ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення автоматизованих систем

Пояснювальна записка

до магістерської роботи
на ступінь вищої освіти магістр

на тему: «**РОЗРОБКА МОДЕЛІ ІНФОРМАЦІЙНОЇ СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ ВИЗНАЧЕННЯ ХАРАКТЕРИСТИК ОБ'ЄКТІВ НА БАЗІ RASPBERRY PI»**

Виконав: студент 6 курсу, групи ІСДМ-61
спеціальності 126 Інформаційні системи та технології
освітня програма «Інформаційні системи та технології»
(шифр і назва спеціальності)
Макарцев М.О.
(прізвище та ініціали)
Керівник Тушич А.М.
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)
Нормоконтроль Данильченко В.М.
(прізвище та ініціали)

Київ – 2021

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення автоматизованих систем
Ступінь вищої освіти - «Магістр»
Спеціальність підготовки 126 Інформаційні системи та технології
Освітня програма «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗАС
_____ К.П.Сторчак
"____" _____2021 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Макарцев Максим Олександрович
(прізвище, ім'я, по батькові)

1.Тема роботи: «Розробка моделі інформаційної системи комп'ютерного зору для визначення характеристик об'єктів на базі raspberry pi»

Керівник роботи: Тушич Аліна Миколаївна, доктор філософії.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від _____року №___

2. Строк подання студентом роботи _____

3. Вхідні дані до роботи :
1. Науково-технічна література
2. Існуючі інструменти для роботи з комп'ютерним зором, розроблені для Raspberry Pi
3. Готові інструменти для роботи з комп'ютерним зором з допомогою Raspberry Pi

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Аналіз існуючих бібліотек, пристроїв та систем
2. Розробка моделі системи комп'ютерного зору

5. Перелік графічного матеріалу
1. Титульний слайд
2. Постановка завдання
3. Компоненти системи

6. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури		
2	Вивчення матеріалів для подальшого використання		
3	Знайомство з програмними інструментами		
4	Налаштування Raspberry Pi		
5	Початкова реалізація		
6	Оптимізація за допомогою зовнішнього процесора		
7	Остаточна реалізація		
8	Вступ, висновки, реферат		
9	Розробка демонстраційних матеріалів		
10	Попередній захист роботи		

Студент_____ Макарцев М.О.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Тушич А.М.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи 80 с., 21 рис., 21 джерело.

Об'єкт дослідження: процес створення моделі системи комп'ютерного зору на базі Raspberry Pi.

Предмет дослідження: реалізація розпізнавання об'єктів з використанням одноплатного комп'ютера Raspberry Pi.

Мета роботи: дослідження ефективних механізмів аналізу зображення з метою розпізнавання об'єктів і взаємодії з ними.

Методи дослідження: методи теорії інформації, методи наукового моделювання, методи дослідження інформаційних систем.

У даній магістерській роботі розроблено модель системи комп'ютерного зору. Проаналізовано сучасні бібліотеки, які дозволяють ефективно і з високою точністю виконувати розпізнавання об'єктів на статичних та динамічних зображеннях, а також відео з камери у реальному часі.

Проведено дослідницьку роботу щодо ефективності їх використання на одноплатному комп'ютері Raspberry Pi 3 Model B. Оглянуто різні способи покращення швидкодії ресурсомістких операцій. Залучено нейронні мережі для підвищення точності розпізнавання об'єктів.

Галузь використання: Системи комп'ютерного зору на базі Raspberry Pi.

Raspberry Pi, Python, OpenCV, DLib, Tensorflow, MobileNet.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

FD	Face Detection	Виявлення обличчя
FR	Face Recognition	Розпізнавання обличчя
RPi	Raspberry Pi	Raspberry Pi
SoC	System On Chip	Система на чіпі
FPS	Frames Per Second	Кадри за секунду
DL	Deep Learning	Глибоке навчання
DNN	Deep Neural Networks	Глибокі нейронні мережі
NN	Neural Networks	Нейронні мережі

ЗМІСТ

ВСТУП

В останні роки досягнення в галузі глибокого навчання та повсюдне використання графічних процесорів призвели до проривних змін та покращень у технології комп'ютерного зору. Проблема в тому, що вони вимагають великих обчислювальних ресурсів і, отже, зазвичай виконуються роботами автономно. У цій роботі представлено розробку та реалізацію алгоритмів комп'ютерного зору, які можуть працювати на Raspberry Pi 3 Model B. Проаналізовано та використано ефективні методи реалізації алгоритмів у таких темах, як розпізнавання осіб, розпізнавання облич, класифікація та відстеження руху на Raspberry Pi для запуску додатків, близьких до реального часу.

За допомогою платформи глибокого навчання Tensorflow для навчання та виконання прогнозів, а також бібліотек OpenCV та dlib для Python, які є широко використовуваними бібліотеками для комп'ютерного зору та машинного навчання відповідно, реалізовано легкі та ефективні алгоритми пам'яті для Raspberry Pi, які мають використовуватись у додатках близьких до реального часу. Крім того, робляться спроби оптимізації цих алгоритмів, які є як алгоритмічними, так і з використанням зовнішнього процесора, такого як сервер або хмарного сервісу, такого як Google Colaboratory.

Системи розпізнавання осіб та відстеження руху будуть використовуватись у роботі, що розглядається у проекті, який використовується, наприклад, для допомоги дітям у навчанні соціальним навичкам. Робота, проведена в рамках цієї роботи, дозволить роботу визначити, чи є людина, яка управляє роботом, дитиною чи дорослою, та відповідно працюватиме в різних режимах.

1 Преамбула

1.1 Мотивація

Комп’ютерний зір — це наука, що дозволяє комп’ютерам автоматично бачити й розуміти візуальну інформацію. Довгий час люди прагнули створити справді розумні машини, і великою перешкодою для цього історично було завдання бачення. Але останні досягнення в апаратному забезпеченні, поява глибокого навчання, поява великих наборів даних і збільшення додатків комп’ютерного зору – все це дало цій галузі вкрай необхідний імпульс, і в останні роки в ній кипить активність.

Комп’ютерний зір має широкий спектр застосувань у таких галузях, як автомобілебудування, спорт, робототехніка, сільське господарство, медицина, безпека, і, за оцінками, до 2022 року ринок комп’ютерного зору становитиме 48,6 мільярдів доларів. Це стимулювало багато інновацій у цій галузі. від таких корпоративних гігантів, як Facebook, Google і Microsoft. Але дослідження та інновації в цій галузі, межі яких зараз розсуваються майже поодинці завдяки глибокому навчанню, були обмежені через його інтенсивну обчислювальну природу та залежність від спеціалізованого обладнання, такого як графічні процесори.

З іншого боку, Raspberry Pi — це серія одноплатних комп’ютерів, які спочатку мали бути дешевою альтернативою комп’ютеру для навчання кодуванню. З моменту свого випуску в 2012 році він став набагато більше, ніж просто, і широко використовується в різноманітних програмах. Очевидно, що обчислювальні можливості Raspberry Pi обмежені і, як правило, непридатні для виконання важкої

роботи, необхідної для комп’ютерного зору. Але Raspberry Pi використовується в багатьох областях, таких як робототехніка, і його універсальність щодо обробки периферійних пристроїв, таких як камери, зробила його ще більш популярним.

Цей проект стосується впровадження та керування алгоритмами комп’ютерного зору, такими як виявлення обличчя або виявлення об’єктів на Raspberry Pi. Тепер, оскільки Raspberry Pi є апаратною системою з обмеженими обчислювальними можливостями, а обчислювальні вимоги алгоритмів комп’ютерного зору, які зазвичай базуються на деякій структурі глибокого навчання, якщо вони мають відповідати належним стандартам точності, є досить високими, ми використовуємо будь-яке, зовнішній процесор, наприклад сервер, підключений до Raspberry Pi через Wi-Fi або хмарну платформу, як-от Google Colaboratory, для підвищення продуктивності.

Робот у проекті допомагає дітям навчитися соціальним навичкам, має Raspberry Pi, вбудований у нього з підключеною камерою. Тепер, якщо ми змогли запустити розпізнання облич на Pi в режимі реального часу, то робот зможе визначити, чи є людина, яка його використовує, дитина чи хтось інший, і, отже, може працювати в різних режимах залежно від цього.

1.2 Обґрунтування

1.2.1 Обґрунтування мети

Після завершення проекту буде сформульована відповідна методологія для ефективного запуску алгоритмів комп’ютерного зору на Raspberry Pi. Цей результат можна використовувати в різних галузях від автомобільної до

робототехніки та медицини. Це, в свою чергу, дозволить роботу працювати в різних режимах залежно від того, дитина чи дорослий його використовує.

Іншим прикладом може бути спостереження, яке зможе зробити система домашнього спостереження краще і швидше, якщо комп'ютер/сервер був увімкнений і підключений до Raspberry Pi.

1.2.2 Обґрунтування технології

Використання глибоких нейронних мереж за останні кілька років здійснило революцію в багатьох областях. Особливо в сферах комп'ютерного зору, які є у центрі уваги цього проекту, таких як класифікація, розпізнавання тощо, глибоке навчання та нейронні мережі значно підвищують точність і широко, майже виключно, використовуються дослідниками для покращення стану техніки.

Існує багато фреймворків глибокого навчання, таких як Tensorflow, Caffe, PyTorch тощо. Останнім часом Tensorflow став найбільш використовуваною та активно розвивається фреймворком глибокого навчання. Таким чином, Tensorflow, розроблений Google, буде найкращою платформою глибокого навчання, яка буде використовуватися для реалізації нейронних мереж. Це також відкритий вихідний код.

Для алгоритмів бачення на Python OpenCV є широко використовуваною бібліотекою з відкритим кодом, яка має безліч алгоритмів. Крім того, dlib — це ще одна бібліотека, написана на C++, яка має багато алгоритмів машинного навчання, які використовувалися для виявлення облич і частин відстеження руху.

Оскільки частини виявлення обличчя та відстеження руху вимагають запуску алгоритму для кожного кадру відео, підхід нейронної мережі виглядає занадто повільно. Отже, підхід не нейронної мережі був використаний з використанням dlib.

Навчання штучних нейронних мереж на Raspberry Pi просто займає занадто багато часу і перегріває Raspberry Pi і, отже, не виконується. Усе навчання або перенавчання здійснюється виключно на зовнішньому процесорі.

1.3 Контекст

У цьому розділі описуються сфери інтересів, а також зацікавлені сторони проекту: цільова аудиторія, користувачі та бенефіціари.

1.3.1 Сфери інтересів

Сферами інтересів тут є підтеми комп’ютерного зору, з якими ми маємо справу: класифікація, розпізнавання облич, розпізнавання обличчя та виявлення/відстеження руху. Оскільки безпосереднє застосування цього проекту відбувається в роботі, усі теми реалізація орієнтована на різні варіанти використання в проекті.

Класифікація

У комп'ютерному баченні класифікація — це процес категоризації візуальних стимулів у скінченний набір класів або міток. Мета полягає в тому, щоб пов'язати об'єкти на зображенні з будь-якими мітками класу.

Цю асоціацію може бути особливо важко зробити, коли об'єкти закриті або коли об'єкти з часом змінюють орієнтацію та масштаб.

У цьому проекті ми розглянемо проблему класифікації віднесення людини до дорослого чи дитини. Ця робота допомогла б роботі в визначити, чи є людина, яка використовує робота, дорослою чи дитиною (без визначення, хто це дитина чи дорослий), і відповідно працювати в різних режимах.

Розпізнавання облич

Розпізнавання обличчя – це процес визначення розташування людського обличчя на зображенні. Програми виявлення облич використовують алгоритми, орієнтовані на виявлення людських облич на більших зображеннях, які можуть містити ландшафти, об'єкти та інші частини людей.

Виявлення облич — це ширший термін, ніж розпізнавання облич. Виявлення обличчя просто означає, що система може визначити, що на зображенні або відео присутнє людське обличчя. Виявлення облич має кілька застосувань, лише одне з яких — розпізнавання обличчя. Виявлення облич також можна використовувати для автоматичного фокусування камер.

За допомогою нерухомих або відеозображень сцени система розпізнавання облич зможе ідентифікувати або перевірити одну або кілька осіб на сцені за допомогою збереженої бази даних. Системи розпізнавання облич побудовані на основі систем розпізнавання об'єктів, оскільки локалізація є важливим кроком у конвеєрі розпізнавання облич.

Розпізнавання облич використовується для перевірки або визначення особистості людини, яка використовує робота в проекті. Це, в принципі, може дати великий простір для персоналізації робота.

Відстеження руху

Відстеження руху — це процес виявлення зміни положення об’єкта відносно його оточення або зміни оточення щодо об’єкта. Відстеження руху – це процес визначення місця розташування рухомого об’єкта (або кількох об’єктів) протягом певного часу за допомогою камери. Виявлення руху можна розглядати як підмножину всієї парадигми відстеження руху.

Взаємодія між роботом і дитиною можна відстежувати, виявляючи, чи дитина використовує робота постійно, чи продовжує рухатися і, отже, не зацікавлена, відстежуючи рух дитини після виявлення. Якщо дитина виходить з поля зору робота, можна подати сигнал тривоги. Це дозволить дорослим відпускати дітей самих у кімнаті без постійного нагляду.

Соціальна робототехніка

Соціальна робототехніка – це дослідження роботів, які здатні взаємодіяти та спілкуватися між собою, з людьми та з навколишнім середовищем, у межах соціальної та культурної структури, що відповідає її ролі. Щоб добре працювати, ці роботи повинні мати можливість добре вчитися у своїх партнерів-людей.

Багато «навчання» відбувається через обробку даних, отриманих за допомогою датчиків. Одним з основних стимулів, які люди використовують як соціальні, є зір, а аналогом цього для робота є візуальні стимули, отримані через камеру. Залежно від типу обробки, яка виконується, робот може використовуватися для виконання різноманітних функцій у відповідь.

Щоб зрозуміти інші функції, засновані на датчиках, візьмемо приклад робота PLEO виробництва Innvo Labs. Точніше візьмемо приклад конкретної версії, PLEO rb.

PLEO — домашній робот із зовнішнім виглядом дитинчата динозавра з розвиненою здатністю імітувати природні рухи та вирази обличчя. Деякі моделі соціальної поведінки, які він має:

- Народжені в характеристиках – ці властивості наведені з моменту виробництва: стать, мужність, вдача, активність і слухняність.
- Атрибути After Born – штучний інтелект (ШІ) створює ці функції на основі середовища роботи та його досвіду. Їх поєднання створює унікальну індивідуальність: Харчування, настрій, емоції, здоров'я, фізичний стан.
- Почуття часу – засноване на його змодельованому біологічному годиннику. Він може навчитися і розпізнати: день, ніч, час спати, час їсти.
- Автономні анімації – залежно від особистості Pleo, взаємодії та оточення, ШІ може генерувати анімацію, можливі такі можливості: голодна послідовність (пастися), послідовність втоми (дремота), жага уваги (сигнал або докучливий буксир), послідовність сновидінь, інші послідовності присутності PLEO rb.

Деякі функції, які він має щодо датчиків системи:

- Зір – Pleo rb має камеру для виконання деяких функцій комп’ютерного зору, таких як: розпізнавання кольорів (обмежено червоним, жовтим і зеленим), визначення кінця треку, спеціальні картки визнання.
- Голос – робот включає мікрофон як частину системи розпізнавання звуку, яка може генерувати: відповіді на конкретні команди, вивчення та запам’ятовування власного імені.
- Рух – завдяки отриманню сигналу від акселерометрів, Pleo rb може класифікувати тип руху, який користувач виконує над ним. Можливі заняття: розмах, трясіння, вплив.

- Відчуття дотику – робот використовує набір тактильних датчиків для досягнення диференціації між пестинням і ударами.
- Реакція на температуру – із вимірювань температури він може вибрати та виконати певну послідовність залежно від стану, як-от чхання або тремтіння, якщо холодно, задихаючись або запаморочення, коли жарко.
- Розпізнавання їжі – додаткові шматки розпізнаються за допомогою RFID як їжа, реакція залежить від настрою, потреб і бажан: їсти, відкидати.

1.3.2 Зацікавлені сторони

Розробка цього проекту може зацікавити людей з різних сфер. Це потенційні зацікавлені сторони та причини, чому вони можуть зацікавитися нашим проектом.

Фірми

Компанії, зацікавлені в пропонуванні продуктів і послуг на основі відеоспостереження або соціальної робототехніки. Необхідно адаптувати інструменти, запропоновані в цьому проекті, до потреб нашої цільової аудиторії, а також до приходу нових технологій.

Студенти

Студенти, які цікавляться робототехнікою та комп’ютерним баченням, можуть використовувати інструменти, створені в цьому проекті, як натхнення, якщо вони вважаються гідними, посібник або потенційну роботу, яку можна розширити та покращити в майбутньому.

Бенефіціари

Бенефіціарами стануть кінцеві користувачі тих продуктів, які запускатимуть алгоритми комп’ютерного зору на Raspberry Pi. Користувачі такої системи можуть отримати вигоду від оптимізації, яку забезпечить цей проект, і можуть мати швидший і плавний досвід використання продукту. Основними вигодами будуть ентузіасти робототехніки.

- Користувачі робота: діти та їхні батьки/дорослі матимуть робота набагато плавніше, оскільки розпізнавання обличчя та руху будуть швидшими та точнішими, і він зможе працювати в різних режимах.
- Люди, які займаються спостереженням: якщо для спостереження використовується Raspberry Pi, тепер вони зможуть запускати розпізнавання в режимі реального часу.
- Любителі автоматизації: у поєднанні з Arduino програми можуть бути нескінченними. Домашня автоматизація, автоматизація роботи може бути виконана ефективно з мінімальними витратами.

1.3 Рівень розвитку

У цьому розділі ми обговорюємо стан справ у різних темах, включених до цього проекту. Оскільки комп'ютерний зір - це область, що швидко розвивається, з надзвичайно активною дослідницькою спільнотою, стан справ покращується кожні кілька місяців. Отже, будь-який майбутній читач повинен розуміти, що описаний тут стан техніки може бути трохи застарілим.

Більш того, сучасний стан у всіх областях включає глибокі нейронні мережі тієї чи іншої форми і, отже, потребує великих обчислювальних ресурсів. Отже, ми даємо схему методів тут, але ми не хотіли б імітувати їх через наші обчислювальні обмеження.

Мережі стиснення та збудження

CNN використовують свої згорткові фільтри для отримання ієрархічної інформації з зображень. Нижні шари знаходите тривіальні елементи контексту, такі як краї або високі частоти, у той час як верхні шари можуть виявляти особи, текст або інші складні геометричні форми. Вони поєднують просторову та канальну інформацію зображення. Різні фільтри спочатку виявляють просторові характеристики у кожному вхідному каналі, а потім додадуть інформацію з усіх доступних вихідних каналів. Але при створенні вихідних карт функцій мережа зважує кожен зі своїх каналів однаково.

Мережі стиснення та збудження (SENets) є будівельним блоком для CNN, який покращує взаємозалежність каналів практично без витрат на обчислення. Основна ідея полягає в тому, що кожен канал у згортковому блоці обробляється по-різному шляхом додавання параметра. Це означає, що мережа може адаптивно регулювати вагу кожної функцій. Цей новий підхід виграв випробування класифікації ILSVRC 2017 та досяг 25% відносного покращення порівняно з переможцем 2016 року.

- У цій статті передбачається, що глобальні середні значення функцій карт містять деякі закономірності, які допомагають при класифікації.
- Розпізнаючи такі закономірності, блок SE зможе «динамічно» масштабувати (помножуючи на 0–1) карти функцій таким чином, щоб полегшити класифікацію (за рахунок залучення відповідних функцій та придушення нерелевантних функцій)

- З іншого боку, масштаб кожної картки об'єктів залежить від масштабів інших карт об'єктів. Отже, блоки SE можна як «моделювання взаємозалежностей між каналами».
- Термін «стиснення» походить від моделі розпізнавання образів у кожному блоці SE, яка є мережею вузьких місць з одним прихованим шаром (ReLU) та логістичними виходами.

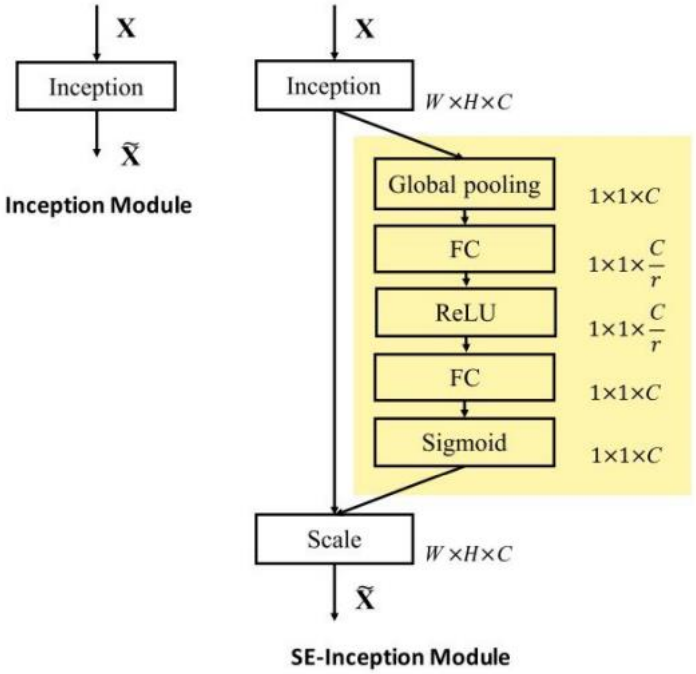


Рисунок 1.1: Vanilla ResNet проти модуля SE-ResNet

- «Стискання» в основному здійснюється з причин масштабування, тому що, якщо немає такого вузького місця, вимоги до параметрів можуть бути величезними (у документі пропонується коефіцієнт зменшення 16).
- Ефект блоку SE незначний на ранніх рівнях, тому що немає хорошого шаблону, який можна було б розпізнати, оскільки майже всі функції однаково поділяються.
- Ефект блоку SE стає більш пізнішими рівнями, тому що тепер кожен шар більш спеціалізований для кожного класу, що означає, що хороші шаблони можна легко знайти. Тож якщо ваша мережа не така глибока, блоки SE

мало допоможуть. У реальній реалізації накладні витрати блоку SE досить малі.

Автори показують, що додаючи SE-блоки ResNet-50, можна очікувати майже такої ж точності, як у ResNet-101. Це вражає моделі, що вимагає лише половини обчислювальних витрат. У документі далі досліджуються інші архітектури, такі як Inception, Inception-ResNet та ResNeXt. Останнє призводить їх до модифікованої версії, яка показує топ-5 помилок 3,79% на ImageNet.

MobileNets

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [9]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [9]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [9]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [43]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [43]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
BN-Inception [14]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [38]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

Рисунок 1.2: Як ResNets покращують існуючі архітектури

MobileNets - це сімейство мобільних моделей комп'ютерного зору для TensorFlow, розроблених для того, щоб ефективно максимізувати точність, пам'ятаючи про обмежені ресурси для вбудованої або вбудованої програми. Мобільні мережі - це невеликі моделі з малою затримкою та низьким енергоспоживанням, параметризовані для відповідності обмежень ресурсів у різних сценаріях використання. Їх можна використовувати для класифікації, виявлення, вбудовування та сегментації, аналогічно тому, як використовуються інші популярні великомасштабні моделі, такі як Inception.

Основний рівень MobileNet - це фільтри з розподілом по глибині, звані «згортка з розділенням по глибині». Мережева структура – ще один фактор підвищення продуктивності. Нарешті, ширину та роздільну здатність можна налаштувати для компромісу між затримкою та точністю.

Розділені по глибині згортки який є формою факторизованих згорток, які розкладають стандартну згортку на згортку по глибині і згортку, звану точковою згорткою. В MobileNet згортка по глибині застосовує один фільтр до кожного вхідного каналу. Потім у точковому згортці застосовується згортка 1x1, щоб поєднати вихідні дані згортки по глибині.

Стандартна згортка має обчислювальну вартість Dk.Dk.M.N.Df .Df,

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

Рисунок 1.3: Порівняння точності MobileNets

тоді як по глибині розгорнуті згортки мають обчислювальну вартість

Dk.Dk.M.Df .Df + M.N.Df .Df, де є N згорткових фільтрів розміру Dk.Dk та глибини M у стандартному випадку та є M згорткових фільтрів розміру Dk.Dk та глибини 1 та N точкових згорток розміру 1.1 та глибини M у випадку Depthwise Separable.

DeepPicar: Недорогий автономний автомобіль на базі глибокої нейронної мережі

Ця робота, виконана безліччю дослідників з Університету Канзасу та Університету Індіани в США, намагається навчити недорогу глибоку нейронну мережу (DNN) автономному водінню на Raspberry Pi 3.

DeepPicar - це невелика копія реального безпілотного автомобіля під назвою Dave-2 від NVIDIA, який їздив дорогами загального користування з використанням глибокої згорткової нейронної мережі (CNN), яка приймає зображення з фронтальної камери як вхідні дані та виробляє рульове керування автомобілем. кути як виведення. DeepPicar використовує майже ту ж мережну архітектуру - 9 рівнів, 27 мільйонів підключень і 250 тис. параметрів - і може бути навчений керувати собою в реальному часі за допомогою веб-камери та скромної чотириядерної платформи Raspberry Pi 3. Використовуючи DeepPicar, вони аналізують обчислювальні можливості 3 для підтримки наскрізного керування автономними транспортними засобами як реального часу на основі глибокого навчання.

Вони застосовують стандартну методику імітаційного навчання, щоб навчити машину рухатися доріжками на землі. Вони збирають дані для навчання та перевірки, вручну керуючи радіокерованим автомобілем і записуючи бачення (з веб-камери, встановленої на радіокерованому автомобілі) і людині, що вводяться, управляючі сигнали.

Потім вони тренують мережу в автономному режимі, використовуючи зібрані дані на настільному комп'ютері, який оснащений графічним процесором NVIDIA GT 210. Нарешті, навчена мережа копіюється назад у Raspberry Pi 3, який потім використовується для виконання операцій логічного виведення – локально на Pi 3 – в основному контур керування автомобіля в режимі реального часу. Використовуючи платформу DeepPicar, вони систематично аналізують її можливості в режимі реального часу в контексті управління в реальному часі на основі глибокого навчання, особливо під час виведення глибоких нейронних мереж у реальному часі.

На основі систематичного дослідження вони роблять низку цікавих спостережень з погляду систем реального часу. По-перше, вони виявляють, що логічний висновок DNN дуже передбачуваний - з точки зору часу - оскільки обсяг обчислень, необхідних для завершення одного виводу, фіксується під час розробки

архітектури DNN і не змінюється під час виконання різних вхідних даних (наприклад, різних кадрів зображення). Така передбачувана тимчасова поведінка, очевидно, є бажаною властивістю для систем реального часу. По-друге, вони виявили, що обробка CNN DeepPicar в реальному часі можлива на сьогоднішніх обчислювальних платформах, що вбудовуються, навіть таких же недорогих, як Raspberry Pi 3. Вони виявили, що цикл управління завершується менш ніж за 33,33 мс або 30 Гц, використовуючи всього два ядра чотириядерний процесор Raspberry Pi 3 (без використання графічного процесора) і може робити це у 100% випадків.

1.4.1 Попереднє дослідження

Перед тим, як розпочати наш проект, ми врахували попередні роботи та пов'язані з ними дослідження. У наступних розділах ми збираємося підбити підсумки найбільш актуальних досліджень, близьких до нашого проекту.

Реалізація алгоритмів комп'ютерного зору для робота

У цьому проекті представлені алгоритми для розпізнавання осіб, виявлення осіб, виявлення руху та визначення виразу обличчя на Raspberry Pi.

Основна мета проекту полягала у виконанні вищезгаданих чотирьох завдань на Raspberry Pi таким чином, щоб вони працювали майже в реальному часі, не споживали багато обчислювальної потужності та не сильно нагрівали Raspberry Pi.

Таблиця 1: Можливі покращення у роботі

Функція	Реалізація	Можливо покращити
Виявлення руху	Фонове виділення, бінаризація	Відстеження руху

Продовження таблиці 1.

Розпізнавання облич	Класифікатор Каскаду Хаара	Швидше, точніше
Розпізнавання облич	Каскад Хаара + Розпізнавання по Фішеру	Використання останньої моделі глибокого навчання
Розпізнавання виразів	Розпізнавання по Фішеру	Фокусування на класифікації

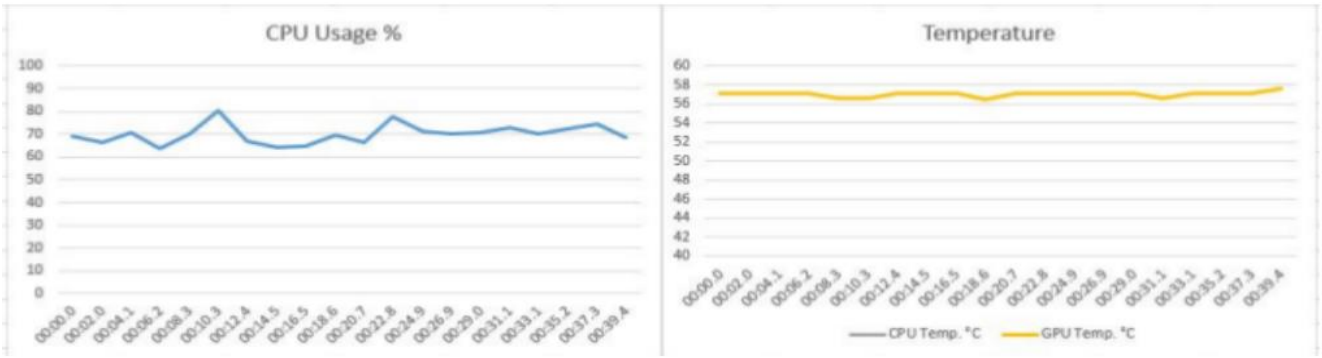


Рисунок 1.4: Використання ЦП та температури для HMI, виявлення руху, облич та виразів обличчя.

Видно, що, можливо, в гонитві за системою, яка не перегрівається та не використовує багато ЦП, представлені алгоритми здаються трохи застарілими за стандартами 2021 року. Більше того, передбачається найкращий варіант використання відстеження та класифікації руху, і, отже, вони будуть вивчені у нашому проекті замість виявлення руху та розпізнавання виразів.

У роботі представлена точність близько 69,5% для систем розпізнавання облич та експресії.

Глибоке навчання на Raspberry Pi для розпізнавання облич у реальному часі

Ця робота була виконана дослідниками з Цюріхського університету прикладних наук у Швейцарії, де вони описали швидкий і точний конвеєр для розпізнавання осіб у реальному часі, що базується на згортковій нейронній мережі (CNN) і потребує лише помірних обчислювальних ресурсів. Після навчання CNN на настільному ПК ми використовували Raspberry Pi, модель B для процедури

класифікації. Там ми досягли продуктивності приблизно 2 кадри на секунду і точності розпізнавання понад 97%. Пропонований підхід перевершує всі алгоритми OpenCV як за точністю, так і за швидкістю, і показує застосування останніх методів глибокого навчання до обладнання з обмеженою обчислювальною продуктивністю.

Оскільки навчання нейронної мережі потребує великих обчислювальних ресурсів, воно проводилося на графічному процесорі NVIDIA GeForce GTX 210 (500 хвилин). Потім навчена модель була перенесена в Raspberry Pi. На етапі навчання вони максимізували логарифмічну ймовірність навчальних даних за допомогою стандартного методу пакетного градієнтного спуску, використовуючи розмір пакета 10 і швидкість навчання = 0,1, яку вони безперервно зменшували в 0,993 рази протягом 1000 епох. Щоб розширити навчальний набір, вони згенерували нові зображення шляхом випадкової зміни вихідних зображень одним з наступних способів: поворот (від 3 до 6 градусів), переклад (від 1 до 2 пікселів), зміна масштабу (з коефіцієнтом від 0,9 до 1.1) та випадкове затемнення 20% пікселів. Як стандарт вони вибрали різні класифікатори OpenCV, і найвища продуктивність була досягнута з використанням класифікатора Fisherface. Вирівнювання зображень перед класифікацією підвищило точність класифікації з 87,5 до 96,9%. Тим не менш, оскільки процедура суміщення була заснована на виявленні обох очей, кількість зареєстрованих зображень зменшилася з N = 278 до N e = 192. Без попередньої обробки CNN показала початкову швидкість розпізнавання 24%. Застосування оператора LBP знизило залежність від освітлення, підвищивши швидкість розпізнавання до 82%. Штучне збільшення навчальних даних ще більше збільшило відсоток збігів до 97,5% всіх 278 тестових зображеннях.

1.5 Проектний вклад

Вклад цього проекту полягає у розробці алгоритмів, які можуть перетворити доступний за ціною елемент комерційного обладнання/комп'ютер з камерою, який легко знайти у будь-якому магазині, на інтелектуальну систему, здатну виявляти та розпізнавати особи та слідкувати за ними. його рух класифікуйте зображення за класами, не перегріваючи і не стикаючись з труднощами, і з хорошою точністю.

Це дозволить роботу в проекті розпізнавати особи людини, яка використовує її, визначати, чи дитина це чи доросла, визначати, хто вона, а потім продовжувати відслідковувати рухи людини. Це, по-перше, дозволить роботу діяти у різних режимах залежно від того, хто його використовує, і, отже, стане набагато персоналізованішим роботом. Відстеження руху також означало б, що дітей можна було б залишити наодинці з роботом без будь-якого нагляду, оскільки попередження могло б подаватися щоразу, коли дитина виходить з поля зору.

Оскільки наш інструмент був створений з навчальними цілями, весь код є відкритим, а також усі фреймворки, пакети та інше програмне забезпечення, яке використовується під час його розробки.

2 Управління

У цьому розділі пояснюється керування проектом. Ми надаємо опис цілей та перешкод, які ми виявили, масштаб проекту та його методологію, планування та аналіз стійкості.

2.1 Цілі

Основна мета проекту: надати Raspberry Pi 3 Model B можливість виявляти людину, класифікувати її дорослу або дитину, визначати, ким є ця людина, а потім точно відстежувати її рухи. Все це потрібно робити без перегріву Raspberry Pi або значного використання потужності процесора. Більше того, якщо можливо, робота повинна бути оптимізована за рахунок використання зовнішнього процесора (портативного комп'ютера), в якому дорогі в обчислювальному відношенні процедури, які мають бути запущені, відправляються на зовнішній процесор (портативний комп'ютер) через Wi-Fi, щоб можна було надсилати лише результати. назад. Це також можна зробити за допомогою хмарних платформ, таких як Google Colaboratory, які надають сервери Jupyter та безкоштовні графічні процесори для використання у хмарі.

Отже, цілі можна поділити на:

- Класифікація зображення на дорослий чи дитячий. Щоб визначити, чи є людина, яка використовує робота у проекті, дорослою чи дитиною.
- Розпізнавання осіб. Як попередник і допомога у процесі розпізнавання осіб

- Розпізнавання особи. Щоб визначити, хто використовує робота, і працювати в різних режимах, залежно від цього, а також включити персоналізацію.
- Виявлення руху. Переключайтеся між режимами, щоб заощадити ресурси, коли немає присутності.
- Відстеження руху. Щоб діти могли використовувати робота у кімнаті без нагляду дорослих.
- Оптимізувати за допомогою зовнішнього процесора
- Оптимізувати за допомогою хмарних сервісів

2.2 Перешкоди

Після опису цілей, яких ми хочемо досягти, необхідно зосередитись на перешкодах та обмеженнях, які ми виявили.

Raspberry Pi дуже повільно навчається

Навчання Raspberry Pi може зайняти дуже багато часу, тому тренуватись на ньому практично марно. Це надзвичайно ймовірна ситуація. Ми намагаємося уникати навчання нейронної мережі з нуля та намагаємось використовувати існуючі моделі за допомогою трансферного навчання. Отже, ми будемо лише перенавчати мережі. Оскільки це також займає занадто багато часу на Raspberry Pi, ми не маємо іншого вибору, крім як зробити це на зовнішньому процесорі.

Немодульовані готові алгоритми

Оскільки деякі стандартні алгоритми не можуть бути розділені на модулі, при цьому обчислювально інтенсивна частина невіддільна від іншої, використання зовнішнього процесора не може покращити продуктивність. У разі весь алгоритм виконується на комп'ютері, і лише результати передаються назад.

Недоступність або надмірна залежність від Wi-Fi

Оскільки всі наші оптимізації за допомогою зовнішнього процесора або хмарної служби залежать від доступності Wi-Fi, схоже, що існує надмірна залежність від Wi-Fi. Єдиний спосіб обійти це – переконатися, що алгоритми працюють досить ефективно всередині без необхідності такої оптимізації.

Зниження точності та ефективність обчислень

Компроміс між точністю алгоритму та обчислювальною ефективністю призводить до дилеми у вибір кращого алгоритму при порівнянні. Ця проблема вирішувалася індивідуально, оскільки кожен випадок був унікальним. Але загалом добре збалансований алгоритм – це те, що ми шукаємо. Те, що працює в розумні терміни і має досить хорошу точність.

Планування

Оскільки на завершення проекту залишалося близько чотирьох місяців, необхідно було розробити належний план і виконати основні етапи. Було встановлено жорсткий, але реалістичний графік та щотижневі зустрічі з наставником проекту, щоб гарантувати, що проект йде за графіком.

Помилки

Глибоке навчання по суті є досить складною областю. На щастя, існує безліч абстракцій, так що програмування нейронних мереж не таке складне, як могло б бути. Але необхідно розуміти, що робота на цих рівнях абстракції не обов'язково означає, що буде легко досягти правильного результату. Важливо писати код без помилок, оскільки помилки можуть мати каскадний ефект та призвести до

подальших ускладнень у майбутньому. Отже, час від часу проводилися модульні тести, щоб уникнути помилок у коді.

2.3 Обсяг

Для досягнення поставленої мети нам потрібно реалізувати кілька програм. Одна програма зможе виявляти та розпізнавати обличчя, потім малювати обмежувальну рамку над ними та ефективно відстежувати їх рух. Ми збираємося реалізувати цю програму на Python, який буде працювати поверх Raspbian OS і використовуватиме бібліотеки dlib та розпізнавання облич.

Ми пишемо іншу програму з використанням Python та бібліотеки Tensorflow для виконання класифікації. Але оскільки ми займатимемося навчанням та прогнозуванням, ми пишемо дві різні програми. Перед цим ми куруємо набір даних із фотографіями дітей та дорослих, і робимо це шляхом вилучення з зображень Google. У першій програмі ми використовуємо ці навчальні зображення навчання нейронної мережі і зберігаємо цю нейронну мережу як графа на диск. Друга програма приймає цей графік як вхідні дані і, використовуючи цей графік, прогнозує зображення, отримане з камери, щоб передбачити, чи є людина дорослою або дитиною.

Також ми збираємося вивчити всі відомі алгоритми виявлення та розпізнавання осіб від області комп'ютерного зору і порівняти їх, щоб знайти ті, які краще підходять для вирішення нашої проблеми або внести зміни в ті, які ближче до нашої нагоди, щоб зробити їх більш відповідними.

Більше того, ми спробуємо оптимізувати ці алгоритми за допомогою зовнішнього процесора (сервера) і надіслати дані з Pi на сервер для обчислень та отримати лише результати. Ми також намагаємося зробити те ж саме, використовуючи Google Colaboratory, хмарний сервіс, що надається Google, в якому ви безкоштовно отримуєте сервери Jupyter, підтримувані середовищем виконання GPU.

Кінцевим результатом буде RPi, який зможе виконувати згадані вище завдання автономно чи іншим чином, але майже реальному часі і без перегріву чи надмірного завантаження ЦП. RPi, який ми вибрали для цього проекту, - це Raspberry Pi 3 B, найдоступніша кожному.

2.4 Методологія та перевірка

Графік розробки проекту за досяжними етапами є ключем до досягнення цілей, що ми показали вище. Щоб досягти нашої мети, нам потрібна методологія роботи, постійний моніторинг, щоб переконатися, що ми робимо все правильно, та нарешті підтвердити остаточні результати, щоб оцінити якість проекту.

2.4.1 Методологія роботи

Перед тим, як почати цей проект, ми запевнили, що зможемо отримати Pi і всі необхідні матеріали, а також перевірити, чи цілі досяжні і чи не утопічні вони. Крім того, оскільки ми хотіли виконати певну інтенсивну обчислювальну роботу, ми подбали про те, щоб RPi міг з цим впоратися.

Після з'ясування цих питань перша частина проекту полягала у вивченні фреймворків глибокого навчання, виборі найбільш придатного для нашого проекту та отриманні додаткових відомостей про нього. Вибравши Tensorflow, ми повинні були отримати належні знання з різних тем, якими ми займаємося в галузі комп'ютерного зору, а також Raspberry Pi.

Після цього ми зробили перший підхід до мети. У цьому першому підході ми помітили деякі несподівані проблеми та деякі непередбачувані обставини, які змусили нас змінитись та шукати альтернативи.

Нарешті, остаточний підхід був реалізований, і ми досягли нашої мети, переслідуючи ту ж мету, але змінивши способи та засоби досягнення нашої мети, і ми провели кілька тестів, щоб перевірити якість проекту.

На рисунку 5 є кругова діаграма з інформацією про важливість звітності, дослідження, реалізації та тестування проекту.

Методологія, що використовується, була аналогічна методології Agile, кожна ітерація мала п'ять різних етапів:

1. Аналіз
2. Виконання
3. Тестування
4. Інтеграція

5. Переглянути розклад

Оскільки це був ітеративний підхід, не потрібно вносити істотних змін до методології, щоб врахувати непередбачувані зміни. Щотижневі зустрічі з супервайзером також дозволили переконатися в тому, що прогрес був досягнутий у правильному напрямку та що проект йшов правильним шляхом.

Короткий цикл розробки

Постановка цілей проводилася короткостроковій основі (щотижня) з ітеративним підходом. Оскільки масштаб проекту досить широкий, кожну частину потрібно було швидко доопрацювати та оптимізувати. У кожному з них були приблизно такі кроки:

1. Прочитати та зрозуміти алгоритм
2. Реалізувати алгоритм
3. Спробувати оптимізувати
4. Оцінка компромісу

Постійний зворотний зв'язок із клієнтами

Оскільки в цьому випадку немає реальних «клієнтів», було проведено постійну та ретельну оцінку проекту керівником проекту та іншими необхідними зацікавленими сторонами, які оцінюватимуть проект. Їхні відгуки зрештою визначили напрям проекту, а також швидкість та глибину охоплення кожної частини обсягу.

2.4.2 Підтвердження результатів

Щоб перевірити, чи Raspberry Pi може виконувати всі завдання, які ми очікуємо від нього, ми провели деяку демонстрацію, щоб перевірити, чи може він без проблем класифікувати, ідентифікувати особи та відслідковувати. Оскільки в ідеалі ми хотіли б робити це в режимі реального часу, але в ході роботи над проектом ми невдовзі зрозуміли, що це занадто утопічна мета, ми встановили контрольну позначку валідації, що дорівнює 2 секундам передбачення та 80%

точності як наш стандарт. Це було встановлено як стандарт перевірки в режимі, близькому до реального часу.

Ще одним простим критерієм перевірки була перевірка того, чи були використання ЦП і тепло, що виділяється менше, ніж повідомлялося в попередніх роботах.

2.5 Планування

У цьому розділі ми поговоримо про тимчасове планування проекту, описавши реалізовані завдання щодо його створення.

2.5.1 Ціль проекту

Мотивувавшись проблемою високого завантаження процесора та перегріву Raspberry Pi під час роботи алгоритми комп'ютерного зору / глибокого навчання, цей проект досліджує можливість використання зовнішнього процесора разом із Raspberry Pi та підключення до нього через Wi-Fi для підвищення продуктивності. Проект охоплює широкий спектр алгоритмів від виявлення та розпізнавання осіб до класифікації та аналізує безліч платформ від вбудованих алгоритмів на OpenCV до нейронних мереж, що настроюються на Tensorflow (обидва засновані на Python).

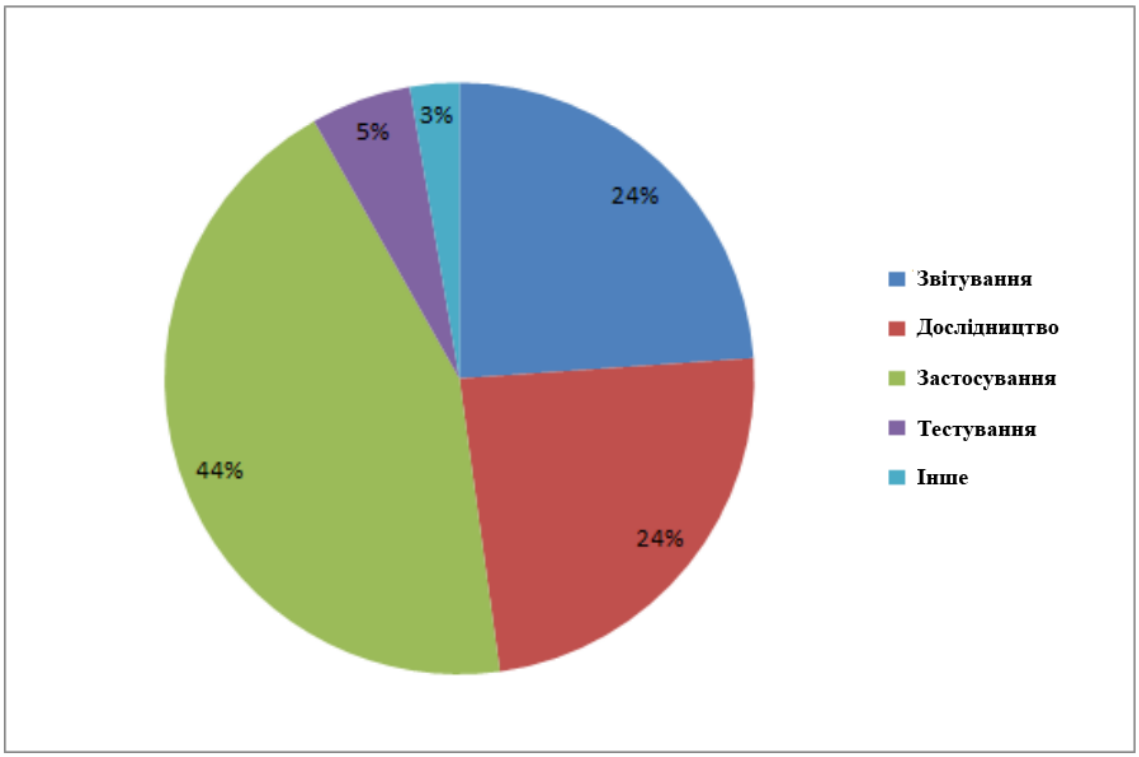


Рисунок 2.1: Кругова діаграма відсоткового змісту різних типів завдань

2.5.2 Планування проекту: Кроки

Набуття необхідних знань у галузі глибоких нейронних мереж, комп'ютерного зору і Raspberry Pi

Перш ніж заглибитись у тему, було важливо отримати необхідний обсяг знань та ознайомитись з темами, які збиралися досліджувати. Потрібні були перепідготовки з глибокого навчання і штучних нейронних мереж, з обробки зображень і комп'ютерного зору, і навіть по Raspberry Pi (ROS / Raspbian тощо. буд.).

Перед реалізацією кожного алгоритму кожної ітерації виконується достатнє отримання базових знань. Ресурси, що використовуються на цьому етапі - це сервер і Raspberry Pi.

Знайомство з програмними інструментами

Знання використовуваних мов програмування (Python) і використовуваних бібліотек (Tensorflow, OpenCV тощо. необхідне плавного просування проекту. Мені також потрібно було познайомитися з Raspbian OS та Linux Terminal. На цьому етапі ми також мали встановити програмне забезпечення Thenstahese на сервері, на якому працювала ОС Ubuntu. Всі апаратні ресурси, що використовуються, - це сервер і Raspberry Pi. Програмні ресурси: Tensorflow, Python, OpenCV, Raspbian.

Сучасний аналіз

На цьому етапі ми аналізуємо та порівнюємо кілька робіт (близько 5-10 різних статей) щодо виконання алгоритмів комп'ютерного зору на Raspberry Pi, а також про виконання алгоритмів комп'ютерного зору. швидко та з дуже невеликими накладними витратами. Сучасний стан алгоритмів комп'ютерного зору, таких як класифікація, є переможцем в останніх випробуваннях ImageNet.

Налаштування Raspberry Pi

Весь проект буде розроблений на конкретній мобільній платформі: Raspberry Pi 3, яка має чотири високопродуктивні процесорні ядра ARM Cortex-A53 з тактовою частотою 1,2 ГГц і графічний процесор VideoCore IV. Незважаючи на те, що є мобільною платформою, вона може працювати під управлінням ОС на базі Linux (Raspbian і навіть Ubuntu). Перший крок – помістити ОС Raspbian на карту пам'яті, а потім встановити її в Raspberry Pi. Потім необхідно встановити OpenCV 3.2 поверх Python із офіційного джерела. Встановити Tensorflow буде трохи складніше, тому що немає офіційної підтримки і потрібно зробити кілька хаків, щоб змусити його працювати. Веб-камера USB повинна бути підключена до Raspberry

Pi, яка діятиме як наша основна камера для захоплення відео та запуску виявлення. Малиновий пі,

Початкова характеристика

Перш ніж застосовувати будь-які оптимізації до існуючих алгоритмів, важливо реалізувати та проаналізувати існуючі алгоритми та зрозуміти, де знаходяться вузькі місця, тобто які частини коду займають більшу частину часу та час виконання. Це називається профільуванням або характеристикою продуктивності. Буде написаний сценарій python для отримання інформації про час виконання та пам'яті. Що стосується енергоспоживання та використання ЦП, у Raspberry Pi є системні змінні, які можна прочитати у певні моменти часу для отримання інформації. Ідея полягає в тому, щоб спочатку ретельно профілювати програму, а потім після кожної оптимізації буде виконуватися менш детальне профільування. Це завдання також вимагає людських ресурсів для написання сценаріїв, які будуть використовуватися для збору даних профілювання та подальшого аналізу даних щодо вузьких місць. Слід зазначити, що ця характеристика має бути зроблена як OpenCV, так Tensorflow. Raspberry Pi, миша, і SD-карта - це апаратні системи, що використовуються.

Оптимізація з використанням зовнішнього процесора та хмарних сервісів

Зовнішній процесор, тобто сервер з графічним процесором, підключений також Wi-Fi, що і Raspberry Pi, так що обчислювально-інтенсивні частини коду можуть бути відправлені на зовнішній процесор для обчислень. Raspberry Pi необхідно синхронізувати та обробляти дані з одним або декількома зовнішніми процесорами, до яких він підключений. Апаратні системи, що використовуються, - це Raspberry Pi, миша, клавіатура, веб-камера, зовнішній процесор і SD-карта. Як альтернативу програму можна було запустити в Google Colaboratory на серверах Jupyter, і лише результати були завантажені в локальну систему.

Фінальний етап

Це завдання полягає у перевірці того, що все працює як належить. Ми підготували все необхідне для здачі проекту, прокоментували код та доопрацювали всю документацію, щоб уникнути помилок та підготували фінальну презентацію.

Проблеми з веб-камерою та Raspberry Pi

Оскільки ми використовували USB-веб-камеру, підключену до Raspberry Pi, в деяких випадках через некоректне закриття програм деякі несанкціоновані процеси звертаються до USB-камери. Це означає, що веб-камера не може використовуватися програмою, яку ми хочемо запустити, тому що вона заблокована процесом, який її використовує.

Було написано bash-скрипт, який автоматично визначає наявність несанкціоновані процесу, що використовує веб-камеру, і вбиває його при кожному відкритті нового терміналу.

2.5.3 Ресурси

Ми ділимо ресурси, які збираємося використовувати для розробки проекту між обладнанням, програмним забезпеченням та іншими ресурсами.

Апаратні ресурси

1. Комп’ютер на базі CPU AMD FX-9590: Високопродуктивний сервер із графічним процесором Nvidia GeForce 210.

   Необхідний, оскільки алгоритми комп'ютерного зору та глибокого навчання надзвичайно дорогі у обчислювальному відношенні. Це також буде зовнішній процесор, підключений до Raspberry Pi через Wi-Fi.
2. Комплект Raspberry Pi 3B: Модель Raspberry Pi має швидкий процесор, а

також, що найважливіше, вбудований модуль Wi-Fi, необхідний для проекту. Комплект також постачається з усіма необхідними кабелями живлення та HDMI, необхідними для його налаштування.

3. USB-клавіатура: Підключений до RPi.
4. USB-миша: Підключений до RPi.
5. Веб-камера USB: Під'єднано до RPi.
6. Карта Micro SD 128 ГБ: Щоб встановити OpenCV та Tensorflow на Raspberry Pi, потрібно багато місця, оскільки це досить громіздкі бібліотеки. Отже, картки на 8 ГБ, що поставляється з Raspberry Pi недостатньо, і потрібна нова картка.

Програмне забезпечення

1. Windows 10 Professional.
2. Visual Studio Code: IDE для розробки у Windows. Середовище, в якому мені найбільш комфортно.
3. Visual Studio 2019 Professional: Для розвитку і налагодження в Windows.
4. Tensorflow: Розробка та впровадження нейронних мереж.
5. OpenCV: використовується у алгоритмах комп'ютерного зору.
6. Microsoft Office: Для цілей документації.

Інші ресурси

1. Електроенергія.
2. Доступ до мережі Інтернет.

2.6 Альтернативи та план дій

Якщо реальна тривалість зазначених завдань відрізняється від очікуваної тривалості, планування буде змінено. У (рідкісному) випадку, якщо тривалість задачі менша за очікувану, наступне завдання буде запущено негайно. З іншого боку, якщо завдання займає більше часу, ніж очікувалося, пізніша задача має бути скорочена або, в гіршому випадку, повністю опущена.

Ітерації такі, що всі кроки будуть виконані одного алгоритму перед переходом до наступного. Крім того, кроки 5 і знаходяться в порядку зменшення важливості, і якщо буде встановлено, що часу недостатньо, останній крок буде пропущений. Оскільки з керівником проекту були організовані щотижневі зустрічі, ви матимете достатньо часу, щоб виявити можливі відхилення від початкового плану та виправити їх.

Можливі проблеми:

1. Складність платформ

Платформу Tensorflow досить важко використовувати навіть на пристрої, на якому вона офіційно підтримується розробниками. Встановлення та робота на пристрої, на якому немає офіційної підтримки, означає, що налагодження помилок буде набагато складнішим, а розробка та виконання алгоритмів можуть бути не зовсім простими. OpenCV, для порівняння, набагато менш складний і не має проблем із переносимістю. Але він, як і раніше, складніший, ніж більшість інструментів розробки, з якими зазвичай працюють люди.

2. Помилки

Враховуючи, що фреймворк глибокого навчання є складним програмним забезпеченням, легко вносити помилки при зміні вихідного коду. Це також може

викликати затримки, якщо помилки не будуть виявлені та виправлені досить швидко.

3. Відсутність комплекту Raspberry Pi

3 Бюджет і сталий розвиток

3.1 Бюджет проекту

У цьому розділі наведено детальний опис витрат на цей проект, включаючи прямі, непрямі та непередбачені витрати. Крім загальних витрат проекту, він розбирає вартість кожного завдання проекту, можливі витрати на непередбачені проблеми, що збільшують бюджет, і непередбачені витрати.

3.1.1 Прямі витрати

Ми ділимо прямі витрати на три категорії: апаратне забезпечення, програмне забезпечення та людські ресурси. Для розрахунку амортизованого значення ми використовуємо загальну вартість розглянутого об'єкта і термін корисної експлуатації об'єкта в роках. Цей проект повинен бути завершений приблизно через півроку, тому амортизовані витрати розраховуються відповідним чином.

Апаратні ресурси

У таблиці 2 показана вартість кожного апаратного продукту, який ми збираємося використовувати при розробці проекту, і загальну вартість.

Таблиця 2: Вартість усіх апаратних ресурсів

Продукт	**Загальна вартість**	**Термін корисної експлуатації**
Преміум-комплект Raspberry Pi 3B	72,00 $	2 роки
Веб-камера USB	33,00 $	2 роки
USB-клавіатура	10,00 $	3 роки
USB-миша	3,00 $	3 роки
Картка MicroSD 128 ГБ	16,00 $	4 роки
Всього	**134,00 $**	

Програмні ресурси

У таблиці 3 показана вартість програмного забезпечення, необхідного для розробки проекту, і загальна вартість.

Таблиця 3: Вартість усіх програмних ресурсів

Продукт	**Загальна вартість**	**Термін корисної експлуатації**
Windows 10 Professional	200,00 $	3 роки
Visual Studio Code для Python 3	0,00 $	3 роки
Visual Studio 2019 Professional	540,00 $	4 роки
OpenCV	0,00 $	2 роки
Tensorflow	0,00 $	2 роки
Всього	**740,00 $**	

Людські ресурси

У таблиці 4 наведені витрати на людські ресурси, необхідні для розробки проекту.

Таблиця 4: Вартість усіх людських ресурсів

Роль	**Ціна / год**	**Кількість годин**	**Витрати**
Керівник проекту	30,00 $	50	1500 $
Розробник ПО	18,00 $	300	5400 $
Тестувальник	11,00 $	75	825 $
Всього		**425**	**7725,00** $

Щоб краще зрозуміти таблицю 4, ми збираємося розібратися в розбивці завдань, але без урахування апаратних і програмних ресурсів, а також без непрямих витрат. Ця інформація представлена в таблиці. 5.

Таблиця 5: Вартість кожного завдання (тільки з людських ресурсів)

Задача	**Витрачено годин**	**Ціна / год**	**Загальна вартість**
Отримання необхідних знань	100	20,00 $	2000,00 $
Знайомство з програмним забезпеченням	50	10,00 $	500,00 $
Сучасний аналіз	30	20,00 $	600,00 $
Налаштування Raspberry Pi	50	20 $	1000,00 $
Початкове наладка	45	25 $	1125,00 $
Оптимізація з використанням зовнішнього процесора	85	25 $	2000,00 $
Кінцева реалізація	25	20 $	500,00 $
Всього			**7725,00 $**

Середня зарплата програміста в Україні за даними thepage.ua становить 2500$. Припускаючи, що людина працює 264 дні на рік і працює 8 годин на день, вартість години становить близько 1,2$.

3.1.2 Непрямі витрати

Як і у всіх комп'ютерних проектах, існують непрямі витрати від використання електроенергії або паперу. Ці витрати підсумовуються в таблиці 6.

Таблиця 6: Вартість усіх непрямих витрат

Продукт	**Ціна**	**Кількість**	**Приблизна вартість**
Электричество	0,06 $ / кВт*г	1500 кВт	90,00 $
Доступ до мережі Інтернет	7,2 $/мес.	4 місяці	28,80 $
Всього			**118,8 $**

3.1.3 Непередбачені витрати

У таблиці 7 показаний найгірший сценарій у разі певних відхилень від плану і необхідності додаткових годин.

Таблиця 7: Непередбачені витрати

Роль	Ціна / год	Кількість годин	Витрати
Керівник проекту	30,00 $	5	150,00 $
Розробник ПО	18,00 $	30	540,00 $
Тестувальник	11,00 $	7,5	82,50 $
Всього		**42**	**772,50 $**

Ми оцінили ймовірність проблем при виконанні завдань на 10%. Ці непередбачені витрати становлять 10% цих цілей і включені в загальний бюджет.

3.1.4 Загальний бюджет

З економічною інформацією про ресурси, представлені в таблицях 2, 3, 4, 6, 7 та пунктом непередбачуваних ситуацій в таблиці 8, загальна вартість проекту деталізована.

3.2 Бюджетний контроль

Щоб контролювати бюджет, в кінці кожного завдання бюджет буде оновлюватися, щоб вказати фактичну кількість годин, вартість використаних ресурсів і вартість непередбачених подій, які могли статися. Ці цифри будуть порівнюватися з попередніми оцінками для отримання показників, які показують величину відхилення від початкового бюджетного планування.

Таблиця 8: Вартість усіх ресурсів

Концепція	**Витрати**
Апаратні ресурси	134,00 $
Програмні ресурси	740,00 $
Людські ресурси	7725,00 $
Непрямі витрати	118,8 $
Непередбачені витрати	772,50 $
Всього	**9490,30 $**

Може бути випадок, коли може знадобитися новий ресурс, такий як програмне забезпечення, про який ми дізнаємося тільки тоді, коли проект рухається в правильному напрямку.

Відхилення вартості = (EC - RC) відхилення споживання x RH = (EH - RH) xEC, де: EH = Розрахунковий годинник, EC = Орієнтовна вартість, RH = Реальний годинник, RC = реальна вартість.

Оскільки непередбачені витрати вже застосовано до загальної вартості, шанси на те, що проект вийде за рамки бюджету, невеликі. Бюджет розглядався як верхня межа, хоча ця вимога переглядалася на кожному етапі проекту. Щотижневі зустрічі з куратором проекту також доповнювали аналіз бюджету та відстеження прогресу. Це також був метод контролю.

3.3 Аналіз стійкості

Екологічність є ключовим фактором у розвитку будь-якого проекту. Ми оцінюємо проект на основі трьох факторів сталого розвитку, а саме економічної стійкості, соціальної стійкості та екологічної стійкості.

3.3.1 Економічний вимір

Проведено детальну кількісну оцінка всіх витрат, пов'язаних з проектом, як матеріальних, так і людських, як показано в попередніх розділах цього документа. Бюджет також був верхньою межею, і дуже малоймовірно, що він буде перевиконаний.

Велика частина програмного забезпечення, що використовується в проекті, є відкритим вихідним кодом, який має нульову вартість продукту, а єдиним програмним забезпеченням, яке поставляється за собівартістю, є операційна система (ОС), без якої ми не можемо створювати програмне забезпечення та програмне забезпечення документації.

Необхідні апаратні витрати - це тільки комп'ютери, які в даний час є обов'язковою частиною будь-якого проекту, і Raspberry Pi, який задуманий як дуже дешевий і доступний SOC. Це рішення не тільки ефективно використовує наявні ресурси, але і значно скорочує час роботи і вироблення тепла. Повторне використання та обмін існуючим обладнанням для підвищення ефективності є центральною темою рішення. Всі існуючі програми глибокого навчання для

комп'ютерного зору на робота або вимагають інтегрованого графічного процесора, який є дорогим, або повільними і, отже, споживають багато енергії. Це рішення також усуває необхідність.

Незважаючи на те, що ми можемо використовувати більш дешеві компоненти і робити той же проект, ми не знаємо, чи отримаємо ми таку ж точність. Також ціна нашого проекту доступна, а більшість використаних ресурсів вже є власністю університету.

3.3.2 Соціальний вимір

Цей проект в кінцевому підсумку може бути реалізований роботом, яким будуть користуватися діти. Той факт, що я можу використовувати технології, щоб допомогти іншим і створити щось, що матиме вплив, був постійним джерелом натхнення. Сучасні реалізації комп'ютерного зору на Raspberry Pi повільні і генерують багато тепла. Оскільки робот повинен використовуватися дітьми, жоден з них не є досконалим. Цей проект дасть мені досвід і неоціненну впевненість у сфері вбудованих систем і робототехніки.

Глибоке навчання та комп'ютерний зір були гарячими темами останнім часом, і в цих областях проводиться багато досліджень. Більш того, соціальна робототехніка також є швидкозростаючим полем. Цей проект, розташований в центрі цих трьох областей, може стимулювати додаткові дослідження, і навіть комерційні проекти, засновані на цих трьох областях, можуть отримати вигоду з цієї роботи.

Використовувана методологія

Запропоноване у цьому проекті є новим і може бути змінений або використаний у багатьох сферах, таких як спостереження, для поліпшення життя людей.

Впровадження технологій, що виконуються в цьому проекті, залишиться прозорим, щоб майбутня робота, яка буде базуватися на цих роботах, могла проводитися без будь-яких серйозних проблем.

Єдиний ризик, пов'язаний з цим, - це конфіденційність зібраних і використовуваних даних. У цьому цифровому світі розпізнавання облич все частіше використовується як метод аутентифікації для різних речей. Тому вкрай важливо, щоб інформація, яку ми зберігаємо, залишалася безпечною і не використовувалася для будь-яких незаконних цілей. Тому в розділі ризику вказується штраф в розмірі -1.

3.3.3 Екологічний вимір

Ресурси, необхідні на кожному етапі проекту описані, а їх вартість показана в кошторисі бюджету.

Хоча на перший погляд здається, що при додаванні додаткового процесора це рішення споживає більше електроенергії, ніж нинішнє, це не так. По-перше, цей проект пропонує зменшити споживання процесора, розсіювання тепла і час роботи алгоритмів на Raspberry Pi. Всі три допомагають зменшити споживання енергії. По-друге, новий комп'ютер не повинен бути повністю присвячений роботі з Raspberry

Pi. Комп'ютер, який увімкнуто та використовується для інших цілей, також може допомогти Raspberry Pi.

З розділів планування та бюджетного планування ми розуміємо, що у нас є комп'ютер, що працює протягом усього проекту. Припускаючи, що кількість споживаної одним комп'ютером енергії становить близько 250 Вт. А враховуючи, що ми витрачаємо на проект 500 годин, витрачена енергія становить 125 кВт.

Сучасний проект довго тренується на одній системі, навіть якщо це графічний процесор. Це призводить до підвищеного розсіювання тепла і використання великих систем охолодження. Більш того, при роботі з такими алгоритмами значно споживається електроенергія.

Навіть через забруднення через енергоспоживання цей проект отримає 8 балів у ресурсній зоні. Це правда, що ми використовуємо багато обладнання, але воно працює на електроенергії, а не на шкідливому та забруднюючому паливі, що робить його більш екологічно чистим.

4 Підготовка

Перш ніж розпочати пояснення розвитку проекту, необхідно мати деяку передісторію з пов'язаних тем проекту. Тому в цьому розділі ми даємо визначення та основні знання методів, алгоритмів та пакетів, що використовуються під час розробки проекту.

4.1 Базова система

Коли ми говоримо про базову систему проекту, ми маємо на увазі Raspberry Pi, який працює на Raspbian (ОС на основі Debian) для Raspberry Pi, та сервер, який використовується як зовнішній процесор і працює під керуванням Ubuntu. 14.04

Цей біном встановить зв'язок між комп'ютером та дроном. Він надасть нашому інструменту канал камери дрону для виконання необхідних алгоритмів, а також здійснюватиме зворотний зв'язок, отримуючи результат обчислення наших алгоритмів і використовуючи його для запуску двигунів для виконання бажаних рухів.

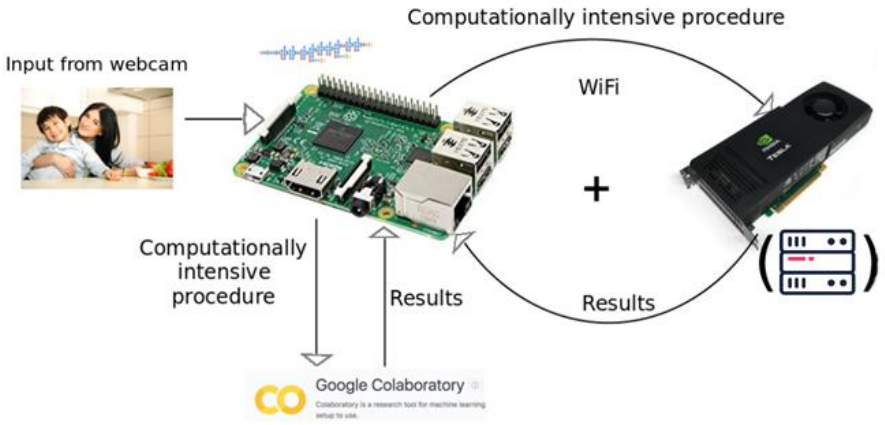


Рисунок 4.1: Огляд базової системи

4.1.1 Raspberry Pi

Raspberry Pi - це недорогий комп'ютер розміром із кредитну картку, який підключається до монітора комп'ютера або телевізора та використовує стандартну клавіатуру та мишу. Це здатний маленький пристрій, який дозволяє людям будь-якого віку вивчати обчислення та вчитися програмувати такими мовами, як Scratch та Python. Він здатний робити все, що ви очікуєте від настільного комп'ютера, від перегляду Інтернету та відтворення відео високої чіткості, створення таблиць, обробка тексту та відтворення гри.

Більше того, Raspberry Pi має здатність взаємодіяти із зовнішнім світом і використовувався у широкому спектрі проектів цифрових виробників, від музичних автоматів та батьківських детекторів до метеостанцій та інших проектів.

Технічні деталі

Raspberry Pi 3 Model B – це Raspberry Pi третього покоління. Він замінив Raspberry Pi 2 Model B+ у лютому 2016 року. Порівняно з Raspberry Pi 2 він має:

• SoC: Broadcom BCM2837 (приблизно на 50% швидше, ніж Pi 2)

• Чотирьохядерний процесор ARM Cortex-A53 із тактовою частотою 1,2 ГГц (набір інструкцій ARM V8)

• 1 ГБ LPDDR2-900 SDRAM

• 802.11n бездротова мережа

• Bluetooth 4.0

	Raspberry Pi 3 Model B	Raspberry Pi Zero	Raspberry Pi 2 Model B	Raspberry Pi Model B+
Introduction Date	2/29/2016	11/25/2015	2/2/2015	7/14/2014
SoC	BCM2837	BCM2835	BCM2836	BCM2835
CPU	Quad Cortex A53 @ 1.2GHz	ARM11 @ 1GHz	Quad Cortex A7 @ 900MHz	ARM11 @ 700MHz
Instruction set	ARMv8-A	ARMv6	ARMv7-A	ARMv6
GPU	400MHz VideoCore IV	250MHz VideoCore IV	250MHz VideoCore IV	250MHz VideoCore IV
RAM	1GB SDRAM	512 MB SDRAM	1GB SDRAM	512MB SDRAM
Storage	micro-SD	micro-SD	micro-SD	micro-SD
Ethernet	10/100	none	10/100	10/100
Wireless	802.11n / Bluetooth 4.0	none	none	none
Video Output	HDMI / Composite	HDMI / Composite	HDMI / Composite	HDMI / Composite
Audio Output	HDMI / Headphone	HDMI	HDMI / Headphone	HDMI / Headphone
GPIO	40	40	40	40
Price	$35	$5	$35	$35

Рисунок 4.2: Порівняння різних моделей Raspberry Pi

• 400 МГц VideoCode IV

• 4 порти USB

• 40 контактів GPIO

• Повний порт HDMI

• Порт Ethernet

• Комбінований 3,5 мм аудіороз'єм та композитне відео

• Інтерфейс камери (CSI)

• Відео-інтерфейс (DSI)

• Слот для картки Micro SD

Оскільки він має процесор ARMv8, він може працювати з повним спектром дистрибутивів ARM GNU/Linux, включаючи Ubuntu Core, а також Microsoft Windows 10.

Raspberry Pi 3 має форм-фактор, ідентичний попередньої (Pi 2) моделі B+, і повністю сумісний із Raspberry Pi 2.

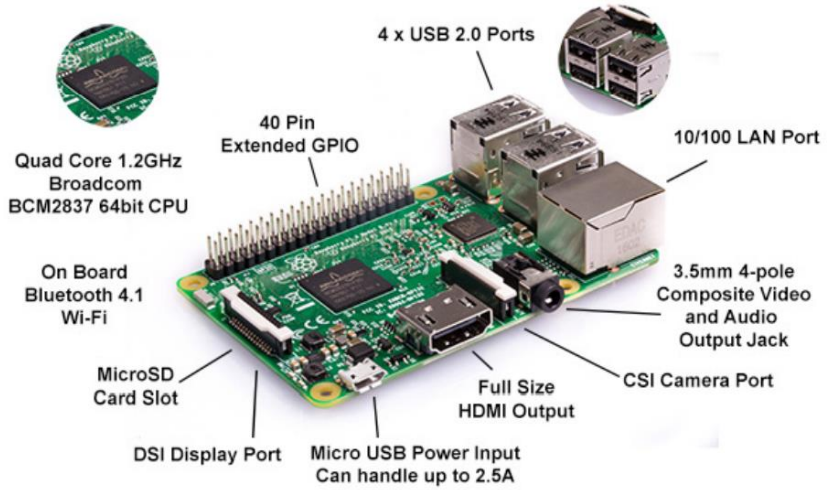


Рисунок 4.3: Raspberry Pi 3B

4.2 Операційна система Raspbian

Raspbian – це безкоштовна операційна система на основі Debian, оптимізована для обладнання Raspberry Pi. Операційна система – це набір основних

програм та утиліт, які запускають Raspberry Pi. Однак Raspbian надає більше, ніж просто ОС: він поставляється з більш ніж 35 000 пакетів, попередньо скомпільованим програмним забезпеченням у зручному форматі для легкої установки на Raspberry Pi.

Raspbian - це неофіційний порт Debian Wheezy armhf з налаштуваннями компіляції, налаштованими для створення оптимізованого коду з жорстким плаваючим кодом, який працюватиме на Raspberry Pi. Це забезпечує значно більш високу продуктивність додатків, що інтенсивно використовують арифметичні операції з плаваючою комою. Інші програми також отримають деяку продуктивність за рахунок використання розширених інструкцій процесора ARMv8 в Raspberry Pi.

Існує кілька версій Raspbian, включаючи Raspbian Stretch та Raspbian Jessie. З 2015 року він був офіційно наданий Raspberry Pi Foundation як основна операційна система для сімейства одноплатних комп'ютерів Raspberry Pi. Raspbian був створений Майком Томпсоном та Пітером Грін як незалежний проект. Початкове складання було завершено в червні 2012 року. Операційна система все ще перебуває у стадії активної розробки. Raspbian дуже оптимізований для низькопродуктивних процесорів ARM з лінійки Raspberry Pi.

Raspbian використовує PIXEL, Pi Improved Xwindows Environment, Lightweight як основне середовище робочого столу з моменту останнього оновлення. Він складається з модифікованого середовища робочого столу LXDE та віконного менеджера стекування Openbox з новою темою та деякими іншими змінами. Дистрибутив поставляється з копією програми комп'ютерної алгебри Mathematica та версією Minecraft під назвою Minecraft Pi, а також із полегшеною версією Chromium останньої версії.

4.3 Обробка зображень та комп'ютерний зір

Обробка зображень – ключовий момент нашого проекту. Ми повинні обчислювати наші алгоритми для кожного кадру, який камера надсилає нам, щоб досягти нашої мети.

У цьому розділі ми збираємося пояснити необхідні концепції для розуміння всього процесу з моменту отримання зображення, його обробки та відображення результатів.

4.3.1 OpenCV

Комп'ютерний зір із відкритим вихідним кодом

OpenCV - це бібліотека функцій програмування для комп'ютерного зору реальному часі. Як випливає з назви, це безкоштовна бібліотека для академічних та комерційних цілей, що випущена під ліцензією BSD з відкритим вихідним кодом. Він написаний на C++ і має інтерфейси C++, C, Python та Java, що працюють у Windows, Linux, Android та Mac. OpenCV має понад 2500 оптимізованих алгоритмів.

OpenCV має модульну структуру, що означає, що пакет включає кілька загальних або статичних бібліотеки. Доступні такі модулі:

• core: Компактний модуль, що визначає основні структури даних, включаючи щільні багатовимірні масив Mat та базові функції, що використовуються рештою всіх модулів.

• imgproc: Модуль обробки зображень, який включає лінійну та нелінійну фільтрацію зображень, геометричні перетворення зображень (зміна розміру, афінне та перспективне спотворення, загальні таблиці-перепризначення на основі), перетворення колірного простору, гістограми тощо.

• video: Модуль відеоаналізу який включає оцінку руху, віднімання фону і алгоритми відстеження об'єкта.

• calib3d: Основні алгоритми геометрії з кількома ракурсами, калібрування одиночної та стереокамери, оцінка положення об'єкта, алгоритми стереовідповідності та елементи тривимірної реконструкції.

• features2d: Детектори характерних ознак, дескриптори та зіставники дескрипторів.

• objdetect: Виявлення об'єктів та екземплярів визначених класів (наприклад, осіб, очі, гуртки, люди, машини тощо).

• highgui: Простий у використанні інтерфейс для захоплення відео, кодеків зображень та відео, а також прості можливості інтерфейсу користувача.

• gpu: Алгоритми із прискоренням на GPU із різних модулів OpenCV.

Ми використовуємо Python-інтерфейс OpenCV для досягнення потрібних нам функцій.

cv2.VideoCapture

Невід'ємною частиною нашого проекту є отримання зображення з веб-камери в реальному часі, його обробка та відображення результатів. OpenCV спрощує це завдання за допомогою класу VideoCapture. Клас VideoCapture має конструктори, які можна використовувати для отримання об'єкта VideoCapture для

пристрою або файлу, з якого хочемо читати. Фігура14 показує необхідні конструктори зі своїми аргументами.

C++: `VideoCapture::`**`VideoCapture`**`()`

C++: `VideoCapture::`**`VideoCapture`**`(const string& `**`filename`**`)`

C++: `VideoCapture::`**`VideoCapture`**`(int `**`device`**`)`

Python: `cv2.`**`VideoCapture`**`() → <VideoCapture object>`

Python: `cv2.`**`VideoCapture`**`(filename) → <VideoCapture object>`

Python: `cv2.`**`VideoCapture`**`(device) → <VideoCapture object>`

Рисунок 4.4: Конструктори VideoCapture

```python
import cv2, numpy as np    Import "cv2" could not be resolved

struct = np.ones((1,1), np.uint8)
vid = cv2.VideoCapture(0)
while True:
    s, frame = vid.read()
    frame = cv2.resize(frame, (400, 300))
    if s:
        frames = [cv2.resize(frame, (200, 150)),
                  cv2.resize(frame[0:200, 200:500], (400, 300)),
                  cv2.GaussianBlur(frame, (9, 9), 0),
                  cv2.cvtColor(frame, cv2.COLOR_BGR2HSV),
                  cv2.Canny(frame, 150, 200),
                  cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY),
                  cv2.dilate(frame, struct, iterations=1),
                  cv2.erode(frame, struct, iterations=1)
                  ]
        for i in range(0,7):
            cv2.imshow("Video" + str(i), frames[i])
        if cv2.waitKey(1) and 0xFF == ord('q'): continue
    else: break
```

Приклад 2. Код на Python для покадрового читання відео з веб-камери та відображення.

4.3.2 Бібліотека Numpy

NumPy – це фундаментальний пакет для наукових обчислень із Python. NumPy додає підтримку великих багатовимірних масивів та матриць, а також великий набір високорівневих математичних функцій для роботи з цими масивами. NumPy націлений на еталонну реалізацію Python CPython, яка не є оптимізуючим інтерпретатором байт-коду. Математичні алгоритми, написані для цієї версії Python, часто працюють набагато повільніше, ніж скомпіловані еквіваленти. NumPy частково вирішує проблему повільності, надаючи багатовимірні масиви, функції та оператори, які ефективно працюють з масивами, вимагаючи переписування деякого коду, переважно внутрішніх циклів, з використанням NumPy.

Використання NumPy в Python дає функціональність, порівнянну з MATLAB, оскільки обидва інтерпретуються, і обидва дозволяють користувачеві писати швидкі програми, поки більшість операцій працює з масивами або матрицями замість скалярів. Для порівняння, MATLAB може похвалитися великою кількістю додаткових наборів інструментів, зокрема Simulink, тоді як NumPy внутрішньо інтегрований з Python, більш сучасною та повною мовою програмування. Додаткові пакети Python доступні; SciPy - це бібліотека, яка додає більше функцій, подібних до MATLAB, а Matplotlib - це пакет для побудови графіків, який надає функціональні можливості побудови графіків, подібні до MATLAB. Всередині і MATLAB і NumPy покладаються на BLAS і LAPACK для ефективних обчислень лінійної алгебри.

Прив'язки Python бібліотеки комп'ютерного зору OpenCV використовують масиви NumPy для зберігання даних і роботи з ними. Оскільки зображення з кількома каналами представлені у вигляді тривимірних масивів, індексація, нарізка або маскування за допомогою інших масивів є дуже ефективними способами

доступу до певних пікселів зображення. Масив NumPy як універсальна структура даних OpenCV для зображень, витягнутих точок характеристик, ядер фільтрів та багато іншого значно спрощує робочий процес програмування та налагодження.

4.3.3 Tensorflow

TensorFlow - це програмна бібліотека з відкритим кодом для високопродуктивних чисельних обчислень. Його гнучка архітектура дозволяє легко розгортати обчислення на різних платформах (процесори, графічні процесори, TPU), а також від настільних комп'ютерів до кластерів серверів та мобільних та периферійних пристроїв. Спочатку розроблений дослідниками та інженерами з групи Google Brain в рамках організації Google AI, він має потужну підтримку машинного навчання та глибокого навчання, а гнучке ядро чисельних обчислень використовується в багатьох інших галузях науки.

TensorFlow надає офіційний Python API та C API; і без гарантії стабільності API: C++, Go та Java. Пакети сторонніх виробників доступні для C #, Haskell, Julia, R, Scala, Rust та OCaml.

Основна ідея програмування Tensorflow - це побудова графів обчислень, відомих як графи DataFlow. Потік даних – це загальна модель програмування для паралельних обчислень. У графі потоку даних вузли представляють одиниці обчислень, а краї представляють дані, які споживаються або вироблені обчисленням.

Простіше кажучи, обчислювальний граф - це серія операцій TensorFlow, організованих у графі вузлів. Кожен вузол приймає нуль або більше тензорів як

вхідні дані і виробляє тензор як вихідні дані. Один тип вузла – постійний. Як і всі константи TensorFlow, він не приймає вхідних даних і виводить значення, яке зберігається всередині.

Наприклад, у графі TensorFlow операція *tf.matmul* буде відповідати одному вузлу з двома вхідними ребрами (матриці, які потрібно помножити) та одним вихідним ребром (результат множення).

Графіки тензорного потоку виконуються всередині сеансу, який насправді зв'язок між клієнтською програмою - зазвичай програмою Python, хоча аналогічний інтерфейс доступний іншими мовами - і середовищем виконання C++.

Перевага використання цих графіків у тому, що вони є інтуїтивно зрозумілий спосіб візуалізації мереж, а виконання зворотного поширення набагато більш інтуїтивно зрозуміло.

Після навчання виконання прогнозу цьому графіку є просто прямий прохід графіка і аналіз отриманих результатів.

В останні роки впровадження Tensorflow було швидким і тому ми використовуємо цю бібліотеку для проектування наших нейронних мереж, навчання та виконання прогнозів. Для отримання додаткової інформації та детальних обговорень, завітайте на офіційний сайт Tensorflow.

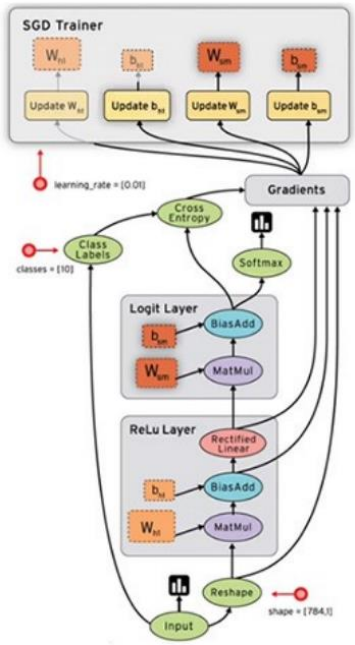


Рисунок 4.5: Приклад потоку TensorFlow

4.3.4 Бібліотека dlib

Dlib – це сучасний набір інструментів C++, що містить алгоритми машинного навчання та інструменти для створення складного програмного забезпечення на C++ для вирішення реальних проблем. Він використовується як у промисловості, так і в академічних колах у широкому діапазоні областей, включаючи робототехніку, вбудовані пристрої, мобільні телефони та великі високопродуктивні обчислювальні середовища. Він також постачається з інтерфейсом Python, який ми використовуємо.

Основні компоненти бібліотеки:

- Алгоритми машинного навчання
- Численні алгоритми

- Алгоритми виведення графічної моделі
- Обробка зображення
- Мережі
- Графічні інтерфейси користувача
- Алгоритми стиснення та цілісності даних
- Тестування

Тут ми використовуємо алгоритми машинного навчання, точніше метод кореляційного трекера. для процедур відстеження руху. Крім того, ми використовуємо бібліотеку розпізнавання облич для завдання розпізнавання облич, і ця бібліотека фактично побудована на основі бібліотеки dlib і використовує сучасну модель розпізнавання облич, створену за допомогою глибокого навчання та має точність 99,38%. у наборі даних Labeled Faces In Wild.

4.4 Алгоритми

У цьому розділі ми описуємо різні алгоритми, які використовувалися для виконання завдань, описаних у проекті.

4.4.1 Класифікація

Для виконання класифікації ми використовуємо концепцію, відому як трансферне навчання. Використовуючи “Transfer Learning”, ми використовуємо

вже існуючу модель, вносимо в неї деякі модифікації, щоб вона відповідала нашій проблемі, перенавчаємо її, а потім виконуємо прогнози по ній.

Сучасні моделі розпізнавання зображень мають мільйони параметрів. Для навчання з нуля потрібно багато розмічених навчальних даних і велика обчислювальна потужність (сотні GPU-годин і більше). Трансферне навчання - це метод, що дозволяє скоротити багато з цього, взявши частину моделі, яка вже була навчена для виконання пов'язаного завдання, та повторно використавши її у новій моделі. У цьому проекті ми повторно використовуємо можливості отримання ознак з потужних класифікаторів зображень, навчених в ImageNet, і просто навчимо новий шар класифікації поверх. Хоча це не так добре, як навчання повної моделі, це напрочуд ефективно для багатьох додатків, працює з помірними обсягами навчальних даних (сотні або тисячі, а не мільйони помічених зображень) і може бути запущено всього за тридцять хвилин на Raspberry Pi.

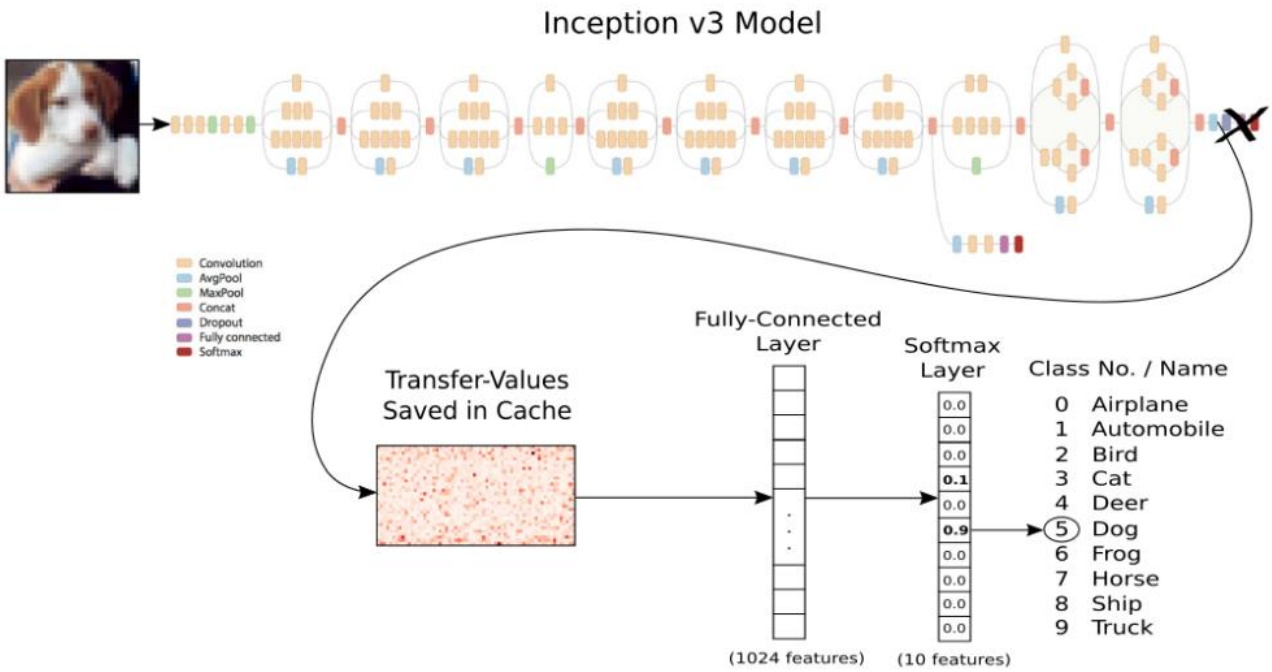


Рисунок 4.6: Зразок перепідготовки Inception

Кроки

Крок 1. Налаштувати набір даних

Зібрати достатню кількість навчальних зображень для необхідних класів, для яких потрібно навчити нейронну мережу. Слід подбати про те, щоб для кожного класу було достатньо прикладів, щоб усунути будь-яку можливу упередженість. Зображення поміщаються в різні папки відповідно до їхніх позначок класів, а ім'я папки збігається з ім'ям класу.

Крок 2. Завантажити модель Vanilla Inception V3

Модель Inception є особливо цікавою, тому що вона пройшла випробування в бойових умовах і показала результати світового класу у широко визнаному конкурсі ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Він також розроблений з розрахунком на ефективність, використовуючи у 12 разів менше параметрів, ніж у інших конкурентів, що дозволяє використовувати Inception у менш потужних системах.

Початок стандартний:

• Вхід 299x299x3, що представляє поле зору з 299 пікселів та 3 колірних (RGB) каналів.

• П'ять шарів ванільної згортки, з кількох операцій, що чергуються, з максимального об'єднання

• Стек модулів запуску

• Вихідний шар softmax в кінці та на проміжному вихідному шарі відразу після змішаного шару 17x17x768

Саме багаторазове накладання модулів Inception робить цю архітектуру «глибокою».

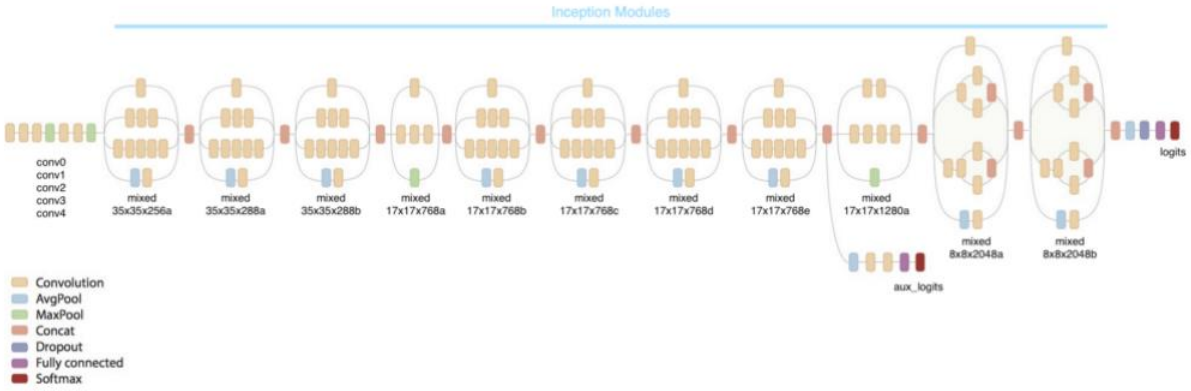


Рисунок 4.7: Стандартний початок

Крок 3. Перенавчити початкову модель

Завдяки Google, ми маємо скрипт retrain.py, який можна запустити прямо зараз. За замовчуванням скрипт завантажить заздалегідь навчену модель Inception V3.

Сценарій перенавчання є основним компонентом нашого алгоритму та будь-якої задачі власної класифікації зображень, яка використовує передачу навчання з Inception v3. Він був розроблений самими авторами TensorFlow для цієї конкретної мети (користувацька класифікація зображень).

Він навчає новий верхній шар (вузьке місце), який може розпізнавати певні класи зображень. Верхній шар отримує як вхідні дані 2048-мірний вектор для кожного зображення. Потім поверх цього уявлення навчається шар softmax. Припускаючи, що шар softmax містить N міток, це відповідає вивченню N + 2048*N (або 1001*N) параметрів моделі, що відповідають вивченим зміщенням та вагам.

Команда для використання сценарію перенавчання з усіма можливими налаштуваннями наведена нижче.

```
python {$your-working_directory}/retrain.py
--bottleneck_dir=/{$your-working_directory}/bottlenecks
--how_many_training_steps 500
--model_dir=/{$your-working_directory}/inception
--output_graph=/{$your-working_directory}/retrained_graph.pb
--output_labels=/{$your-working_directory}/retrained_labels.txt
--image_dir /{$your-working_directory}/${your_training_data_path}
```

Після того, як навчання вузьким місцям закінчиться, останній шар буде навчений, і ви побачите точність перевірки, як показано нижче. Наприкінці ви побачите "Остаточна точність тесту", і у вашому робочому каталозі будуть ще два файли: "retrained graph.pb" та "retrained labels.txt".

Вузькі місця зберігаються, тому на випадок, якщо ми захочемо знову перенавчити модель, вузькі місця можна використовувати безпосередньо, а не обчислювати заново.

Крок 4. Тестування нового набору даних

Тепер, коли ми маємо повторно навчений граф, записаний на диск як «retrained_graph.pb», і ми знаємо, що «retrained_labels.txt» містить імена різних міток, все, що залишається зробити, - це виконати один прямий прохід цього графік з новим зображенням, і він дасть вам позначку класу, до якого він належить. Код для маркування виглядає так:

```
import tensorflow as tf , sys    Import "tensorflow" could not be resolved
image_path = sys.argv [1]
image_data = tf.gfile.FastGFile(image_path,'rb').read()
label_lines = [line.rstrip() for line
    in tf.gfile.GFile( "D:/tensorflow_work/retained_labels.txt")]
with tf.gfile.FastGFile("D:/tensorflow_work/retrained_graph.pb",'rb') as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f. read ())
        _ = tf.import_graph_def(graph_def , name = '')
with tf.Session() as sess:
    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')
    predictions = sess.run(softmax_tensor , {'DecodeJpeg/contents:0': image_data })
    top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]
    for node_id in top_k:                (variable) node_id: Any
        human_string = label_lines[node_id]
        score = predictions[0][node_id]
        print('\%s(score=\%.5f)', (human_string , score ))
```

Скажімо, наприклад, ми розглядаємо випадок віднесення золотистих ретріверів до хаски і перенавчаємо початкову модель за тією ж. Якщо все піде правильно і ми додамо на графік зображення золотистого ретрівера під час тесту, ми повинні отримати щось подібне до того, що показано нижче:

```
golden retriever (score = 0.99933)
husky (score = 0.00067)
```

Рисунок 4.8: Результати тестування

4.4.2 Виявлення та розпізнавання облич

Для елементів виявлення та розпізнавання облич ми використовуємо бібліотеку розпізнавання облич, яка, в свою чергу, залежить від бібліотеки dlib, яка виконує виявлення рис обличчя та вбудовування облич із використанням ResNet з 29 сіточними шарами. По суті, це версія мережі ResNet-34 зі статті He, Zhang, Ren

та Sun Deep Residual Learning for Image Recognition з кількома віддаленими шарами та зменшеною удвічі кількістю фільтрів на шар. Мережа була навчена з нуля на наборі даних приблизно з 3 мільйонів осіб. Нижче показано псевдокод для досить спрощеного представлення алгоритму розпізнавання осіб.

Алгоритм 1. Псевдокод алгоритму виявлення та розпізнавання облич

Input: Video object of webcam, known face encodings
Import libraries
while True **do**
Grab a frame from webcam
Resize it to one-fourth for faster computation
Find the face locations in the image
unknown_encoding = Get the face encodings of that location
Find closest match to *unknown_encoding* in the *known_face_encodings*
Get name of the matched *known_face_encoding*
Draw a bounding box over the face
Write the name of the person on top of the box
Display image
end while

Пошук облич

Розпізнавання обличчя стало масовим явищем на початку 2000-х років, коли Пол Віола та Майкл Джонс винайшли спосіб виявлення осіб, який був досить швидким, щоб працювати на дешевих камерах. Проте набагато надійніше рішення існують зараз. Збиралися використовуйте метод, винайдений у 2005 році, який називається гістограмою орієнтованих градієнтів або скорочено HOG.

Щоб знайти обличчя на зображенні, ми почнемо з того, що зробимо наше зображення чорно-білим, тому що нам не потрібні дані про колір для пошуку облич. Потім ми подивимося на кожен піксель зображення по черзі. Для кожного окремого пікселя ми хочемо подивитися на пікселі, що безпосередньо оточують

його. Наша мета - з'ясувати, наскільки темнішим є поточний піксель у порівнянні з пікселями, що безпосередньо оточують його. Потім ми хочемо намалювати стрілку, яка показує, в якому напрямку зображення темніє. Якщо ви повторите цей процес для кожного пікселя зображення, ви закінчите тим, що кожен піксель буде замінено стрілкою. Ці стрілки називаються градієнтами.

Але збереження градієнта для кожного пікселя дає надто багато деталей. Зрештою, ми нудьгуємо за лісом за деревами. Було б краще, якби ми могли просто побачити основний потік світла/темряви на більш високому рівні, щоб ми могли побачити основний візерунок зображення.

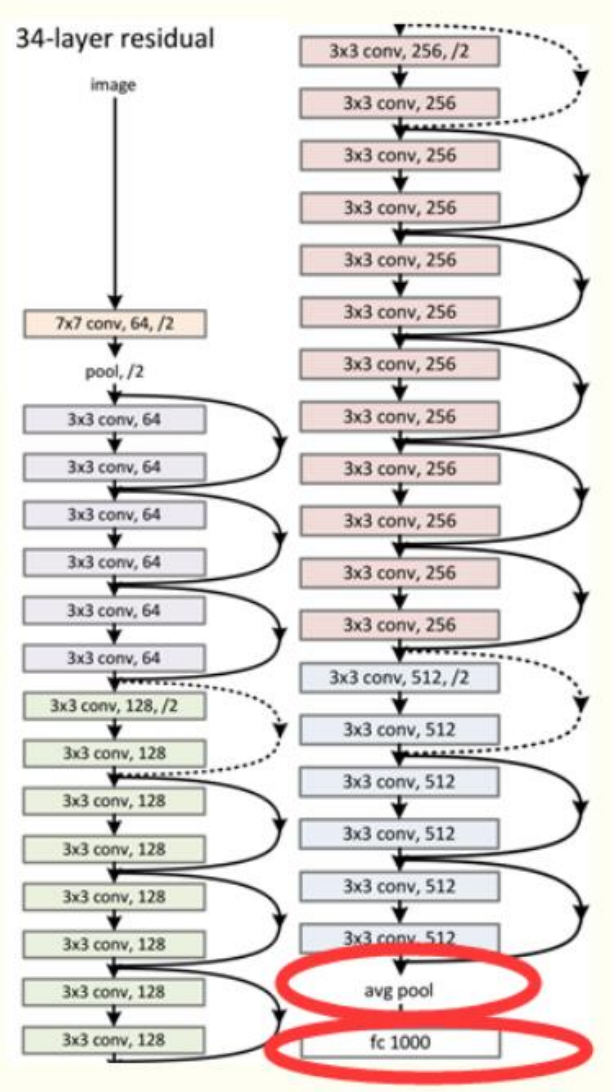


Рисунок 4.9: 34-шарова залишкова сітка

Для цього ми розділимо зображення на маленькі квадрати розміром 16×16 пікселів кожен. У кожному квадраті ми підрахуйте, скільки градієнтів вказує в кожному з основних напрямків (скільки точок вгору, вправо вгору, вправо і т. д.). Потім ми замінимо цей квадрат на зображенні стрілками, які були найсильнішими.

Щоб знайти обличчя на цьому зображенні HOG, все, що нам потрібно зробити, це знайти частину нашого зображення, яка найбільше схожа на відомий шаблон HOG, який був вилучений з багатьох інших навчальних осіб.

Розпізнавання обличчя

Рисунок 15: Вихідне зображення перетворене на представлення HOG, яке фіксує основні особливості зображення незалежно від яскравості зображення.

Щоб розпізнавати обличчя, нам потрібно мати достовірне зображення. Але оскільки порівняння реальних зображень досить складно і займає багато часу, ми використовуємо вбудовування обличчя, при якому ущільнюємо всі зображення

обличчя у вектор, а потім продовжуємо порівнювати тільки ці вектори, отримані з різних зображень, один з одним, а не саме зображення. .

Які виміри ми повинні збирати для кожного обличчя, щоб створити нашу базу даних відомих облич? Виявляється, що вимірювання, які здаються очевидними для нас, людей (наприклад, колір очей), насправді не мають сенсу для комп'ютера, що дивиться на окремі пікселі зображення. Дослідники виявили, що найбільш точний підхід – дозволити комп'ютеру обчислити виміри та зібрати їх самостійно. Глибоке навчання краще за людей допомагає зрозуміти, які частини обличчя важливо виміряти.

Рішення - навчити глибоку згорткову нейронну мережу. Але замість навчання мережі Щоб класифікувати зображення, як ми це робили раніше, ми збираємося навчити його генерувати 128 вимірювань для кожної особи. Саме для цього ми використовуємо архітектуру ResNet 34. Зазвичай цей вектор має розмір 128 і, отже, ущільнюємо зображення обличчя до 128 чисел. Втрата трійки використовується, коли відстань між двома кодуваннями однієї людини зводиться до мінімуму, а відстань між різними людьми максимізується.

Цей останній крок насправді найпростіший крок у всьому процесі. Все, що нам потрібно зробити, це знайти людину в нашій базі даних відомих людей, розміри якої найбільш близькі до нашого зображення. Для цього ми використовуємо простий лінійний класифікатор SVM.

4.4.3 Відстеження руху

Ми виконуємо відстеження руху за допомогою засобу відстеження кореляції, який надає бібліотека dlib. Трекери кореляції - як випливає з їхньої назви - працюють, зіставляючи набір пікселів від одного кадру до іншого. Цей інструмент є реалізацією методу, описаного в статті: Danelljan, Martin, et al. "Точна оцінка масштабу для надійного візуального відстеження". Праці Британської конференції з машинного зору BMVC.

Приклад коду для вибору рамки, що обмежує, і подальшого відстеження об'єкта в ній наведено нижче. Оскільки сам код читається та добре коментується, він наведений нижче замість алгоритму. Наступний код відстежує лише один об'єкт.

```
def run(source=0, dispLoc = False):
    cam = cv2.VideoCapture(source)
    if not cam.isOpened():
        print("Source not avaliable")
        exit()

    print('press key p to pause the video to start tracking')
    while True :
        retval , img = cam . read ()
        if not retval :
            print("Device unreachable")
            exit()
        if(cv2.waitKey(10)==ord('p')):
            break
        cv2.namedWindow("Image", cv2.WINDOW_NORMAL)
        cv2.imshow ("Image", img)
    cv2.destroyWindow ("Image")

    points = get_points.run(img)
    if not points:
        print('No obj to be tracked')
        exit()

    cv2.namedWindow("Image", cv2.WINDOW_NORMAL)
    cv2.imshow("Image", img)

    tracker = dlib.correlation_tracker()
    tracker.start_track(img, dlib.rectangle(* points [0]))

    while True :
        retval, img = cam.read()
        if not retval :
            print("Device unreachable")
            exit()
        tracker.update(img)
        rect = tracker.get_position()
        pt1 = (int(rect.left()), int(rect.top()))
        pt2 = (int(rect.right()), int(rect.bottom()))
        cv2.rectangle(img, pt1, pt2, (255 , 255 , 255) ,3)
        print ("Object tracked at [ { } , { } ] \ r " . format ( pt1 , pt2 ))
        if dispLoc :
            loc = (int(rect.left()), int(rect.top() -20))
            txt = "Object tracked at [ { } , { } ] " . format ( pt1 , pt2 )
            cv2.putText(img, txt, loc, cv2.FONT_HERSHEY_SIMPLEX, .5, (255 ,255 ,255) , 1)
        cv2.namedWindow("Image", cv2.WINDOW_NORMAL)
        cv2.imshow("Image", img)
        if cv2.waitKey(1) == 27:
            break
```

Тут клас отримання точок використовується для отримання точок прямокутника, обраних користувачем як область інтересу і які слід відстежувати. Він повертає координати прямокутника, який намальований користувачем.

Вищезгадана функція очікує, поки користувач перейде в режим відстеження, і як тільки це буде зроблено, користувачеві необхідно намалювати прямокутник над областю, яку захопила програма. Потім ініціалізується засіб відстеження кореляції, і об'єкт в області, що цікавить, відстежується шляхом кореляції набору пікселів від одного кадру до іншого.

5 Остаточна реалізація

У цьому розділі ми представляємо остаточну реалізацію проекту. Деякі з наведених тем не мають широкого пояснення, щоб спростити відстеження реалізації. Усі основні знання пояснюються у розділі «Попередні відомості», і ми рекомендуємо прочитати її перед тим, як розпочати цей розділ.

5.1 Класифікація

Спочатку ми зайнялися проблемою класифікації. Наша мета полягала в тому, щоб відрізнити дитину від дорослої, щоб система знала, хто використовує робота, і, отже, могла діяти у різних режимах.

5.1.1 Курування набору даних

Першим кроком було отримання розміченого набору даних про дітей та дорослих. На жаль, для цього не було готового набору даних, тому нам довелося вручну зробити це самостійно з пошуку картинок Google.

Тому був написаний власний скрипт javascript і python для автоматичного завантаження всіх зображень з результатів пошуку.

Нижче наведений код javascript використовується для завантаження всіх URL-адрес зображень, показаних в результаті пошуку зображень Google. Це зберігає весь код у текстовому файлі під назвою urls.txt.

```
varscript = document.createElement('script');
script.src = "https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js";
document.getElementsByTagName('head')[0].appendChild(script);
varurls = $('.rg_di.rg_meta').map(function(){returnJSON.parse($(this).text()).ou;});

vartextToSave = urls.toArray().join('\n');
varhiddenElement = document.createElement('a');
hiddenElement.href = 'data:attachment/text,'+encodeURI(textToSave);
hiddenElement.target = '_blank';
hiddenElement.download = 'urls.txt';
hiddenElement.click();
```

Після цього ми пишемо python-скрипт download.py для автоматичного завантаження всіх зображень з їх вихідні веб-сторінки одну за одною в бажану папку.

```
from imutils import paths
import argparse
import requests
import cv2
import os

ap = argparse.ArgumentParser()
ap.add_argument("-u","--urls",required=True, help="path to file containing image URLs")
ap.add_argument("-o","--output",required=True, help="path to output directory of images")
args=vars(ap.parse_args())
rows=open(args["urls"]).read().strip().split("\n")
total=0
for url in rows:
    try:
        r=requests.get(url,timeout=60)
        p=os.path.sep.join([args["output"],"{}.jpg".format(
        str(total).zfill(8))])
        f=open(p,"wb")
        f.write(r.content)
        f.close()
        print("[INFO]downloaded:{}".format(p))
        total+=1
    except:
        print("[INFO]error downloading {} ...skipping".format(p))

for imagePath in paths.list_images(args["output"]):
    delete=False
    try:
        image=cv2.imread(imagePath)
        if image is None:
            delete=True
    except:
        print("Except")
        delete=True
    if delete:
        print("[INFO]deleting{}".format(imagePath))
        os.remove(imagePath)
```

Ми повторюємо те саме для зображень дітей, а потім зображень дорослих. Оскільки ми використовуємо Google Пошук, є ймовірність того, що деякі зображення можуть бути помилковими, і, отже, необхідно спробувати відсіяти їх.

Це простий і ефективний спосіб створення налаштованого набору даних. Заради цього проекту ми куруємо стандартний набір даних зображень 500 дорослих та 500 дітей. Як тестовий набор ми завантажуємо набір даних приблизно 200 дітей, а також деякі інші фотографії дорослих.

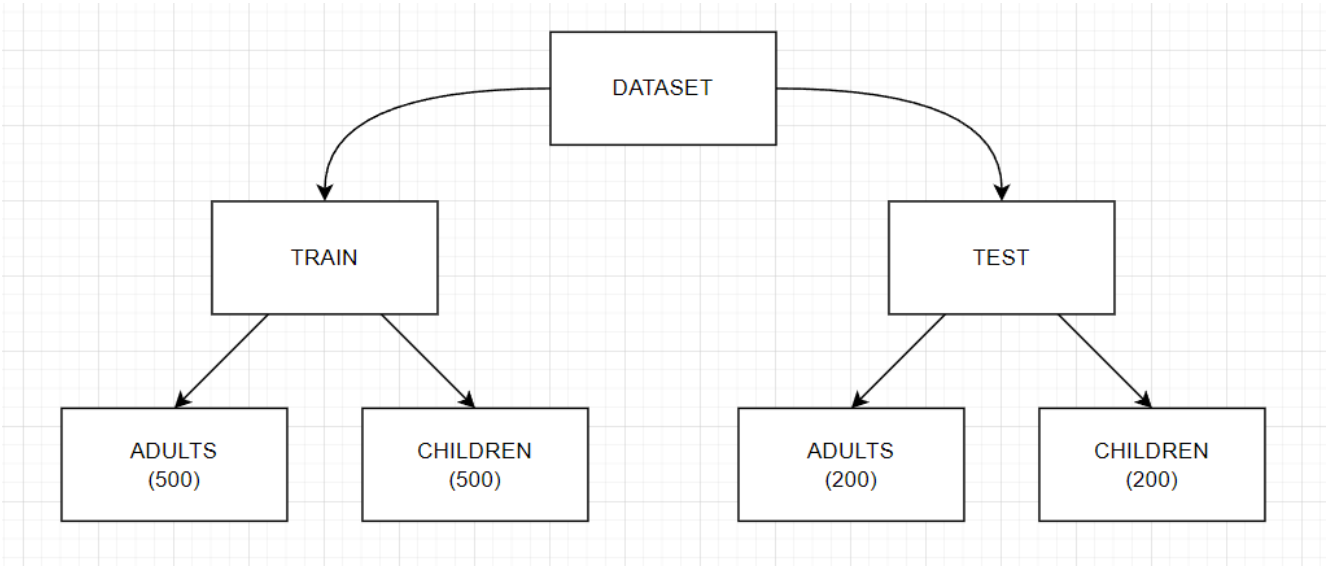


Рисунок 5.1: Використання набору даних.

5.1.2 Перенавчання та передбачення

Після завершення курування бази даних виконується перенавчання мережі Inception V3 для відповідності нашій задачі класифікації. Як тільки це буде зроблено, ми отримаємо повторно навчений граф "retrained_graph.pb" та " retrained_label.txt".

В результаті ми отримуємо графік розміром близько 85 МБ. Виконуючи прогнози, ми отримуємо точність 92% як на сервері, так і Raspberry Pi. Однак час прогнозування становить близько 1 с на сервері, тоді як на Raspberry Pi воно становить близько 15-17 с, і, отже, повільніше.

5.1.3 Використання MobileNet замість Inception V3

MobileNets – це клас згорткових нейронних мереж, розроблений дослідниками Google. Вони створені «в першу чергу для мобільних пристроїв», оскільки з самого початку спроектовані таким чином, щоб забезпечувати зручність використання ресурсів і швидко запускатися прямо на вашому телефоні або пристрої, що вбудовується.

Основна відмінність між архітектурою MobileNet і «традиційними» CNN полягає в тому, що один згортковий шар 3x3, за яким слідує пакетна норма і ReLU, MobileNets розділіть згортку на 3x3 по глибині і 1x1 по поточковому згортку.

Використовуючи той же набір даних та виконуючи прогнози в MobileNet замість Inception, точність знижується до 80% як на сервері, так і Raspberry Pi. Це займає близько 0,4 с прогноз на сервері, тоді як на Raspberry Pi потрібно близько 8-10 секунд. Розмір результуючого графіка становить близько 2 МБ.

5.2 Виявлення, розпізнавання та відстеження осіб

Ми намагаємось об'єднати всі три теми в єдину програму, щоб вона була корисною у проекті.

Ми виконуємо розпізнавання облич із подальшим відстеженням руху цієї особи замість розпізнавання облич. на кожному кадрі, щоб відстежувати його, тому що перший набагато менш тяжкий з точки зору обчислень і, отже, підходить для запуску на Raspberry Pi.

Для розпізнавання облич зображення відомих осіб повинні бути надані або під час виконання, або заздалегідь, щоб їх вкладення можна було вивчити.

Алгоритм 2. Псевдокод виявлення, розпізнавання та відстеження облич

Input: Video_object_of_webcam, known_face_encodings
Import libraries
while True **do**
Grab a frame from webcam
Resize it to one-fourth for faster computation
faces? ← Find if there are face locations in the image
if faces? is True **then**
unknown_encoding = Get the face encodings of that location
Find closest match to *unknown_encoding* in the *known_face_encodings*
name ← Get name of the matched known face encoding
bounding_box ← bounding box of detected face
break
end if
end while
Using the bounding box track the face and keep displaying the name

Продовження алгоритму 2.

if face exits screen **then**
Rise alert
end if

Отже, тут немає ручного втручання користувача, і все відбувається автоматично.

По-перше, система чекає появи обличчя, а доти просто чекає. Як тільки він виявляє обличчя, він одразу розпізнає, хто це, і знає своє становище. А потім він одразу ж переходить у режим відстеження, у якому постійно відстежується особа.

Якщо особа виходить за межі екрана, це означає, що дитина перестала користуватися роботом і, отже, попередження може бути надіслано якесь на зразок будильника або повідомлення.

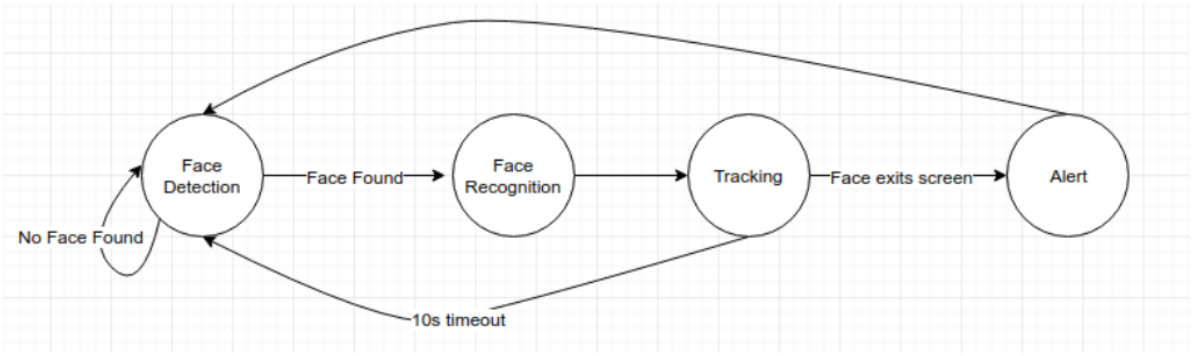


Рисунок 5.2: Діаграма станів запропонованої моделі

Кожні 10 секунд з модуля відстеження виконується повернення до стану виявлення особи, оскільки відстеження особи, яку ми виконуємо, в основному відстежує все, що знаходиться в рамці, що обмежує, повертається модулем виявлення осіб. Можливо, хоч і малоймовірний, що система відстеження руху може вибрати якийсь інший об'єкт у межах рамки, що обмежує, і відстежити його, а не обличчя. Отже, щоб мати страхову сітку, ми використовуємо цей цикл. Якщо є особа і вона відслідковується належним чином, цей цикл повинен завершитися, і управління має бути повернено модулю відстеження майже в найкоротші терміни, оскільки виявлення та розпізнавання відбуватимуться досить швидко.

5.3 Оптимізація з використанням зовнішнього процесора

Ключовою проблемою тут є розробка правильної методології відправлення даних з Raspberry Pi на сервер Wi-Fi. На щастя, простота та привабливість роботи в мережі з використанням сокетів, яку я дізнався під час курсу «Мережа», все ще свіжа у моїй пам'яті.

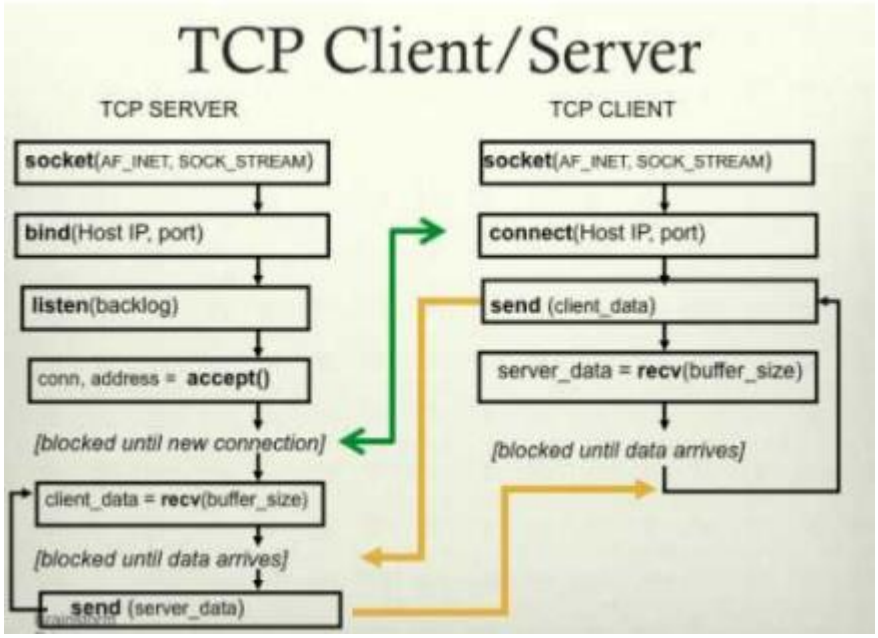


Рисунок 5.3: Структура програмування сокету

Сервер чекає, поки Raspberry Pis підключиться до нього. Після підключення він може отримати будь-які дані в байтах. Отже, клієнт Pi перетворює зображення на байти і відправляє його на сервер. Портативний комп'ютер відновлює зображення з його байтової форми, виконує необхідну обробку, знову пакує його в байти і відправляє назад клієнту, де воно відображається.

Тепер головне питання: скільки часу це займе?

Що ж, це сильно залежить від Wi-Fi, але в моделюванні, проведеному в лабораторії, майже немає затримок у потоковій передачі відео з Pi на сервер, коли відправляються Jpeg-файли з якістю 50% (що достатньо для виконання обробки.

Що нам потрібно). Надсилання кадру займає менше 0,1 с. Це означає, що при використанні цього методу буде загальна затримка близько 0,2 с. Враховуючи той факт, що сервер може працювати з нейронними мережами у 6-8 разів швидше, ніж у Raspberry Pi, є реальний сенс використати цей метод.

5.4 Оптимізація за допомогою Google Colaboratory

Незважаючи на те, що це був один із найбільш потенційно цікавих методів, виявлених у ході проекту, він не дав дуже добрих результатів. Частково це пов'язано з тим, що ще дуже рано повністю покладатися на Google Colaboratory, оскільки вона все ще досить недавня і безкоштовна.

Виникли проблеми за допомогою цього методу:

1. Іноді безкоштовні графічні процесори недоступні

2. Спільна робота продовжує давати збій

3. Браузер Chromium на Raspberry Pi продовжує давати збій

4. Іноді, що дивно, повільніше, ніж обчислення на Raspberry Pi.

Отже, це скоріше ставка на майбутнє, оскільки час від часу ми отримуємо перспективні результати, такі як малий час для обчислень.

5.5 Відстеження кількох об'єктів

Той самий алгоритм, який був запропонований для відстеження окремих осіб, може бути розширений і для відстеження кількох осіб. Це напрочуд добре працює як на сервері, так і на Raspberry Pi без перегріву.

Єдине застереження у тому, що запропонований нами раніше конвеєр об'єднання виявлення осіб, розпізнавання осіб і відстеження руху негаразд тривіально розширений у разі кількох об'єктів. Наші модулі виявлення та розпізнавання осіб повинні вміти обробляти кілька осіб та правильно малювати рамки, що обмежують. Хоча це можливо, у цьому проекті це не було втілено, але це хороша ідея, яку варто зробити.

У цьому рішенні рамки, що обмежують, намальовані вручну для осіб, які ми хотіли б відстежувати, і видно, що відстеження осіб у таких випадках відбувається досить добре.

Рисунок 5.4: Відстеження кількох осіб

5.6 Перешкоди

5.6.1 Несанкціонований процес з використанням USB-камери на Pi

При запуску завжди є несанкціонований процес, який тримає веб-камеру USB заблокованою та використовуваною. Якби це було так, потрібно було б увійти до root і визначити, ким був процес, і вручну вбити його, інакше камера була б непридатна для використання звичайними програмами.

Сценарій bash був написаний, щоб робити це автоматично при кожному відкритті нового терміналу. Таким чином, щоразу, коли відкривається термінал, перевіряється, чи використовується камера будь-яким процесом, і якщо це так, цей процес автоматично ідентифікується та завершується.

5.6.2 Фактори системного рівня, що впливають на продуктивність у реальному часі

При використанні платформи Raspberry Pi 3 необхідно враховувати кілька факторів системного рівня, а саме джерело живлення та температуру для досягнення стабільної та високої продуктивності в реальному часі. У всіх наших експериментах з Raspberry Pi 3 процесор працював із переважною тактовою частотою 1,2 ГГц. Однак без обережності ЦП може працювати на нижчій частоті.

Важливим фактором є теплове регулювання процесора, яке може вплинути на тактову частоту процесора, якщо температура процесора є надто високою (мікропрограма Pi 3 налаштована на регулювання на 85°C). Операції моделі DNN вимагають великих обчислювальних ресурсів, тому температура процесора може швидко підвищитись. Це може бути особливо проблематично у ситуаціях, коли кілька моделей DNN одночасно працюють на Pi 3. Якщо температура досягає порогового значення, спрацьовує теплове дроселювання Pi 3 та знижує тактову частоту до 600 МГц – половину максимальної частоти 1,2 ГГц. – щоб температура процесора залишалася на безпечному рівні. Ми виявили, що без належних рішень для охолодження (радіатора чи вентилятора) тривале використання системи призведе до зниження частоти процесора, що може вплинути на оцінку.

Ще один фактор, який слід враховувати, – це джерело живлення. На наш досвід, дроселювання частоти Pi 3 також спрацьовує, коли джерело живлення не може забезпечити необхідний мінімум струму в 2 А. В експериментах, проведених з блоком живлення, який давав тільки 1 А, Pi не зміг підтримувати тактову частоту 1,2 ГГц і замість цього коливався між роботою на частотах 600 МГц та 1,2 ГГц. В результаті необхідно або, принаймні, рекомендується, щоб джерело живлення, що використовується для Raspberry Pi 3, могло видавати 2 А, в іншому випадку оптимальна продуктивність не гарантується.

5.7 Остаточні результати

5.7.1 Класифікація

Класифікація з використанням моделі початку глибокого навчання була досить повільною Raspberry Pi. Це зайняло близько 15-17 секунд, тоді як на сервері знадобилося всього 1 с. Але завантаження ЦП у цей час є досить низьким, як показано нижче. З використанням цього досягається точність 92%.

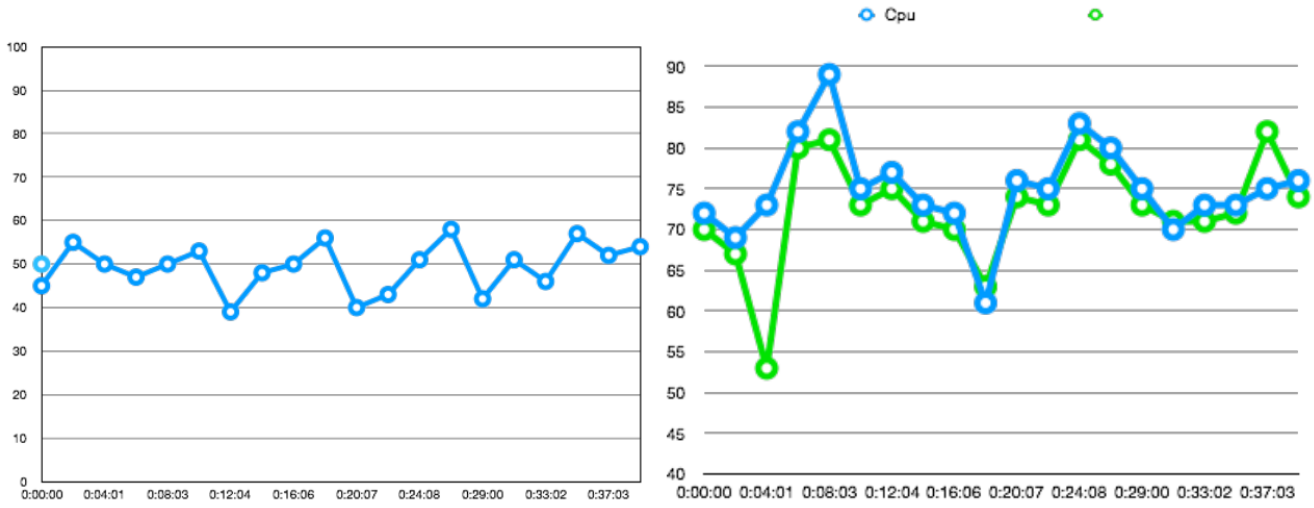


Рисунок 5.5: Завантаження ЦП та температура для класифікації лише за RPi

Тепер, завдяки оптимізації з використанням зовнішнього процесора, загальний час, необхідний класифікації зображення на Raspberry Pi, становить 4 секунди. 2 секунди йде на передачу кадру з Pi на сервер, а ще 2 секунди сервер використовує для виконання прогнозів на зображенні. Це чотириразове покращення продуктивності.

При використанні цього завантаження ЦП також нижче, ніж у попередніх порівняних роботах.

5.7.2 Виявлення, розпізнавання та відстеження облич

Вкладення осіб відомих людей потрібно вивчити лише один раз. Потім його можна розмістити на диску та зберегти. При кожному наступному використанні його можна легко прочитати з диска. Середній час розпізнавання особи становить близько 0,4 с на сервері та 2 с на Raspberry Pi. Відстеження руху більш-менш точне і в будь-якому разі оновлюється кожні 10 секунд для підвищення продуктивності. Середній час, що витрачається на вивчення осіб (яке у будь-якому випадку виконується лише один раз), становить близько 1 с на сервері та близько 2 с на Raspberry Pi.

Цей метод надійний і добре працює без будь-яких оптимізацій. Більше того, видно, що оптимізація цього з використанням зовнішнього процесора має багато проблем і не така проста. Отже, це було оптимізовано з використанням зовнішнього процесора і залишено як є.

Також було виконано відстеження кількох об'єктів, і Raspberry Pi може впоратися із цим без особливого дискомфорту.

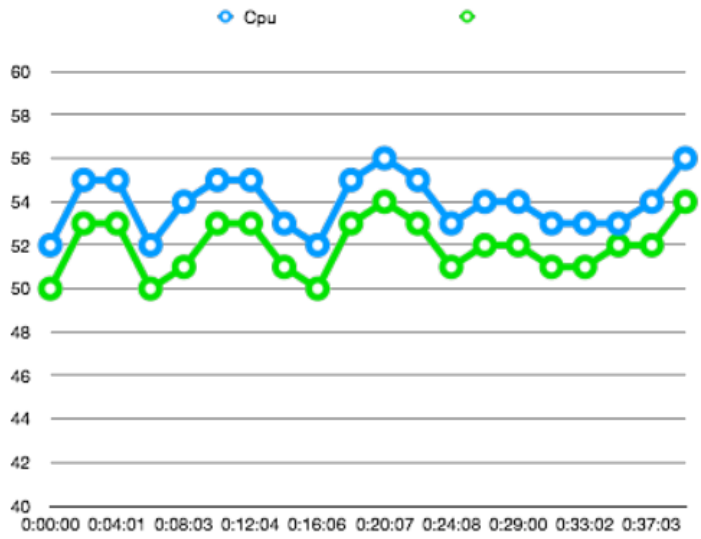


Рисунок 5.6: Зміна температури на RPi

ВИСНОВКИ

Розроблено інструмент, який використовує Raspberry Pi 3 B, який здатний виконувати різні операції комп'ютерного зору, такі як класифікація, відстеження руху, розпізнавання облич та розпізнавання облич.

Базова система обробляється операційною системою Raspbian та зовнішнім процесом (сервером), оптимізовано продуктивність, обмінюючись даними між ними.

Для завдання класифікації використано добре відому модель Inception V3 та моделі MobileNet та Transfer Learning з використанням інфраструктури Tensorflow. Цей метод дає хороші результати, хоча на Raspberry Pi він трохи повільний. Рекомендується використовувати зовнішній процесор разом з Pi, щоб покращити продуктивність.

Для задач виявлення, розпізнавання та відстеження руху використано бібліотеку OpenCV разом з бібліотеками dlib та розпізнавання облич та створюємо конвеєр робочого процесу, який автоматично виконує всі три операції бажаним чином.

Наступним завданням виконується відстеження кількох об'єктів. Хоча це не було досліджено в цілому, багатообіцяюче з'ясувати, що він дійсно може бути ефективно виконаний як на Raspberry Pi, так і на сервері.

Оптимізація з використанням зовнішнього процесора дуже перспективна, тому що затримок у відправленні майже немає між зображенням з камери від Pi до сервера. Це основний метод оптимізації, який дає дуже добрі результати.

Оптимізація з використанням Google Colaboratory не дала задовільних результатів тому. Наразі сервіс знаходиться в активній розробці та тестуванні. По

закінченню цього циклу сервіс модливо буде використати на повну, що дасть значний приріст у швидкості обробки данних.

Незважаючи на те, що інструмент має хорошу продуктивність, можна розробити безліч покращень, щоб оптимізувати програму.

ПЕРЕЛІК ПОСИЛАНЬ

1. Video tracking [Електронний ресурс] // Wikipedia, The Free Encyclopedia. – 2021. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Video_tracking.
2. Visual Navigation for the Parrot AR.Drone. [Електронний ресурс] // Technische Universitat Munchen. – 2019. – Режим доступу до ресурсу: https://vision.in.tum.de/data/software/tum_ardrone.
3. OpenTLD [Електронний ресурс] / G. Nebehay. – 2011. – Режим доступу до ресурсу: https://github.com/gnebehay/OpenTLD.
4. About ROS [Електронний ресурс] // Open Source Robotics Fundation. – 2020. – Режим доступу до ресурсу: https://www.ros.org/about-ros.
5. ROS Tutorials [Електронний ресурс] // Open Source Robotics Fundation. – 2019. – Режим доступу до ресурсу: https://wiki.ros.org/ROS/Tutorials.
6. Ardrone Autonomy Docs [Електронний ресурс] // Autonomy Lab. – 2018. – Режим доступу до ресурсу: https://ardrone-autonomy.readthedocs.org/en/latest/.
7. Node.js homepage [Електронний ресурс] // Joyent. – 2021. – Режим доступу до ресурсу: https://nodejs.org/.
8. Ardrone [Електронний ресурс] // Nautical pea maker. – 2019. – Режим доступу до ресурсу: https://www.npmjs.com/package/ar-drone.
9. Parallel Tracking and Mapping for Small AR Workspaces [Електронний ресурс] / G. Klein, D. Murray // University of Oxford. – 2018. – Режим доступу до ресурсу: https://www.robots.ox.ac.uk/gk/publications/Slides_KleinMurray2007ISMAR.pdf.
10. OpenTLD [Електронний ресурс] // Austrian Institute of Technology – Режим доступу до ресурсу: https://www.gnebehay.com/tld/.
11. Raspberry Pi [Електронний ресурс] // The Raspberry Pi Foundation. – 2021. – Режим доступу до ресурсу: https://www.raspberrypi.org/.
12. Field and Service Robotics [Електронний ресурс] / L. Mejias, P. Corke, J.Roberts // Results of the 9th International Conference. – 2015. – Режим доступу до ресурсу: https://www.researchgate.net/publication/321572149_Field_and_Service_Robotics_Results_of_the_9th_International_Conference.

13. View-based Maps [Електронний ресурс] / K. Konolige, J. Bowman, J.D. Chen та ін.] // The International Journal of Robotics Research – Режим доступу до ресурсу: https://ijr.sagepub.com/publishes/robo/last/viewbasedmaps.
14. Parallel Tracking and Mapping [Електронний ресурс] – Режим доступу до ресурсу: https://ewokrampage.wordpress.com/about.
15. H. Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms [Електронний ресурс] / H. Durrant-Whyte, T. Bailey // IEEE Robotics & Automation Magazine – Режим доступу до ресурсу: https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/readings/Durrant-Whyte_Bailey_SLAM-tutorial-I.pdf.
16. International Symposium on Mixed and Augmented Reality [Електронний ресурс] // ISMAR – Режим доступу до ресурсу: http://www.ismar07.org/.
17. The vSLAM Algorithm for Navigation in Natural Environments [Електронний ресурс] / N. Karlsson, L. Goncalves, P. Pirjanian, M. Munich // Evolution Robotics – Режим доступу до ресурсу: https://www.semanticscholar.org/paper/The-vSLAM-Algorithm-for-Robust-Localization-and-Karlsson-Bernardo/be15fdcd4aa6d795af5dee94f8f625b86452e0c0.
18. Raspbian [Електронний ресурс] // Raspbian – Режим доступу до ресурсу: https://www.raspbian.org/.
19. Ardrone Autonomy [Електронний ресурс] // AutonomyLab – Режим доступу до ресурсу: https://github.com/AutonomyLab/ardrone_autonomy.
20. Cvg Ardrone2 Ibvs [Електронний ресурс] // Vision4UAV – Режим доступу до ресурсу: https://github.com/Vision4UAV/cvg_ardrone2_ibvs.
21. Using cv_bridge to convert between ROS images and OpenCV images [Електронний ресурс] // Open Source Robotics Fundation – Режим доступу до ресурсу: https://wiki.ros.org/cv_bridge/Tutorials/UsingCvBridgeToConvertBetweenROSImagesAndOpenCVImages.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ
МАГІСТЕРСЬКА РОБОТА НА ТЕМУ:
«РОЗРОБКА МОДЕЛІ ІНФОРМАЦІЙНОЇ СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ ВИЗНАЧЕННЯ ХАРАКТЕРИСТИК ОБ'ЄКТІВ НА БАЗІ RASPBERRY PI»
ВИКОНАВ: СТУДЕНТ ГРУПИ ІСДМ-61 МАКАРЦЕВ МАКСИМ ОЛЕКСАНДРОВИЧ
КЕРІВНИК: ТУШИЧ АЛІНА МИКОЛАЇВНА

RaspberryPi

OpenCV

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)

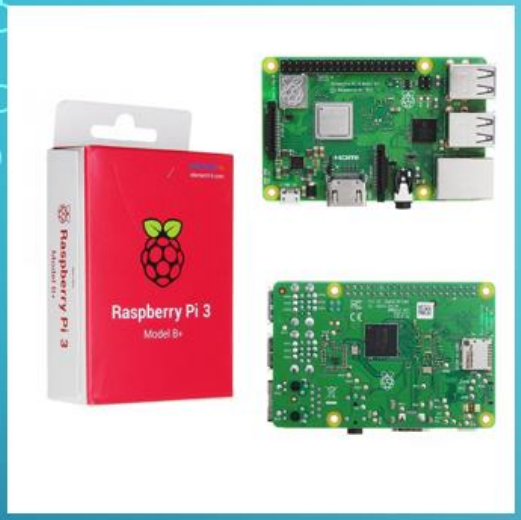
1)МЕТА МАГІСТЕРСЬКОЇ РОБОТИ - ДОСЛІДЖЕННЯ ЕФЕКТИВНИХ МЕХАНІЗМІВ АНАЛІЗУ ЗОБРАЖЕННЯ З МЕТОЮ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ І ВЗАЄМОДІЇ З НИМИ.

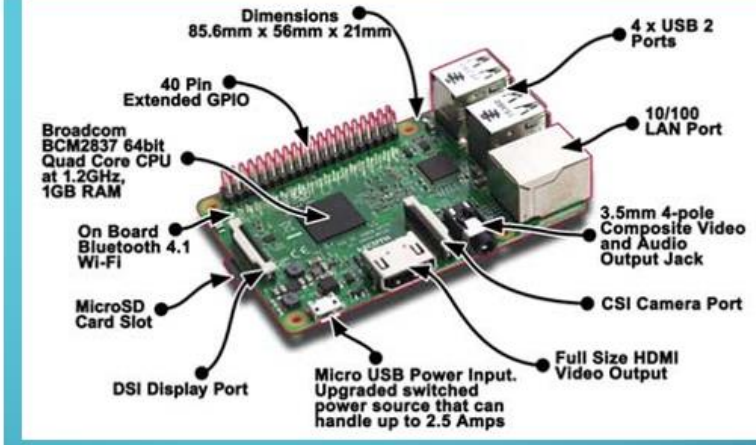
2)ОБ'ЄКТ ДОСЛІДЖЕННЯ – АПАРАТНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ.

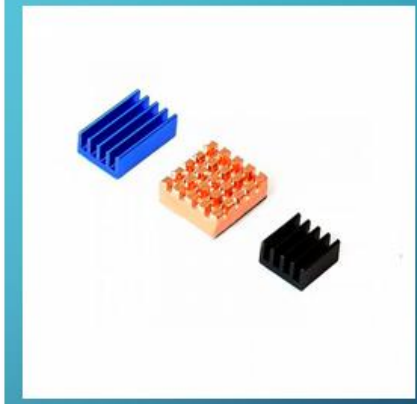
3)ПРЕДМЕТ ДОСЛІДЖЕННЯ – СТРУКТУРИ, СЦЕНАРІЇ РОБОТИ З АНАЛІЗОМ ЗОБРАЖЕНЬ.

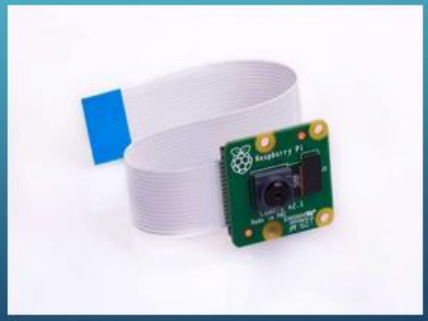
4)НАУКОВА НОВИЗНА МАГІСТЕРСЬКОЇ РОБОТИ - ПОЛЯГАЄ В ВИБОРІ ЕФЕКТИВНИХ МЕТОДІВ ТА АПАРАТНИХ ПЛАТФОРМ У АНАЛІЗІ ЗОБРАЖЕНЬ І ВІДЕО ДЛЯ ПОДАЛЬШОЇ ВЗАЄМОДІЇ З НИМИ.

ВИБІР ПЛАТФОРМИ

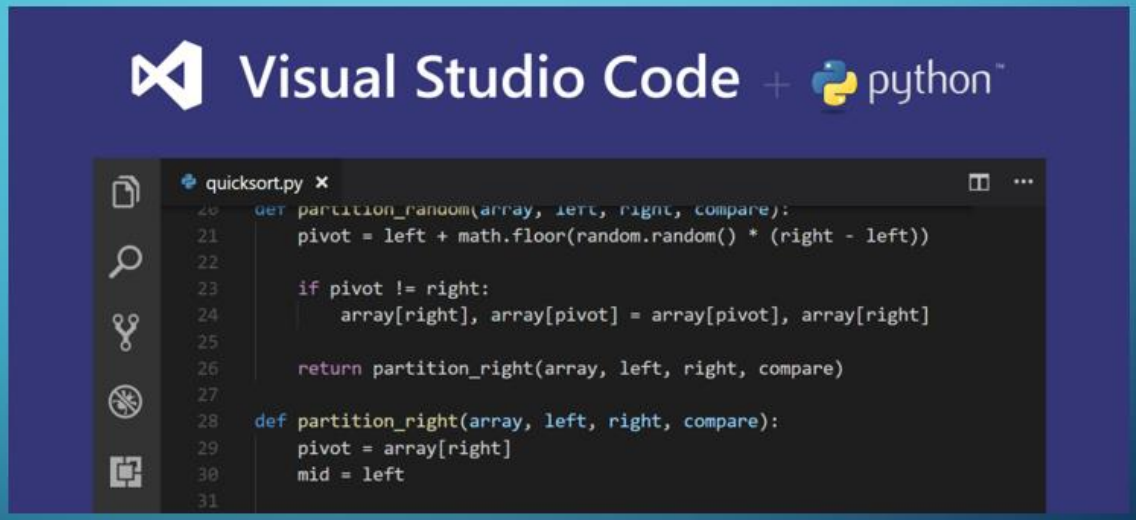




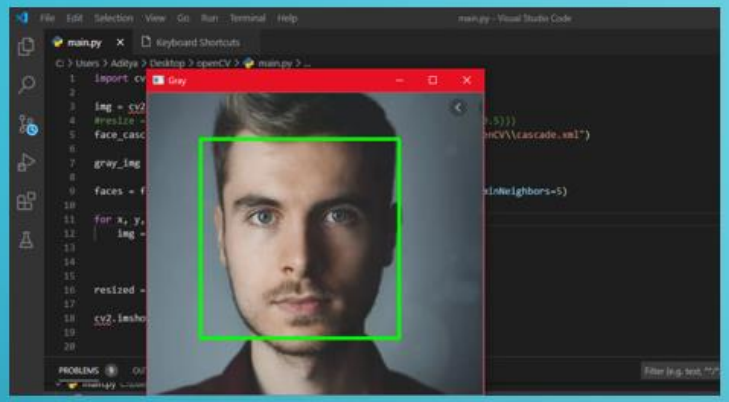


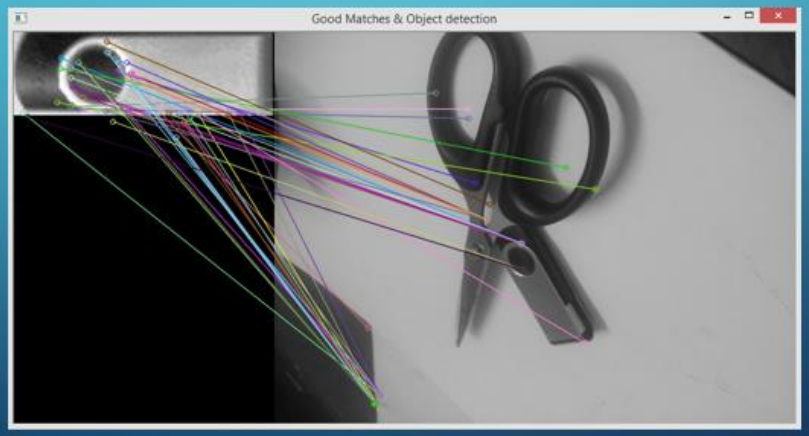


Python та IDE



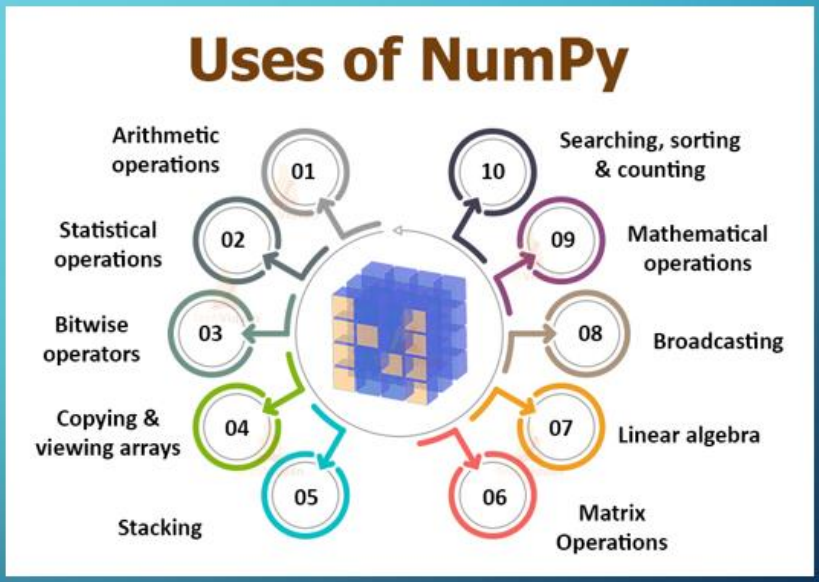
Open CV


main.py
Keyboard Shortcuts
Gray

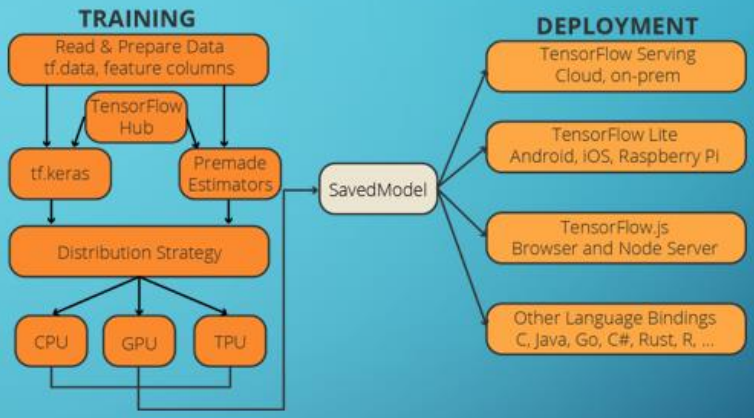

Good Matches & Object detection

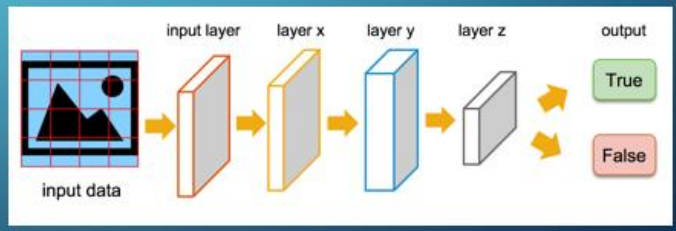
NumPy

NumPy

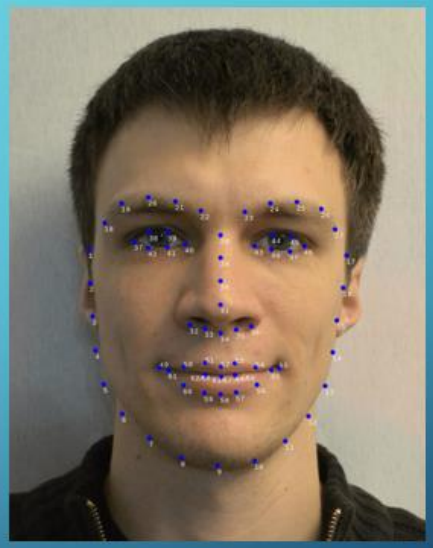

Uses of NumPy
Arithmetic operations
01
Statistical operations
02
Bitwise operators
03
Copying & viewing arrays
04
Stacking
05
Matrix Operations
06
Linear algebra
07
Broadcasting
08
Mathematical operations
09
Searching, sorting & counting
10

TensorFlow

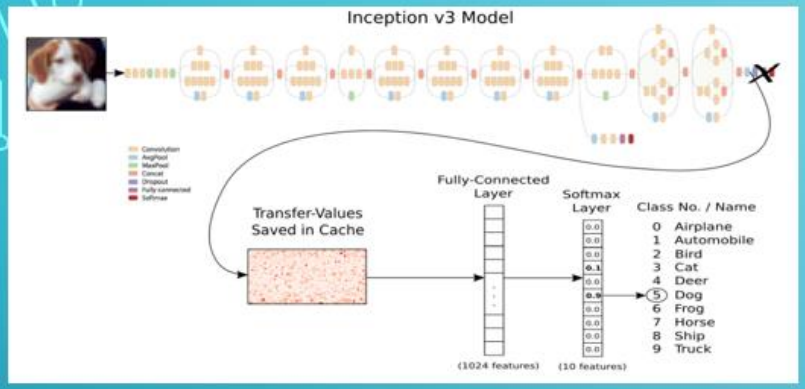

TRAINING
Read & Prepare Data
tf.data, feature columns
TensorFlow Hub
tf.keras
Premade Estimators
Distribution Strategy
CPU
GPU
TPU
SavedModel
DEPLOYMENT
TensorFlow Serving
Cloud, on-prem
TensorFlow Lite
Android, iOS, Raspberry PI
TensorFlow.js
Browser and Node Server
Other Language Bindings
C, Java, Go, C#, Rust, R, ...


input layer
layer x
layer y
layer z
output
True
False
input data

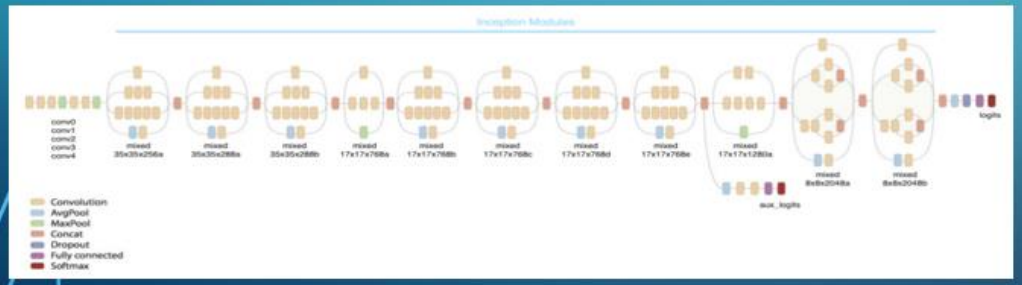
Dlib



Interception V3




```python
import tensorflow as tf , sys    Import "tensorflow" could not be resolved
image_path = sys.argv [1]
image_data = tf.gfile.FastGFile(image_path,'rb').read()
label_lines = [line.rstrip() for line
    in tf.gfile.GFile( "D:/tensorflow_work/retained_labels.txt")]
with tf.gfile.FastGFile("D:/tensorflow_work/retrained_graph.pb",'rb') as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f. read ())
        _ = tf.import_graph_def(graph_def , name = '')
with tf.Session() as sess:
    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')
    predictions = sess.run(softmax_tensor , {'DecodeJpeg/contents:0': image_data })
    top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]
    for node_id in top_k:          (variable) node_id: Any
        human_string = label_lines[node_id]
        score = predictions[0][node_id]
        print('\%s(score=\%.5f)', (human_string , score ))
```




```
golden retriever (score = 0.99933)
husky (score = 0.00067)
```

Виявлення та розпізнавання облич

Алгоритм 1. Псевдокод алгоритму виявлення та розпізнавання осіб

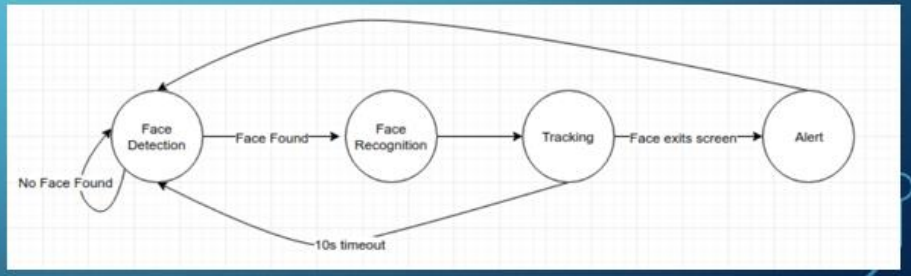
```
Input: Video object of webcam, known face encodings
  Import libraries
  while True do
    Grab a frame from webcam
    Resize it to one-fourth for faster computation
    Find the face locations in the image
    unknown_encoding = Get the face encodings of that location
    Find closest match to unknown_encoding in the known_face_encodings
    Get name of the matched known_face_encoding
    Draw a bounding box over the face
    Write the name of the person on top of the box
    Display image
  end while
```

Алгоритм 2. Псевдокод виявлення, розпізнавання та відстеження облич

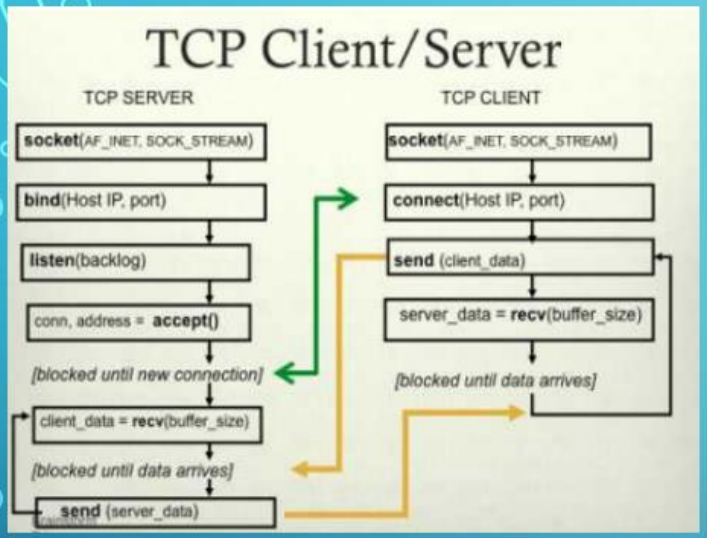
```
Input: Video_object_of_webcam, known_face_encodings
  Import libraries
  while True do
    Grab a frame from webcam
    Resize it to one-fourth for faster computation
    faces? ← Find if there are face locations in the image
    if faces? is True then
    unknown_encoding = Get the face encodings of that location
    Find closest match to unknown_encoding in the known_face_encodings
    name ← Get name of the matched known face encoding
    bounding_box ← bounding box of detected face
    break
  end if
end while
Using the bounding box track the face and keep displaying the name
if face exits screen then
  Rise alert
end if
```

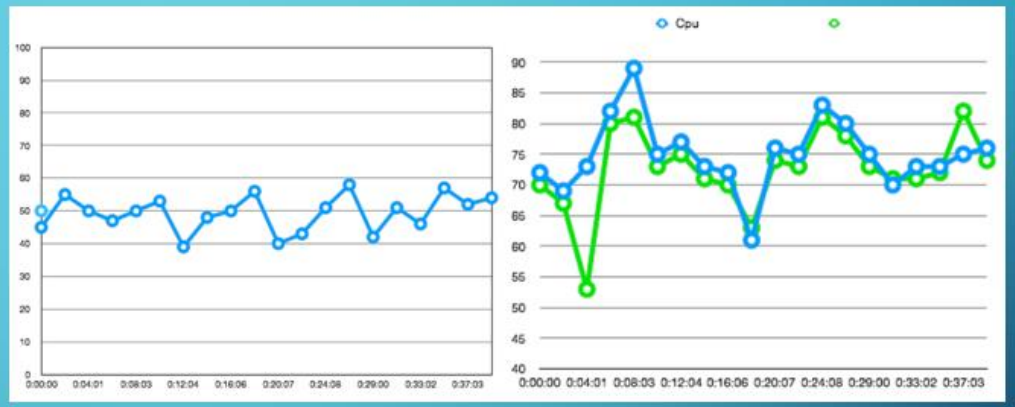
Реалізація

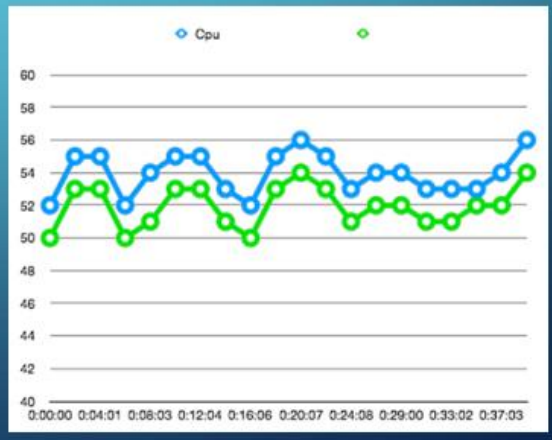
```
def run(source=0, disploc = False):
    cam = cv2.VideoCapture(source)
    if not cam.isOpened():
        print("Source not avaliable")
        exit()

    print('press key p to pause the video to start tracking')
    while True :
        retval , img = cam . read ()
        if not retval :
            print("Device unreachable")
            exit()
        if(cv2.waitKey(10)==ord('p')):
            break
        cv2.namedWindow("Image", cv2.WINDOW_NORMAL)
        cv2.imshow ("Image", img)
    cv2.destroyWindow ("Image")

    points = get_points.run(img)
    if not points:
        print('No obj to be tracked')
        exit()

    cv2.namedWindow("Image", cv2.WINDOW_NORMAL)
    cv2.imshow("Image", img)

    tracker = dlib.correlation_tracker()
    tracker.start_track(img, dlib.rectangle(* points [0]))

    while True :
        retval, img = cam.read()
        if not retval :
            print("Device unreachable")
            exit()
        tracker.update(img)
        rect = tracker.get_position()
        pt1 = (int(rect.left()), int(rect.top()))
        pt2 = (int(rect.right()), int(rect.bottom()))
        cv2.rectangle(img, pt1, pt2, (255 , 255 , 255) ,3)
        print ("Object tracked at [ {} , {} ] \ r " . format ( pt1 , pt2 ))
        if disploc :
            loc = (int(rect.left()), int(rect.top() -20))
            txt = "Object tracked at [ {} , {} ] " . format ( pt1 , pt2 )
            cv2.putText(img, txt, loc, cv2.FONT_HERSHEY_SIMPLEX, .5, (255 ,255 ,255) , 1)
        cv2.namedWindow("Image", cv2.WINDOW_NORMAL)
        cv2.imshow("Image", img)
        if cv2.waitKey(1) == 27:
            break
```



Реалізація з обробкою на зовнішньому сервері







Висновок

- В даный роботы було спроектовано інструмент, який використовує Raspberry Pi 3 B, який здатний виконувати різні операції комп'ютерного зору, такі як класифікація, відстеження руху, розпізнавання облич.
- Базова система обробляється операційною системою Raspbian та зовнішнім процесом, птимізуємызуючи продуктивність, обмінюючись даними між ними.
- Для завдання класифікації ми використовуємо добре відому модель Inception V3 та моделі MobileNet та виконуємо Transfer Learning з використанням інфраструктури Tensorflow. Цей метод дає хороші результати, хоча на Raspberry Pi він трохи повільний. Ми можемо використовувати зовнішній процесор разом з Pi, щоб покращити тут продуктивність.
- Для задач виявлення, розпізнавання та відстеження руху ми використовуємо бібліотеку OpenCV разом з бібліотеками dlib та розпізнавання облич та створюємо конвеєр робочого процесу, який автоматично виконує всі три операції бажаним чином.
- Виконується відстеження кількох об'єктів. Хоча це не було досліджено в цілому, багатообіцяюче з'ясувати, що він дійсно може бути ефективно виконаний як на Raspberry Pi, так і на сервері.
- Оптимізація з використанням зовнішнього процесора дуже перспективна, тому що затримок у відправленні майже немає. кадри зображення від Pi до сервера. Це основний метод оптимізації, який дає дуже добрі результати.

АПРОБАЦІЯ

- Опубліковано тезу «АРХІТЕКТУРА МЕРЕЖІ INTERNET OF NANO-THINGS (IoNT)» та посилання
- Опубліковано статтю «ПРОБЛЕМИ, ТРУДНОЩІ ТА МОЖЛИВОСТІ ІОТ ТА ХМАРНИХ ОБЧИСЛЕНЬ» у фаховому виданні МОН «ЗВ'ЯЗОК» № 4 (2021) (готується до друку) та посилання

Дякую за увагу!