ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до магістерської роботи
на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЯВЛЕННЯ ОБЛИЧЧЯ НА БАЗІ МІКРОКОНТРОЛЕРА ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ»**

Виконав: студент 6 курсу, групи ПДМ-61 спеціальності

121 Інженерії програмного забезпечення

(шифр і назва спеціальності)

Шефкін Б.В.

(прізвище та ініціали)

Керівник Сторчак К.П.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Нормоконтроль _____

(прізвище та ініціали)

КИЇВ – 2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій
Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Магістр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення Негоденко О.В.
"____" _____2020року

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

ШЕФКІНУ БОГДАНУ ВОЛОДИМИРОВИЧУ
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення для виявлення обличчя на базі мікроконтролера за допомогою штучного інтелекту

Керівник роботи: Сторчак Каміла Павлівна, завідувач кафедри ІСТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «13» жовтня 2020 року №230.
2. Строк подання студентом роботи 24.12.2020р.
3. Вихідні дані до роботи
Методи проектування та розробки нейронних мереж для виявлення обличчя;
Науково-технічна література з питань, пов'язаних з програмним забезпеченням щодо інтеграції нейронних мереж у мікроконтролери.

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
4.1 Аналіз систем та алгоритмів виявлення обличчя.
4.2 Аналіз технологій.
4.3 Програмна реалізація системи.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Аналіз аналогів
2. Технічне завдання
3. Спеціалізовані технології
4. Архітектура TensorFlow
5. Розрахунки точності в роботі NN
6. Тренування NN
7. Виявлення обличчя
8. Схема роботи системи
9. Практичне застосування
10. Наукова новизна та практична значимість, Висновки

6. Дата видачі завдання 02.11.2020р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	02.11-10.11	
2	Вимоги до системи	11.11-13.11	
3	Аналіз стану питання	14.11-16.11	
4	Аналіз нейронних мереж	17.11-20.11	
5	Конфігурація та результати моделювання	21.11-1.12	
6	Вступ, висновки, реферат	2.12-11.12	
7	Розробка презентації	12.12-15.12	

Студент _____

Керівник роботи _____

РЕФЕРАТ

Пояснювальна записка містить: сторінок – 78, рисунків – 45, джерел – 19.

Об'єкт дослідження – системи штучного інтелекту, нейронні мережі, мікроконтролери.

Предмет дослідження – виявлення обличчя.

Мета роботи – розробка система, яка вирішуватиме проблеми: контролю доступу до об’єкту, дому, споруди; малобюджетність, компактність, ефективність, мінімізація ціни для кінцевого користувача.

Методи дослідження – математичні методи моделювання та оптимізації, системне програмування, моделювання IoT, розрізнювальне моделювання, алгоритми машинного навчання.

Для досягнення поставленної мети були виконані наступні дії:

- Проаналізовано алгоритми нейронних мереж та машинного навчання.
- Проаналізовано мікроконтолери для реалізації кінцевої системи.
- Оптимізовано алгоритм нейронної мережі для використання в мікроконтролері.
- Інтегровано поштовий протокол SMTP для передачі зображень як один з методів безпечної передачі інформації.

Галузь використання – сучасні системи розумних будинків, системи відеоспостереження та фіксації, системи захисту приватних територій.

НЕЙРОННА МЕРЕЖА, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, TENSORFLOW, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МІКРОКОНТРОЛЕР, ВИЯВЛЕННЯ ОБЛИЧЧЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI	Artificial Intelligence	Штучний інтелект
ANN	Artificial Neural Network	Штучна нейронна мережа
API	Application Programming Interface	Прикладний програмний інтерфейс
CNN	Convolutional neural network	Згорткова нейронна мережа
MCU	Microcontroller Unit	Мікроконтролер
PCA	Principal Component Analysis	Метод головних компонент
RCNN	Region Based Convolutional Neural Network	Регіональна згорткова нейронна мережа
TF	TensorFlow	Відкрита бібліотека для машинного навчання

ВСТУП

Актуальність роботи. В останній час широке розповсюдження отримує відеоаналітика – технологія, яка використовує комп'ютерний зір для автоматизованого збору інформації обличч людей у послідовності кадрів, які отримуються з відеокамер в реальному часі чи з відеозаписів. Дана технологія може бути застосована в відеоспостереженні, системах безпеки, транспорті, у домашніх розумних системах охорони. Сьогодні існує велика кількість різних рішень на ринку пристроїв охорони, контролю доступу. Але існуючі рішення на ринку систем відеонагляду мають лише базовий функціонал або крім камер потрібно комбінувати з системами аналітики на основі штучного інтелекту.

Метою даної магістерської роботи є розробка системи, яка вирішуватиме відразу декілька задач, а саме: контроль доступу до об’єкту, дому, споруди; малобюджетність та компактність кінцевого продукту; мінімізації ціни для кінцевого користувача.

Розроблений продукт на базі мікроконтролера дозволить виявляти людину у відео потоці, інтегрувати його до елементів розумного будинку для спостереження у домашніх умовах або в інших б’єктах.

Об'єкт дослідження – системи штучного інтелекту, нейронні мережі, мікроконтролери.

Предмет дослідження – виявлення обличчя.

Методи дослідження – математичні методи моделювання та оптимізації, системне програмування, моделювання IoT, розрізнювальне моделювання, алгоритми машинного навчання.

Наукова новизна дослідження полягає в оптимізації бібліотеки TensorFlow для розробки нейронної мережі для виявлення обличчя людини та інтегрування даної системи в малобюджетний та малопотужний мікроконтролер.

Практична значимість дослідження полягає в створенні продукту на базі мікроконтолера, який дозволить виявляти людське обличчя, будучи при цьому ефективним, компактним, легким в інтегруванні в системи охорони та малобюджетним рішенням.

Одна з причин підвищеної уваги до біометричних технологій являється існування великої кількості соціальних і комерційних додатків, де можливі рішення названої проблеми будуть сприйняті досить успішно. Дане рішення дасть змогу отримувати зображення на поштову скриньку з відеокамер зовнішнього спостереження за доступною ціною. Кожен зможе встановити в себе біля будинку або вічка двері, або в інших місцях.

1 АНАЛІЗ СИСТЕМ ТА АЛГОРИТМІВ ВИЯВЛЕННЯ ОБЛИЧЧЯ

1.1 Використання нейронних мереж для розпізнавання обличчя

Нейронна мережа в найпростішому випадку – математична модель, що складається з декількох шарів елементів, що виконують паралельні обчислення. Спочатку така архітектура була створена за аналогією з дрібними обчислювальними елементами людського мозку – нейронами. Мінімальні обчислювальні елементи штучної нейронної мережі теж називаються нейронами. Нейронні мережі, зазвичай, складаються з трьох або більше шарів: вхідного шару, прихованого шару (або шарів) і вихідного шару (рисунок 1.1), у деяких випадках вхідний і вихідний шари не враховуються, і тоді кількість шарів в мережі вважається за кількістю прихованих шарів. Такий тип нейронної мережі називається перцептрон.

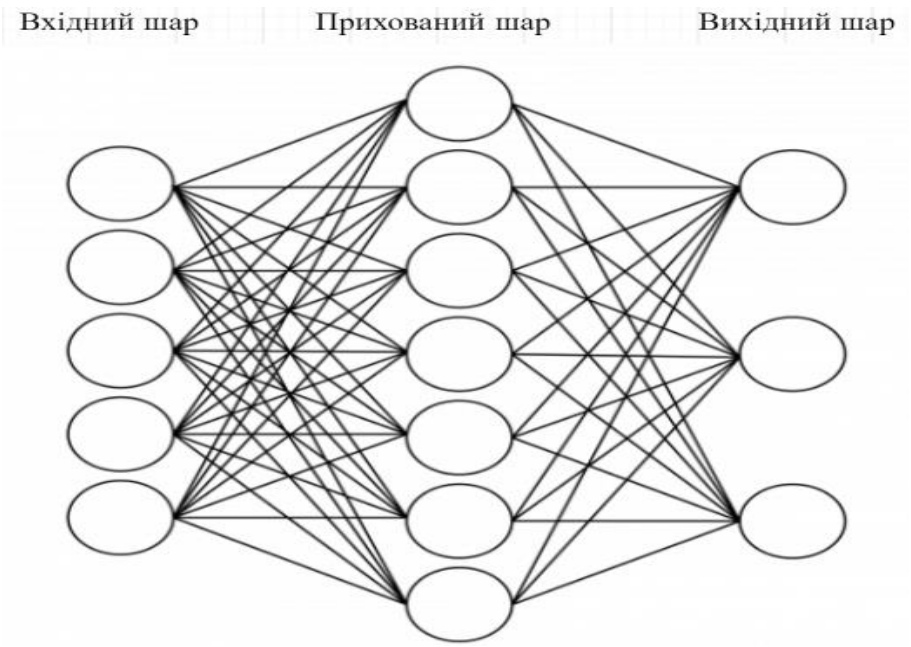

Рисунок 1.1 – Нейронна мережа

Важливою особливістю нейронної мережі є її вміння навчатися на прикладах, це називається навчанням з учителем. Нейронна мережа навчається на великій кількості прикладів, що складаються з пар вхід-вихід (відповідні один одному вхід і вихід). В задачах розпізнавання об'єктів такою парою буде вхідне зображення і відповідний йому лейбл – назва об'єкта. Навчання нейронної мережі – ітеративний процес, що зменшує відхилення виходу мережі від заданого «відповіді вчителя» – лейбла, яке відповідає даному зображенню (рисунок 1.2). Цей процес складається з кроків, які називаються епохами навчання (вони зазвичай обчислюються тисячами), на кожному з яких відбувається підгонка «ваг» нейронної мережі – параметрів прихованих шарів мережі. По завершенні процесу навчання якість роботи нейронної мережі зазвичай досить гарне для виконання завдання, під яку вона була навчена, хоча оптимальний набір параметрів, ідеально розпізнає всі зображення, часто підібрати неможливо.

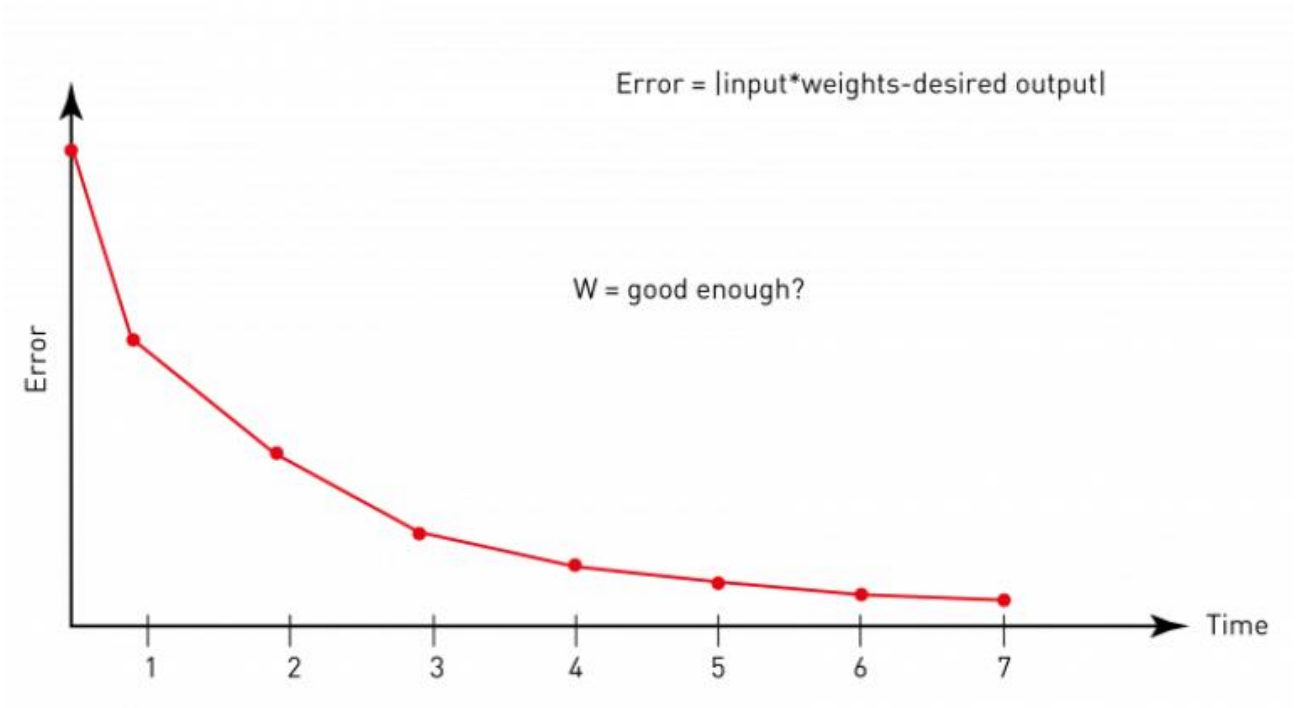

Рисунок 1.2 – Навчання нейронної мережі.

Глибокі, або глибинні, нейронні мережі - це нейронні мережі, що складаються з декількох прихованих шарів (рисунок 1.3). Даний рисунок являє собою зображення глибинної нейронної мережі, що дає читачеві загальне уявлення про те, як виглядає нейронна мережа. Проте, реальна архітектура глибинних нейронних мереж набагато складніше.

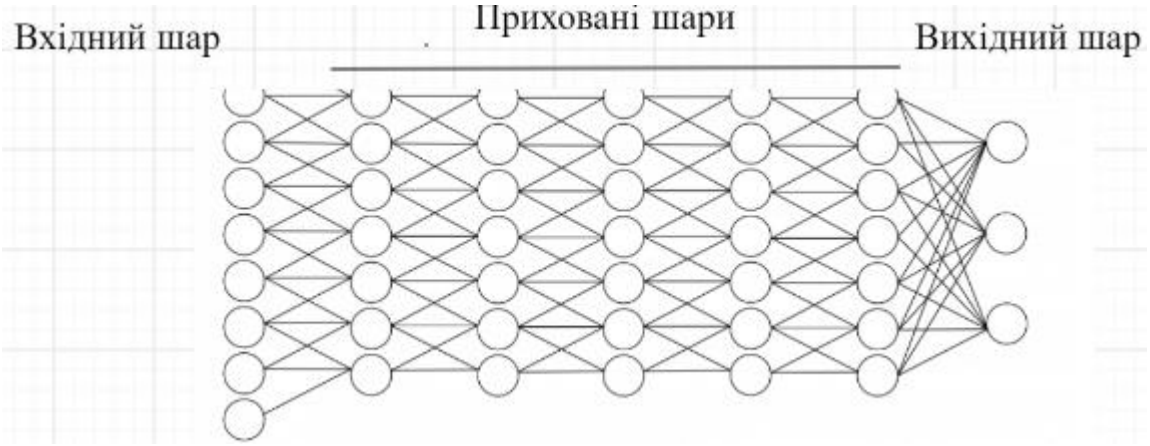

Рисунок 1.3 – Нейронна мережа с декількома прихованими шарами.

Творці згортаємих нейронних мереж, звичайно, спочатку надихнулися біологічними структурами зорової системи. Перші обчислювальні моделі, засновані на концепції ієрархічної організації візуального потоку примата, відомі як Неокогнітрон Фукушіма (рисунок 1.4). Сучасне розуміння фізіології зорової системи схоже з типом обробки інформації в свёрточних мережах, по крайней мере, для швидкого розпізнавання об'єктів.

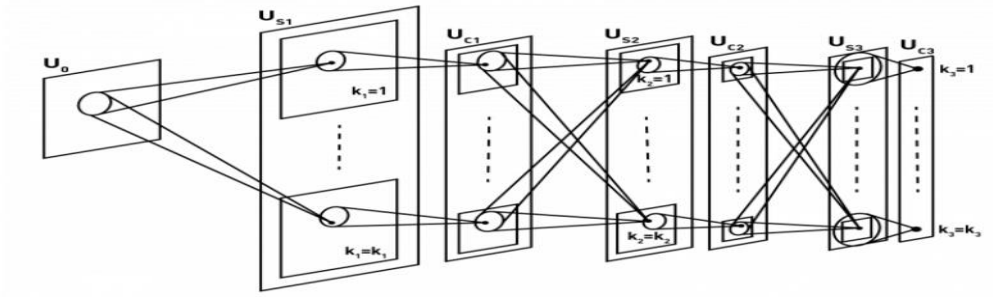

Рисунок 1.4 – Діаграма зв’язку між шарами в моделі Неокогнітрон.

Пізніше ця концепція була реалізована канадським дослідником Яном ЛеКуном в його згортаємої нейронної мережі, створеної ним для розпізнавання рукописних символів. Дана нейронна мережа складалася з шарів двох типів: свёрточних шарів і субдіскретізірующіх (subsampling) шарів (або шарів підвибірки-pooling). У ній кожен шар має топографічну структуру, тобто кожен нейрон пов'язаний з фіксованою точкою вихідного зображення, а також з рецептивних полем (областю вхідного зображення, яка обробляється даними нейроном). У кожному місці кожного шару існує цілий ряд різних нейронів, кожен зі своїм набором вхідних ваг, пов'язаних з нейронами в прямокутному фрагменті попереднього шару. Різні вхідні прямокутні фрагменти з однаковим набором ваг пов'язані з нейронами з різних локацій.

Загальна архітектура глибинної нейронної мережі для розпізнавання образів представлена на рисунку 1.5. Вхідний зображення представлено у вигляді набору пікселів або невеликих ділянок зображення (наприклад, 5-на-5 пікселів)

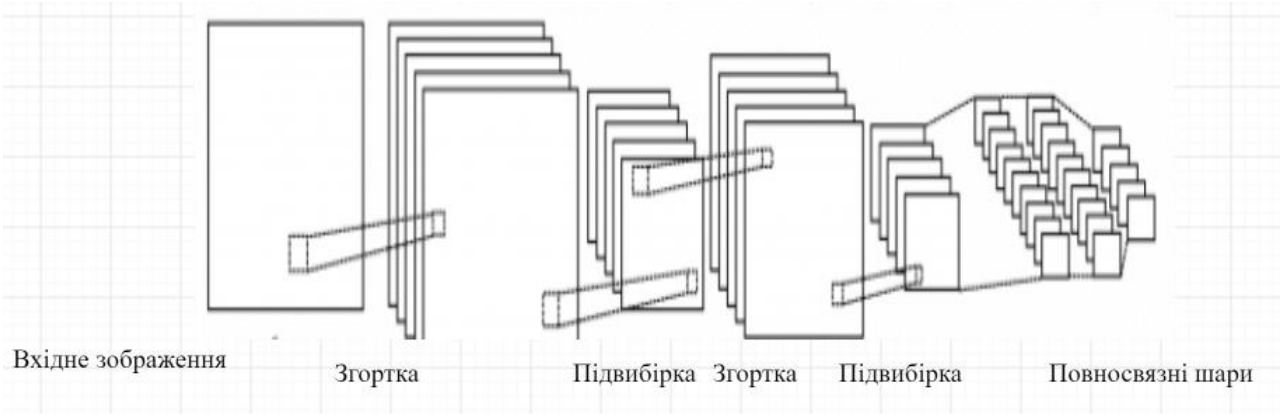

Рисунок 1.5 – Діаграма згортаємої нейронної мережі.

Як правило, глибинні нейронні мережі зображені в спрощеному вигляді: як стадії обробки, які іноді називають фільтрами. Кожна стадія відрізняється від іншої

рядом характеристик, таких як розмір рецептивного поля, тип ознак, який мережу вчить розпізнавати в даному шарі, і тип обчислень, виконуваних на кожній стадії. Сфери застосування глибинних нейронних мереж, в тому числі згортаємих мереж, не обмежуються розпізнаванням осіб. Вони широко використовуються для розпізнавання мови і аудіо-сигналів, обробки показань з різного типу сенсорів або для сегментації складних багатошарових зображень (таких як супутникові карти) або медичних зображень.

Для досягнення високої точності розпізнавання нейронна мережа предобучается на великому масиві зображень, наприклад, такому, як в базі даних MegaFace. Це основний метод навчання для розпізнавання осіб.

Після того, як мережа навчена розпізнавати обличчя, процес розпізнавання особи може бути описаний таким чином (рисунок 1.6). Спочатку зображення обробляється за допомогою детектора особи: алгоритму, який визначає прямокутний фрагмент зображення з обличчям. Цей фрагмент нормалізується для того, щоб легше оброблятися нейронною мережею: найкращий результат буде досягнутий, якщо все вхідні зображення будуть однакового розміру, кольору і т.д. Нормалізоване зображення подається на вхід нейронної мережі для обробки алгоритмом. Даний алгоритм зазвичай є унікальною розробкою компанії для підвищення якості розпізнавання, однак існують і «стандартні» рішення для даного завдання. Нейронна мережа будує унікальний вектор ознак, який потім переноситься в базу даних. Пошукова система порівнює його з усіма векторами ознак, що зберігаються в базі даних і дає результат пошуку у вигляді певного числа імен або профілів користувачів зі схожими лицьовими ознаками, кожному з яких присвоюється певна кількість. Це число є мірою схожості нашого вектора ознак з знайденим в базі даних.

Коли ми вибираємо, який алгоритм застосувати до задачі розпізнавання об'єкта або особи, ми повинні мати засіб порівняння ефективності різних алгоритмів. Оцінка якості роботи системи розпізнавання осіб проводиться за допомогою набору метрик,

які відповідають типовим сценарієм використання системи для аутентифікації за допомогою біометрії.

Як правило, робота будь-якої нейронної мережі може бути виміряна з точки зору точності: після настройки параметрів і завершення процесу навчання мережу перевіряється на тестовому безлічі, для якого ми маємо відгук вчителя, але який відділений від навчального набору. Як правило, цей параметр є кількісною мірою: число (часто у відсотках), яке показує, наскільки добре система здатна розпізнавати нові об'єкти. Ще одна типова міра - це помилка (може бути виражена як у відсотках, так і в числовому еквіваленті). Проте, для біометрії існують більш точні заходи.

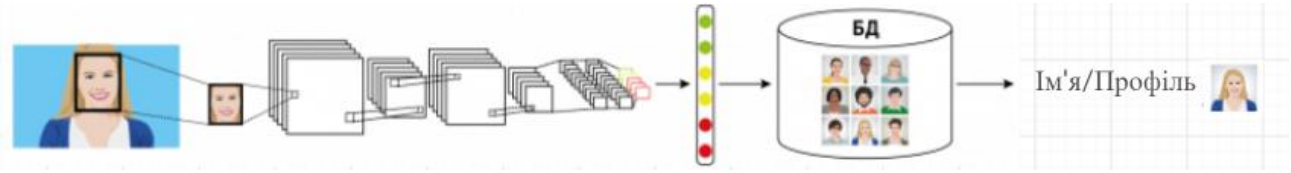

Рисунок 1.6 – Процес розпізнавання обличчя.

У біометрії взагалі і біометрії для розпізнавання осіб, зокрема, існує два типи програм: верифікація та ідентифікація. Верифікація являє собою процес підтвердження певної особистості шляхом порівняння зображення індивіда (вектора ознак особи або іншого вектора ознак, наприклад, сітківки або відбитків пальців) з одним або декількома раніше збереженими шаблонами. Ідентифікація – це процес визначення особистості індивіда. Біометричні зразки збирають і порівнюють з усіма шаблонами в базі даних. Існує ідентифікація в замкнутому безлічі ознак, якщо передбачається, що людина існує в базі даних. Таким чином, розпізнавання об'єднує один або обидва терміни - верифікацію та ідентифікацію.

Часто крім безпосередньо результату порівняння, потрібно оцінити рівень «впевненості» системи в своєму рішенні. Таке значення називають терміном «рівень подібності» (або подібності, similarity score). Більший показник подібності вказує на те, що два порівнюваних біометричних зразка більш схожі.

1.2 Аналіз цілей використання систем розпізнавання обличчя

Прихильники називають розпізнавання обличчя "найбільш природним з усіх біометричних вимірювань" і кажуть, що ця технологія є надзвичайно точною, з "нульово малими різницями в показниках хибнопозитивних чи помилково негативних показань у демографічних групах". Критики, навпаки, стверджують, що це "найбільша помилка техніки": вона прислухається до дистопічного світу Чорного дзеркала і дає занадто багато помилкових спрацьовувань - невинних людей, яких визнають винними. Отже, хто правий?

По суті, розпізнавання обличчя передбачає ідентифікацію або перевірку когось за допомогою характеристик та рис його обличчя. Існують різні технології, включаючи тривимірну, судинну та теплову схему та аналіз текстури шкіри. У найбільш поширеному типі розпізнавання обличчя алгоритми визначають певні точки на обличчі, наприклад, форму підборіддя, і створюють шаблон для цієї людини. Найточніше, коли користувачі добровільно подають свої зображення в обмежену базу даних, наприклад, в мережу компанії, для використання, наприклад, у контролі доступу за допомогою обличчя. Коли людина наближається до сканера обличчя, її зображення в реальному часі фіксується та перетворюється у шаблон, який потім порівнюється із шаблонами, що зберігаються у базі даних. Збіг при скануванні дозволяє користувачеві здійснити певну діяльність, наприклад, пройти через двері або увійти в комп'ютерну мережу.

Які переваги систем розпізнавання обличчя? По-перше, безпека. Технологію розпізнавання обличчя використовували незліченну кількість разів для ідентифікації, пошуку та захоплення підозрюваних у кримінальних злочинах. У 2019 році поліція Нью-Йорка використовувала розпізнавання обличчя, щоб відшукати імовірного ґвалтівника протягом 24 годин. Система також використовується для оформлення документів, що посвідчують особу, під час прикордонних перевірок, для виявлення зниклих дітей, пошуку жертв торгівлі людьми тощо. Однак, його найбільш

швидкозростаюче застосування – це частина системи доступу до дверей для розпізнавання обличчя.

Зручність. При використанні у фізичних або логічних системах контролю доступу, розпізнавання обличчя забезпечує пасивне, безперешкодне та комфортне рішення для доступу. Користувачам не потрібно турбуватися про те, щоб не забути свій значок або комбінацію символів. Їм потрібно лише на мить зупинитись біля камери, а не зупинятися і вставляти руку чи палець у пристрій. Це абсолютно безконтактна система, коли двері налаштовані на автоматичне відкривання.

Обслуговування клієнтів. Роздрібна торгівля, театри, стадіони та інші заклади використовують розпізнавання обличчя, щоб пристосувати відвідувача чи клієнта та задовольнити VIP-персон, таких як надання послуг консьєржа. З його допомогою можна створювати розумні цифрові вивіски та спрощувати процес оформлення замовлення, як це відбувається в магазинах Amazon GO, наприклад.

Досягнення в охороні здоров'я. Дослідники Національного дослідницького інституту геному людини використовували програмне забезпечення для розпізнавання обличчя для точного діагностування синдрому ДіГеорджа, хромосомного розладу, що викликає багато проблем зі здоров'ям. Дослідники порівняли 156 латиноамериканців, африканців, кавказців та азіатів із синдромом Ді-Джорджа проти інших без хвороби. Розпізнавання обличчя дало майже 97% показника правильного діагнозу. Інші програми охорони здоров'я включають програму, яка гарантує, що пацієнти приймають ліки відповідно до приписів. І організації використовують його у битві проти коронавірусу. Наприклад, компанія CLEAR, яка допомагає пришвидшити подорожі через безпеку в аеропортах США, випускає інструмент, який використовує розпізнавання обличчя, щоб визначити, чи є у співробітника Covid і чи повинен він залишатися вдома.

Проблеми, заперечення та недоліки щодо розпізнавання обличчя. Відповідні помилки. Точність систем розпізнавання обличчя зменшується, коли всесвіт користувачів необмежений. Розглянемо роздрібну мережу, яка використовує

розпізнавання обличчя, щоб зіставити покупців із базою даних відомих магазинів. По мірі того, як все більше людей заходить у їхні магазини, стає більш імовірним, що хтось буде схожий на відомого торговця крамницями і буде неправильно підібраний. Помилки можуть також виникати через погане зображення, зображення, зроблені під поганим кутом, та погане освітлення.

Проблеми конфіденційності. Розпізнавання обличчя втручається у вашу приватність? Регулярно трапляються масові порушення даних, і перспектива вкрасти зображення обличчя викликає занепокоєння. Наприклад, минулого року митна служба та прикордонна служба США повідомили, що хакери викрали майже 100 000 фотографій мандрівників. Крім того, технологія розпізнавання обличчя може бути використана для відстеження людей, і Американський союз громадянських свобод застерігає від його використання для "загальних, без підозр систем спостереження".

Зловживання технологією розпізнавання обличчя. Багато громадян побоюються зловживання цією технологією. В опитуванні Pew 2019 року половина дорослих американців заявила, що не довіряє технологічним компаніям відповідально використовувати дані розпізнавання обличчя. Майже 30 відсотків мали подібні сумніви щодо правоохоронних органів. І дві третини не довіряли цій технології в руках рекламодавців. Крім того, Panda Security зазначає, що багато баз даних розпізнавання обличчя є загальнодоступними. "Це означає, що будь-яка особа, навіть із зловмисними намірами, може знайти вас у базі даних і відстежити".

Як працює технологія розпізнавання облич? Сучасне розпізнавання обличчя розроблено в галузі машинного навчання та комп'ютерного зору. З 2015 року, завдяки відродженню глибокого навчання, комп’ютери змогли витягувати інформацію із зображень із надлюдською точністю. Технологія розпізнавання обличчя використовує всю RGB (червону, зелену, синю кольорову модель) фотографію людини та запускає її за допомогою алгоритму глибокого навчання (AI), який обробляє зображення за допомогою мільйонів обчислень із плаваючою точкою, щоб сформувати унікальний підпис обличчя . Цей унікальний підпис

використовується для порівняння фотографії користувача в майбутньому та перевірки наявних підписів.

Чи безпечне розпізнавання обличчя? Фізично розпізнавання обличчя є абсолютно безпечним. Сканування не завдає шкоди обличчю, і воно не є нав'язливим. Як звичайні, так і інфрачервоні камери використовують частоти світла, які існують у сонячному світлі, а інфрачервоні камери за своєю силою схожі на телевізійний пульт дистанційного керування. Він також є безконтактним, що є ключовим фактором під час пандемії коронавірусу.

Чи можна системи розпізнавання обличчя обдурити фотографіями чи відео? Погано впроваджену систему розпізнавання обличчя можна обдурити за допомогою методів підробки, тримаючи фотографію чи відео людини або одягаючи тривимірну маску, що становить ризик для безпеки. Важливо використовувати системи, що реалізують заходи проти підробки.

Як системи розпізнавання обличчя запобігають підробці? Будь-які якісні системи розпізнавання облич повинні застосовувати заходи проти підробки, щоб запобігти неправильному використанню та несанкціонованому доступу до системи. Такі заходи включають виявлення обличчя "живого" (наприклад, за допомогою швидкості мигання), аналіз текстури та вимоги до руху користувача (виклик / відповідь).

Чи працює розпізнавання обличчя з масками для обличчя? Системи розпізнавання обличчя використовують всю геометрію обличчя та створюють унікальний підпис обличчя. Маски запобігають високій точності систем розпізнавання обличчя, обмежуючи аналіз обличчя лише відкритими ділянками обличчя. Важливо, щоб системи розпізнавання облич надавали доступ лише тоді, коли люди знімають маску, інакше можуть виникнути проблеми з точністю та авторизованим доступом. Поки компанії люто працюють над технологіями, які можуть забезпечити надійні результати для людей, які носять маски, ці системи ще не настільки ефективні.

Чи працює розпізнавання обличчя в окулярах та сонцезахисних окулярах? А як щодо капелюшків та пов’язок? Розпізнавання обличчя працює з високою точністю з більшістю окулярів, сонцезахисних окулярів, капелюхів та пов’язок. Однак, якщо людина носить надзвичайно великі сонцезахисні окуляри, щільно розтягнуту шапочку або поєднання шапки, окулярів та / або пов'язки на голову, система може їх не розпізнати.

Чи упереджене розпізнавання обличчя на основі раси та етнічної приналежності? Системи розпізнавання облич потрапили під пильну перевірку з метою зниження точності кольорових людей та різних етнічних груп. Використання розпізнавання обличчя для розслідування злочинів є проблематичним, оскільки деякі правоохоронні системи розпізнавання осіб невірно класифікують темношкірих як злочинців. Системи, побудовані з широкого спектру предметів різної раси, етнічної приналежності та зовнішності, не страждають від проблем з меншою точністю.

Як уникнути упередженості в системах розпізнавання облич? Упередження точності розпізнавання обличчя можна усунути, використовуючи такі найкращі практики:

- Хороші дані тренувань: Важливо тренувати розпізнавання облич, використовуючи різні обличчя за різними статями, віковими групами, етнічними групами та кольорами шкіри.
- Додайте тривимірне розпізнавання: компанії повинні додати тривимірне розпізнавання облич, використовуючи тривимірні камери як другу функцію розпізнавання.
- Зображення з високою роздільною здатністю: системи розпізнавання облич повинні використовувати зареєстровані зображення з високою роздільною здатністю. Камери відеоспостереження, які претендують на розпізнавання обличчя, часто страждають від низької точності, оскільки вони використовують камери з низьким роздільною здатністю і високим полем зору, встановлені далеко на стелі або в далекому кутку.

Чому варто розглянути можливість використання розпізнавання обличчя для контролю доступу? Багато компаній, що займаються контролем доступу, покладаються на картки ключів, які можна загубити чи забути - і ними часто діліться, що викликає занепокоєння під час пандемії. Розпізнавання облич використовується мільярдами людей завдяки високій точності, що забезпечується досягненнями в галузі комп'ютерного зору. В епоху Ковіда безконтактні системи входу в двері важливі як ніколи. Використовуючи системи контролю доступу до розпізнавання обличчя, організації тепер можуть впроваджувати безконтактний, безконтактний доступ до будівель, на заходах, у школах та на багатьох інших майданчиках. Ця технологія забезпечує вищий рівень безпеки та доступ до тертя для кінцевих користувачів.

Чи можна розпізнавання обличчя використовувати для вимірювання температури? Щонайменше 15 компаній пропонують термінали – або кіоски – де технологія розпізнавання обличчя вимірює температуру тіла людини. Деякі системи також можуть використовуватися для контролю доступу до об'єкта чи району, підтвердження того, що людина носить маску для обличчя, проведення масового скринінгу, застосування соціального дистанціювання, пропонування цифрових повідомлень або відстеження двонаправлених рухів.

Підсумок. Зрештою, розпізнавання обличчя швидко вдосконалюється і пропонує широкий і розширюючий спектр переваг. З розумним використанням, з належним контролем, який обмежує масштабне використання правоохоронними органами та корпораціями, розпізнавання обличчя є ключовим елементом у безпеці, безпеці, охороні здоров’я та багатьох інших сферах.

1.3 Порівняльний аналіз систем

Технологія не перестає дивувати нас. Розпізнавання зразків та обробка зображень – це одна з галузей, яка сьогодні широко обговорюється та досліджується.

Еволюція рішень з біометричної автентифікації запропонувала протиотруту незручним, незграбним та легкозабутим паролям. Це допомогло підприємствам, школам, будинкам, офісам та житловим кварталам захистити свої приміщення за допомогою правильних технологій безпеки. Розпізнавання облич – одне з найвизначніших біометричних рішень, яке передбачає ідентифікацію людських облич на цифрових зображеннях та відео. Після виявлення обличчя ШІ надає інформацію про його розмір, позу та місце розташування. Все це підкріплено передовими програмними засобами розпізнавання облич та полегшено відповідно до алгоритмів технології розпізнавання обличчя (FRT).

Концепція вступила в контекст у 2001 році, коли Пол Віола та Майкл Джонс представили систему виявлення об'єктів. Структура, спрямована на виявлення об’єктів у режимі реального часу та проблема виявлення обличчя, в першу чергу мотивована результатом необхідного рішення. Сьогодні ринок програмного забезпечення для розпізнавання обличчя, ймовірно, досягне 11,30 мільярда доларів до 2026 року.

Що таке методи виявлення обличчя? Експерти класифікували метод виявлення обличчя на чотири категорії.

На основі знань: Людські знання для виявлення обличчя, тобто обличчя повинно мати ніс, очі та рот на певній відстані та в положенні одне з одним.

На основі функцій: Цей метод допомагає знаходити обличчя, витягуючи структурні риси обличчя. Ідея цього полягає в тому, щоб подолати межі інстинктивного знання облич, і він має 94% успіху.

Збіг шаблонів: Цей метод використовує параметризовані або заздалегідь визначені шаблони обличчя для виявлення обличчя, встановлюючи кореляцію між вхідними зображеннями та шаблонами.

На основі зовнішнього вигляду: Цей метод залежить від набору моделей облич, а також застосовується для вилучення об’єктів для розпізнавання обличчя. Зображено рисунок 1.7 категорій виявлення обличчя.

Як запустити програмне забезпечення для виявлення обличчя в режимі реального часу (веб-камера)? Якщо ви закінчите з усіма необхідними системними налаштуваннями, вам доведеться вибрати відповідне програмне забезпечення, яке здається ідеальним для ваших бізнес-потреб.

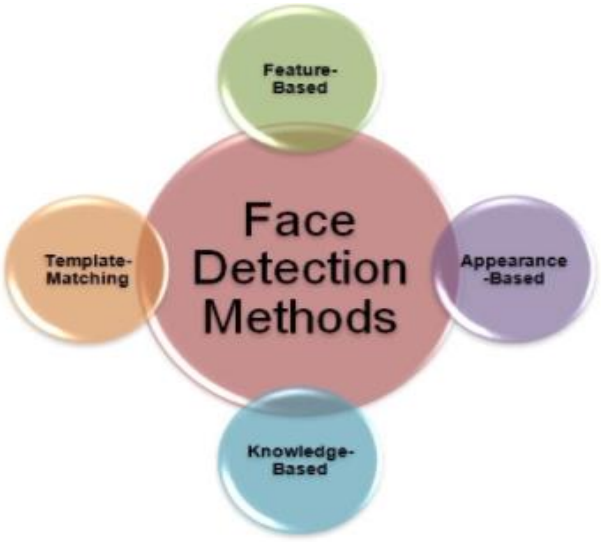

Рисунок 1.7 – схема категорій виявлення обличчя.

Якщо ви великий діловий дім, ви можете планувати скористатися усіма розширеними функціями, доступними у власних програмах. Однак, якщо у вас обмежений бюджет, завжди краще вибрати безкоштовні програмні рішення для розпізнавання обличчя з відкритим кодом. Ось як ви можете запускати програмне забезпечення для виявлення обличчя в режимі реального часу:

- Імпорт бібліотек.
- Імпортний класифікатор для виявлення обличчя та очей.
- Перетворити зображення у відтінки сірого.
- Дайте координати, щоб визначити місце розташування обличчя та очей за допомогою рентабельності інвестицій.
- Налаштування веб-камери для виявлення обличчя.
- Коли все зроблено, відпустіть знімок.

Список безкоштовного програмного забезпечення для виявлення обличчя з відкритим кодом:

List of Face Detection Software Solutions	Deployment	Device Supported	Unique Feature
OpenBR	Open-API, On-Premise	Windows, Mac, Linux	A complete NIST compliant software that evaluates facial recognition, detection, and land-marking.
Flandmark	Open-API	Windows, Linux	A structured output classifier that relies on Deformable Part Models (DPM).
OpenFaceTracker	Open-API	Windows	It is developed as a modular library. Hence, it can either enable or disable some part of the software.
OpenEBTS	Open-API	Windows, Web-Based	It includes two open-source APIs: OpenEBTS API and OpenM1 API.
Bioenable Tech - iFace	Open-API, On-Premise	Web-Based	It is a multi-biometric identification program integrated with a 630MHz high-speed multi bio processor.
Bioenable Tech - vFace	Open-API, On-Premise	Web-Based	vFace system is made available in elegant ergonomic design and requires 3inch TFT touch screen, nine digits user ID, and T9 input.
FacePlusPlus	Cloud-Hosted	Windows, Mac, Web-Based	Integrated C+E solutions, excellent accuracy, robust anti-spoofing technique, and frequent model updating.
DeepFace	Cloud-Hosted	Windows, Mac, Web-Based	It facilitates creating a database from faces and search based-on a given image or name.

Рисунок 1.8 – Список безкоштовного ПЗ для виявлення обличчя.

OpenBR – це провідна система виявлення обличчя та біометричного розпізнавання, яка підтримує розробку відкритих алгоритмів та відтворюваних оцінок. Стабільна версія 1.1.0 програмного забезпечення випущена 29 вересня 2019 року. Вона працює в операційних системах на базі Windows, Linux, OS X та Raspbian. Проект ліцензований згідно Apache 2.0.

Система OpenBR – основа для дослідження нових методів, вдосконалення існуючих алгоритмів, взаємодії з комерційними системами, вимірювання продуктивності розпізнавання та розгортання автоматизованих біометричних систем. Проект розроблений для полегшення швидкого прототипування алгоритмів і має зрілий основний фреймворк, гнучку систему плагінів та підтримку розробки з

відкритим та закритим кодом. Готові алгоритми також доступні для конкретних способів, включаючи розпізнавання обличчя, оцінку віку та оцінку статі:

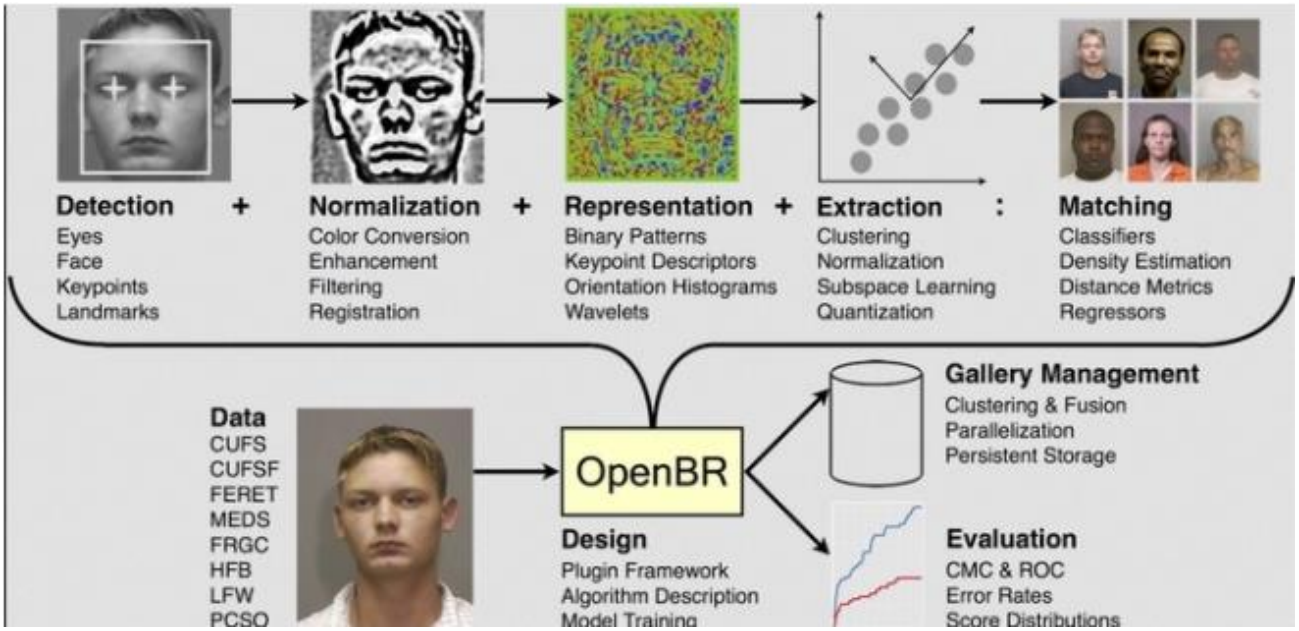

Рисунок 1.9 – Архітектура роботи системи OpenBR.

- OpenBR надає API C ++, який можна вбудувати у власні програми.
- Це повне програмне забезпечення, сумісне з NIST, яке оцінює розпізнавання обличчя, виявлення та нанесення мітки.
- Він реалізує алгоритм 4SF2 для розпізнавання обличчя.
- Алгоритми програм також працюють для оцінки віку та статі.

Flandmark – це бібліотека з відкритим кодом, яка реалізує виявлення орієнтирів обличчя на статичних зображеннях. Eydea Recognition Ltd – компанія, яка надає Flandmark детектор обличчя. Програмне забезпечення працює на Windows, Linux та Mac OS і ліцензується відповідно до версії 3. GNU / GPL. Початкова версія 1.02 програмного забезпечення вийшла 10 серпня 2011 року:

- Flandmark (версія 1.06) також може використовуватися мовою Python.
- В основному, це написано на C, C ++ та MATLAB.

- Кожен кадр обробляється окремо програмним забезпеченням для орієнтиру обличчя.
- Він використовує структурований класифікатор вихідних даних, який спирається на моделі деформованих деталей (DPM).

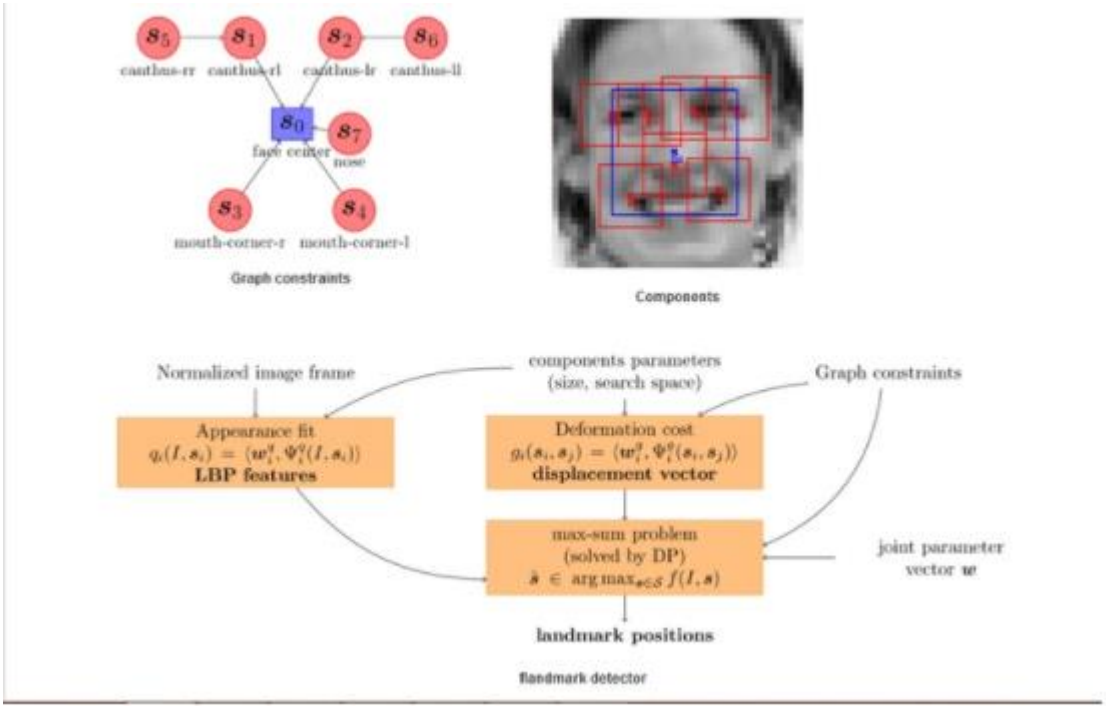

Рисунок 1.10 – Архітектура роботи системи Flandmark.

Як одна з провідних програм розпізнавання облич - OpenFaceTracker виявляє одне або кілька облич на відео чи зображенні та ідентифікує їх через базу даних. Це програмне забезпечення для виявлення обличчя з відкритим кодом, яке постачається з ліцензією lGPLv3 та стабільною версією 3.0. Винахід програмного забезпечення натхненний американським телешоу "Person of Interest":

- Отримання та обробка зображень у режимі реального часу з подальшою ідентифікацією, запасом та доступними даними для друку.
- Для розгортання потрібен Open-API і може працювати в системі на базі Windows.

- Програмне забезпечення потребує встановлення OpenCV3.2 та QT4 на вашому комп’ютері перед установкою програмного забезпечення OpenFaceTracker.
- Програмне забезпечення OpenFaceTracker розроблено як модульна бібліотека. Отже, він може або вмикати, або вимикати якусь частину програмного забезпечення.

DeepFace – це система виявлення та розпізнавання облич, створена дослідницькою групою у Facebook, яка може легко виявляти та знаходити людські обличчя на даному зображенні чи відео. Він використовує високонадійні методи глибокого навчання з найсучаснішими технологіями для отримання відповіді в реальному часі для реальних програм:

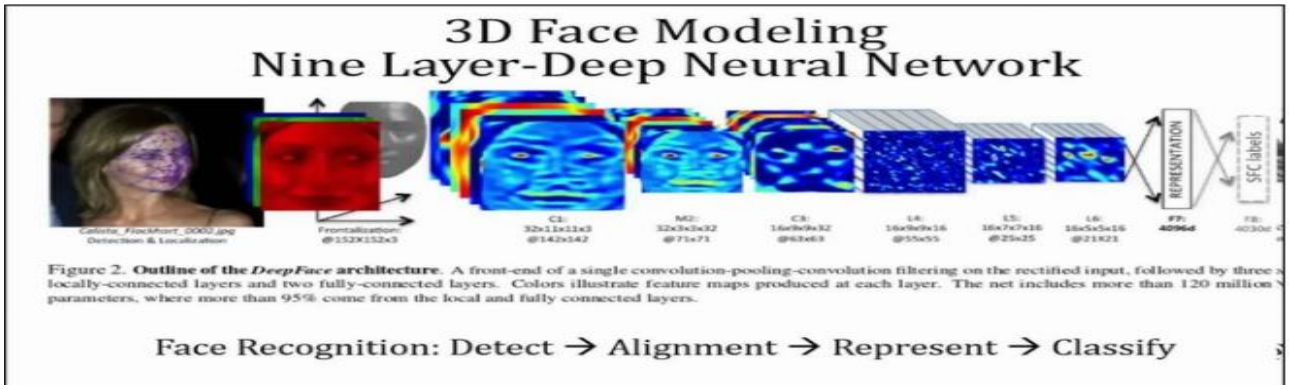

Figure 2. **Outline of the *DeepFace* architecture**. A front-end of a single convolution-pooling-convolution filtering on the rectified input, followed by three locally-connected layers and two fully-connected layers. Colors illustrate feature maps produced at each layer. The net includes more than 120 million parameters, where more than 95% come from the local and fully connected layers.

Рисунок 1.11 – Архітектура роботи системи DeepFace.

- Миттєве виявлення та розпізнавання обличчя в Інтернеті полегшується простим завантаженням фотографії з комп’ютера або веб-камери.
- Швидкі результати в реальному часі відображаються під виявленням обличчя, аналізом обличчя, перевіркою обличчя та аналізом емоцій.
- Використання різних атрибутів полегшується шляхом завантаження місцевого зображення, а аналіз проводиться на основі віку, статі, пози голови, стану очей та кольору шкіри.

- Програмне забезпечення полегшує створення бази даних з граней та пошук на основі заданого зображення чи імені.

Висновки. Минуло десять плідних років відтоді, як було винайдено перше програмне забезпечення для виявлення обличчя, яке може виявляти та розпізнавати людські обличчя. Зараз ринок, наповнений програмними рішеннями та програмами для виявлення обличчя, доступний для індивідуального та ділового використання в усьому світі.

У звіті зазначається, що лише в США понад 62 мільйони камер безпеки встановлені за допомогою біометричних програм розпізнавання обличчя та автентифікації, а мільйони знаходяться в кишенях людей як смарт-камери на основі гаджетів.

Markets and Markets – відома дослідницька фірма, прогнозує, що попит на камери та програмне забезпечення, обладнані ШІ, становитиме 7,76 млрд доларів у найближчі три роки. Це покращить технологію виявлення відбитків обличчя та пальців у корпоративному, урядовому та приватному секторах, включаючи торгові центри, спортивні майданчики, громадські збори.

Технологія розпізнавання обличчя має великий потенціал, щоб стати переважною в майбутньому. Імовірність помилок просто менша, якщо задіяні передові програми. Він сек'юритизує людські особи та дані в системах, доки вони там зберігаються.

1.4 Аналіз методології реалізації систем

Протягом останніх кількох років розпізнаванню обличчя приділяли значну увагу і вважали його одним із найуспішніших додатків у галузі аналізу зображень. Людські обличчя представляють складний, багатовимірний, значущий візуальний стимулятор. Розробка обчислювальної моделі для розпізнавання обличчя складна.

Розпізнавання обличчя можна розглядати як основну частину систем розпізнавання облич відповідно до його здатності фокусувати обчислювальні ресурси на частині зображення, що містить обличчя. Процес розпізнавання обличчя на зображеннях складний із-за мінливості в людських обличчях, таких як: поза, положення та орієнтація, колір шкіри, наявність окулярів або волосся на обличчі, відмінності в посиленні камери, умови освітлення, та роздільна здатність зображення. Аналіз виразу обличчя був в першу чергу напрямком досліджень для психологів в останні роки. У той же час, досягнення в багатьох сферах, таких як: виявлення обличчя відстеження та визнання, розпізнавання образів і обробка зображень суттєво сприяла дослідженню автоматичного розпізнавання виразів обличчя. Розпізнавання обличчя слід виконувати перед системою розпізнавання. Це робиться для отримання відповідної інформації для аналізу обличчя та виразу обличчя. Два класи методів представлення обличчя та вилучення відповідної інформації. А витяг геометричних ознак базується на таких параметрах, як от очі, рот і ніс. Водночас обличчя є представлений як масив значень інтенсивності пікселів, попередньо оброблених підходами на основі зовнішнього вигляду (текстура). Потім цей масив порівнюється із шаблоном обличчя за допомогою відповідної метрики. Дослідження порівнювало ефективність цих технік подання при розпізнаванні облич. Тому, згідно зі складністю процесу розпізнавання обличчя, нещодавно було розроблено багато додатків, заснованих на виявленні людського обличчя, таких як: системи спостереження, цифровий моніторинг, інтелектуальні роботи, ноутбуки, камери для ПК, цифрові камери та мобільні телефони 3G.

Нейронна мережа, підключена до сітківки (RCNN). Роулі, Балуя та Канаде (1996) представили систему виявлення обличчя, засновану на нейронній мережі, підключеній до сітківки (RCNN), яка досліджує маленькі вікна зображення, щоб вирішити, чи містить кожне вікно обличчя. На рисунку 1.12 показано цей підхід. Система робить арбітраж між багатьма мережами для поліпшення продуктивності в одній мережі. Вони використовували алгоритм завантажувального ремінця в ході

навчання для навчальних мереж, щоб додати помилкові виявлення до навчального набору. Це усуває складне завдання ручного відбору прикладів тренувань, які не стосуються обличчя, і які потрібно обирати так, щоб охоплювати весь простір зображень без обличчя. Спочатку до вікна зображення застосовується етап попередньої обробки, адаптований. Потім вікно передається через нейронну мережу, яка вирішує, чи містить вікно грань. Вони використали три навчальні набори зображень. Тестовий набір A, зібраний в CMU: складається з 42 відсканованих фотографій, газетних зображень, зображень, зібраних з WWW, та телевізійних зображень (169 фронтальних виглядів облич і вимагає ANN для вивчення 22 053 124 вікон розміром 20 × 20 пікселів). Тестовий набір B складається з 23 зображень, що містять 155 граней (9 678 084 вікон). Test Set C схожий на Test Set A, але містить зображення із більш складним фоном і без будь-яких граней для вимірювання швидкості виявлення помилок: містить 65 зображень, 183 обличчя та 51 368 003 вікна. Коефіцієнт виявлення цього підходу дорівнює 79,6% граней протягом двох великих тестових наборів та невеликої кількості помилкових спрацьовувань.

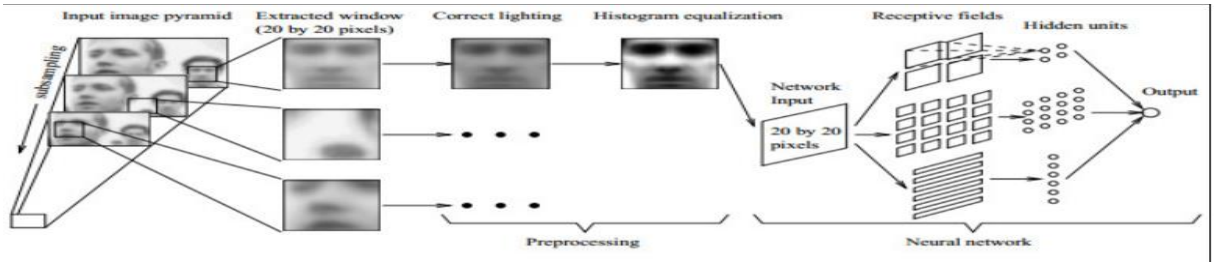

Рисунок 1.12 – RCNN мережа для виявлення обличчя.

Інваріантна нейронна мережа (RINN). Роулі, Балуя та Канаде (1997) представили систему виявлення обличчя на основі нейронної мережі. На відміну від подібних систем, які обмежуються виявленням вертикальних, фронтальних граней, ця система виявляє грані при будь-якому ступені обертання в площині зображення. На рисунку 1.13 показано підхід RINN. Система використовує кілька мереж: перший – це мережа «маршрутизатора», яка обробляє кожне вікно введення для визначення

його орієнтації, а потім використовує цю інформацію для підготовки вікна до однієї або кількох детекторних мереж. Ми представляємо методи навчання для обох типів мереж. Ми також проводимо аналіз чутливості в мережах і представляємо емпіричні результати на великому наборі тестів. Нарешті, ми представляємо попередні результати для виявлення граней, які повертаються за межі площини зображення, таких як профілі та напівпрофілі.

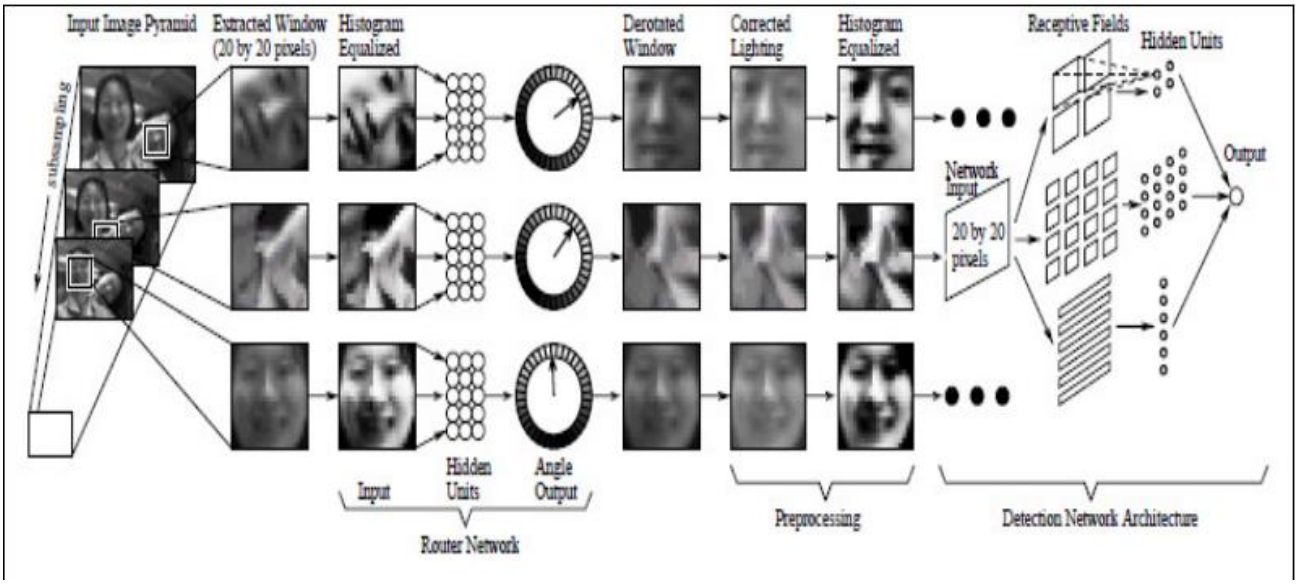

Рисунок 1.13 – RINN мережа для виявлення обличчя

Аналіз основних компонентів за допомогою ANN (PCA & ANN). Джеффрі Норріс (1999) використовував аналіз основних компонентів (PCA) із специфічною для класу лінійною проекцією для виявлення та розпізнавання облич у потоці відео в реальному часі. На рисунку 1.14 показано PCA та ANN для виявлення обличчя. Система надсилає команди на автоматичні розсувні двері, синтезатор мови та сенсорний екран через сервер управління клієнтськими дверима. Для розробки системи використовували Matlab, C та Java. Кроки системи для пошуку обличчя на зображенні:

1. Виберіть кожні 20 × 20 області вхідного зображення.
2. Використовуйте значення інтенсивності його пікселів як 400 входів до ANN.

3. Значення подачі вперед через ANN.

4. Якщо значення вище 0,5, область представляє грань.

5. Повторіть кроки (1..4) кілька разів, кожен раз у зміненій версії вихідного вхідного зображення, щоб шукати обличчя в різних масштабах.

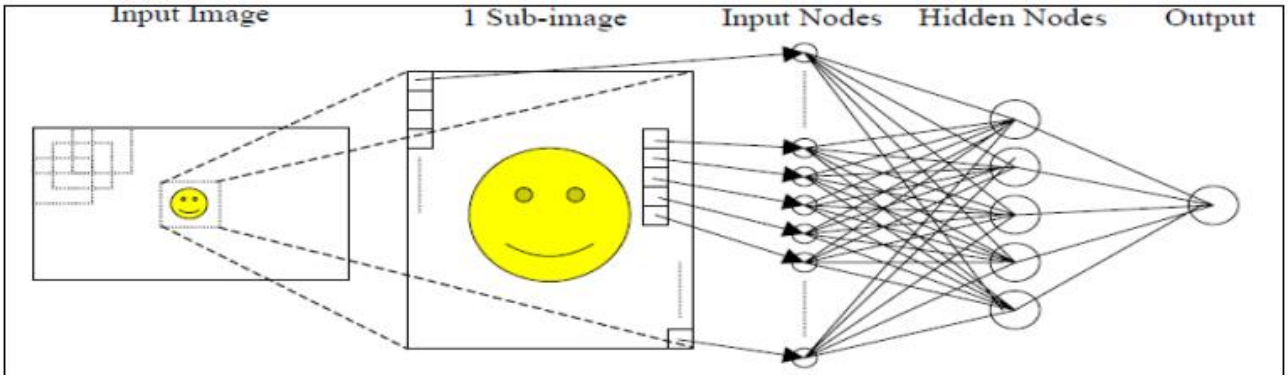

Рисунок 1.15 – PCA та ANN мережі для виявлення обличчя.

Швидкі нейронні мережі (FNN). Хазем Ель-Бакрі (2002) запропонував підхід до швидких нейронних мереж (FNN) для зменшення часу обчислення для виявлення людських облич. Кожне зображення ділиться на невеликі допоміжні зображення, а потім кожне тестується окремо за допомогою швидкого ANN. Експериментальні результати порівняння із звичайними нейронними мережами показали, що висока швидкість досягається при застосуванні FNN.

Поліноміальна нейронна мережа (PNN). Lin-Lin Huang та ін. (2003) запропонував метод виявлення обличчя за допомогою поліноміальної нейронної мережі (PNN). Локальні регіони у багатомасштабних розсувних вікнах класифікуються PNN до двох класів (обличчя та нелице) для визначення людських облич на зображенні. PNN приймає біноми проекції локального зображення на підпростір ознак, засвоєний аналізом основних компонентів (PCA), як вхідні дані. Вони досліджували вплив PCA як на зразки обличчя, так і на об'єднані зразки обличчя та обличчя.

Згорткова нейронна мережа (CNN). Масакадзу Мацугу (2003) описав алгоритм, заснований на правилах, для надійного розпізнавання виразів обличчя у поєднанні з розпізнаванням обличчя за допомогою згорткової нейронної мережі (CNN). На рисунку 1.16 показано підхід CNN. У цьому дослідженні було розглянуто проблему незалежності суб'єкта та перекладу, обертання та незмінності масштабу при розпізнаванні виразу обличчя.

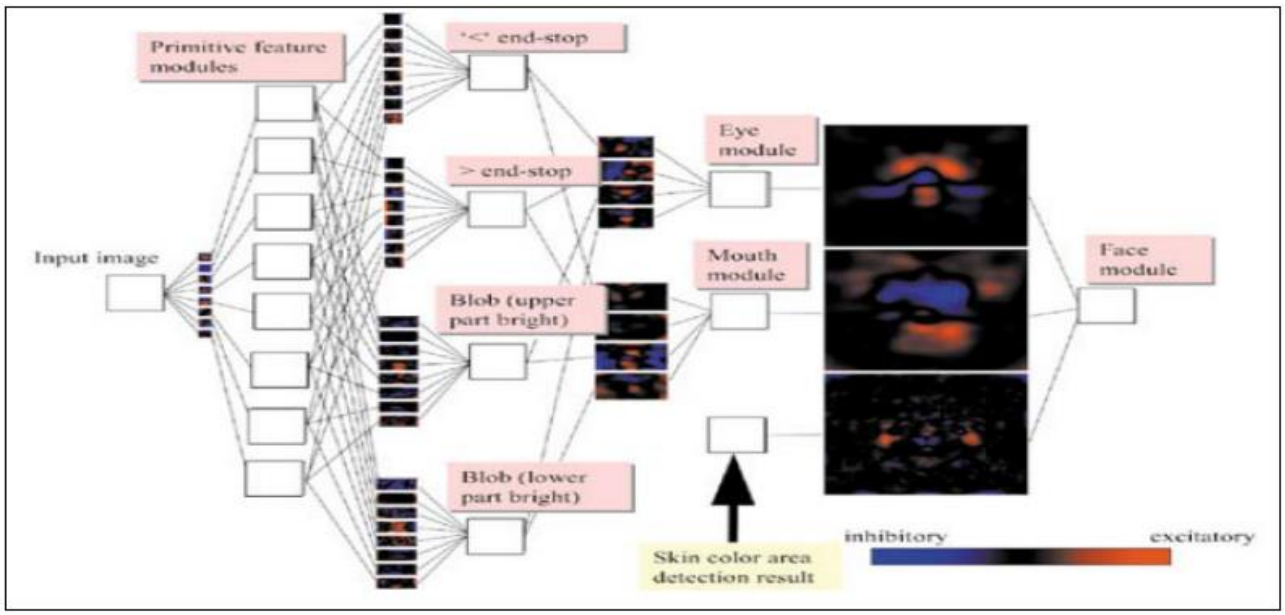

Рисунок 1.16 – CNN для виявлення обличчя.

Еволюційна оптимізація нейронних мереж. Стефан та співавт. (2004) використовували ANN для отримання рішення, чи представляє попередньо оброблена область зображення людське обличчя чи ні. Вони описали оптимізацію цієї мережі гібридним алгоритмом, що поєднує еволюційні обчислення та навчання на основі градієнта. Розроблені рішення працюють значно швидше, ніж архітектура, розроблена експертом, без втрати точності. Запропонований гібридний алгоритм вирішує проблему зменшення кількості прихованих нейронів мережі виявлення обличчя без втрати точності виявлення. Швидкість класифікації, чи відповідає

область зображення обличчю чи ні, може бути покращена приблизно на 30%. На рисунку 1.17 показано еволюційну оптимізацію ANN.

Як ми можемо спостерігати, яку кількість ітерацій проходить гібрідний алгоритм для визначення кінцевого об’єкту. Також наочно зображено архітектуру побудови вхіодних та польових зв’язків.

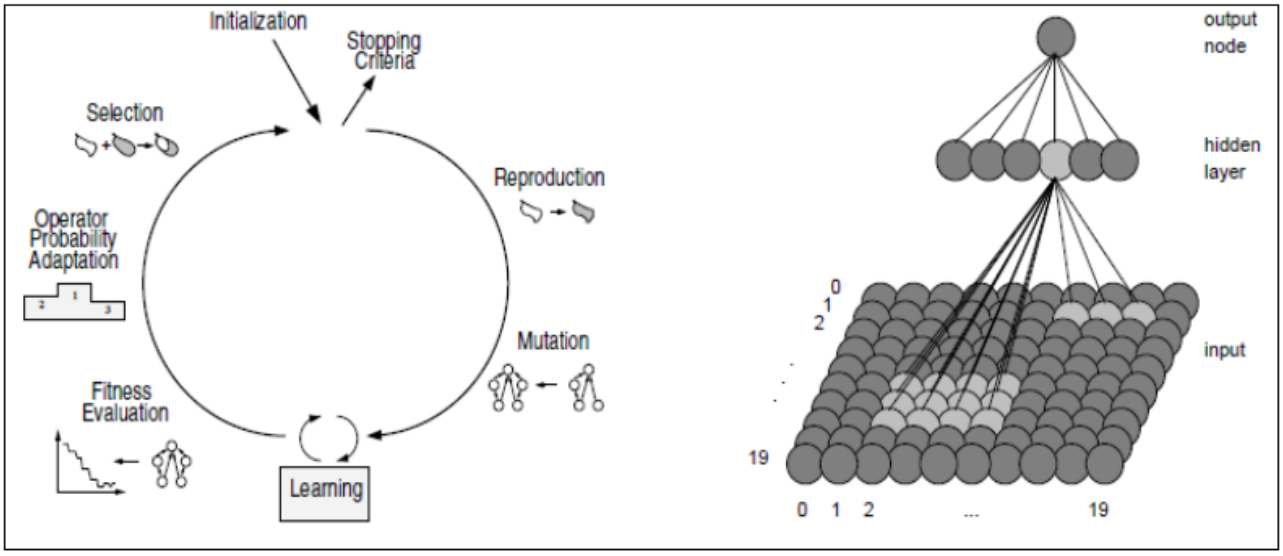

Рисунок 1.17 – (a): Гібридний алгоритм. (b): Візуалізація вхідних та польових зв’язків.

Нарешті, ми можему зазначити, що залежно від виду нейронної мережі показник виявлення відрізняється у всіх варіантах. Це можно спостерігати у данній таблиці**.** На базі таблиці з результатим дослідження можно скласти графік залежностей, на якому ми можем спостерігати наглядну показники топологій

Як ми можемо спостерігати, швидкість виявлення в різних підходах дещо відрізняються між собою. Так, найвищий рівень виявлення обличчя можна отримати за допомогою CNN – вище 95 відсотків. Близько до найкращого результату підійшов алгоритм BPNN, який поступається найкращому лише декількома відсотками. Далі йдуть алгоритми RCNN та MLP – приблизно 91-93%, що також вважається дуже впевненим та ефективним результатом. Відносно попередніх зазначених моделей

близько 90 відсотків має алгоритм Gabor, що також показую досить впевнений результат, але дещо поступається своїм суперникам.

Ну й найнижчий результат у виявленні обличчя людини демонструє алгоритм – PNN. Як видно на графіку, він отримав результат лише 85%, але даний результат все ще можна вважати достатньо позитвним з впененою оцінкою «добре».

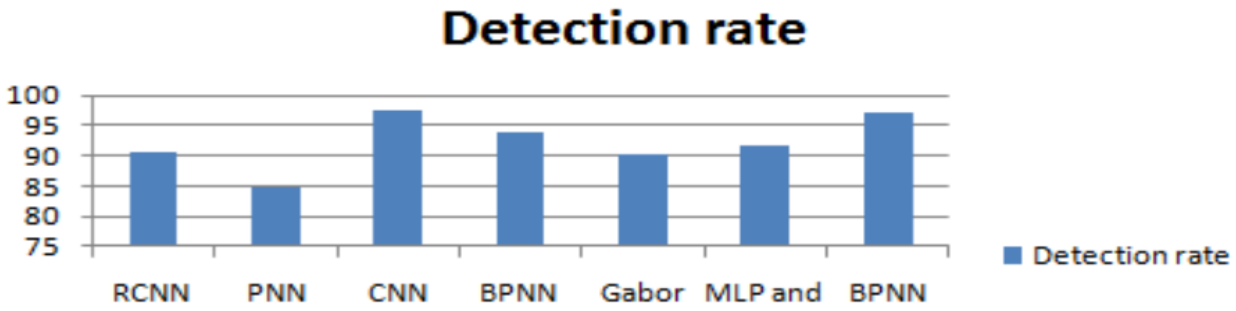

Рисунок 1.18 – Графік залежностей показників топологій.

Topology	Data Base: Training & Testing	Performance
Retinal connected neural network	Three training sets of images. Test SetA: 42 scanned photographs Test SetB: 23 images contain 155 faces Test SetC: 65 images, 183 faces (images with more complex backgrounds and without faces to measure false detection)	Detect 78.9% - 90.5% of faces in a set of 130 test images Acceptable number of false detections.
PCA with ANN	Select 700 pictures in Kah-Kay Sung's data set of 1488 faces to train ANN with 700 random noise pictures as negative examples remaining 788 faces in Kah-Kay's data set, followed by 788 random noise pictures	1.2% error after training for 50 epochs 1566 examples, 35 mis classifications made (2.23% error).
PNN	First set: 3257 images downloaded from several websites (384×384), with one face in each image. Second set: 130 images downloaded from website of CMU	Detection rate = 84.6% False rate=3:51 × 10−6
CNN	training of CNN, the number of facial fragment images used is 2900 for the FD2 layer, 5290 for the FD3, and 14,700 (face) for the FD4 layer, respectively. Number of non-face images, also used for the FD4 layer, is 137.	Recognition rate = 97.6% for 5600 still images of more than 10 subjects
BPNN	Training set contains 12000 face images collected from various face DBs. These samples also include the scaled versions at the same face with factor (0.8 - 1.12)	Detection rates measured for separate test set of 500 faces and 4000 non-faces. Performance=94%.
Gabor wavelet with ANN	ORL dataset: 400 frontal faces: 10 tightly cropped (92×112) with 256 grey images of 40 individuals with variations in pose, illumination, ..etc	Detect (77.9% - 90.3%) of faces in a set of 130 test images
MLP and MRC	Training set: face images from MIT DB. Images (scaled to 20×20) Test set: 2000 face/non-face images from MIT DB. Non-face patterns generated at different locations and scales.	Detection rate = 91.6% Error rate = 7.54%
BPNN	50 real images taken under different lighting conditions (digital camera images and web images from several websites).	Detect 97.3% of faces in a set of 50 real images. Processing Time (s) of image (63×180)= 3.8 Processing Time (s) of image (200×219)= 6.2

Рисунок 1.19 – Результати дослідженних топологій.

Треба зазначити, що при дослідженні данних топологій були різні вимоги щодо архітектур, мови програмування, вимоги до процесора, пам’яті, баз данних для навчання тощо.

1.5 Проблематика розробки систем штучного інтелекту на базі обмежених обчислюваних ресурсів

Яке відношення має крихітний мікроконтролер до могутнього світу штучного інтелекту (ШІ) і його технологічним відгалуженням, таким як машинне навчання і глибоке навчання? Зрештою, проекти ШІ, в основному, асоціюються з потужними процесорами, графічними процесорами і FPGA.

Чи є це так? Хоча це поняття недалеко від реальності, воно швидко змінюється з переходом ШІ від хмари до периферії, де обчислювальні механізми ШІ дозволяють мікроконтролерам розширювати межі можливого для вбудованих додатків. Це, в першу чергу, дозволяє вбудованим конструкцій поліпшити швидкість реагування в реальному часі і захистити пристрою від кібератак.

На рисунку зображено аналіз, проведений у хмарі, наближається до зондування та дій, і це зумовлює потребу в мікроконтролерах з підтримкою ШІ; це зменшує необхідну пропускну здатність для передачі даних та економить можливості обробки хмарних серверів.

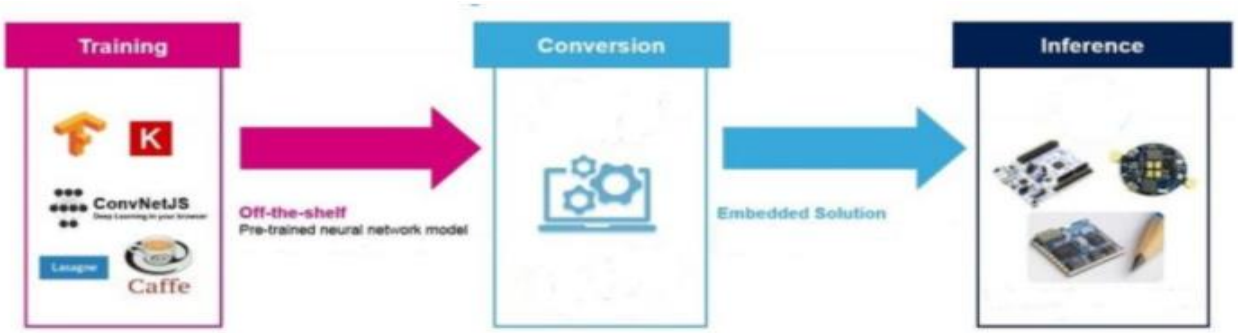

Рисунок 1.20 – Робота хмарних технологій у купі з мікроконтролерами.

MCU, обладнані алгоритмами AI, беруть на себе програми, які охоплюють такі функції, як розпізнавання об’єктів, послуги з підтримкою голосу та обробка природної мови. Вони також сприяти більшій точності та конфіденційності даних для пристроїв, що працюють від акумулятора, в Інтернеті речей (IoT), переносимих, медичних додатках.

Отже, як мікроконтролери реалізують функції ШІ в конструкціях ребер і вузлів? Нижче наведено огляд трьох основних підходів, що дозволяють мікроконтролерам виконувати прискорення ШІ на межі мережі IoT.

Перший, і, мабуть, найпоширеніший підхід, стосується перетворень моделей для ряду нейронних мереж (NN), таких як Caffe 2, TensorFlow Lite та Arm NN для розгортання хмарних моделей та механізмів виведення на MCU. Існують програмні засоби, які беруть заздалегідь навчені нейронні мережі з хмари та оптимізують їх для мікроконтролерів, перетворюючи їх у С-код.

Оптимізований код, що працює на мікроконтролерах, може виконувати функції ШІ в програмах виявлення голосу, зору та аномалій. Інженери можуть завантажувати ці інструменти в конфігурацію MCU і робити висновки оптимізованих нейронних мереж. Ці набори інструментів ШІ також пропонують приклади коду для додатків ШІ на основі нейронної мережі.

Рисунок 1.21 – Набори інструментів AI виконують перетворення моделей для запуску висновків оптимізованих нейронних мереж на недорогих та малопотужних мікроконтролерах.

Другий підхід обходить необхідність у попередньо навчених нейромережевих моделях, запозичених у хмарі. Дизайнери можуть інтегрувати бібліотеки ШІ в мікроконтролери та включати в свій код можливості локального навчання та аналізу ШІ.

Згодом розробники можуть створювати моделі даних на основі сигналів, отриманих від датчиків, мікрофонів та інших вбудованих пристроїв на краю, і запускати такі програми, як прогнозне обслуговування та розпізнавання шаблонів.

По-третє, наявність ко-процесорів, призначених для штучного інтелекту, дозволяє постачальникам MCU прискорити розгортання функцій машинного навчання. Співпроцесори, такі як Arm Cortex-M33, використовують такі популярні API, як CMSIS-DSP, щоб спростити перенесення коду, дозволяючи MCU, тісно поєднаним з копроцесорами, прискорювати функції ШІ, такі як співвідношення та матричні операції.

Вищезазначені програмні та апаратні платформи демонструють, як впровадити функціональність ШІ в недорогі мікроконтролери за допомогою механізмів виведення, розроблених відповідно до вбудованих вимог дизайну. Це має вирішальне значення, оскільки мікроконтролери з підтримкою штучного інтелекту цілком можуть перетворити конструкції вбудованих пристроїв у IoT, промислові, розумні будівлі та медичні програми.

2 АНАЛІЗ ТЕХНОЛОГІЙ

2.1 Аналіз мережевих протоколів у мікроконтролерах

З появою Інтернету речей (IoT) як у споживчих, так і в комерційних програмах, велика увага приділяється зв’язкам. IoT – це все, що стосується підключення пристроїв разом, щоб забезпечити досвід вищого порядку та зручність використання.

Першим прикладом замовлення може бути офісна кімната для переговорів, де світло та обладнання можуть автоматично вмикатись та вимикатись залежно від того, коли хтось перебуває в кімнаті для економії енергії. Функцією другого порядку може бути відстеження всієї інформації про використання приміщення, надання менеджерам змоги побачити, які кімнати переговорів активно використовуються, а які ні, оптимізувати робочий простір та вирішити, коли приміщення потребує для чищення через інтенсивне використання.

Хоча ми можемо піти ще далі, і ці дані можна використовувати для підготовки залів засідань заздалегідь, увімкнувши проектори та інше обладнання, не тільки на основі запланованих зустрічей, але й на основі очікуваної діяльності на основі виміряних даних. Цей довідковий документ розгляне ключові будівельні блоки, необхідні для того, щоб зробити подібні сценарії можливими, а також те, як їх можна інтегрувати у дедалі менші та енергоефективніші мікроконтролери.

Розглянемо мережеві бездротові технлогії передачі даних:

1. Wi-Fi – найбільш швидкісна бездротова технологія з пропускною спроможністю до 54 Мбіт / с. Базується на стандартах IEEE 802.11a, 802.11b, 802.11g і 802.11n і використовує радіоканал 2,4 ГГц і / або 5 ГГц. Wi-Fi застосовується в широкому спектрі додатків, таких як медичні прилади (системи моніторингу, діагностичні установки і так далі),

споживча електроніка (планшети, смартфони та інше), промислова і домашня автоматика, системи безпеки, відеоспостереження тощо.

2. ZigBee – стандарт, застосовуваний для створення пористих мереж, пристроїв і датчиків. Використовує робочу частоту радіосигналу 2,4 ГГц. ZigBee широко поширений в системах автоматизації в промисловості, медицині, логістиці, системах обліку споживання енергії і так далі.
3. Bluetooth – одна з найбільш популярних технологій для зв'язку пристроїв, розташованих на невеликих відстанях аж до декількох сотень метрів. Bluetooth дозволяє працювати як в режимі «точка-точка», так і в режимі «зірка». Проте, цей стандарт найчастіше використовується для обміну інформацією між двома пристроями. Для передачі використовується діапазон 2,4 ... 2,485 ГГц. Bluetooth застосовується в портативній електроніці («розумні годинник», смартфони та інше), в датчиках, побутовій електроніці, медичних приладах і так далі.
4. Near Field Communication (NFC) – бездротова технологія, яка працює на відстані до декількох сантиметрів. У NFC використовується сигнал 13,56 МГц. Обмін даними відбувається між двома пристроями: ініціатором (зчитувачем) і метою (транспондером). При цьому ініціатор здатний не тільки передавати і отримувати дані, але і забезпечувати харчування транспондера. З цієї причини пасивні NFC-пристрої взагалі не містять елементи живлення.

За допомогою схем розгялнемо дальність роботи, енерноемність та пропускну здатність бездротових технологій. Перед тим, розглянемо деяку технічну специфікацію стосовно Near Field Communication. За цим специфікаціям існує наступні способи зв'язку для пристроїв NFC: NFC-A, NFC-B, NFC-F, і п'ять типів NFC-міток. Пристрої NFC можуть бути активною або пасивною комунікації і підтримувати один (або декілька) з 3 режимів роботи. Тип зв'язку NFC-A заснований на стандарті ISO / IEC 14443A для безконтактних карт. Типи зв'язку відрізняються

використовуваними режимами кодування сигналу і модуляції. NFC-A використовує код Міллера і амплітудну модуляцію. Двійкові дані передаються зі швидкістю близько 106 Кбіт / с, сигнал повинен змінюватися від 0% до 100%, щоб розрізняти двійкову 1 і двійковий 0. Тип зв'язку NFC-B заснований на стандарті ISO / IEC 14443B для безконтактних карт. NFC-B використовує метод манчестерського кодування. Двійкові дані також передаються зі швидкістю близько 106 Кбіт / с. Тут замість 100% використовується 10% -е зміна амплітуди для двійкового 0 (тобто низького рівня) і 100% для двійковій 1 (тобто високого). У манчестерському кодуванні перехід з низького на високий рівень являє двійковий 0, а перехід з високого на низький рівень являє двійкову 1. Тип зв'язку NFC-F заснований на стандарті FeliCA JIS X6319-4, також відомий як просто FeliCa. Стандарт регулюється японської JICSAP. Там ця технологія, і найбільш популярна. Швидкість передачі даних 212/424 Кбіт / с, використовується манчестерське кодування і амплітудна модуляція. У режимі активної комунікації кожен пристрій генерує своє електромагнітне поле. Пристрій-ініціатор генерує електромагнітне поле на частоті 13,56 МГц, використовує амплітудну модуляцію для відправки команди, а потім відключає поле. Пристрій-мета у відповідь генерує своє електромагнітне поле і точно також модулюючи його відправляє відповідь. Щоб уникнути зіткнень, тільки відправляє пристрій випромінює електромагнітне поле. Цей вид комунікації використовується тільки в режимі P2P.

Зображено схему енергоемності:

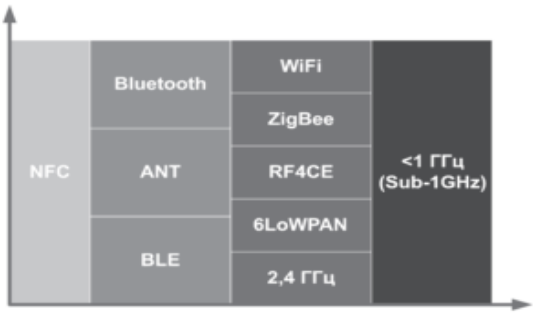

Рисунок 2.1 – Схема дальності роботи бездротових технологій

NFC-транспондери можуть і зовсім обходитися без власного елемента живлення, використовуючи для роботи енергію радіовипромінювання ведучого.

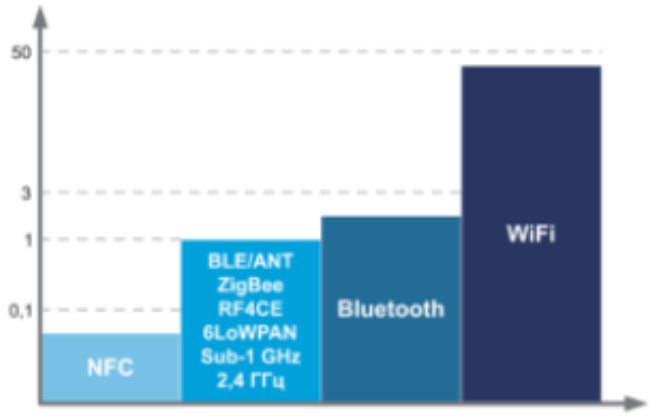

Рисунок 2.2 – Пропускна здатність бездротових технологій

NFC програє всім іншим технологіям за швидкістю і дальністю дії, але бере впевнений реванш за рівнем споживання.

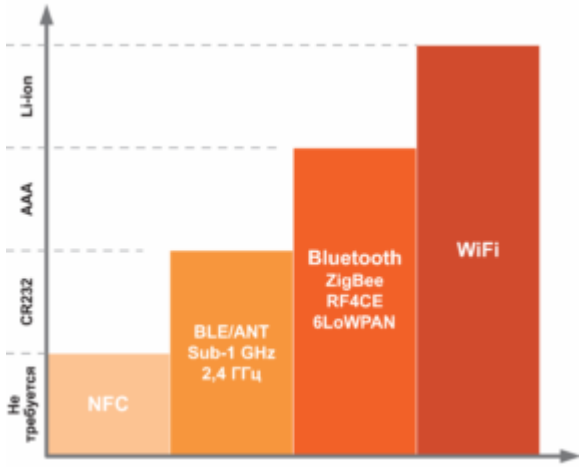

Рисунок 2.3 – Схема енергоемності бездротових технологій

При виборі оптимальної технології необхідно враховувати і таку особливість, як складність реалізації. Наприклад, якщо мова йде про створення захищеного Wi-Fi-пристрою, то розробники повинні мати досить високу кваліфікацію. Це відноситься

як до схемотехніки, так і до програмістів. Будемо відверті – далеко не всім під силу впоратися з цими завданнями.

Існує, також, технологія бездротової роботи – LTE (4G). Це стільникова технологія, яка підтримує високу швидкість передачі даних та дальність роботи. На відміну від 2G, використовуючи TDMA і FDMA, і 3G, використовуючи CDMA, LTE використовує OFDMA (ортогональний мультиплексний доступ з частотним поділом) як технологію багаторазового доступу. На OFDM (ортогональне мультиплексування з розподілом частоти) сигнал, що передається, розкладається на сигнал `` n " (це можна зробити за допомогою послідовного / паралельного перетворювача)
Іншим досягненням систем LTE є використання методу множинних антен. LTE – це перша глобальна мобільна стільникова система, що використовує технологію MIMO (Multiple-Input Multiple-Output). Використання декількох антен може значно покращити ефективність зв'язку завдяки більшій пропускній здатності каналу.

У наведеній нижче таблиці порівняно LTE та LTE-A. Перед тим, пару слів про LTE-Advanced – можливість агрегування спектра є, мабуть, найголовнішою характерною особливістю LTE-Advanced і забезпечує додаткову гнучкість використання спектра, закладену в системі LTE в формі набору каналів з масштабированной шириною.

System Performance		LTE-Advanced	LTE
Peak rate	Uplink	1000Mbps@100MHz	100Mbps@20MHz
	Downlink	500Mbps@100MHz	50Mbps@20MHz
Control-plane delay	Idle to connected	<50ms	<100ms
	Dormant to active	<10ms	<50ms
User-plane delay (without load)		Lower than that of LTE	<5ms
Spectral efficiency	Peak	Downlink: 30 bps/Hz @≤ 8×8, Uplink: 15bps/Hz @≤4×4	Downlink: 5bps/Hz @ 2×2, Uplink: 2.5bps/Hz @1×2
	Average	Downlink: 3.7bps/Hz/cell @ 4×4, Uplink: 2.0 bps/Hz/cell @ 2×4	Downlink: 3 to 4 times of R6 HSPA @ 2×2, Uplink: 2 to 3 times of R6 HSPA @1×2
	Cell edge	Downlink: 0.12bps/Hz/cell/user @4×4 Uplink: 0.07 bps/Hz/cell/user @ 2×4	N/A
Mobility		≤350km/h, ≤500km/h@freq band	≤350km/h
Flexible bandwidth deployment		Continuous spectrum @>20MHz, Spectral convergence	1.4, 3, 5, 10, 15, 20MHz Support paired spectrum and unpaired spectrum

Рисунок 2.4 – Порівняльна таблиця технологій LTE та LTE-A

Як ми можемо спостерігати, LTE-A має вижчі показники по всім параметрам, але до даної технології запобігаються більш жорсткі вимоги щодо інсталяції, порівняно із молодшими технологіями. Тому й сама реалізація такой системи дорожча за LTE.

Аналізуючи дані факти, підводимо підсумок даних стильникових рішень – значним недоліком є ціна. Мікроконтроллери з підтримкою LTE значно дорощі. Тому для даного бюджетного проекту ця стільникова технологія не підходить.

Проаналізувавши дані технології, було вирішено використовувати технологію Wi-Fi, оскільки дана технологія найбільш розповсюджена, проста в роботі, має високий показник швидкості передачі даних та охоплення.

2.2 Аналіз ресурсів мікроконтролера

Мікроконтролер – це одночипний мікрокомп’ютер, виготовлений завдяки виробництву VLSI. Мікроконтролер також називають вбудованим контролером, оскільки мікроконтролер та його опорні схеми часто вбудовані в пристрої, якими вони керують. Мікроконтролер доступні у різній разрядності, наприклад мікропроцесори (4-бітні, 8-бітні, 16-бітні, 32-бітні, 64-бітні та 128-бітні мікроконтролери доступні сьогодні). На рисунку 2.5 зображено один із моделей мікроконтролерів:

Рисунок 2.5 –Мікроконтолер

1) Мікроконтролер в основному містить один або кілька таких компонентів:
   - Центральний процесор (CPU);
   - Оперативна пам’ять (RAM);
   - Пам'ять лише для читання (ПЗУ);
   - Порти вводу / виводу;
   - Таймери та лічильники;
   - Елементи управління перериваннями;
   - Аналого-цифрові перетворювачі;
   - Цифрові аналогові перетворювачі;
   - Порти послідовної взаємодії;
   - Коливальні ланцюги;

2) Мікроконтролер внутрішньо складається з усіх функцій, необхідних для обчислювальної системи, і функціонує як комп'ютер, не додаючи в нього жодних зовнішніх цифрових частин.

3) Більшість висновків мікросхеми мікроконтролера може бути програмованим користувачем.

4) Мікроконтролер має безліч інструкцій щодо обробки бітів, які програміст може легко зрозуміти.

5) Мікроконтролер здатний обробляти булеві функції.

6) Більша швидкість і продуктивність.

7) Структура ПЗУ мікросхеми мікроконтролера забезпечує кращий захист мікропрограми.

8) Простий в проектуванні з низькою вартістю та невеликими розмірами.

Для повного розуміння роботи мікроконтролера потрібно дослідити його побудову, тому нижче представлено схему структури мікроконтролера:

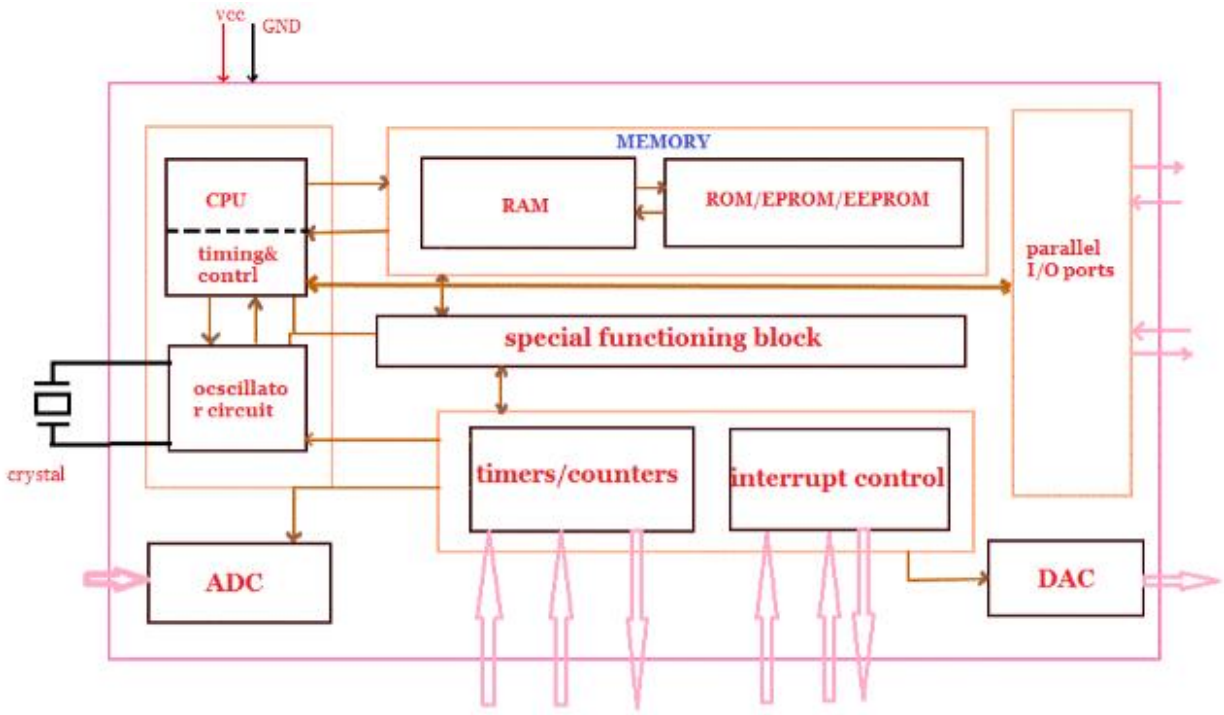

Рисунок 2.6 – Структура мікроконтролера

Розберемо детально кожен елемент схеми, для чого він потрібен, яка його задача.

1. Процесор – це мозок мікроконтролера. Процесор відповідає за отримання інструкції, декодує її, а потім остаточно виконує. Процесор з’єднує кожну частину мікроконтролера в єдину систему. Основною функцією ЦП є отримання та декодування інструкцій. Інструкція, отримана з пам'яті програми, повинна бути декодована процесором.
2. Пам'ять. Функція пам'яті в мікроконтролері така ж, як у мікропроцесора. Вона використовується для зберігання даних та програм. Мікроконтролер, як правило, має певну кількість оперативної пам’яті та ПЗУ (EEPROM, EPROM тощо) або флеш-пам’яті для зберігання вихідних кодів програм.
3. Паралельні вхідні / вихідні порти. Порти паралельного введення / виведення в основному використовуються для керування / інтерфейсу

різних пристроїв, таких як LCD, LED дісплеї, принтери, пам'яті тощо, до мікроконтролера.

4. Послідовні порти забезпечують різні послідовні інтерфейси між мікроконтролером та іншими периферійними пристроями, такими як паралельні порти.
5. Таймери / лічильники. Це одна з корисних функцій мікроконтролера. Мікроконтролер може мати більше одного таймера та лічильників. Таймери та лічильники забезпечують усі функції синхронізації та відліку всередині мікроконтролера. Основними операціями цього розділу є виконання тактових функцій, модуляція, генерація імпульсів, вимірювання частоти, здійснення коливань тощо. Це також може бути використано для підрахунку зовнішніх імпульсів.
6. Аналого-цифровий перетворювач (АЦП). Перетворювачі АЦП використовуються для перетворення аналогового сигналу в цифрову форму. Вхідний сигнал у цьому перетворювачі повинен бути в аналоговій формі (наприклад, вхід датчика), а вихід з цього пристрою – у цифровій формі. Цифровий вихід можна використовувати для різних цифрових програм (наприклад, вимірювальних пристроїв).
7. Цифрово-аналоговий перетворювач (ЦАП). ЦАП виконує операцію зворотного перетворення АЦП, ЦАП перетворює цифровий сигнал в аналоговий формат. Зазвичай він використовується для управління аналоговими пристроями, такими як двигуни постійного струму, різні приводи та ін.
8. Управління перериваннями. Управління перериваннями, що використовується для забезпечення переривання (затримки) робочої програми. Переривання може бути зовнішнім (активується за допомогою штифта переривання) або внутрішнім (за допомогою інструкції щодо переривання під час програмування).

9. Спеціальний функціонуючий блок. Деякі мікроконтролери використовують лише деякі спеціальні програми (наприклад, космічні системи та робототехніку), ці контролери містять додаткові порти для виконання таких спеціальних операцій. Це розглядається як спеціальний функціонуючий блок.

Нижче приведено пункти, які коротко характеризують дані мікросхеми.

Мікропроцесор:

1. Це лише загальний комп'ютерний процесор.
2. Пам'ять, порти вводу / виводу, таймери, переривання недоступні всередині мікросхеми.
3. У ньому повинно бути багато додаткових цифрових компонентів для його роботи.
4. Системи стають громіздкішими та дорожчими.
5. Неможливо обробляти булеві функції.

Мікроконтролер:

1. Це сам мікрокомп'ютер.
2. Пам'ять, порти вводу / виводу, таймери, переривання інтегровані всередині мікросхеми мікроконтролера.
3. Може функціонувати як мікрокомп'ютер без будь-яких додаткових компонентів.
4. Робить систему простою, економічною та компактною.
5. Обробляє булеві функції.

Для того, щоб наглядно продемонструвати різницю архітектур мікропроцесора та мікроконтролера, пропоную звернутися до наступної порівняльної схеми двох пристроїв.

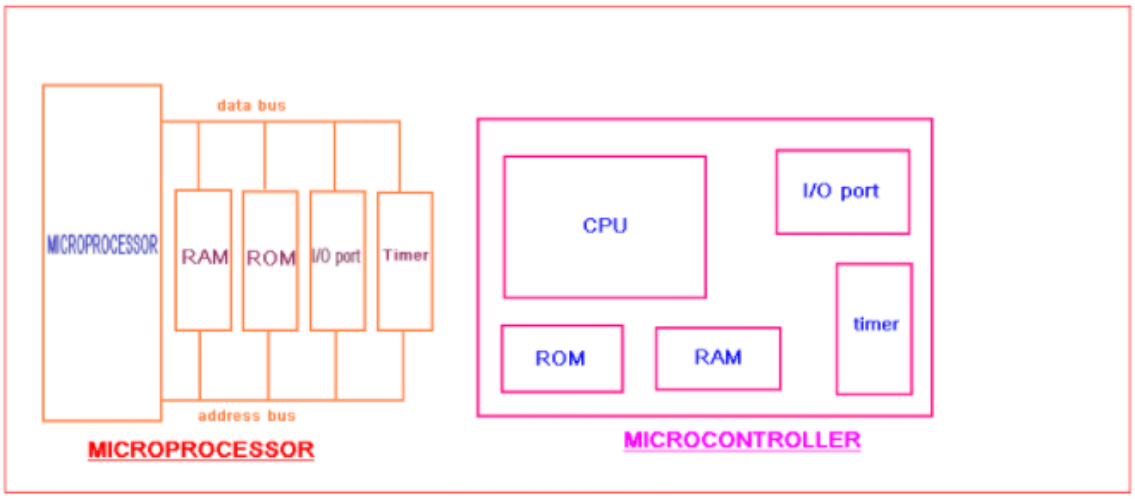

Рисунок 2.7 – Порівняльна схема структур мікропроцесора та мікроконтролера

Розглянемо переваги та недоліки мікронтролера. Мікроконтролер має наступні переваги:

1. Мікроконтролери діють як мікрокомп'ютер без будь-яких цифрових деталей.
2. Оскільки більш висока інтеграція всередині мікроконтролера, що зменшує вартість та розміри системи.
3. Простота використання мікроконтролера, легко усунути несправності та підтримувати систему.
4. Більшість контактів є програмовані користувачем для виконання різних функцій.
5. Легко взаємодіяти з додатковими портами оперативної пам'яті, ПЗУ, вводу-виводу.
6. Малий час, необхідний для виконання операцій.

Далі розглянемо недоліки схеми:

1. Мікроконтролери мають більш складну архітектуру, ніж мікропроцесори.
2. Можливо виконувати лише обмежену кількість операцій одночасно.
3. В основному використовується в мікрообладнанні.
4. Не вдається безпосередньо взаємодіяти з пристроями великої потужності.

У наші дні ви можете знайти мікроконтролери у всіх видах електронних пристроїв. Будь-який пристрій, який вимірює, зберігає, контролює, обчислює або відображає інформацію, повинен мати мікросхему мікроконтролера всередині. Найбільше прикладів використання мікроконтролерів застосовується в автомобільній промисловості (мікроконтролери широко використовуються для управління двигунами та регуляторами потужності в автомобілях). Ви також можете знайти мікроконтролери всередині клавіатур, миші, модемів, принтерів та інших периферійних пристроїв. У тестовому обладнанні мікроконтролери дозволяють легко додавати такі функції, як можливість зберігати вимірювання, створювати та зберігати підпрограми користувачів, а також відображати повідомлення та сигнали. Споживчі товари, що використовують мікроконтролери, включають цифрові відеокамери, оптичні програвачі, рідкокристалічні / світлодіодні дисплеї тощо. І це лише кілька прикладів.

Нижче преведено загальні приклади роботи мікроконтролера:

1. Використовується в біомедичних інструментах.
2. Широко використовується в системах зв'язку.
3. Використовується як периферійний контролер у ПК.
4. Використовується в робототехніці.
5. Використовується в автомобільній сфері.

Перед тим, як вибирати мікроконтролер для продукту, потрібно розробити блок-схему, яка зображуватиме всі різноманітні функції, необхідні для всієї системи. Блок-схему можно назвати попередній дизайн продукту. Треба схематично зобразити що саме потрібно для підключення до мікроконтролера, які протоколи зв’язку будуть застосовані, яка кількість контактів для вводу та виводу, скільки комунікаційних портів необхідно? Зображено приклад блок-схеми попереднього дизайну мікроконтролера (абстрактна модель):

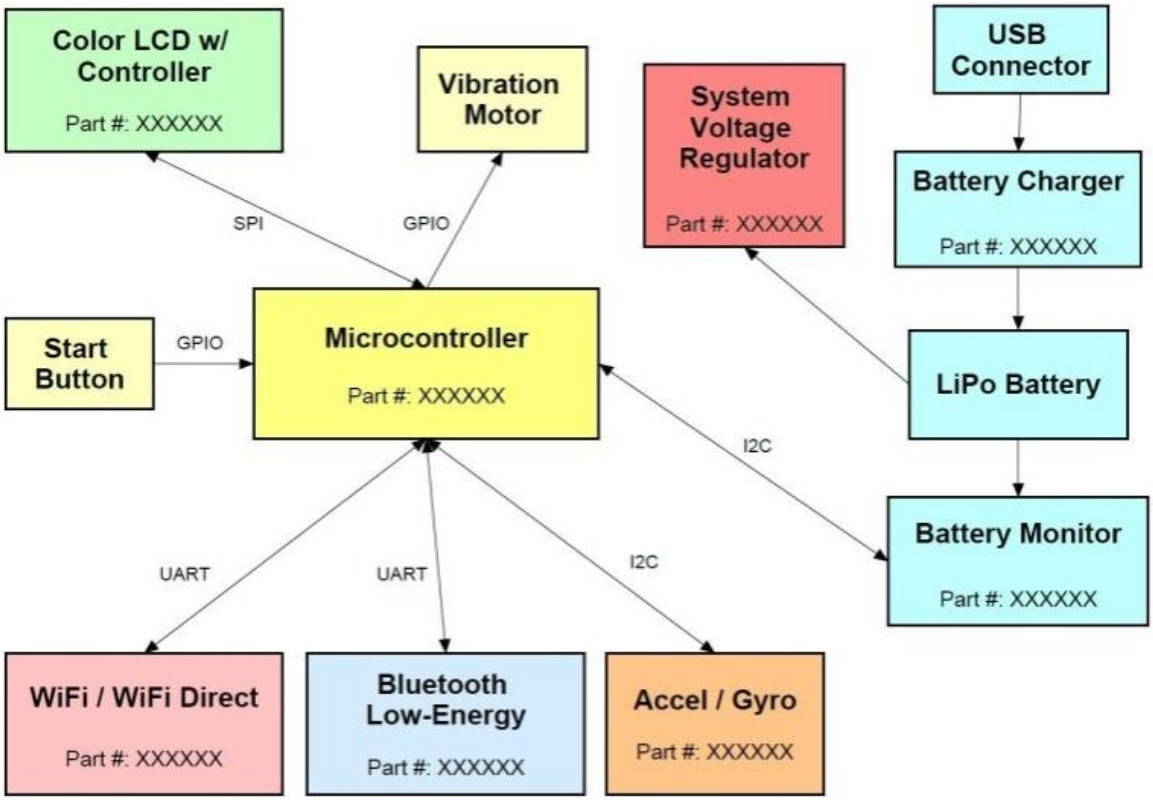

Рисунок 2.8 – Абстрактна модель-блок-схема мікроконтролера

Ця інформація потрібна для чіткого вибору конкретного мікроконтролера під конкретну чітко визначену задачу. У противному випадку, якщо вибрати мікроконтролера, який містить занадто велику продуктивність та велику кількість функцій, які будуть не потрібні, тим самим ускладнити задачу розробки системи. Або з іншого боку, обрати пристрій із надто малим потенціалом для розробки, що в подальшому приведе до тупику в розробці системи.

2.3 Аналіз мікроконтролерів та модулей камер

Розробників все частіше просять додати можливості аналізу зображень та відео до поточних мереж та нових продуктів, особливо для програм безпеки. Розпізнавання обличчя та автомобіля є особливо бажаним і може додати значну

цінність, але вивчення та програмування деталей моделей розпізнавання обличчя для системи на базі мікроконтролера може зайняти багато часу та складно.

Як це часто буває, розробникам потрібно якомога швидше почати експериментувати та розробляти. Тому, перед розробкою системи потрібно обрати комбінацію мікроконтролера та камери для реалізації системи розпізнавання обличчя. Аналіз готових комбінацій допоможе розібратися детальніше у тому, яка схема буде остаточною та найбільш доцільною для данної системи.

Більшість розробників вже знайомі з виявленням обличчя. Це процес аналізу зображення та визначення місця присутності обличчя. Багато смартфонів мають таку можливість. Розпізнавання обличчя – наступний етап після виявлення обличчя. Це процес аналізу обличчя та розробки математичної моделі з використанням даних, отриманих із відносної відстані рис обличчя. Сюди входять відстань між очима та відносне розташування очей до носа, щелепи, вух. Також визначається кут обличчя щодо камери розпізнавання обличчя, оскільки це важливо при визначенні відносних відстаней. Тон шкіри також можна виявити і використовувати для розпізнавання.

Потім цю модель порівнюють із базою даних збережених математичних моделей відомих граней, і алгоритм визначає найкращий збіг. Існує ймовірність (у відсотках), що виявлене обличчя відповідає збереженому. Альтернатива полягає в тому, що не існує збігу, і тому виявлене обличчя невідоме.

Перша комбінація, яка розглянемо – STM32L-DISCOVERY та модуль камери LinkSprite JPEG Color Camera.

Плата STM32L-DISCOVERY – на ній встановлено мікроконтролер STM32L152RBT6 спеціально розроблений для реалізації пристроїв з низьким енергоспоживанням, що згодилося б для реалізації підключення камери від акумулятора. Нижче можна спостерігати дану плату на рисунку.

Рисунок 2.9 – Зображено плату STM32L-DISCOVERY

Модуль LinkSprite JPEG Color Camera, що працює по інтерфейсу UART. Даний модуль має вже готову камеру, видає фотографії в форматі JPEG по UART. Налагодження та управління також здійснюються по цьому інтерфейсу. Приведено характеристики модуля:

1. Підтримується розширення фото: VGA/QVGA/160*120.
2. Швидкість от 9600 до 115200 (за замовчуванням 38400).
3. Живлення 5V.
4. Розмір 32mm X 32mm.
5. Струм 80-100mA.

Нижче зображено модуль камери:

Рисунок 2.10 – Модуль камери LinkSprite JPEG Color

Для підключення даної схеми для ПК потрібен адаптер USB-UART.

Модуль підключення SD карт від LC STUDIO, мікроконтролер буде взаємодіяти з ним по інтерфейсу SPI.

Далі будемо аналізувати мікроконтролер ESP32-CAM.

ESP32-CAM – це недорога плата розробки на основі ESP32 з вбудованою камерою, невелика в розмір. Це ідеальне рішення для застосування IoT, побудови прототипів та саморобних проектів. Плата інтегрує Wi-Fi, традиційний Bluetooth і BLE з низьким енергоспоживанням, з 2 високопродуктивними 32-розрядними процесорами LX6. Він приймає 7-ступінчасту архітектуру трубопроводу, вбудований датчик, Датчик Холла, датчик температури тощо, а також основні діапазони регулювання частоти від 80 МГц до 240 МГц. Повністю сумісний зі стандартами WiFi 802.11b / g / n / e / i та Bluetooth 4.2, його можна використовувати як головний режим для побудови незалежного мережевого контролера або як ведений для іншого хоста MCU для додавання мережевих можливостей до існуючих пристроїв. У даної плати є молодший брат ESP8266. Є порівняльна таблиця даних плат, яка представлена нижче:

Specifications	ESP8266	ESP32
MCU	Xtensa Single-Core 32-bit L 106	Xtensa Dual-Core 32-bit LX6 600 DMIPS
802.11 b/g/n Wi-Fi	Yes, HT20	Yes, HT40
Bluetooth	N/A	Bluetooth 4.2 and below
Typical Frequency	80 MHz	160 MHz
SRAM	160 kBytes	512 kBytes
Flash	SPI Flash up to 16 MBytes	SPI
GPIO	17	36
Hardware / Software PWM	None / 8 Channels	1 / 16 Channels
SPI / I2C / I2S / UART	2/1/2/2	4/2/2/2
ADC	10-bit	12-bit
CAN	N/A	1
Ethernet MAC Interface	N/A	1
Touch Sensor	N/A	Yes
Temperature Sensor	N/A	Yes
Working Temperature	-40° C – 125° C	-40° C – 125° C

Рисунок 2.11 – Порівняльна характеристка ESP32 та ESP8266

Як ми можемо спостерігати, ESP32 на голову вища за молодшу модель, тому для даного проекту може бути розгялунти більш потужнішу варіацію ESP плати.

Щодо модуля камери, то для ESP32-CAM можна використовувати OV2640, OV7670, OV5640.

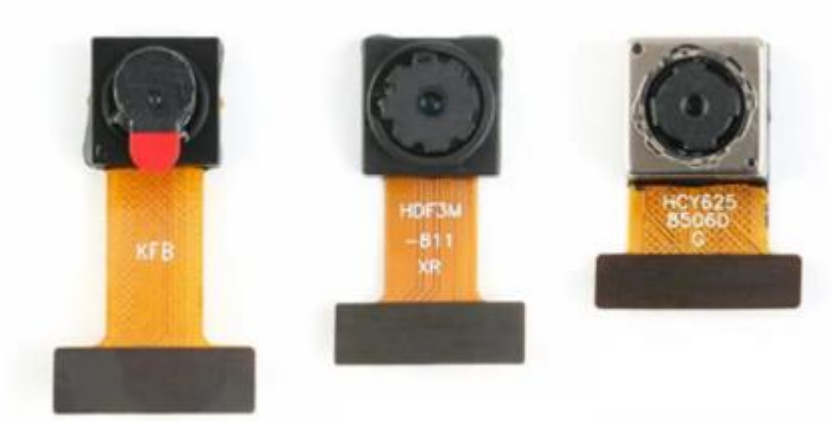

Рисунок 2.12 – Зображено модулі OV7670-OV2640-OV5640

Модуль OV7670:

1. Датчик зображення OV7670, невеликий за розміром і легкий в експлуатації, забезпечує всі функції однієї камери VGA та процесора зображення. За допомогою управління шиною SCCB він може виводити 8-бітові дані зображення з різною роздільною здатністю у цілому кадрі, субодискретизації та вікні.

2. Виріб має зображення VGA до 30 кадрів в секунду. Користувачі мають повний контроль над якістю зображення, форматом даних та передачею. Всі функції обробки зображень, включаючи гамма-криву, баланс білого, насиченість, кольоровість тощо, можна програмувати через інтерфейс SCCB.

3. Датчики зображення використовують унікальну сенсорну технологію для поліпшення якості зображення та зменшення різких і стабільних кольорових зображень за рахунок зменшення або усунення оптичних або електронних дефектів, таких як шум з фіксованим малюнком, хвости і плавання.

Модуль OV2640:

1. OV2640 - це 1/4-дюймовий CMOS UXGA (1632 * 1232) датчик зображення. Невеликі розміри датчика і низька робоча напруга забезпечують всі функції однієї камери UXGA та процесора зображення.

2. За допомогою керування шиною SCCB він може виводити 8/10-бітові дані зображення різної роздільної здатності, такі як повнокадрове зображення, субодискретизація, масштабування та віконне оформлення. Зображення UXGA цього виробу може досягати 15 кадрів в секунду (до 30 кадрів для SVGA та 60 кадрів для CIF). Користувачі мають повний контроль над якістю зображення, форматом даних та передачею.

3. Усі функції обробки зображень, включаючи гамма-криву, баланс білого, контраст, кольоровість тощо, можна програмувати через інтерфейс SCCB. Датчики зображення використовують унікальну технологію датчиків для поліпшення якості зображення та зменшення різких і стабільних кольорових зображень за рахунок зменшення або усунення оптичних або електронних дефектів, таких як шум з фіксованим малюнком, змазування та плавання.

Модуль OV5640:

1. OV5640 - це 1/4-дюймовий CMOS QSXGA (2592 * 1944) датчик зображення, який забезпечує повне рішення піксельної камери 500 Вт із вбудованою функцією автофокусування (AF). Співвідношення ціна / продуктивність.

2. Невеликі розміри датчика та низька робоча напруга забезпечують всі функції однієї камери QSXGA та процесора зображення. За допомогою контролера шини SCCB він може виводити 8/10-бітові дані зображення з різною роздільною здатністю, такі як повнокадрове зображення, субодискретизація, масштабування та віконне оформлення. Зображення QSXGA цього виробу може досягати 15 кадрів в секунду (до 30 кадрів для зображень 1080P, 60 кадрів для зображень 720P та 120 кадрів для роздільної здатності QVGA).

3. Користувачі мають повний контроль над якістю зображення, форматом даних та передачею. Всі функції обробки зображень, включаючи гамма-криву, баланс білого, контраст, кольоровість тощо, можна програмувати через інтерфейс SCCB.

4. Датчики зображення використовують унікальну сенсорну технологію для поліпшення якості зображення та зменшення різких і стабільних кольорових зображень за рахунок зменшення або усунення оптичних або електронних дефектів, таких як шум з фіксованим малюнком, змазування та плавання.

Усі представлені модулі підходять для підключення до ESP32. Залежно відбюджету та задачі можна використовувати будь-який. Опираючись на аналіз даних вище, я вирішив використовувати комбінацію – ESP32-CAM у купі з модулем OV2640. Як ми можемо спостерігати з попередніх даних, дана плата має великий потенціал при розробці ПЗ для розпізнавання обличчя завдяки потужним та ефективним комплектуючим, які містить дана плата. Щодо модуля камери, було прийнято рішення використовувати OV2640, тому що пропорція ціни/якості є найоптимальнішою згідно із конкретними поставленими задачами.

2.4 Аналіз роботи нейронних мереж

Мозок людини є найскладнішим органом в організмі людини. Це допомагає нам думати, розуміти та приймати рішення. Секрет його сили – нейрон.

Ще з 1950-х років вчені намагаються імітувати функціонування нейрона і використовувати його для створення розумніших і кращих роботів. Після численних спроб і помилок люди нарешті створили комп’ютер, який міг розпізнавати людську мову. Лише після 2000 року люди змогли освоїти глибинне навчання (підмножина ШІ), яке могло бачити та розрізняти різні зображення та відео.

Перш ніж детально розглянути, що таке нейронна мережа, потрібно зрозуміти глибинне навчання.

Deep Learning – це підмножина машинного навчання, яка вимагає від комп'ютерів робити те, що природно для людини: вчитися на прикладі.

Машина навчається за зображеннями як прикладами. Цей процес сильно відрізняється від підключення комп'ютерної програми, щоб вона щось розпізнавала і вчилася. Ви не контролюєте, як воно вчиться; ви контролюєте аспекти, які в ньому входять. На основі зображень, поданих раніше, комп'ютер ідентифікує об'єкт.

Вченим вдалося побудувати штучну форму нейрону (біологічну), яка керує будь-якою машиною глибокого навчання.

Після короткого обговорення глибинного навчання перейдемо до питання «Що таке нейронна мережа?»

Щоб зрозуміти, як працює штучний нейрон, ми повинні знати, як працює біологічний нейрон.

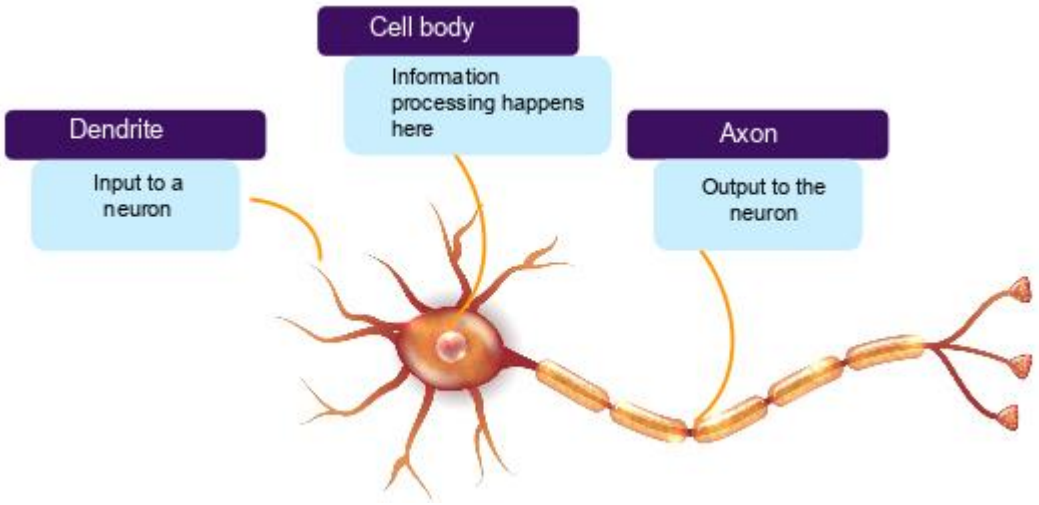

Рисунок 2.13 – Схема людського нейронну

Дендрити: вони отримують інформацію або сигнали від інших нейронів, які до неї підключаються.

Клітинне тіло: обробка інформації відбувається в клітинному тілі. Вони беруть всю інформацію, що надходить з різних дендритів, і обробляють цю інформацію.

Аксон: він надсилає вихідний сигнал іншому нейрону для потоку інформації. Тут кожен з фланців з'єднується з дендритом або волосками на наступному.

На рисунку нижче зображено принцип роботи ANN. Мережа починається з рівня введення, який отримує введення у вигляді даних. Лінії, з'єднані з прихованими шарами, називаються вагами, і вони складаються на прихованих шарах. Кожна точка в прихованому шарі обробляє вхідні дані, і вона поміщає вихідні дані в наступний прихований шар і, нарешті, у вихідний рівень.

Переглядаючи два вищезазначені зображення, ви можете спостерігати, як ANN відтворює біологічний нейрон:

- Вхід до нейрона - вхідний рівень;
- Нейрон - прихований шар;
- Вихід на наступний нейрон - вихідний рівень.

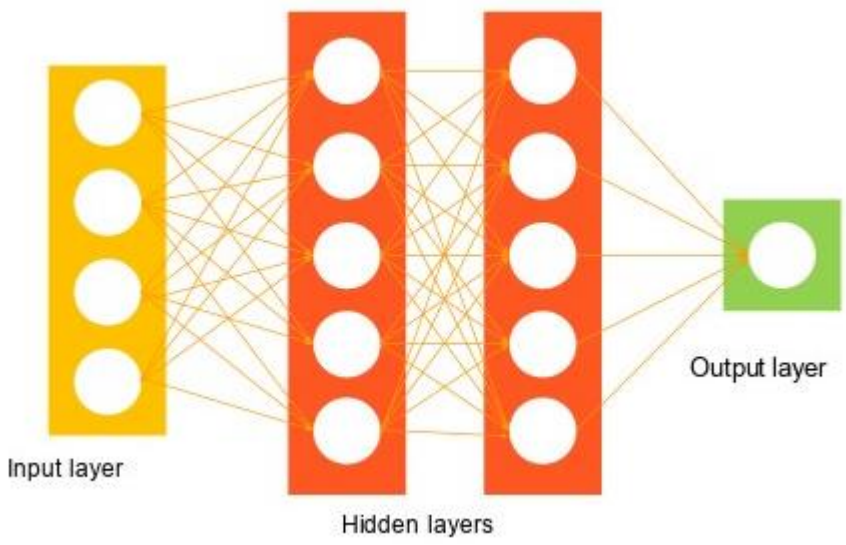

Рисунок 2.14 – Схема роботи нейронної мережі

Нейронна мережа – це система апаратного чи програмного забезпечення, створена за шаблоном після роботи нейронів в мозку людини. Нейронні мережі, які також називають штучними нейронними мережами, є способами досягнення глибинного навчання.

Як насправді працює нейронна мережа?

Ви коли-небудь задавали питання Сірі? Пристрій відповідає точно. Давайте зрозуміємо, як цей віртуальний помічник здійснює розпізнавання мови.

Розглянемо нейронну мережу, показану нижче. У мережі є вхідні, приховані та вихідні рівні. Речення, яке мережа повинна розпізнати, таке: який час? Тут кожне слово входить як зразок звуку. Речення отримує вибірки для дискретних звукових хвиль.

Давайте розглянемо перше слово: Що:

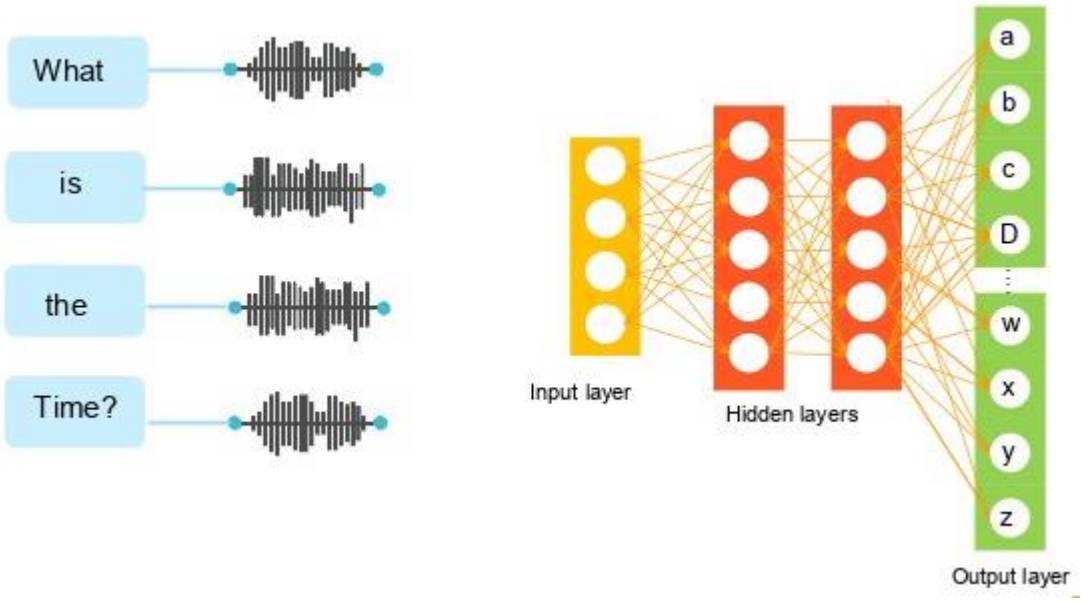

Рисунок 2.15 – Нейронна мережа

Ви можете бачити, що форма сигналу розділена на основі частини для кожної літери. Коли ми аналізуємо букву «W», амплітуда змінюється в залежності від звукової хвилі. Так, на схемі зображено, що кожен сегомент слова, тобто кожна

літера має своє значення амплітуди. Амплітуда сегментів варіюється від 0.5 до 1.9. Так, кожна літера проходить через декілька шарів мережі – початковий, прихований (складається з двох частин) та вихідний шар. Це можна спостерігати на рисунку 2.16:

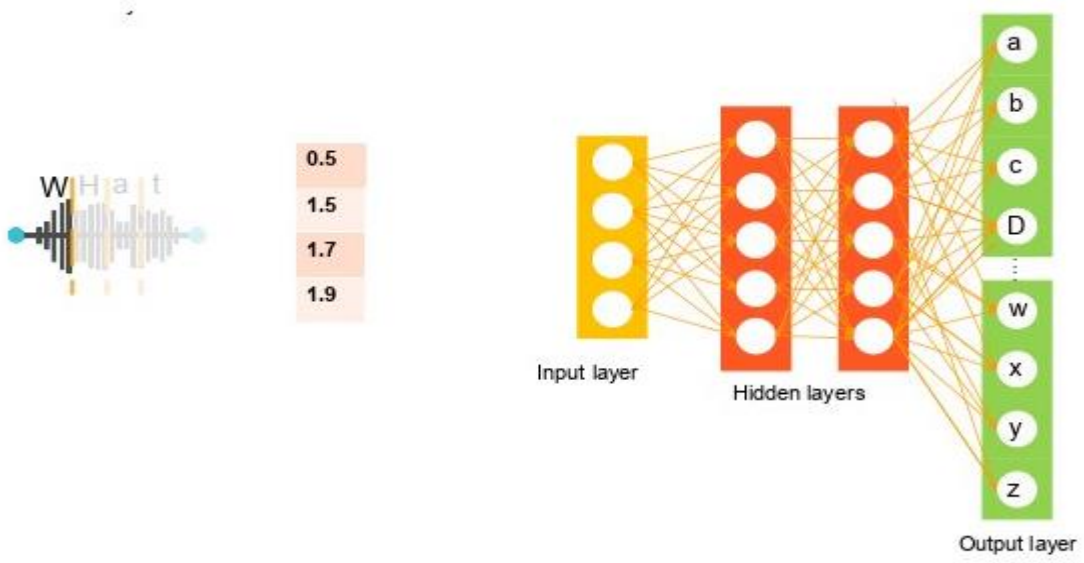

Рисунок 2.16 – Ділення слова на сегменти

Ми збираємо значення з інтервалом і формуємо масив. Для різних літер подаються різні амплітуди, і ми подаємо масив амплітуд на вхідний шар.

Випадкові ваги призначаються кожному взаємозв'язку між вхідним та прихованим шарами. Ми завжди починаємо з випадкового ключа, оскільки присвоєння заданого значення вагам займає значну кількість часу під час навчання моделі.

Ваги множаться з входами, і зміщення додається для формування передавальної функції.

Ваги призначаються взаємозв'язку між прихованими шарами. Вихід передавальної функції подається як вхід для функції активації. Вихід з одного прихованого шару стає входом для наступного прихованого шару.

Акустична модель містить статистичне представлення кожного окремого звуку, що створює слово. Ми починаємо будувати ці акустичні моделі, і оскільки ці

шари розділяють їх, вони почнуть вивчати, що різні моделі представляють для різних букв.

Лексикон містить дані для різних вимов кожного слова. Лексикон знаходиться в кінці, де ми опиняємося ABCD, і він визначає різні букви там.

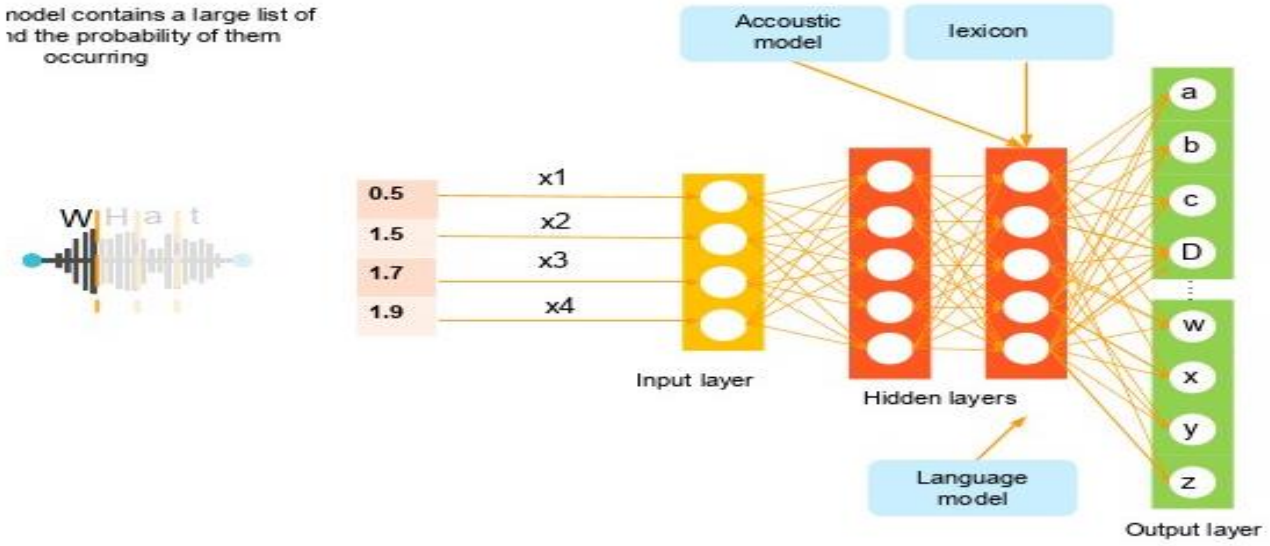

Рисунок 2.17 – Фінальна стадія аналізу

Нарешті, ми отримуємо наш вихідний лист. Слідуючи одному і тому ж процесу для кожного слова та літери, нейронна мережа розпізнає речення, яке ви сказали, або запитання, яке ви задали.

Розглянемо переваги нейронних мереж. Вихідні дані ANN не обмежуються повністю вхідними даними та результатами, наданими їм спочатку експертною системою. Ця здатність стане в нагоді робототехніці та системам розпізнавання образів.

Ця мережа має високу стійкість до несправностей і здатна самостійно налагоджувати або діагностувати мережу. ANN може переглядати тисячі файлів журналів від компанії та сортувати їх. В даний час це втомливе завдання, яке виконують адміністратори.

Нелінійні системи можуть знаходити ярлики для досягнення обчислювально дорогих рішень. Ми бачимо це в банківській справі, де у них є таблиця Excel, а потім вони починають будувати коди навколо цього аркуша. Через 20 років вони можуть створити репертуар усіх цих функцій, і нейронна мережа пропонує ті самі відповіді, зроблені за кілька днів, тижнів або навіть місяця для великого банку.

Завдяки величезній кількості реалізацій додатків щодня, зараз самий підходящий час дізнатись про програми нейронних мереж, машинне навчання та штучний інтелект.

Нейронні мережі використовуються для перетворення рукописних символів у цифрові символи, які машина може розпізнати.

Біржу важко відстежити і важко зрозуміти. Багато факторів впливає на фондовий ринок. Нейронна мережа може вивчити безліч факторів і щодня прогнозувати ціни, що допомогло б біржовим маклерам.

Ідея нейронної мережі стиснення даних полягає у збереженні, шифруванні та відтворенні фактичного зображення заново. Ми можемо оптимізувати розмір наших даних за допомогою нейронних мереж стиснення зображень. Це ідеальний додаток для економії пам’яті та її оптимізації.

Ви повинні знати, що на американській фондовій біржі щодня надходить більше трьох терабайт даних. Це багато даних, щоб перекопати, і вам доведеться їх розібрати, перш ніж почати зосереджуватись хоча б на одному запасі.

У майбутньому нейронні мережі будуть набагато швидшими, і інструменти нейронних мереж можуть бути вбудовані в кожну поверхню дизайну. У нас вже є маленька міні-нейронна мережа, яка підключається до недорогої плати обробки або навіть до вашого ноутбука. Зосередження уваги на апаратному забезпеченні, а не на програмному забезпеченні, зробить пристрої ще швидшими.

Нейронні мережі знайдуть застосування в галузі медицини, сільського господарства, фізики, відкриттів та всього іншого, що ви можете собі уявити. Нейронні мережі також використовуються в спільних системах передачі даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Бібліотека TensorFlow

Машинне навчання – складна дисципліна. Але впровадження моделей машинного навчання набагато менш страшне і складне, ніж було раніше, завдяки механізмам машинного навчання, таким як Google TensorFlow, які полегшують процес отримання даних, навчальних моделей, обслуговування прогнозів та уточнення майбутніх результатів.

Створений командою Google Brain, TensorFlow – це бібліотека з відкритим кодом для чисельних обчислень та масштабного машинного навчання. TensorFlow поєднує в собі безліч моделей машинного навчання та глибокого навчання (він же нейронних мереж) та алгоритмів, і робить їх корисними за допомогою загальної метафори. Він використовує Python, щоб забезпечити зручний інтерфейсний API для побудови додатків з фреймворком, виконуючи ці програми в високопродуктивному C ++.

TensorFlow може навчати та запускати глибокі нейронні мережі для рукописної класифікації цифр, розпізнавання зображень, вбудовування слів, періодичних нейронних мереж, моделей послідовності в послідовність для машинного перекладу, обробки природних мов та моделювання на основі PDE (рівняння часткових диференціальних процесів). Найкраще, TensorFlow підтримує масштабне прогнозування виробництва, використовуючи ті самі моделі, що використовуються для навчання.

Як працює TF? TensorFlow дозволяє розробникам створювати графіки потоку даних – структури, які описують, як дані рухаються через графік або серію вузлів обробки. Кожен вузол на графіку представляє математичну операцію, а кожне з'єднання або ребро між вузлами є багатовимірним масивом даних або тензором.

TensorFlow забезпечує все це для програміста за допомогою мови Python. Python простий у вивченні та роботі з ним, і він пропонує зручні способи висловити, як абстракції високого рівня можна поєднати. Вузли та тензори в TensorFlow – це об’єкти Python, а програми TensorFlow – самі програми Python.

Однак фактичні математичні операції в Python не виконуються. Бібліотеки перетворень, доступні через TensorFlow, записані як високопродуктивні двійкові файли C ++. Python просто спрямовує трафік між фрагментами та забезпечує абстракції програмування високого рівня, щоб з'єднати їх між собою.

Програми TensorFlow можна запускати на будь-якій зручній цілі: на локальній машині, кластері у хмарі, пристроях iOS та Android, центральних процесорах або графічних процесорах. Якщо ви використовуєте власну хмару Google, ви можете запустити TensorFlow на спеціальному кремнію Google TensorFlow Processing Unit (TPU) для подальшого прискорення. Отримані в результаті моделі, створені TensorFlow, можуть бути розгорнуті на більшості будь-яких пристроїв, де вони будуть використовуватися для обслуговування прогнозів.

TensorFlow 2.0, випущений у жовтні 2019 року, багато в чому переробив структуру, базуючись на відгуках користувачів, щоб полегшити роботу (наприклад, за допомогою порівняно простого API Keras для навчання моделей) та підвищив продуктивність. Розподілене навчання легше запускати завдяки новому API, а підтримка TensorFlow Lite дає змогу розгортати моделі на більшій кількості різноманітних платформ. Однак код, написаний для попередніх версій TensorFlow, повинен бути переписаний – іноді лише незначно, іноді суттєво – щоб максимально скористатися новими можливостями TensorFlow 2.0.

Найбільшою перевагою TensorFlow для розвитку машинного навчання є абстракція. Замість того, щоб займатися дрібницею деталей реалізації алгоритмів або з'ясовувати правильні способи приєднання виходу однієї функції до входу іншої, розробник може зосередитись на загальній логіці програми. TensorFlow піклується про деталі за кадром.

TensorFlow пропонує додаткові зручності для розробників, яким потрібно налагодити та проаналізувати програми TensorFlow. Режим енергійного виконання дозволяє оцінювати та модифікувати кожну операцію з графіком окремо та прозоро, замість того, щоб створювати весь графік як єдиний непрозорий об'єкт і оцінювати його всі відразу. Набір візуалізації TensorBoard дозволяє перевіряти та формувати графіки роботи графіків за допомогою інтерактивної веб-панелі інструментів.
TensorFlow також отримує багато переваг завдяки підтримці комерційного спорядження зі списку A у Google. Google не тільки підживив швидкі темпи розвитку за проектом, але створив багато важливих пропозицій навколо TensorFlow, які полегшують розгортання та полегшують використання: вищезгаданий кремній TPU для прискореної роботи в хмарі Google; Інтернет-центр для обміну моделями, створеними за допомогою фреймворку; втілення фреймворку в браузері та для мобільних пристроїв; і набагато більше.

Одне застереження: деякі деталі реалізації TensorFlow ускладнюють отримання повністю детермінованих результатів навчання моделей для деяких навчальних робіт. Іноді модель, навчена в одній системі, буде дещо відрізнятися від моделі, навченої в іншій системі, навіть коли їм подаються однакові дані. Причини цього слизькі - наприклад, як зароджуються випадкові числа і де, або певна недетермінована поведінка при використанні графічних процесорів). Тим не менш, ці проблеми можна обійти, і команда TensorFlow розглядає додаткові засоби контролю, щоб вплинути на детермінованість у робочому процесі.

TensorFlow Lite – це набір інструментів, які допомагають розробникам запускати моделі TensorFlow на мобільних, вбудованих та IoT-пристроях. Це дозволяє зробити висновок машинного навчання на пристрої з низькою затримкою та невеликим двійковим розміром.

TensorFlow Lite складається з двох основних компонентів:

1. Інтерпретатор TensorFlow Lite, який використовує спеціально оптимізовані моделі на багатьох різних типах обладнання, включаючи мобільні телефони, вбудовані пристрої Linux та мікроконтролери.
2. Перетворювач TensorFlow Lite, який перетворює моделі TensorFlow в ефективну форму для використання інтерпретатором, і може запровадити оптимізацію для поліпшення бінарного розміру та продуктивності.

TensorFlow Lite розроблений, щоб спростити машинне навчання на пристроях, "на краю" мережі, замість того, щоб надсилати дані вперед і назад із сервера. Для розробників виконання машинного навчання на пристрої може допомогти покращити:

1. Затримка: немає зворотного шляху до сервера.
2. Конфіденційність: дані не повинні залишати пристрій.
3. Підключення: підключення до Інтернету не потрібно.
4. Споживання енергії: мережеві з’єднання потребують енергії.
5. TensorFlow Lite працює з величезним набором пристроїв, від крихітних мікроконтролерів до потужних мобільних телефонів.

3.2 Hardware та Software складові системи

Для розробки системи виявлення обличчя використовувалися такі інструменти:

Hardware – ESP32-CAM + OV2670 camera, USB-TTL Adapter Waveshare, кабель для виводу додаткового живлення 5 вольт на мікроконтроллер, кабелі інсталювання для плати. Зображено схему підключення мікроконтроллера для подальшої роботи з ним. Як ми можемо спостерігати, для підключення використовуються такі контакти:

- GPIO1 – U0TXD – підключається до Rx UART.
- GPIO3 – U0RXD – підключається до Tx UART.

Важлива деталь, для піключення мікроконтроллера для подальшого програмування його та налаштування потрібно використати два контакти GPIO0 та GND, які потрібно замкнути між собою. Пілся успішної інсталяції скетча потрібно убрати перемичку GPIO0 та GND для того, щоб запустити систему:

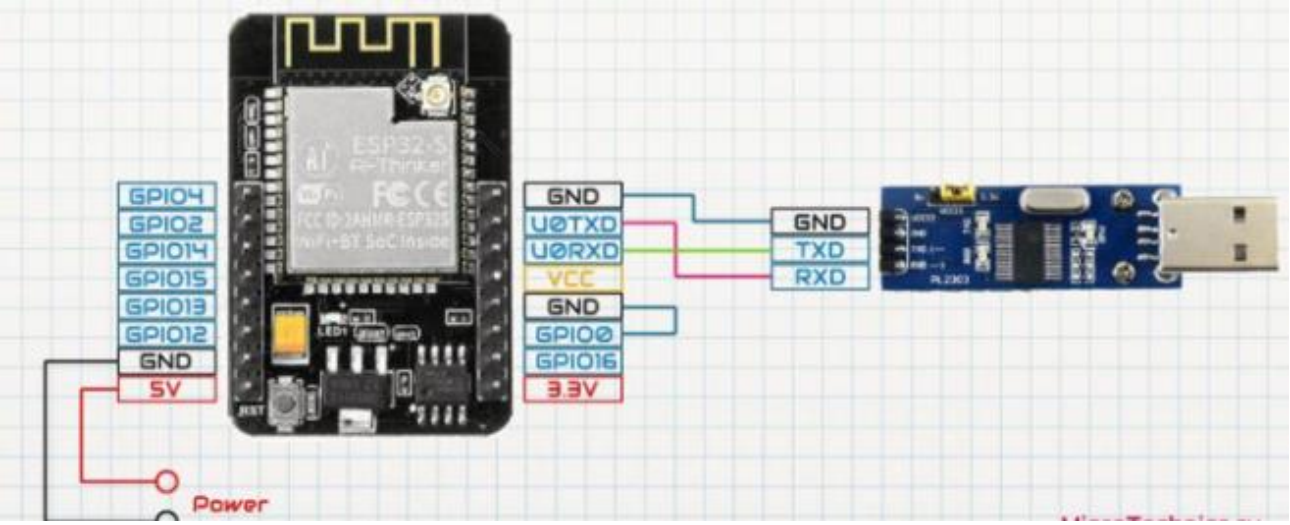

Рисунок 3.1 – Схема підключення ESP32

Також потрібно зазначити, що при підключенні живлення 3.3В спостерігалися велика кількість різних проблем з мікроконтролером, а саме: спостерігалися помилки при завантаженні скетчів, сам модуль не працював потрібним чином через недостатнє живлення. Для уникнення подібних проблем потрібно надавати платі додаткове живлення 5В для подальшої роботи з ним. Але потрібно пам’ятати, що напруга на логічних рівнях на підключеннях не повинна перевищувати 3.3В, для уникнення фізичних пошкоджень плати.

Software – Toolchain для компіляції коду для мікроконтролеру, Build tools (Cmake та Ninja) для розробки системи, ESP-IDF (набор бібліотек та інструментів для програмування мікроконтролеру), python, tensorflow.

Також для реалізації розробки можна використовувати Eclipse та VS Code. Нижче зображено схему розробки систем для мікроконтролеру ESP32 за допомогою зазначених інструментів:

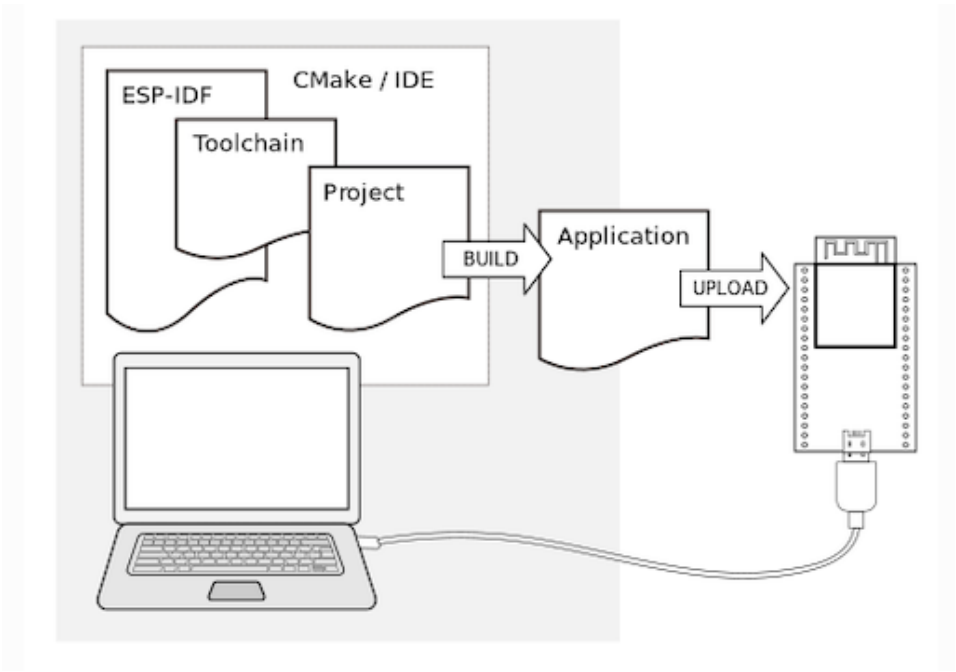

Рисунок 3.2 – Схема розробки системи

Для інсталювання скетчів на мікроконтролер додатково було завантажено та налагоджено WSL (Windows Subsystem for Linux). Підсистема Windows для Linux дозволяє розробникам запускати середовище GNU / Linux – включаючи більшість інструментів командного рядка, утиліти та програми – безпосередньо в Windows, не модифіковану, без накладних витрат на традиційну віртуальну машину або налаштування подвійного завантаження. Після інсталювання підсистеми Убунту на ПК потрібно налагодити інструменти розробки. Перш за все, потрібно завантажити ESP-IDF набір утиліт. Для цього використовувалася команда – git clone --recursive https://github.com/espressif/esp-idf.git. Інструменти завантажимуться в директорію /esp/esp-idf. Потім потрібно інсталювати завантажені утиліти, для цього використовувалися наступні команди – cd ~/esp/esp-idf, ./install.sh. Встановлені

інструменти ще не додані до змінної середовища PATH. Щоб зробити інструменти придатними для використання з командного рядка, необхідно встановити деякі змінні середовища. Дані маніпуляції виконувалися такою командою –
. $HOME/esp/esp-idf/export.sh .

Після вижче зазначених операцій, потрібно конфігурувати проект. Для цього потрібно вибрати каталог, у якому буде створено скетч. Надалі запуск інсталювання буде проходити з данної директорії. Для конфігурування використовувалися наступні команди – idf.py set-target esp32. Встановлення цілі за допомогою idf.py set-target esp32 слід зробити один раз, після відкриття нового проекту. Якщо проект містить деякі існуючі збірки та конфігурацію, вони будуть очищені та ініціалізовані. Ціль може бути збережена у змінній середовища, щоб взагалі пропустити цей крок. Далі потрібно завантажити конфігуратор, за допомогою якого встановлюється «тонке» налагодження параметрів мікроконтолера (частоти процесора, кількість використовуваної пам’яті, підключення до Wi-Fi та інше). Нижче зображено скріншот конфігуратора:

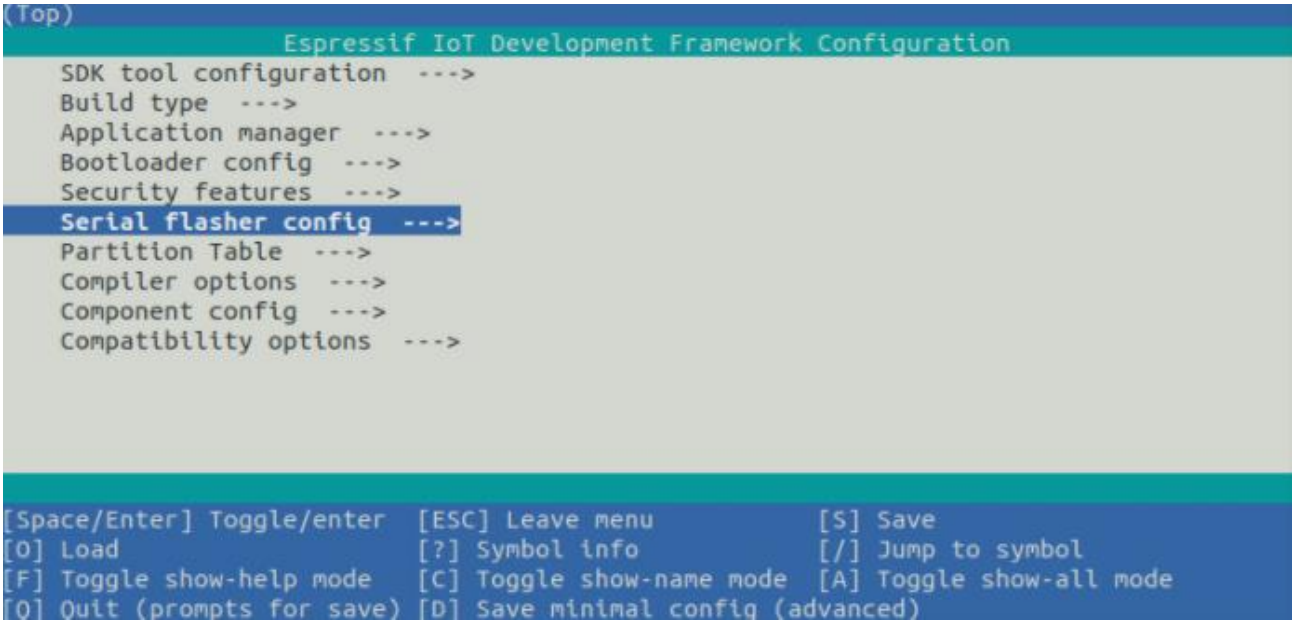

Рисунок 3.3 – Скріншот конфігуратора IDF

Після конфігурування мікроконтролера, потрібно завантажити build проекту. Для даного кроку потрібно використовувати команду – idf.py build. Ця команда скомпілює програму та всі компоненти ESP-IDF, потім генерує завантажувач, таблицю розділів та двійкові файли додатків. Якщо помилок не відбулося, ми матимемо наступний вивід у терміналі:

```
$ idf.py build
Running cmake in directory /path/to/doorbellcam/build
Executing "cmake -G Ninja --warn-uninitialized /path/to/ doorbellcam "...
Warn about uninitialized values.
-- Found Git: /usr/bin/git (found version "2.17.0")
-- Building empty aws_iot component due to configuration
-- Component names: ...
-- Component paths: ...
... (more lines of build system output)
[527/527] Generating doorbellcam
esptool.py v2.3.1
Project build complete. To flash, run this command:
../../../components/esptool_py/esptool/esptool.py -p (PORT) -b 921600 write_flash --flash_mode dio --flash_size detect --flash_freq 40m 0x10000 build/ doorbellcam.bin build 0x1000 build/bootloader/bootloader.bin 0x8000 build/partition_table/partition-table.bin
or run 'idf.py -p PORT flash'
```

Після успішного конфігурування build проекту можна завантажити скетч на мікроконтролер. Для цього використовувалася команда – idf.py -p /dev/ttyS3 -b 921600 flash. Слід зазначити, що параметри –p та –b потрібні для уточнення послідовного порту (порт підключення мікроконтролеру до ПК) та швидкості запису (921600 Бод). Без даних параметрів команда не відпарцює та процес завантаження скетчу буде припинено.

Якщо при застосуванні даної команди спостерігаються такі помилки, як «Не вдалося підключитися», причин може бути кілька. Однією з причин можуть бути проблеми, з якими стикається esptool.py, утиліта, що викликається системою збірки

для скидання мікросхеми, взаємодії із завантажувачем ПЗУ та прошивки флеш-пам'яті. Одне просте рішення, яке можна спробувати, – це ручне скидання. esptool.py автоматично скидає ESP32, затверджуючи лінії керування DTR і RTS мікросхеми USB до послідовного перетворювача, тобто FTDI або CP210x. Лінії управління DTR і RTS, у свою чергу, підключені до штифтів GPIO0 та CHIP_PU (EN) ESP32, таким чином, зміни рівнів напруги DTR і RTS завантажать ESP32 у режим завантаження мікропрограми. Якщо операція пройшла успішно ми отримаємо наступний вивід у терміналі:

```
esptool.py --chip esp32 -p /dev/ttyS3 -b 460800 --before=default_reset --after=hard_reset write_flash --flash_mode dio --flash_freq 40m --flash_size 2MB 0x8000 partition_table/partition-table.bin 0x1000 bootloader/bootloader.bin 0x10000 door_bell_cam.bin
esptool.py v3.0-dev
Serial port /dev/ttyUSB0
Connecting........_
Chip is ESP32D0WDQ6 (revision 0)
Features: WiFi, BT, Dual Core, Coding Scheme None
Crystal is 40MHz
MAC: 24:0a:c4:05:b9:14
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Compressed 3072 bytes to 103...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.0 seconds (effective 5962.8 kbit/s)...
Hash of data verified.
Compressed 26096 bytes to 15408...
Writing at 0x00001000... (100 %)
Wrote 26096 bytes (15408 compressed) at 0x00001000 in 0.4 seconds (effective 546.7 kbit/s)...
Hash of data verified.
Compressed 147104 bytes to 77364...
```

```
Writing at 0x00010000... (20 %)
Writing at 0x00014000... (40 %)
Writing at 0x00018000... (60 %)
Writing at 0x0001c000... (80 %)
Writing at 0x00020000... (100 %)
Wrote 147104 bytes (77364 compressed) at 0x00010000 in 1.9 seconds (effective 615.5 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...
Done
```

Як ми можемо спостерігати, підключення до мікроконтролера та подальше завантаження скетчу пройшло успішно. Це означає, що систему завантажено на мікроконтролер та пристрій готовий до використання. Для перевірки практичної працездатності усієї системи потрібно застосувати команду, яка викликає монітор послідовного порту. Для цього потрібна команда – idf.py –p /dev/ttyS3 monitor, де /dev/ttyS3 – це точна вказівка на послідовний порт, який операційна система призначила нашій платі ESP-32. Для будь-яких манупіляцій із пристроєм у терміналі потрібно пам’ятати, що потрібно завжди вказувати потрібний послідовний порт, у іншому випадку підключитися до мікроконтолера не відбудеться. При зміні фізичного порту USB при підключенні USB-TTL адаптеру потрібно також пам’ятати, що операційна система може змінити послідовний порт для плати, тому при інсталюванні и подальшого використання завжди потрібно перевіряти, який насправді послідовний порт був застосований. Далі представлено інформацію із терміналу монітору порту:

```
$ idf.py -p /dev/ttyUSB0 monitor
Running idf_monitor in directory [...]/esp/door_bell_cam/build
Executing "python [...]/esp-idf/tools/idf_monitor.py -b 115200 [...]/esp/ door_bell_cam /build/ door_bell_cam.elf"...
--- idf_monitor on /dev/ttyS3 115200 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
...
Hello world!
```

```
Restarting in 10 seconds...
This is esp32 chip with 2 CPU core(s), WiFi/BT/BLE, silicon revision 1, 2MB external flash
Minimum free heap size: 298968 bytes
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
```

Як можна спостерігати, підключення відбулося успішно та вивід «Hello World!» для перевірки працездатності можно побачити у лозі терміналу після відпрацювання команди.

3.3 Конфігурування нейронної мережі для розпізнавання обличчя

Розпізнавання обличчя – це проблема комп’ютерного зору, пов’язана з локалізацією одного чи кількох облич на фотографії.

Розміщення обличчя на фотографії означає позначення координати обличчя на зображенні, тоді як локалізація стосується розмежування протяжності обличчя, часто через обмежувальну рамку навколо обличчя.

Виявлення облич на фотографії легко вирішується людьми, хоча це історично складно для комп’ютерів, враховуючи динамічний характер облич. Наприклад, обличчя слід розпізнавати незалежно від орієнтації та кута, на який вони спрямовані, рівня освітленості, одягу, аксесуарів, кольору волосся, волосся на обличчі, макіяжу, віку тощо.

Для розробки повноцінної нейронної мережі, яку можна застосовувати на практиці для розпізнавання людського обличчя, потрібно не тільки розробити працездатний алгоритм нейронних обчислень, а й реалізувати так зване тренування нейронної мережі за допомогою набору даних. У нашому випадку виступатимуть цими даними – фотографії облич людей.

Модель тренування нейронної мережі дуже ресурсомісткий процес для обчислювальної системи, навіть для найпотужніших стаціонарних ПК або цілих

серверів, вже не кажучи за мікроконтролери. Цей процес може займати декілька днів на базі потужної системи. Тому для ефективного розподілу часу та ресурсів при тренуванні моделі використовувалися ресурси хмарного сервісу Google Cloud Deep Learning VM. Google Cloud Platform – це інструмент, наданий Google, який можна використати для створення масштабних рішень. Ця платформа нещодавно набула великої популярності завдяки легкому доступу до графічних процесорів. Крім того, вони також безкоштовно надають вам кредити на суму 300 доларів з терміном дії, який залежно від виду обробки, який вам потрібно зробити, може тривати до року. Тому для тренування нескладної моделі розпізнавання обличчя цей варіант повністю підходить. Після конфігурування машини в хмарній платформі для подальшого користування, потрібно застосувати бібліотеку для побудови моделі TensorFlow.
Keras – рекомендований інтерфейс для побудови моделей у TensorFlow, але коли створювалася модель детектора людини, вона ще не підтримувала всіх необхідних нам функцій. З цієї причини був обрабний старіший інтерфейс tf.slim. Він все ще широко використовується, але застарілий, тому майбутні версії TensorFlow можуть не підтримувати цей підхід.

Для того, щоб навчити модель детектора людини, нам потрібна велика колекція зображень, що маркуються залежно від того, чи є в них люди. Дані про тисячу класів ImageNet, які широко використовуються для навчання класифікаторів зображень, не містять міток для людей, але, на щастя, набір даних COCO містить. Ви також можете завантажити ці дані без реєстрації вручну, і Slim пропонує зручний скрипт для їх автоматичного захоплення:

```
! chmod +x models/research/slim/datasets/download_mscoco.sh
! bash models/research/slim/datasets/download_mscoco.sh coco.
```

Набір даних призначений для використання для навчання моделей локалізації, тому зображення не позначені категоріями "містить особу", "Не містить особу", для яких ми хочемо тренуватися. Натомість кожне зображення постачається зі списком обмежувальних полів для всіх об’єктів, які воно містить. "Особа" - одна з цих

категорій об'єктів, тому, щоб дійти до потрібних міток класифікації, ми повинні шукати зображення з обмежуючими полями для людей. Щоб переконатися, що вони не надто крихітні, щоб їх можна було впізнати, нам також потрібно виключити дуже маленькі обмежувальні рамки. Slim містить сценарій для перетворення обмежувального вікна в мітки:

```
! python models/research/slim/datasets/build_visualwakewords_data.py
--logtostderr \
--train_image_dir=coco/raw-data/train2014 \
--val_image_dir=coco/raw-data/val2014 \
--train_annotations_file=coco/raw-data/annotations/instances_train2014.json \
--val_annotations_file=coco/raw-data/annotations/instances_val2014.json \
--output_dir=coco/processed \
--small_object_area_threshold=0.005 \
--foreground_class_of_interest='person'.
```

Цей процес займає близько двадцяти хвилин. Після завершення, створюється набір TFRecords в coco/processed, що містить позначену інформацію про зображення. Ці дані були створені Aakanksha Chowdhery і відомі як набір даних Visual Wake Words. Він розроблений для порівняльного тестування вбудованого комп’ютерного зору.

Наступний етап – це тренування моделі. Однією з приємних речей використання tf.slim для обробки тренінгу є те, що параметри, які зазвичай потрібно змінити, доступні як аргументи командного рядка, тому ми можемо просто викликати стандартний скрипт train_image_classifier.py для навчання нашої моделі.

```
! python models/research/slim/train_image_classifier.py \
   --train_dir=vww_96_grayscale \
   --dataset_name=visualwakewords \
   --dataset_split_name=train \
   --dataset_dir=coco/processed \
   --model_name=mobilenet_v1_025 \
   --preprocessing_name=mobilenet_v1 \
   --train_image_size=96 \
   --input_grayscale=True \
   --save_summaries_secs=300 \
```

```
--learning_rate=0.045 \
--label_smoothing=0.1 \
--learning_rate_decay_factor=0.98 \
--num_epochs_per_decay=2.5 \
--moving_average_decay=0.9999 \
--batch_size=96 \
--max_number_of_steps=1000000.
```

Даний процес займає пару днів на екземплярі одного графічного процесора v100, щоб виконати всі мільйонні кроки, але є можливість отримати досить точну модель через кілька годин для подальших експерементів з нею. Після завершення тренування моделі, ми можемо спостерігати наступні дані:

```
INFO:tensorflow:Evaluation [406/406]
I0929 22:52:59.936022 140225887045056 evaluation.py:167] Evaluation [406/406]
eval/Accuracy[0.717438412]eval/Recall_5[1].
```

Важливим числом тут є точність. Він показує частку зображень, які були класифіковані правильно, що в цьому випадку становить 72% після перетворення у відсотки. Якщо ви будете слідувати прикладу сценарію, слід очікувати, що повністю навчена модель досягне точності близько 84% за один мільйон кроків і покаже втрату близько 0,4. Після цього настає етап квантування. Квантування є складним та залученим процесом, і це все ще дуже активна область досліджень, тому отримання плаваючого графіку, який ми до цього навчали, і перетворення його у вісім біт займає досить багато коду. Більшість коду готує приклади зображень для подачі у навчену мережу, щоб можна було виміряти діапазони шарів активації при типовому використанні. Ми покладаємось на клас TFLiteConverter, щоб обробляти квантування та перетворення у файл плоского буфера TensorFlow Lite, який нам потрібен для механізму виведення.

```
import tensorflow as tf
import io
import PIL
import numpy as np
def representative_dataset_gen():
```

```
  record_iterator =
tf.python_io.tf_record_iterator(path='coco/processed/val.record-00000-of-00010')
  count = 0
  for string_record in record_iterator:
    example = tf.train.Example()
    example.ParseFromString(string_record)
    image_stream =
io.BytesIO(example.features.feature['image/encoded'].bytes_list.value[0])
    image = PIL.Image.open(image_stream)
    image = image.resize((96, 96))
    image = image.convert('L')
    array = np.array(image)
    array = np.expand_dims(array, axis=2)
    array = np.expand_dims(array, axis=0)
    array = ((array / 127.5) - 1.0).astype(np.float32)
    yield([array])
    count += 1
    if count > 300:
        break
converter =
tf.lite.TFLiteConverter.from_frozen_graph('vww_96_grayscale_frozen.pb',
['input'], ['MobilenetV1/Predictions/Reshape_1'])
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = representative_dataset_gen
tflite_quant_model = converter.convert()
open("vww_96_grayscale_quantized.tflite", "wb").write(tflite_quant_model).
```

Конвертер виписує файл, але більшість вбудованих пристроїв не мають файлової системи. Щоб отримати доступ до серіалізованих даних нашої програми, ми повинні скомпілювати їх у виконуваний файл і зберегти у Flash. Найпростіший спосіб зробити це – перетворити файл у масив даних C.

```
# Install xxd if it is not available
!apt-get -qq install xxd
# Save the file as a C source file
!xxd -i vww_96_grayscale_quantized.tflite > person_detect_model_data.cc.
```

Після цього був створенний файл person_detect_model_data.cc, у якому міститься наша тренована модель для розпізнавання обличчя. Він становить велику частину всієї системи та є найбільш містким щодо часу та складності реалізації.

3.4 ESP32 Camera Driver

ESP32 Camera Driver – дана частина проекту потрібна для роботи із модулем камери на мікроконтролері. Мається на увазі виклик модуля камери для захвату зображення та подальшої роботи зі зображенням. Тобто модуль в інтервалі кількох секунд сканує кадр на предмет обличчя. Якщо обличчя з’являється – потрібно захватити зображення, якщо ні – не виконувати дію.

Також даний драйвер додатково містить деякі інструменти, які дозволяють конвертувати зображення у різні формати, наприклад – BMP або JPEG.

Потрібно пам’ятати деякі важливі речі, при впровадженні даного драйверу до системи, а саме:

1. За винятком випадків, коли у форматі JPEG використовується CIF або нижча роздільна здатність, драйвер вимагає встановлення та активації PSRAM.
2. Використання YUV або RGB сильно напружує мікросхему, оскільки запис у PSRAM не особливо швидкий. У результаті, дані зображення можуть бути відсутніми. Особливо, якщо увімкнено WiFi. Якщо вам потрібні дані RGB, рекомендується захопити JPEG, а потім перетворити на RGB за допомогою fmt2rgb888 або fmt2bmp / frame2bmp.
3. Коли використовується 1 буфер кадру, драйвер буде чекати закінчення поточного кадру (VSYNC) і запустить I2S DMA. Після отримання кадру I2S буде зупинено, а буфер кадрів повернуто до програми. Цей підхід дає

більше контролю над системою, але призводить до більш тривалого часу, щоб отримати кадр.

4. Коли використовуються 2 або більше буферів кадрів, I2S працює в безперервному режимі, і кожен кадр переводиться в чергу, до якої програма може отримати доступ. Цей підхід створює більше навантаження на центральний процесор / пам'ять, але дозволяє подвоїти частоту кадрів.

Один із ефективних способів – пропустити версію реєстру бібліотеки platform.io та зв’язати репозиторій git як підмодуль. (тобто використання коду поза платформою. управління бібліотекою io). У цьому випадку ми встановимо це як підмодуль всередині папки platform.io $ project / lib:

```
cd $project\lib
git submodule add -b master https://github.com/espressif/esp32-camera.git.
```

Потім потрібно додати до platformio.ini наступні дані:

```
build_flags =
  -I../lib/esp32-camera
```

Щодо Kconfig, товін використовується платформою .io menuconfig (доступ до якої виконується: pio run -t menuconfig) для інтерактивного управління різними операторами #ifdef у всьому espidf та підтримуючих бібліотеках (тобто це репо: esp32-camera та arduino-esp32.git). Процес menuconfig генерує файл sdkconfig, який в кінцевому підсумку використовується за кадром процесом espidf compile + build.

Далі наведено екземпляр коду, у якому реалізовано команди для захвату та обробки зображення в кадрі за допомогою інструментів, які містить драйвер ESP32 Camera Driver.

```
esp_err_t camera_init(){
   //power up the camera if PWDN pin is defined
   if(CAM_PIN_PWDN != -1){
      pinMode(CAM_PIN_PWDN, OUTPUT);
      digitalWrite(CAM_PIN_PWDN, LOW);
   }
   //initialize the camera
```

```
    esp_err_t err = esp_camera_init(&camera_config);
    if (err != ESP_OK) {
        ESP_LOGE(TAG, "Camera Init Failed");
        return err;
    }
    return ESP_OK;
}
esp_err_t camera_capture(){
    //acquire a frame
    camera_fb_t * fb = esp_camera_fb_get();
    if (!fb) {
        ESP_LOGE(TAG, "Camera Capture Failed");
        return ESP_FAIL;
    }
    //replace this with your own function
    process_image(fb->width, fb->height, fb->format, fb->buf, fb->len);
    //return the frame buffer back to the driver for reuse
    esp_camera_fb_return(fb);
    return ESP_OK;
}
```

Після того, як Kconfig (або Kconfig.projbuild) запрацює, можна вибрати конфігурації відповідно до наших налаштувань, або бібліотеки камери будуть скомпільовані. Потрібно також видалити каталог .pio / build до появи опцій. Якщо проігнорувати даний крок, модуль камери скомпілюється, але логіка камери всередині бібліотеки буде порожньою, оскільки Kconfig встановлює правильні оператори #ifdef під час процесу збірки для ініціалізації вибраних камер.

ESP32 вже використовується в ряді проектів розумного будинку / підключеного пристрою з різноманітними датчиками та виконавчими механізмами, підключеними до мікроконтролера, щоб відчувати навколишнє середовище та діяти відповідно. З TensorFlow Lite для мікроконтролерів, що виконується на ESP32, це відкриває сценарії для всіх типів випадків використання, які ініціюються локально.

Підсумовуючи виконану роботу по розробці даної системи, можна сказати, що запланований проект був успішно спроектований та реалізований на базі мікроконтролера ESP32-CAM. В ході розробки системи було оптимізовано бібліотеки для імплементування складної нейронної мережі у малопотужну плату. Дана система блескуче порається з поставленими задачами по детекції обличчя людини у кадрі модуля камери та відправки на електрону поштову скриньку власника.

3.5 Практичне застосування системи виявлення обличчя

Після великої кількості пройдених етапів по проектуванню та реалізації системи розпізнавання обличчя, а також розробки та тренування нейронної мережі для даної системи, було успішно завершено роботу, результатом якої є – повністю працездатна, компактна та малобюджетна плата, здатна розпізнавати людське обличчя серед захопленого кадру модуля камери за допомогою нейронної мережі. Далі ми розглянемо роботу системи на пракцтиці.

Перш за все підключення до джерела живлення, напруга якого не перевищує 5В, а також підключення плати до USB-TTL адаптеру. Нижче зображено кадр підлючення системи до ПК:

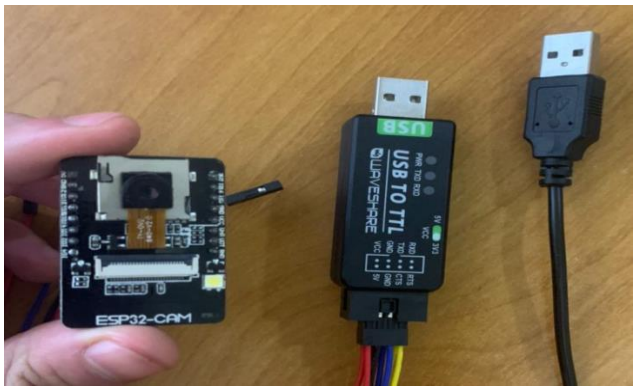

Рисунок 3.4 – Схема підключення мікроконтолеру через адаптер до ПК

Джерелом живлення плати – USB порт на ПК. Адаптер підключено до плати по RX-TX інтерфейсу. Підтяжка GPIO0-GND не потрібна, тому що мікроконтролер вже прошито скетчем, на разі її потрібно відімкнути від плати для подальшого користування системою. Далі безпосередньо демонстрація роботи самої системи.
Після сканування обличчя за допомогою модуля камери, система виконую захват кадру, аналізує його на вміст людського обличчя у ньому (якщо перед модулем немає обличчя – система ігнорує кадр). Система сканує з інтервалом 5 секунд. Після того, як я дав можливість мікроконтролеру виконати захват кадру та відсканувати обличчя, можна спостерігати на моніторі послідовного порту наступну інформацію:

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:9720
ho 0 tail 12 room 4
load:0x40080400,len:6364
entry 0x400806b8

WELLCOME
SPIFFS mounted successfully
Saved file to path: \b§?□
Connecting.
WiFi connected.

Preparing to send email

Connecting to SMTP server...
SMTP server connected, wait for response...
Identification...
Authentication...
Sign in...
Sending Email header...
Sending Email body...
Sending attachments...
/captured_person.jpg
Finalize...
Finished
Email sent successfully
----------------
```

Рисунок 3.5 – Лог терміналу монітору послідовного порту

Як ми можемо спостерігати, підключення до ПК пройшло успішно, живлення є. Внутрішня пам’ять мікроконтролера спрацювала та зберегла кадр для подальшого аналізу. Плата була успішно підключено до мережі інтернет. Подальший етап – авторизація та аутентифікація у поштовій системі для генерації письма з зображенням обличчя. Підключення до поштової скриньки відбулося успішно. Система створює листа з зображенням та наступним текстом – «There is someone close to door». Далі можемо спостергіати як це відбувається на практиці:

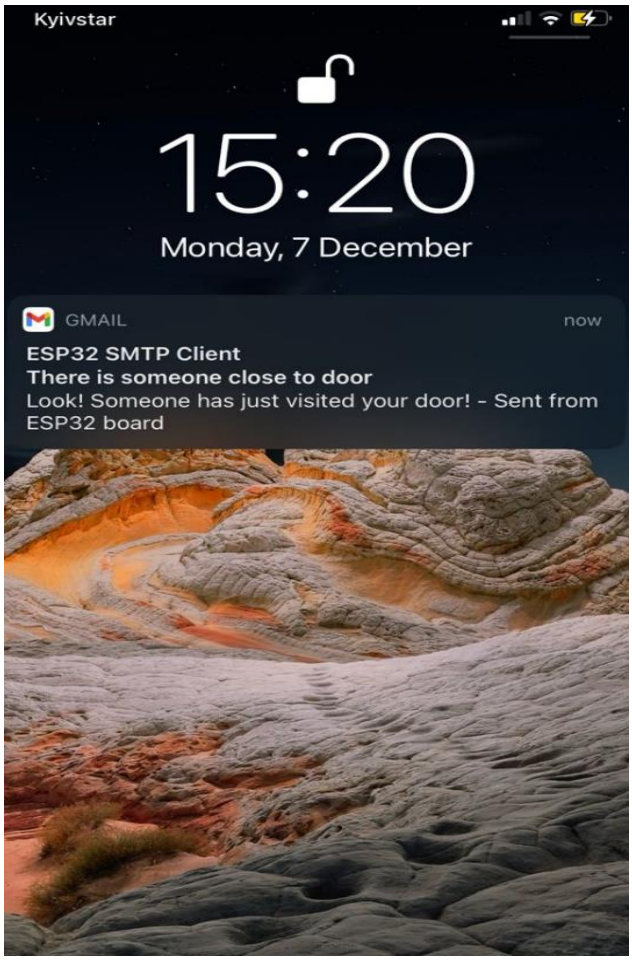

Рисунок 3.6 – Скріншот зі смартфону. Лист від ESP32

Як можемо побачити, я отримав листа від SMTP клієнту. Далі перевіремо, що саме ми побачимо у листі.

Переходимо до поштової скриньки. Перевіряємо вхідні листи, які потрапили останніми. Бачимо, що останнім листом є повідомлення від ESP32 SMTP Cleint. Відкриваємо листа від ESP32.

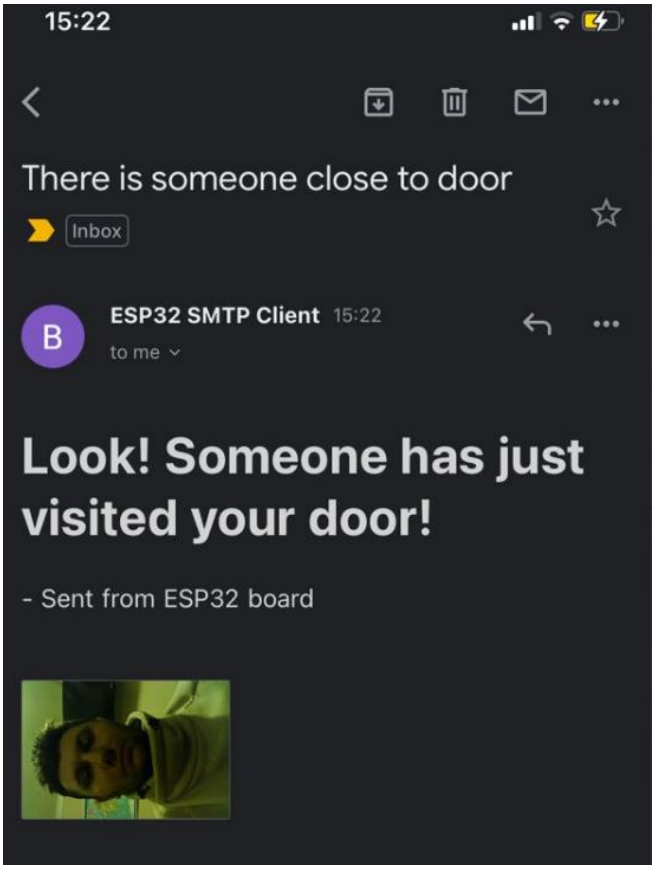

Рисунок 3.7 – Скріншот зі смартфону. Огляд листа від ESP32

Так, насправді, ми отримали листа, у якому застереження, що біля дверей хтось є. Підкріплено зображення із фотографією обличчя. Позначка, що надіслано від плати ESP32.

Отже, підсумовуючи дану демонстрацію, можна зробити висновок. Мікроконтролер у купі зі скетчем успішно відпрацював, а саме було здійснено захват людського обличчя з подальшою відправкою кадра через поштовий сервіс. Задачу виконано на сто відсотків.

ВИСНОВКИ

У ході роботі було досягнуто зменшення необхіодних обчислювальних ресурсів для інтегрування бібліотеки TensorFlow у мікроконтролерах. Для цього виконано наступні дії:

1. Проаналізовано існуючі моделі та методи програмного моделювання, направлені для вирішення подібних задач. Даний аналіз став підгрунттям у виборі методик та платформ для проектування та реалізації даної системи.
2. Обрана існуюча модель, яка найбільше відповідає поставленим задачам по розробці системи.
3. Описано проектування та розробку системи, вибір технічних засобів (Hardware), платформ та інструментів по розробці програмного забезпечення (Software).
4. Зображено процес розробки нейронної мережі для виявлення обличчя людини у кадрі.
5. Розроблено систему для виявлення обличчя людини на базі мікроконтролера за допомогою нейронної мережі на основі оптимізованної бібліотеки TensorFlow.

Дипломна робота має практичну цінність, що кожна людина матиме змогу встановити систему виявлення обличчя з ціллю підвищення безпеки приватної території.

Розроблена система дасть змогу створювати продукти для контролю території, продукти для розумних будинків, інтеграція в системи безпеки, створення автономної системи нагляду за стратегічними об’єктами.

Дана система може бути модефікована для ідентифікації обличчя. Тому в майбутньмо планується модернізувати дану систему для можливості ідентифікувати особистість спираючись на локальну базу даних, заповнено власноруч.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Gao, J., Dolan, B., and Nie, J.-Y. A neural network approach to context-sensitive generation of conversational responses. In Proceedings of NAACL, 2015.
2. Tiedemann, J. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In Nicolov, N., Bontcheva, K., Angelova, G., and Mitkov, R. (eds.), Recent Advances in Natural Language Processing, volume V, pp. 237–248. John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria, ISBN 978 90 272 4825 1, 2009.
3. W. Zhao, et al “Face recognition: a literature survey”, Technical Report CAR-TR-948, University of Maryland, October 2000.
4. Bouchra Abboud, et “Facial expression recognition and synthesis based on an appearance model”, Signal Processing: Image Communication, Vol. 19, Issue. 8, pp723-740, 2004.
5. P. Viola & M.J. Jones “Robust real-time object detection”, Technical Report CRL/2001/01, Cambridge Research Laboratory, USA, February 2001
6. Brunelli R & Poggio T “Face recognition: features versus templates”, IEEE Transaction Pattern Analysis and Machine Intelligence, Vol. 15, No.10, pp1042–1052, 2010.
7. Hochreiter, S. and Schmidhuber, J. Long short-term memory. Neural Computation, 1997.
8. Jean, S., Cho, K., Memisevic, R., and Bengio, Y. On using very large target vocabulary for neural machine translation. CoRR, abs/1412.2007, 2014.
9. Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In NIPS, 2014.
10. Abdenour Hadid, Matti Pietik¨ainen & Timo Ahone “A Discriminative Feature Space for Detecting and Recognizing Face”, Proceedings of the 2011 IEEE

Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, 2011.

11. Elise Arnaud et al “A Robust And Automatic Face Tracker Dedicated To Broadcast Videos, IEEE International Conference On Image Processing, 2005.
12. Zhonglong Zheng, Jie Yang & Yitan Zhu “Face Detection and Recognition using Colour Sequential Images”, Journal of Research and Practice in Information Technology, Vol. 38, No. 2, pp.135-149, May 2006.
13. Zuo F. and P.H.N. de With “Automatic Human Face Detection for a Distributed Video Security System”, Proceedings of the Progress workshop on Embedded Systems, pp269–274, Oct. 2010.
14. Bernd Menser & Michael Brunig “Face Detection and Tracking for Video Coding
15. Application”, Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 2004.
16. Pedro Alexandre Dias Martin “Active Appearance Models for Facial Expression, Recognition and Monocular Head Pose Estimation”, master thesis, Dept. of Electrical and Computer Eng., Faculty of Sciences and Technology, University of Coimbra, 2011.
17. Hjelmas and Low “Face Detection: A Survey,” Computer Vision and Image Understanding, vol. 83, pp236-274, doi:10.1006/cviu.2001.0921, 2012.
    [Електронний ресурс]: Режим доступу: http://www.idealibrary.com.
18. Jain A, Ross A & Prabhakar S “An introduction to biometric recognition”, IEEE Transactions on Circuits and Systems for Video Technology, Vol.14, No.1, pp4–20, Jan. 2005.
19. Henry Rowley, Baluja S. & Kanade T. “Neural Network-Based Face Detection, Computer Vision and Pattern Recognition”, Neural Network-Based Face Detection, Pitts-burgh, Carnegie Mellon University, PhD thesis, 2007.

ДОДАТОК А

Наведено найголовніші програмні модулі системи стисло.

Detection responder:

```
#include "tensorflow/lite/micro/examples/doorbell_camera/detection_responder.h"
#include "tensorflow/lite/micro/examples/doorbell_camera/image_provider.h"
#include "smtp_client.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "esp_log.h"
#include "img_converters.h"
/* min time (in ms) between two consecuting email notifications */
#define HOLD_TIME 5000
static int64_t elapsed_time = 0;
uint8_t *jpeg_image = NULL;
size_t jpeg_img_size = 0;
camera_fb_t* camera_fb;
static const char *TAG = "tf_responder";
void RespondToDetection(tflite::ErrorReporter* error_reporter,
                        uint8_t person_score, uint8_t no_person_score) {
  if (elapsed_time == 0) {
    elapsed_time = esp_timer_get_time();
  }
  if (person_score > 240) {
    if (((esp_timer_get_time() - elapsed_time)/1000) >= HOLD_TIME) {
      camera_fb = (camera_fb_t*)image_provider_get_camera_fb();
      TF_LITE_REPORT_ERROR(error_reporter, "person detected");
      free(jpeg_image);
      bool ret = frame2jpg(camera_fb, 80,  &jpeg_image, &jpeg_img_size);
      if (ret != true) {
        TF_LITE_REPORT_ERROR(error_reporter,"jpeg compression failed");
      }
      esp_err_t esp_ret = smtp_client_send_email(jpeg_image, jpeg_img_size);
      if (esp_ret != ESP_OK) {
        ESP_LOGE(TAG, "Failed to send the email, returned %02X", esp_ret);
      }
      elapsed_time = 0;
    }
```

```
  }
}
```

DoorBellTest:

```
#include "tensorflow/lite/c/common.h"
#include "tensorflow/lite/micro/examples/doorbell_camera/model_settings.h"
#include "tensorflow/lite/micro/examples/doorbell_camera/no_person_image_data.h"
#include "tensorflow/lite/micro/examples/doorbell_camera/person_detect_model_data.h"
#include "tensorflow/lite/micro/examples/doorbell_camera/person_image_data.h"
#include "tensorflow/lite/micro/kernels/micro_ops.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/micro/micro_mutable_op_resolver.h"
#include "tensorflow/lite/micro/micro_optional_debug_tools.h"
#include "tensorflow/lite/micro/testing/micro_test.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"
constexpr int tensor_arena_size = 93 * 1024;
uint8_t tensor_arena[tensor_arena_size];
TF_LITE_MICRO_TESTS_BEGIN
TF_LITE_MICRO_TEST(TestInvoke) {
  tflite::MicroErrorReporter micro_error_reporter;
  tflite::ErrorReporter* error_reporter = &micro_error_reporter;
  const tflite::Model* model = ::tflite::GetModel(g_person_detect_model_data);
  if (model->version() != TFLITE_SCHEMA_VERSION) {
    TF_LITE_REPORT_ERROR(error_reporter,
                         "Model provided is schema version %d not equal "
                         "to supported version %d.\n",
                         model->version(), TFLITE_SCHEMA_VERSION);
  }
  PrintModelData(model, error_reporter);
  tflite::MicroMutableOpResolver<3> micro_op_resolver;
  micro_op_resolver.AddBuiltin(
      tflite::BuiltinOperator_DEPTHWISE_CONV_2D,
      tflite::ops::micro::Register_DEPTHWISE_CONV_2D());
  micro_op_resolver.AddBuiltin(tflite::BuiltinOperator_CONV_2D,
                               tflite::ops::micro::Register_CONV_2D());
  micro_op_resolver.AddBuiltin(tflite::BuiltinOperator_AVERAGE_POOL_2D,
                               tflite::ops::micro::Register_AVERAGE_POOL_2D());
  tflite::MicroInterpreter interpreter(model, micro_op_resolver, tensor_arena,
```

```
                                        tensor_arena_size, error_reporter);
interpreter.AllocateTensors();
TfLiteTensor* input = interpreter.input(0);
TF_LITE_MICRO_EXPECT_NE(nullptr, input);
TF_LITE_MICRO_EXPECT_EQ(4, input->dims->size);
TF_LITE_MICRO_EXPECT_EQ(1, input->dims->data[0]);
TF_LITE_MICRO_EXPECT_EQ(kNumRows, input->dims->data[1]);
TF_LITE_MICRO_EXPECT_EQ(kNumCols, input->dims->data[2]);
TF_LITE_MICRO_EXPECT_EQ(kNumChannels, input->dims->data[3]);
TF_LITE_MICRO_EXPECT_EQ(kTfLiteUInt8, input->type);
const uint8_t* person_data = g_person_data;
for (int i = 0; i < input->bytes; ++i) {
  input->data.uint8[i] = person_data[i];
}
TfLiteStatus invoke_status = interpreter.Invoke();
if (invoke_status != kTfLiteOk) {
  TF_LITE_REPORT_ERROR(error_reporter, "Invoke failed\n");
}
TF_LITE_MICRO_EXPECT_EQ(kTfLiteOk, invoke_status);
TfLiteTensor* output = interpreter.output(0);
TF_LITE_MICRO_EXPECT_EQ(4, output->dims->size);
TF_LITE_MICRO_EXPECT_EQ(1, output->dims->data[0]);
TF_LITE_MICRO_EXPECT_EQ(1, output->dims->data[1]);
TF_LITE_MICRO_EXPECT_EQ(1, output->dims->data[2]);
TF_LITE_MICRO_EXPECT_EQ(kCategoryCount, output->dims->data[3]);
TF_LITE_MICRO_EXPECT_EQ(kTfLiteUInt8, output->type);
uint8_t person_score = output->data.uint8[kPersonIndex];
uint8_t no_person_score = output->data.uint8[kNotAPersonIndex];
TF_LITE_REPORT_ERROR(error_reporter,
                     "person data.  person score: %d, no person score: %d\n",
                     person_score, no_person_score);
TF_LITE_MICRO_EXPECT_GT(person_score, no_person_score);
const uint8_t* no_person_data = g_no_person_data;
for (int i = 0; i < input->bytes; ++i) {
  input->data.uint8[i] = no_person_data[i];
}
invoke_status = interpreter.Invoke();
if (invoke_status != kTfLiteOk) {
  TF_LITE_REPORT_ERROR(error_reporter, "Invoke failed\n");
}
```

```
  TF_LITE_MICRO_EXPECT_EQ(kTfLiteOk, invoke_status);
  output = interpreter.output(0);
  TF_LITE_MICRO_EXPECT_EQ(4, output->dims->size);
  TF_LITE_MICRO_EXPECT_EQ(1, output->dims->data[0]);
  TF_LITE_MICRO_EXPECT_EQ(1, output->dims->data[1]);
  TF_LITE_MICRO_EXPECT_EQ(1, output->dims->data[2]);
  TF_LITE_MICRO_EXPECT_EQ(kCategoryCount, output->dims->data[3]);
  TF_LITE_MICRO_EXPECT_EQ(kTfLiteUInt8, output->type);
  person_score = output->data.uint8[kPersonIndex];
  no_person_score = output->data.uint8[kNotAPersonIndex];
  TF_LITE_REPORT_ERROR(
      error_reporter,
      "no person data.  person score: %d, no person score: %d\n", person_score,
      no_person_score);
  TF_LITE_MICRO_EXPECT_GT(no_person_score, person_score);
  TF_LITE_REPORT_ERROR(error_reporter, "Ran successfully\n");
}
TF_LITE_MICRO_TESTS_END
```

ImageProvider:

```
#include "tensorflow/lite/micro/examples/doorbell_camera/image_provider.h"
#include "tensorflow/lite/micro/examples/doorbell_camera/model_settings.h"
TfLiteStatus GetImage(tflite::ErrorReporter* error_reporter, int image_width,
                      int image_height, int channels, uint8_t* image_data) {
  for (int i = 0; i < image_width * image_height * channels; ++i) {
    image_data[i] = 0; }
  return kTfLiteOk; }
```

PersonDetectModel:

```
#include "tensorflow/lite/micro/examples/doorbell_camera/person_detect_model_data.h"
#ifdef __has_attribute
#define HAVE_ATTRIBUTE(x) __has_attribute(x) #else
#define HAVE_ATTRIBUTE(x) 0 #endif
#if HAVE_ATTRIBUTE(aligned) || (defined(__GNUC__) && !defined(__clang__))
#define DATA_ALIGN_ATTRIBUTE __attribute__((aligned(4)))
#else
#define DATA_ALIGN_ATTRIBUTE #endif
const unsigned char g_person_detect_model_data[] DATA_ALIGN_ATTRIBUTE = {
  0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x0e, 0x00, 0x18, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00,
  0x10, 0x00, 0x14, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00…};
const int g_person_detect_model_data_len = 236072;
```