися фотограмметричні мітки, які наклеюють на об'єкти спостереження або калібрувальні дошки для вирішення завдань визначення внутрішніх параметрів і зовнішньої орієнтації камер. Природно, необхідність наклеювати користувачеві такі мітки на предмети інтер'єру істотно погіршила б споживчі якості побутових роботів.

Показовим є те, що перші роботи-пилососи, оснащені системами комп'ютерного зору, з'явилися лише в 2006 році. До цього моменту використання мобільних процесорів типу сімейства ARM з частотою 200 МГц дозволяло домогтися зіставлення зображень тривимірних сцен всередині приміщень на основі інваріантних дескрипторів ключових точок з метою сенсорної локалізації робота з частотою близько 5 кадрів на секунду. Використання зору для визначення

роботом свого місця розташування стало економічно виправданим, хоча ще нещодавно для цих цілей виробники використовувати сонари.

Подальше підвищення продуктивності мобільних процесорів дозволяє ставити нові завдання для систем комп'ютерного зору в побутових роботах, число продажів яких по всьому світу обчислюється вже мільйонами примірників на рік [16]. Крім завдань навігації, від роботів, призначених для персонального використання, може знадобитися рішення задач розпізнавання людей їх рух, траєкторії в залежності від типу завдання, розв'язуваного роботом. Багато з цих завдань далекі від повного вирішення і є перспективними з інноваційної точки зору.

Таким чином, сучасна робототехніка вимагає вирішення широкого кола завдань комп'ютерного зору, що включає, зокрема:

– набір завдань, пов'язаних з орієнтацією в зовнішньому просторі (наприклад, завдання одночасної локалізації і картографування – Simultaneous Localization and Mapping, SLAM), визначенням відстаней до об'єктів і т.п;

– завдання з розпізнавання різних об'єктів і інтерпретації сцен в цілому;

– завдання з виявлення людей, розпізнаванню їх осіб і аналізу траєкторії руху.

Крім побутових роботів, методи комп'ютерного зору знайшли широке застосування в системах допомоги водієві. Роботи з детектування розмітки, перешкод на дорозі, розпізнаванню знаків і т.п. Активно велися і в 90-х роках. Однак достатнього рівня (як по точності і надійності самих методів, так і по продуктивності процесорів, здатних в масштабі реального часу виконувати відповідні методи) вони досягли переважно в останньому десятилітті.

Одним з показових прикладів є методи стереозору, використовувані для виявлення перешкод на дорозі. Ці методи можуть бути дуже критичні до надійності, точності і продуктивності. Зокрема, з метою виявлення пішоходів може вимагатися побудова щільної карти дальності в масштабі, близькому до реального часу. Ці методи можуть вимагати сотень операцій на піксель і точності, що досягається при розмірах зображень не менше мегапікселя, тобто при сотнях

мільйонів операцій на кадр (декількох мільярдів і більше операцій в секунду).

Повертаючись до систем допомоги водієві, не можна не згадати про сучасні методи детектування пішоходів, зокрема, на основі гістограм орієнтованих градієнтів [17]. Сучасні методи машинного навчання, про які ще буде сказано пізніше, вперше дозволили комп'ютера краще людини вирішувати таку досить загальну зорову завдання, як розпізнавання дорожніх знаків, але не завдяки використанню спеціальних засобів формування зображень, а завдяки алгоритмам розпізнавання, які отримували на вхід в точності ту ж інформацію, що і людина. Одним з істотних технічних досягнень став безпілотний автомобіль Google, який, однак, використовує багатий набір сенсорів крім відеокамери, а також не працює на незнайомих (заздалегідь не знятих) дорогах і при поганих погодних умовах.

Таким чином, для систем допомоги водієві потрібно рішення різних завдань комп'ютерного зору, включаючи:

– стереозір;

– виявлення перешкод на дорогах;

– розпізнавання дорожніх знаків, розмітки, пішоходів і автомобілів;

– завдання, також вимагають згадки, пов'язані з контролем стану водія.

Ще більш масовими в порівнянні з побутовою робототехнікою і системами допомоги водієві є завдання комп'ютерного зору для персональних мобільних пристроїв, таких як смартфони, планшети і т.п. Зокрема, число мобільних телефонів неухильно зростає і вже практично перевищило за чисельністю населення Землі. При цьому основна частка телефонів випускається зараз з камерами. У 2009 р кількість таких телефонів перевищила мільярд, що створює колосальний за розмірами ринок для систем обробки зображень і комп'ютерного зору, який далекий від насичення, незважаючи на численні R&D-проекти, що проводяться як самими фірмами – виробниками мобільних пристроїв, так і великим числом стартапів.

Частина завдань з обробки зображень для мобільних пристроїв з камерами збігається із завданнями для цифрових відеокамер. Основна відмінність по-

лягає в якості об'єктивів і в умовах зйомки. Клас задач комп'ютерного зору, рішення яких можуть бути застосовані в мобільних додатках, вкрай широкий. Великий набір додатків є у методів зіставлення зображень, в тому числі з оцінкою тривимірної структури сцени і визначенням зміни орієнтації камери і методів розпізнавання об'єктів, а також аналізу осіб людей. Однак може бути запропоновано необмежено велике число мобільних додатків, для яких буде вимагатися розроблення спеціалізованих методів комп'ютерного зору.

Методи глибокого навчання вимагають великих обчислювальних ресурсів, і навіть для адаптації розпізнавання обмеженого класу об'єктів можуть вимагатися кілька днів роботи на обчислювальному кластері.

На даний момент, аналізування дорожньої ситуації під час руху є дуже важливою і необхідною задачею для сучасних розробників. Кількість автомобілів зростає з кожним днем, аварій стає більше, люди порушують правила дорожнього руху, що невблаганно несе за собою людські жертви.

Системи контролю руху можуть бути не тільки внутрішніми – встановленими на самому автомобілі або з використанням додаткових приладів допомоги водієві, контролю сліпих зон, контролю смуги, відстані до об'єктів і екстренного гальмування, але і зовнішні. Під зовнішніми мається на увазі системи контролю ситуації в місті за допомогою безлічі камер встановлених на дорогах міст і мегаполісів. Дані системи вже існують у багатьох країнах світу і контролюють дотримання швидкісного режиму, правил дорожнього руху, а так само фіксують і реагують на події. Наприклад при порушенні правил, перевищенні швидкості або проїзді на червоне світло порушнику виписується штраф або висилається патрульна машина для затримання, за умови серйозних порушень. Так само як і у випадку аварійних ситуацій система може сама викликати швидку допомогу за потрібною адресою, що помітно скорочує час реакції на ту чи іншу ситуацію.

Це лише найбільш поширені програми комп'ютерного зору для масового користувача. Однак існує і безліч інших, менш типових програм. Наприклад, методи комп'ютерного зору можуть бути використані в військової промислово-

сті для самонаведення знарядь, ракет з можливістю самонаведення і переслідування, космічній сфері при аналізі рухомих космічних тіл, розрахунку їх швидкості, траєкторії і попередженні про небезпеку.

Далі за допомогою розробленого програмного продукту планується створити конкретний комерційний додаток для водіїв, який буде за допомогою голосових команд попереджати про різні рухоми об'єкти на дорозі, такі як автомобілі, тролейбуси, автобуси, вантажний транспорт, мотоцикли, велосипеди, а найголовніше люди. Програма будет прогнозувати траєкторії їх руху, фокусуватися на різних об'єктах перефеійного зору, а так само запам'ятовувати номера автомобілів попереду, та повідомляти водія про можливу небезпеку.

За допомогою проведених випробувань і розробки повноцінного мобільного продукту який можна поширити за допомогою інтернету великої аудиторії користувачів ми зможемо зробити корисний додаток який зробить повсякденне життя користувачів комфортніше, та безпечніше.

ВИСНОВКИ

В рамках даної атестаційної роботи були вдосконалені знання роботи технологій машинного навчання і комп'ютерного зору на прикладі створення моделі і програми розпізнавання рухомих об'єктів в реальному часі. Основна різниця технологій полягає в складності реалізації і в необхідності попередньої обробки даних перед розпізнаванням. Також складність при розробці програми полягала в зібранні якісних попередніх данних для тренування моделі, та отримання координат виявленного рухомого об'єкту в кадрі.

Для експерименту в конкретному завданні були реалізовані обидві технології розпізнавання спрямовані на виявлення рухомих об'єктів таких як пішохід та автомобіль які є головними учасниками дорожнього руху. Модель розпізнавання була створена на основі великої вибірки експериментальних даних зібраних з кількох десятків відео з автомобільних відео реєстраторів знайдених на просторах інтернет мереж. Програма була написана для мобільного пристрою на мові Swift.

Отримані результати продемонстрували абсолютну коректність роботи механізму розпізнавання кількох рухомих об'єктів у різних сценаріях використання та різних вхідних даних.

Найбільш оптимальною в даному випадку є технологія Vision, яка не вимагає витратних операцій по попередній обробці даних для розпізнавання кількох рухомих об'єктів, та дозволяє отримувати хороші показники частоти оновлення кадрів в секунду що є критично важливим для роботи створеної програми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Історія розвитку машинного навчання. URL : https://hi-news.ru /technology/trendy-mashinnoe-obuchenie.html (дата звернення: 14.03.2019).

2. Kirichenko L., Radivilova T., Zinkevich I. Comparative Analysis of Conversion Series Forecasting in E-commerce Tasks. URL : https://link.springer.com/ chapter/10.1007/978-3-319-70581-1_16 (дата звернення: 14.10.2019).

3. Лапыгин Ю. Н. Теория организации. Москва : "Инфра-М", 2007. 311 с.

4. Киселев Д. Ю., Киселев Ю. В., Макарьев В. Д. Структурный анализ потоков данных (Data Flow Diagrams – DFD) : метод. указания. Самара : Изд-во СГАУ, 2014. 12 с.

5. Домингос П. Верховный алгоритм: как машинное обучение изменит наш мир. Москва : Манн, Иванов и Фербер, 2016. 480 с.

6. Хайкин С. Нейронные сети: полный курс. Москва : Издательский дом Вильямс, 2008. 1103 с.

7. Kirichenko L., Radivilova T., Bulakh V. Binary Classification of Fractal Time Series by Machine Learning Methods. URL : https://link.springer.com/ chapter/10.1007/978-3-030-26474-1_49 (дата звернення: 20.10.2019).

8. Kirichenko L., Radivilova T., Bulakh V. Machine Learning in Classification Time Series with Fractal Properties. URL : https://www.mdpi.com/2306-5729/4/1/5/htm (дата звернення: 20.10.2019).

9. Технічна документація технології Core ML. URL : https://developer. apple.com/documentation/coreml/core_ml_api (дата звернення: 20.05.2019).

10. Технічна документація технології Vision. URL : https://developer. apple.com/documentation/vision (дата звернення: 05.06.2019).

11. Neuburg M. iOS 10. Programming Fundamentals with Swift. O'Reilly Media, 2016. 620 p.

12. Rodriguez-Canosa G. R., Thomas S., Cerro J. et al. A real-time method to detect and track moving objects (DATMO) from unmanned aerial vehicles (UAVs) using a single camera // Remote Sensing. 2012. No 4. P. 1090–1111.

13. Bouguet J. Y. Pyramidal implementation of the Lucas–Kanade feature tracker. Description of the algorithm; Technical report; Intel Corporation Microprocessor Research Lab: Santa Clara, CA, USA, 1999.

14. Klein G., Murray D. Parallel tracking and mapping for small AR workspaces // Proceedings of 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. 2007. P. 225–234.

15. Bechtel W. The Cardinal Mercier Lectures at the Catholic University of Louvain: An Exemplar Neural Mechanism: The Brain's Visual Processing System. Ch. 2, P. 1. 2003. 15 p.

16. Юревич Е. И. Основы робототехники. Санкт-Петербург : БХВ-Петербург, 2007. 252 с.

17. Viola P., Jones M. Robust Real-time Object Detection // Workshop on Statistical and Computation Theories Vision. July, 2001. 30 p.