Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ МЕТОДАМИ ШТУЧНОГО ІНТЕЛЕКТУ НА ПЛАТФОРМАХ ARDUINO

IMAGE RECOGNITION BY ARTIFICIAL INTELLIGENCE METHODS ON ARDUINO PLATFORMS

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІс-21
Сальнік Дмитро Русланович

(підпис)

Керівник:
к.т.н., доцент
Мельник Катерина Вікторівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 06 » червня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т. Терлецький
« 10 » 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Сальніку Дмитру Руслановичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Розпізнавання зображень методами штучного інтелекту на платформах Arduino*

Керівник роботи *к.т.н., доцент Мельник Катерина Вікторівна*

затверджені наказом закладу вищої освіти від «04» січня 2025 року № 11/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 10.06.2025р.

3. Вихідні дані до роботи *джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз проблематики розпізнавання зображень
Аналіз методів та технологій для розпізнавання зображень
Проєктування та розробка системи
Тестування системи
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Існуючі рішення: системи розпізнавання зображень на основі мікроконтролерів
Використані технології: алгоритми обробки зображень на основі штучного інтелекту
Архітектура системи: схема взаємодії між мікроконтролером Arduino, камерою ESP32 CAM та серверною частиною для обробки зображень
Інтерфейс системи: графічний інтерфейс користувача

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Теоретичні основи розпізнавання зображень за допомогою штучного інтелекту на основі Arduino	*Мельник К.В., доцент*		
Аналіз методів та технологій для розпізнавання зображень	*Мельник К.В., доцент*		
Розробка та тестування системи розпізнавання зображень на основі Arduino	*Мельник К.В., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,ст.викладач*		

7. Дата видачі завдання 10.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	до 10.02.2025 р.	Виконано
2.	*Аналіз проблеми за темою роботи та постановка завдань дослідження*	до 02.03.2025 р.	Виконано
3.	*Теоретичне дослідження та практична реалізація.*	до 02.04.2025 р.	Виконано
4.	*Висновки та пропозиції*	до 10.04.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 15.04.2025 р.	Виконано
6.	*Формування додатків*	до 02.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 10.05.2025 р.	Виконано
8.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	до 15.05.2025 р.	Виконано
9.	*Нормоконтроль*	до 30.05.2025 р.	Виконано
10.	*Інструментальна перевірка на академічний плагіат*	до 03.06.2025 р	Виконано
11.	*Представлення кваліфікаційної роботи бакалавра до захисту*	до 10.06.2025 р.	Виконано

Здобувач вищої освіти _____ Сальнік Д.Р.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Мельник К.В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Сальнік Д.Р. Розпізнавання зображень методами штучного інтелекту на платформах Adruino. Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків.

Перший розділ присвячено теоретичним основам розпізнавання зображень. У ньому розглядаються основи обробки зображень, принципи роботи штучного інтелекту в задачах комп'ютерного зору, а також особливості використання платформи Arduino для цих цілей.

Другий розділ містить аналіз методів та технологій, що застосовуються в розпізнаванні зображень. Розглядаються класичні підходи, такі як SIFT, SURF, ORB, а також методи на основі підтримки векторних машин (SVM). Окремо аналізується використання нейронних згорткових мереж (CNN).

Третій розділ присвячено розробці та тестуванню системи розпізнавання зображень на основі Arduino. У цьому розділі здійснено проєктування апаратної частини системи, розробку програмного забезпечення з використанням бібліотек YOLO, OpenCV, NumPy.

Ключові слова: Arduino, ESP32, CNN, нейрона мережа, кадр, розпізнавання зображення, YOLO, OpenCV.

ANNOTATION

Salnik D. Inage recognition by artifical intelligence methods on arduino platforms. Bachelor's thesis of the educational program "Computer Engineering", specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

Qualification work consists of an introduction, three chapters, conclusions, a list of references and appendices.

The first chapter is devoted to the theoretical foundations of image recognition. It discusses the basics of image processing, the principles of artificial intelligence in computer vision tasks, and the features of using the Arduino platform for these purposes.

The second chapter provides an analysis of methods and technologies used in image recognition. Classical approaches such as SIFT, SURF, ORB are considered, as well as methods based on support vector machines (SVM). The use of convolutional neural networks (CNN) is also analyzed.

The third chapter is dedicated to the development and testing of an image recognition system based on Arduino. This chapter covers the design of the hardware part of the system, as well as the development of software using the YOLO, OpenCV, and NumPy libraries.

Keywords: Arduino, ESP32, CNN, neural network, frame, image recognition, YOLO, OpenC

ЗМІСТ

ВСТУП

Актуальність теми. У останні роки системи розпізнавання зображень знайшли широке застосування в таких сферах, як автоматизація, безпека, медицина та робототехніка. Використання штучного інтелекту для обробки та аналізу зображень є напрямом, що активно розвивається. Розпізнавання об'єктів на зображеннях стало можливим завдяки розвитку мікроконтролерів і сенсорних технологій. Застосування таких систем в умовах обмежених ресурсів, зокрема на платформі Arduino, дозволяє створювати компактні рішення для різних завдань.

Метою роботи є дослідження і розробка системи розпізнавання зображень за допомогою штучного інтелекту на основі платформи Arduino, а також інтеграція з камерою ESP32-CAM для збору зображень і подальшого їх аналізу.

Об'єкт дослідження –розпізнавання та виявлення об'єктів на зображенні.

Предмет дослідження – система розпізнавання об'єктів на зображенні на основі Arduino та методів штучного інтелекту.

Завдання, які необхідно виконати:

– дослідити можливості використання платформи Arduino для систем розпізнавання зображень;

– спроектувати систему, що включає інтеграцію ESP32-CAM з Arduino для збору зображень;

– розробити алгоритм для мікроконтролеру для отримання зображень і взаємодії з клієнтською частиною системи;

– розробити алгоритм обробки зображень за допомогою штучного інтелекту.

У процесі виконання роботи буде реалізована система, здатна розпізнавати об'єкти на зображеннях, отриманих через камеру ESP32-CAM. Для цього буде використано штучне інтелектуальне середовище для обробки зображень. Технічне рішення, що включає інтеграцію мікроконтролера Arduino та камери ESP32-CAM, дозволить створити компактну систему для розпізнавання та ідентифікації об'єктів у реальному часі.

Дослідження включатиме аналіз існуючих технологій та методів розпізнавання зображень, а також їх адаптацію для обмежених ресурсів мікроконтролерів. У результаті розробки буде створена робоча система, здатна виконувати завдання, що стосуються ідентифікації об'єктів в умовах реального часу, зокрема з використанням обмежених апаратних ресурсів.

Робота включатиме вивчення бібліотек для роботи з камерою ESP32-CAM та розпізнавання зображень на основі алгоритмів штучного інтелекту, таких як YOLO. Застосування цих бібліотек дозволить зібрати зображення та обробити їх з метою подальшого аналізу.

У рамках роботи передбачено тестування працездатності прототипу в умовах обмежених апаратних ресурсів. Очікується, що результати підтвердять можливість використання розробленої системи для розпізнавання об'єктів у реальному часі. Такий пристрій має потенціал застосування у сфері автоматизованого моніторингу, контролю доступу та обробки відеопотоків. Запропонований підхід орієнтований на реалізацію системи розпізнавання зображень із використанням доступних та бюджетних апаратних засобів.

Розробка таких рішень є актуальною в умовах зростаючого попиту на компактні та енергоефективні інтелектуальні системи. Застосування недорогих платформ відкриває нові можливості для впровадження технологій комп'ютерного зору у побутові, промислові та безпекові системи. У роботі також розглядаються особливості оптимізації алгоритмів під обмежені обчислювальні ресурси.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ НА ОСНОВІ ARDUINO

1.1 Основи розпізнавання зображень

Розпізнавання зображень є процесом обробки та аналізу візуальних даних для ідентифікації об'єктів, патернів або характеристик. Цей напрямок знаходиться в області комп'ютерного зору, що використовує алгоритми для обробки графічної інформації. В основі таких методів лежать математичні та статистичні моделі, що дозволяють виконувати класифікацію та сегментацію зображень.

Обробка зображень включає кілька етапів. На першому етапі здійснюється оцифрування, що передбачає перетворення аналогового сигналу в цифровий формат. Далі застосовуються методи фільтрації для зменшення шуму та підвищення якості вихідних даних. Наступним етапом є виділення ознак, що може включати виявлення контурів, кутів, текстурних характеристик або кольорових складових.

Зображення представляється у вигляді двовимірного масиву, де кожен елемент містить інформацію про інтенсивність або колірні характеристики пікселя. В залежності від типу зображення використовується градація яскравості для монохромних даних або багатоканальні представлення для кольорових зображень. При обробці зображень можуть застосовуватися алгоритми гістограмної нормалізації, підвищення контрастності або адаптивної бінаризації [29].

Як видно з рисунку 1.1, процес обробки зображень передбачає проходження через кілька етапів трансформації, що дозволяє отримати структуровану інформацію для подальшої обробки.

Ці трансформації відіграють ключову роль у підготовці зображень для розпізнавання за допомогою нейронних мереж. Завдяки цьому забезпечується висока точність і надійність виявлення об'єктів у складних умовах зйомки.

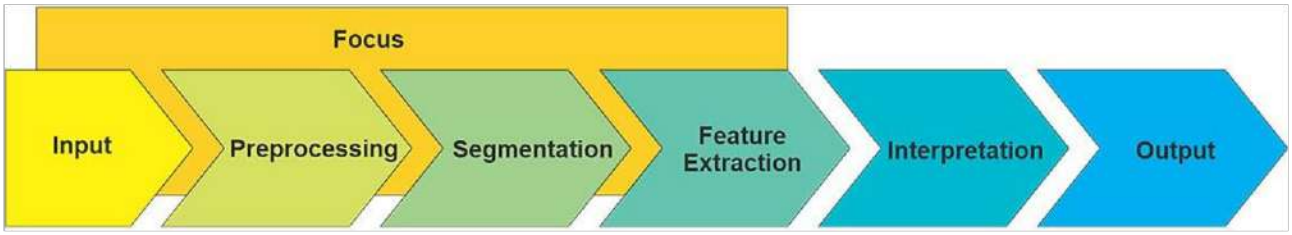

Рисунок 1.1 – Процес обробки зображення [24]

Початковим етапом є Input, що передбачає отримання вхідного зображення з камери, датчика або іншого джерела. Далі здійснюється Preprocessing – попередня обробка, яка може включати нормалізацію яскравості, видалення шумів та покращення контрастності.

Наступним етапом є Segmentation, що передбачає розділення зображення на окремі області або об'єкти для подальшого аналізу. Це може виконуватися за допомогою методів порогової обробки, кластеризації або морфологічних операцій. Після сегментації здійснюється Feature Extraction – виділення ознак, які є характерними для об'єктів на зображенні. Це можуть бути контури, текстурні властивості, колірні характеристики або просторові співвідношення.

Процеси Input, Preprocessing, Segmentation та Feature Extraction взято в блок Focus, що означає його основну роль у підготовці вхідних даних для подальшої інтерпретації.

Наступний етап – Interpretation, під час якого алгоритми класифікації або нейронні мережі аналізують виділені ознаки та виконують розпізнавання об'єктів. Завершується обробка етапом Output, що передбачає виведення результату у вигляді міток, координат об'єктів або прийнятих рішень.

У процесі аналізу зображень використовуються методи класичного комп'ютерного зору, що базуються на застосуванні алгоритмів, а також методи, засновані на штучних нейронних мережах. Перший підхід включає традиційні методи, такі як порогова сегментація, морфологічні операції та аналіз градієнтів. Другий підхід передбачає використання багаторівневих моделей для автоматичного навчання ознак та класифікації.

Методи, засновані на глибокому навчанні, включають згорткові нейронні мережі, що показують високу точність при обробці візуальних даних. Такі мережі містять шари згортки, активації та об'єднання, що дозволяє автоматично знаходити характерні ознаки зображення.

Одним із підходів до розпізнавання є використання попередньо навчених моделей, що можуть бути адаптовані до нових задач за допомогою донавчання. Це дозволяє зменшити вимоги до обчислювальних ресурсів та часу тренування.

Застосування алгоритмів розпізнавання залежить від специфіки задачі. Наприклад, для детектування об'єктів використовуються моделі, що поєднують регресійні та класифікаційні підходи. Однією з таких моделей є YOLO (You Only Look Once), яка дозволяє обробляти зображення в реальному часі, розбиваючи його на сітку та визначаючи координати об'єктів [32].

У випадку роботи на платформі Arduino використовуються оптимізовані моделі, оскільки обчислювальні можливості таких пристроїв обмежені. Для роботи з нейронними мережами на мікроконтролерах застосовуються спеціалізовані бібліотеки, такі як TensorFlow Lite або OpenCV, що дозволяють запускати спрощені версії моделей [17].

Розпізнавання зображень має застосування у різних сферах, таких як безпека, медицина, робототехніка та автоматизація. Наприклад, у системах відеоспостереження використовуються алгоритми виявлення та ідентифікації об'єктів, що дозволяє виконувати аналіз у реальному часі.

1.2 Штучний інтелект в задачах комп'ютерного зору

Штучний інтелект у задачах комп'ютерного зору застосовується для автоматизованого аналізу, розпізнавання та класифікації візуальних даних. Основою таких систем є математичні моделі, що дозволяють здійснювати обробку зображень та приймати рішення на основі отриманих результатів.

Нейронні мережі є однією з основних технологій, що використовуються у штучному інтелекті для комп'ютерного зору. Вони складаються з набору

взаємопов'язаних нейронів, організованих у кілька шарів. Кожен нейрон отримує вхідні дані, виконує певну математичну операцію і передає результат наступному рівню [33].

Процес навчання нейронної мережі базується на використанні великої кількості зображень із відомими мітками. Під час тренування модель коригує вагові коефіцієнти нейронів, щоб мінімізувати похибку при класифікації об'єктів. Чим більший і якісніший набір навчальних даних, тим точнішим стає алгоритм.

Як видно з рисунку 1.2, структура нейронної мережі складається з вхідного шару, одного або кількох прихованих шарів і вихідного шару. Вхідний шар приймає зображення або його числове представлення. Приховані шари виконують операції над цими даними, застосовуючи нелінійні перетворення, а вихідний шар формує кінцевий результат [28].

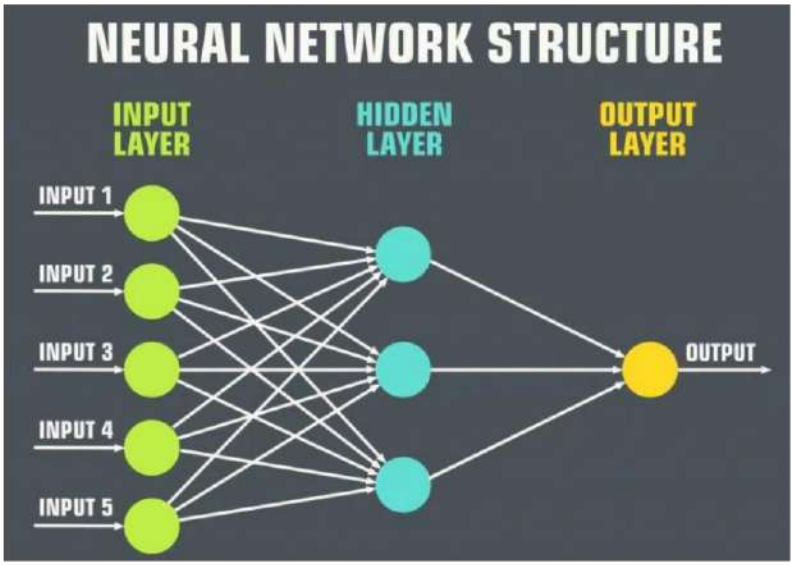

Рисунок 1.2 – Структура нейронної мережі [10]

Одним із поширених типів моделей для аналізу зображень є згорткові нейронні мережі. Вони використовують згорткові шари для виділення ознак, які є стійкими до змін освітлення, масштабу або ракурсу. Завдяки цьому нейронна мережа автоматично формує набір ознак без необхідності явного програмування критеріїв розпізнавання.

На етапі передбачення (інференсу) мережа отримує нове зображення та виконує послідовний аналіз у прихованих шарах. В результаті формується ймовірнісний розподіл класів, до яких може належати об’єкт. Модель вибирає клас із найбільшою ймовірністю як остаточний результат.

Для покращення роботи нейронних мереж застосовуються методи регуляризації, такі як нормалізація ваг, дропаут та використання великих навчальних вибірок. Це дозволяє зменшити та підвищити узагальнюючу здатність моделі.

Одним із викликів у застосуванні штучного інтелекту для комп’ютерного зору є оптимізація обчислювальних витрат. Глибокі нейронні мережі вимагають значних ресурсів для тренування та інференсу. Для вирішення цієї проблеми використовуються спрощені моделі, квантовані ваги та апаратні прискорювачі.

Окрім класифікації, штучний інтелект застосовується у задачах детекції об’єктів та сегментації зображень. У детекції модель визначає координати об’єкта на зображенні, а в сегментації – розбиває зображення на області, які відповідають різним класам.

У комп’ютерному зорі використовуються попередньо навчені моделі, що дозволяють скоротити час навчання та покращити точність. Ці моделі можуть адаптуватися до нових задач за допомогою донавчання на специфічних наборах даних.

Процес прийняття рішень у штучному інтелекті базується на аналізі ознак, отриманих у ході обробки. Окрім традиційних алгоритмів, можуть використовуватися ансамблеві методи, що поєднують кілька моделей для підвищення точності розпізнавання [26].

Розвиток штучного інтелекту в комп’ютерному зорі спрямований на підвищення автономності систем, що працюють у реальному часі. Оптимізація алгоритмів дозволяє використовувати нейронні мережі навіть на пристроях з обмеженими обчислювальними ресурсами.

Таким чином, штучний інтелект у задачах комп’ютерного зору базується на принципах автоматичного аналізу зображень, використання багаторівневих

моделей та адаптації до нових умов роботи. Застосування нейронних мереж дозволяє автоматизувати процес розпізнавання та підвищити точність обробки візуальних даних.

1.3 Особливості використання платформи Arduino для розпізнавання зображень

Платформа Arduino широко використовується для реалізації систем автоматизації та обробки даних, включаючи задачі комп'ютерного зору. Завдяки доступності, відкритому програмному середовищу та підтримці численних периферійних пристроїв, мікроконтролери Arduino можуть застосовуватися для базового розпізнавання зображень у вбудованих системах.

Обмежені обчислювальні ресурси стандартних мікроконтролерів Arduino ускладнюють реалізацію глибоких нейронних мереж без використання додаткових обчислювальних пристроїв. Для роботи з зображеннями зазвичай застосовуються модулі на основі більш продуктивних процесорів, таких як ESP32-CAM або окремі співпроцесори для прискорення обчислень [30].

Основним завданням системи комп'ютерного зору на базі Arduino є отримання, попередня обробка та передача зображень для подальшого аналізу. Як видно з рисунку 1.3, мікроконтролер може використовувати камеру, зчитувати вхідні дані, виконувати базові операції фільтрації та передавати інформацію на інші пристрої для обробки.

Такі системи можуть бути застосовані для простих задач, наприклад, виявлення руху, розпізнавання кольору або відстеження об'єктів. У поєднанні з бездротовими модулями можливе створення автономних пристроїв для моніторингу навколишнього середовища. Подальше вдосконалення апаратної бази дозволяє розширити функціональність таких систем і наблизити їх до більш складних рішень у сфері штучного інтелекту.

Завдяки цьому платформа Arduino залишається популярним вибором для прототипування недорогих і енергоефективних систем комп'ютерного зору.

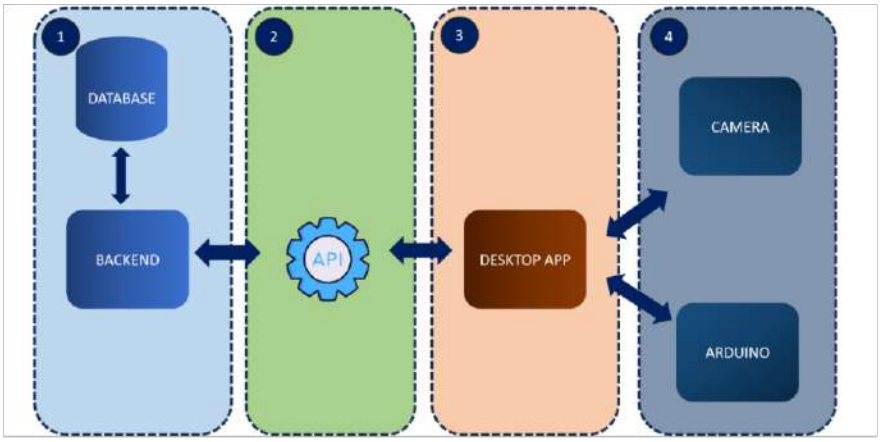

Рисунок 1.3 – Архітектура системи розпізнавання зображень на основі Arduino [4]

Для підключення камери до платформи Arduino використовуються спеціалізовані модулі, такі як OV7670 або ESP32-CAM. Вони дозволяють отримувати зображення з певною роздільною здатністю та передавати його через серійний інтерфейс або бездротове з'єднання для подальшого аналізу.

Обробка зображень безпосередньо на мікроконтролері зазвичай обмежується базовими операціями, такими як перетворення кольорового простору, згладжування або порогова сегментація. Через обмеження оперативної пам'яті виконання складних згорткових нейронних мереж на стандартних платах Arduino є неможливим, тому розпізнавання часто виконується на зовнішніх серверах або додаткових модулях обчислення.

Для передавання зображень та отримання результатів класифікації мікроконтролери можуть використовувати бездротові технології, зокрема Wi-Fi або Bluetooth. Наприклад, використання ESP32-CAM дозволяє передавати відеопотік через HTTP або MQTT-протоколи для подальшої обробки на сервері.

У випадках, коли обробка виконується безпосередньо на пристрої, застосовуються оптимізовані моделі, адаптовані для виконання на обмежених обчислювальних ресурсах. Наприклад, використання TensorFlow Lite дозволяє запускати спрощені нейронні мережі на мікроконтролерах з мінімальним споживанням пам'яті [23].

Платформа Arduino також підтримує використання додаткових периферійних пристроїв, таких як датчики руху, світлові індикатори та сервоприводи. Це дозволяє створювати інтерактивні системи, які реагують на розпізнані об’єкти та виконують відповідні дії.

Застосування Arduino для комп’ютерного зору часто обмежене низькою тактовою частотою та невеликим обсягом пам’яті. Проте завдяки поєднанню мікроконтролера з більш продуктивними обчислювальними пристроями можлива реалізація складніших алгоритмів розпізнавання.

Розвиток вбудованих систем на базі Arduino відкриває можливості для створення компактних та енергоефективних пристроїв, що здатні виконувати базову обробку зображень у режимі реального часу. Вдосконалення апаратних можливостей мікроконтролерів та програмних бібліотек сприяє розширенню сфер їхнього застосування у задачах комп’ютерного зору.

Так, у відкритому доступі існують проєкти систем комп’ютерного зору на базі Arduino, орієнтовані на розв’язання конкретних прикладних задач.

Наприклад, проєкт на GitHub від користувача MAzewai [1] демонструє реалізацію системи виявлення об’єктів з використанням Arduino Uno, ультразвукового сенсора та сервоприводу. В основі лежить застосування методів машинного навчання – ансамблевих алгоритмів, зокрема Random Forest та Decision Tree.

Навчання моделі відбувається на зовнішньому комп’ютері із використанням Python та бібліотеки Scikit-learn. Для збору навчальних даних сервомотор здійснює поворот датчика у межах 0–180 градусів, при цьому проводиться вимірювання відстані до об’єктів за допомогою ультразвукового сенсора. Отримані дані передаються на комп’ютер, де формуються навчальні вибірки. Побудована модель експортується у форматі, сумісному з Arduino, шляхом генерації відповідного заголовкового файлу (model.h), що містить код моделі на C++ з використанням бібліотеки EloquentTinyML. У мікроконтролер інтегрується модуль класифікації, який працює у реальному часі, приймаючи

нові вимірювання як вектор ознак та передаючи їх на вхід функції predict(), що повертає клас присутності або відсутності об’єкта.

Залежно від результату класифікації, сервопривід може змінювати положення або здійснювати інші дії. Проєкт містить повний цикл: генерацію даних, навчання моделі, експорт моделі до заголовкового файлу, інтеграцію до прошивки Arduino та демонстрацію роботи системи.

Інший приклад – демонстраційний проєкт «Arduino AI Fun With TensorFlow Lite» від Trevor Lee [12]. У цій реалізації застосовано концепцію розподіленої обробки даних. Arduino UNO або ESP32 виконує роль пристрою управління і комунікації, тоді як основні обчислювальні задачі пов’язані з комп’ютерним зором делегуються на мобільний пристрій Android. Комунікація між мікроконтролером та смартфоном здійснюється через бібліотеку DumbDisplay, яка забезпечує передачу команд, зображень та результатів обробки.

Arduino ініціює завантаження зображення з мережевого ресурсу на мобільний застосунок, після чого DumbDisplay виконує обробку зображення за допомогою моделі TensorFlow Lite. Модель детектує об’єкти на зображенні, результати передаються назад на Arduino та відображаються графічно на екрані мобільного пристрою.

Таким чином реалізується взаємодія, де Arduino виступає як клієнт, а мобільний застосунок – як виконавець моделей глибокого навчання. ідентифікатором BT32.

РОЗДІЛ 2
АНАЛІЗ МЕТОДІВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

2.1 Огляд методів розпізнавання зображень на базі штучного інтелекту

Класичні методи комп'ютерного зору являють собою набір алгоритмів, які використовуються для обробки та аналізу зображень без застосування глибоких нейронних мереж. Вони базуються на математичних моделях та операціях, які дозволяють виділяти ознаки зображення, такі як контури, текстури, кольори тощо. Ці методи здебільшого використовують стандартні підходи до обробки зображень, які застосовуються на етапах попередньої обробки, сегментації та класифікації.

Одним із найбільш популярних класичних методів є порогова сегментація, яка полягає у виділенні пікселів зображення, що відповідають певному порогу яскравості чи кольору.

Цей підхід підходить для завдань, де об'єкти на зображенні чітко контрастують із фоном, але в складних умовах освітлення або при наявності шуму його точність може знижуватися.

Наступним методом є аналіз контурів. За допомогою цього методу можна виділяти межі об'єктів на зображенні. Поширеними алгоритмами для цього є метод Собеля, що виявляє горизонтальні й вертикальні градієнти, та метод Кенні, який гарантує точніше виявлення країв і демонструє стійкість до шуму. Обидва алгоритми дозволяють отримувати контури, що використовуються в подальшому для більш детального аналізу [9].

Іншим класичним методом є морфологічні операції, які дозволяють проводити аналіз форми об'єктів. Серед основних операцій можна виділити ерозію та дилатацію, які дозволяють змінювати форму об'єктів на зображенні для покращення якості сегментації. Ці операції використовуються для видалення шумів, заповнення прогалин або виділення специфічних об'єктів.

Для виділення ознак зображення застосовуються алгоритми, такі як SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features) та ORB (Oriented FAST and Rotated BRIEF). Вони дозволяють знаходити ключові точки на зображенні, які використовуються для порівняння та ідентифікації об'єктів, незалежно від змін масштабу та орієнтації [31].

Одним із методів, який застосовується для класифікації об'єктів, є метод опорних векторів (SVM) [20]. Принцип роботи цього методу проілюстровано на рисунку 2.1.

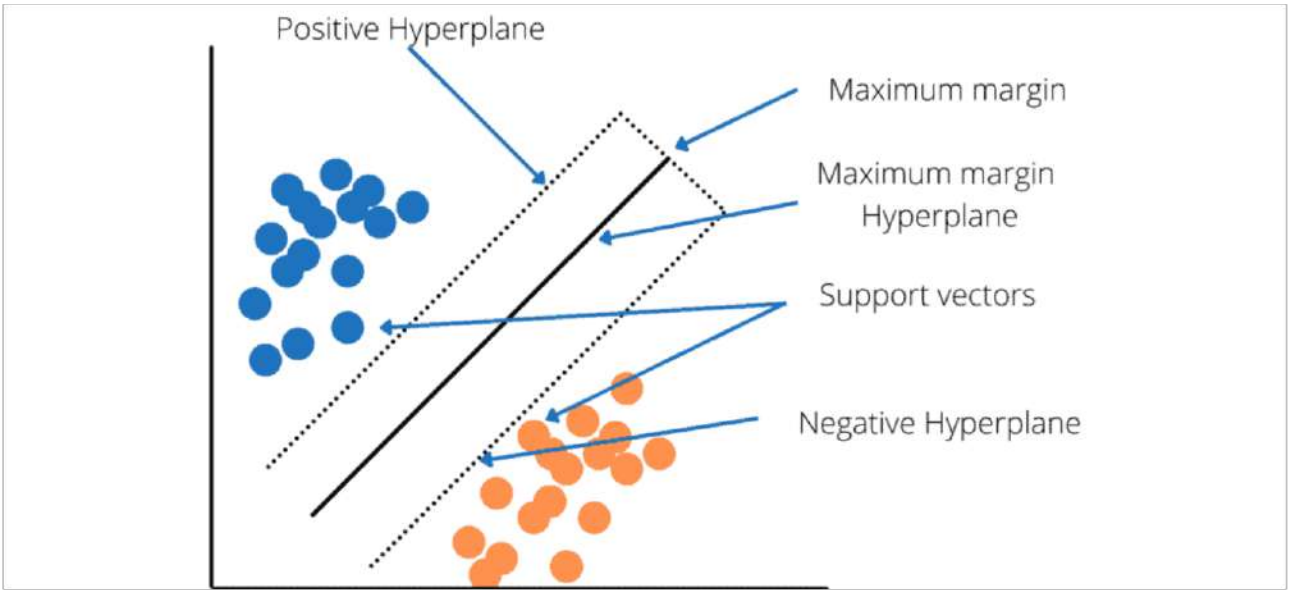

Рисунок 2.1 – Ілюстрація гіперплощини SVM [14]

Метод опорних векторів використовує побудову гіперплощини, яка максимально відокремлює різні класи об'єктів на основі виділених ознак. Support Vectors – це точки даних, які знаходяться найближче до гіперплощини. Ці точки визначають лінію розмежування, обчислюючи маржі, і мають велике значення для побудови класифікатора.

Гіперплощина є планом прийняття рішень, який розділяє об'єкти, що належать до різних класів.

Margin – це відстань між двома лініями, що проходять через найтісніше розташовані точки кожного класу. Вона обчислюється як перпендикулярна

відстань від лінії до опорних векторів або найближчих точок. Широкий марж між класами є бажаним, тоді як вузький марж може свідчити про менш точне розмежування класів.

Класичні методи комп'ютерного зору мають низькі вимоги до обчислювальних ресурсів, що дозволяє використовувати їх у реальному часі. Однак ці методи мають обмеження в точності, особливо при обробці складних зображень з наявністю шумів або високої варіативності. У таких ситуаціях їхня продуктивність може знижуватися.

Хоча класичні методи досі використовуються для базових завдань комп'ютерного зору, їх застосування в обробці складних зображень є обмеженим. Часом їх використовують на ранніх етапах обробки зображень перед тим, як переходити до більш складних підходів, заснованих на штучному інтелекті та глибоких нейронних мережах.

Згорткові нейронні мережі (CNN) є однією з основних архітектур, що використовуються для розпізнавання зображень [6].

Згорткові нейронні мережі здатні автоматично виділяти релевантні ознаки зображень, що значно підвищує точність розпізнавання, особливо у випадках складної структури даних. На відміну від класичних алгоритмів, CNN не потребують ручного проектування ознак, що робить їх гнучкішими і ефективнішими у різноманітних умовах. Крім того, такі мережі демонструють високу стійкість до шумів і змін освітлення, що є критично важливим у практичних застосуваннях.

Процес роботи глибоких нейронних мереж з зображеннями можна побачити на рисунку 2.2, що ілюструє основні етапи обробки зображень за допомогою згорткових нейронних мереж.

У сучасних дослідженнях CNN активно застосовуються для розпізнавання облич, медичної діагностики, автономного керування транспортом та інших сфер. Їхня здатність до навчання на великих обсягах даних дозволяє досягати результатів, близьких до рівня людського сприйняття. Подальший розвиток

апаратного забезпечення сприятиме ще ширшому впровадженню глибокого навчання у повсякденні технології.

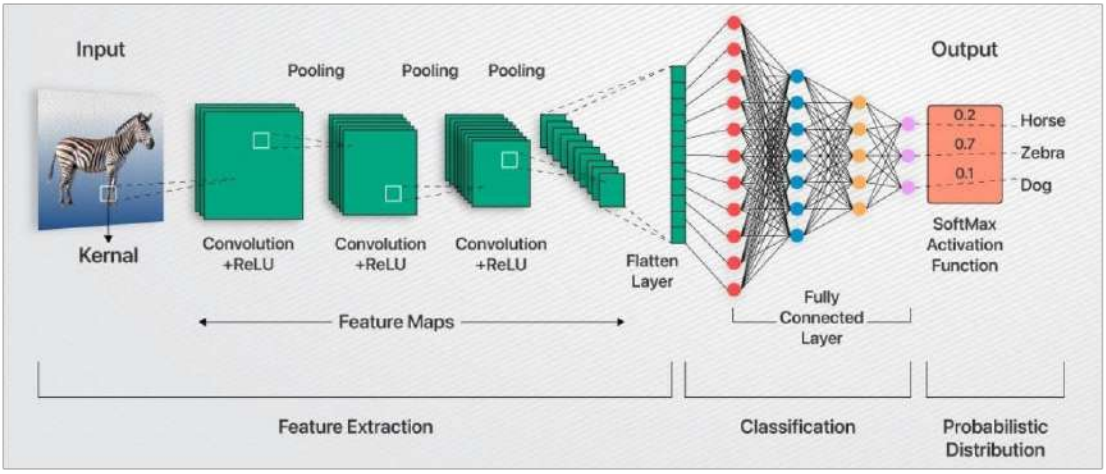

Рисунок 2.2 – Архітектура згорткової нейронної мережі [5]

Згорткові нейронні мережі (CNN) складаються з кількох типів шарів, кожен з яких виконує певну задачу при обробці зображень. Першим шаром є згортковий шар (Convolution Layer), який виконує виділення ознак з вхідних даних. Цей шар використовує фільтри (ядра) з певними розмірами (MxN) для обробки вхідного зображення. Фільтри містять ваги, які навчаються на етапі тренування мережі. Вони переміщуються по зображенню, обчислюючи скалярний добуток між фільтром і частиною зображення, що накривається фільтром, створюючи карту ознак, яка відображає елементи зображення, наприклад, краї та кути. Згортковий шар дозволяє мережі визначати значущі риси без потреби в ручному налаштуванні.

Для зменшення розміру даних застосовується шар підвибірки (Pooling Layer). Після кожного згорткового шару зазвичай використовується шар підвибірки, що зменшує розмір карти ознак, зберігаючи лише найбільш значущі інформаційні елементи. Це дозволяє знижувати ризик перенавчання та покращувати продуктивність моделі. Для підвибірки можуть застосовуватися різні методи, зокрема максимальна підвибірка (max pooling), яка вибирає

максимальне значення в кожному вікні, або середнє підвибірка (average pooling), яке обчислює середнє значення в області.

Далі, після того як було виділено всі основні ознаки, зображення перетворюється в одномірний вектор через шар згладжування (Flatten Layer). Цей шар призначений для того, щоб перетворити багатовимірний масив ознак у простий вектор, який потім може бути переданий до повнозв'язного шару (Fully Connected Layer).

Повнозв'язний шар (Fully Connected Layer) обробляє цей вектор, де кожен нейрон одного шару з'єднаний з усіма нейронами наступного шару. Кілька таких шарів виконують остаточну обробку ознак, отриманих на попередніх етапах, для класифікації зображення. Вони визначають характеристики, на основі яких буде здійснена класифікація.

Нарешті, після проходження через всі повнозв'язні шари, результат передається на вихідний шар (Output Layer), який зазвичай використовує логістичні функції (наприклад, softmax або sigmoid) для оцінки ймовірностей належності об'єкта до конкретного класу. Цей шар надає ймовірнісну оцінку кожного класу, що дозволяє класифікувати зображення.

Процес навчання згорткових мереж полягає в тому, що на кожному етапі мережа намагається знайти оптимальні фільтри для виділення ознак. Це дозволяє мережі автоматично навчатися, що значно знижує потребу в ручному налаштуванні параметрів. Оскільки глибоке навчання потребує великих обсягів даних для тренування, CNN використовують великі набори зображень з мітками для навчання.

Однією з складових глибокого навчання є зворотний зв'язок, який використовується для корекції помилок під час навчання. В результаті використання зворотного зв'язку нейронна мережа здатна коригувати ваги та покращувати результати класифікації.

Розрізняють різні типи глибоких нейронних мереж, серед яких є рекурентні нейронні мережі (RNN), які можуть обробляти дані з часом, і

генеративні змагальні мережі (GAN), які використовуються для створення нових зображень на основі вже існуючих.

Методика навчання нейронних мереж вимагає великих обчислювальних ресурсів. Це може бути досягнуто за допомогою використання графічних процесорів (GPU), які значно прискорюють обчислення порівняно з традиційними центральними процесорами (CPU). Технології, як CUDA (Compute Unified Device Architecture), розроблені компанією NVIDIA, дозволяють використовувати можливості GPU для обчислень, значно скорочуючи час тренування моделей [27].

Приклади застосування глибоких нейронних мереж включають не тільки розпізнавання зображень, але й задачу генерації зображень, сегментації об'єктів, виявлення аномалій та багато інших напрямків.

Глибоке навчання дозволяє значно підвищити точність розпізнавання зображень порівняно з класичними методами, особливо в умовах складних або змінних умов освітлення, присутності шуму або варіативності об'єктів.

Застосування глибоких нейронних мереж для задач розпізнавання зображень продовжує розвиватися, і з кожним роком технології покращуються, що дозволяє досягати більш високих результатів.

Автономні моделі для вбудованих систем – це спеціалізовані підходи та алгоритми, що дозволяють пристроям з обмеженими обчислювальними потужностями виконувати задачі машинного навчання або комп'ютерного зору без необхідності підключення до потужних серверів або хмарних ресурсів. Ці моделі використовуються в реальному часі для прийняття рішень на основі аналізу даних, що надходять від датчиків або камер.

Вбудовані системи є частиною багатьох сучасних пристроїв, таких як розумні камери, робототехніка, автономні транспортні засоби, а також системи для моніторингу стану здоров'я або управління виробничими процесами. Вони працюють на спеціалізованих мікропроцесорах з обмеженими ресурсами: обчислювальною потужністю, пам'яттю та енергоспоживанням. Тому для їхньої

роботи використовуються оптимізовані моделі, які мають мінімальні вимоги до обчислювальних ресурсів.

Одним із напрямків розвитку автономних моделей є оптимізація для вбудованих систем. Для цього використовуються різні методи, що дозволяють зменшити розмір моделі та вимоги до пам'яті, при цьому зберігаючи її працездатність. Такі моделі часто застосовують квантизацію, пакетну нормалізацію, ф'южн моделі та моделі на основі прямих інтерпретацій. Квантизація, наприклад, дозволяє зменшити точність числових значень (наприклад, з 32-бітних значень до 8-бітних), що значно знижує вимоги до пам'яті та пришвидшує обчислення, без значної втрати точності.

Також використовуються невеликі нейроні мережі або мобільні архітектури, такі як MobileNet, Tiny YOLO та SqueezeNet, які створені спеціально для роботи в реальному часі на обмежених ресурсах. Такі мережі забезпечують баланс між точністю та швидкістю виконання задач, зокрема, на пристроях, які мають обмежену потужність обчислень і пам'яті [13].

Основними перевагами автономних моделей для вбудованих систем є здатність працювати без підключення до інфраструктури, швидкість обробки даних без затримок, а також збереження конфіденційності, оскільки всі обчислення відбуваються локально.

Проте, основними викликами для таких моделей є обмеження в обчислювальних потужностях, енергоспоживанні, пам'яті та необхідність підтримки високої точності моделей на обмежених даних. Щоб подолати ці виклики, активно використовуються методи компресії та оптимізації моделей, а також розробляються нові архітектури, спеціально адаптовані до вимог вбудованих систем.

2.2 Аналіз апаратних засобів для розпізнавання зображень

Апаратнізасоби для розпізнавання зображень варіюються за обчислюваль ними можливостями та призначенням. Вибір відповідної платформи залежить

від складності завдання, вимог до обчислювальних потужностей, енергоспоживання та бюджету. Розглянемо кілька популярних варіантів: Raspberry Pi, NVIDIA Jetson Nano, ESP32-CAM з Arduino Uno.

Raspberry Pi є одноплатним комп'ютером, який здобув популярність завдяки своїй доступності, компактним розмірам та гнучкості в застосуванні. Це апаратне забезпечення використовується для багатьох різноманітних задач, серед яких – обробка зображень, розпізнавання об'єктів, а також розробка вбудованих систем для робототехніки, автоматизації та навчання. Зовнішній вигляд й розміщення елементів наведено на рисунку 2.3. Для ілюстрації було обрано Raspberry Pi 5, як новішу модель.

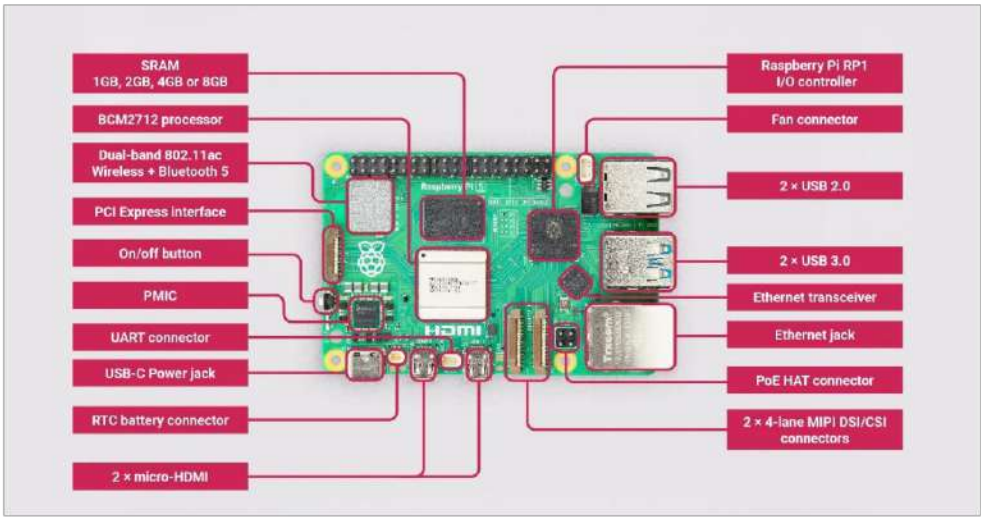

Рисунок 2.3 – Розміщення елементів на Raspberry Pi 5 [18]

Raspberry Pi оснащений процесором ARM, який має достатню продуктивність для виконання простих до середньо складних задач. Однак для задач обробки зображень та виконання алгоритмів машинного навчання потрібен оптимізований підхід, оскільки для обробки зображень стандартного центрального процесора (CPU) недостатньо. Зазвичай Raspberry Pi використовує відкриті бібліотеки, такі як OpenCV, для розпізнавання зображень.

Процесор в Raspberry Pi має кілька варіантів, залежно від моделі, зокрема процесор ARM Cortex-A53, що дозволяє працювати з роздільною здатністю до 4K, а також підтримує HDMI, камери та інші периферійні пристрої.

Однією з основних переваг Raspberry Pi є можливість підключення різних периферійних пристроїв. Він має порти для підключення камер, таких як офіційна камера Raspberry Pi Camera Module або сторонні камери через GPIO-піни. Це робить його зручним для створення систем спостереження, відеонагляду або навіть для розпізнавання об’єктів у реальному часі.

Однією з основних проблем, що виникають при використанні Raspberry Pi для обробки зображень, є обмежені апаратні ресурси. Для складніших завдань, таких як виконання нейронних мереж або складної обробки відео, доцільно підключати зовнішні пристрої, наприклад, графічні прискорювачі (GPU) або спеціалізовані чіпи для машинного навчання. У таких випадках можна використовувати модулі, які підключаються до Raspberry Pi через інтерфейс USB або через інші порти [19].

Для навчання нейронних мереж на Raspberry Pi треба оптимізувати моделі, адже потужності пристрою можуть бути недостатньо для виконання великих моделей глибокого навчання без застосування апаратного прискорення. В такому випадку моделі часто навчаються на більш потужних системах, а потім передаються на Raspberry Pi для подальшої роботи з інференсом.

Raspberry Pi підтримує роботу з популярними фреймворками для машинного навчання, такими як TensorFlow, PyTorch і OpenCV, що дає можливість використовувати його для різних задач, від простого розпізнавання об’єктів до складних застосувань в робототехніці та автономних системах. Крім того, є можливість використовувати Raspberry Pi для реалізації моделей на основі машинного зору, таких як детекція обличчя, виявлення руху, класифікація об’єктів та багато іншого.

Крім того, Raspberry Pi має низьке енергоспоживання, що робить його підходящим для використання в автономних системах, таких як роботи, бездротові системи відеоспостереження та інші мобільні пристрої. Платформа також має велику спільноту, що дозволяє швидко знайти рішення для типових завдань.

NVIDIA Jetson Nano – це одноплатна платформа для розробки інтелектуальних систем, зокрема для задач комп'ютерного зору, робототехніки та штучного інтелекту. Вона є частиною сімейства Jetson від NVIDIA і призначена для виконання обчислень, пов'язаних з глибоким навчанням, штучним інтелектом та обробкою зображень. Використання цієї платформи дозволяє досягати високої продуктивності при виконанні алгоритмів машинного навчання та нейронних мереж завдяки наявності спеціалізованих CUDA-ядер.

Однією з головних особливостей NVIDIA Jetson Nano є наявність потужного графічного процесора (GPU), який підтримує технології NVIDIA CUDA. Завдяки підтримці CUDA-ядер, Jetson Nano здатний значно прискорювати виконання паралельних обчислень, що робить його ідеальним для задач, що вимагають високої обчислювальної потужності, таких як обробка зображень у реальному часі, розпізнавання об'єктів, а також виконання алгоритмів глибокого навчання.

Розміщення основних елементів NVIDIA Jetson Nano наведено на рисунку 2.4.

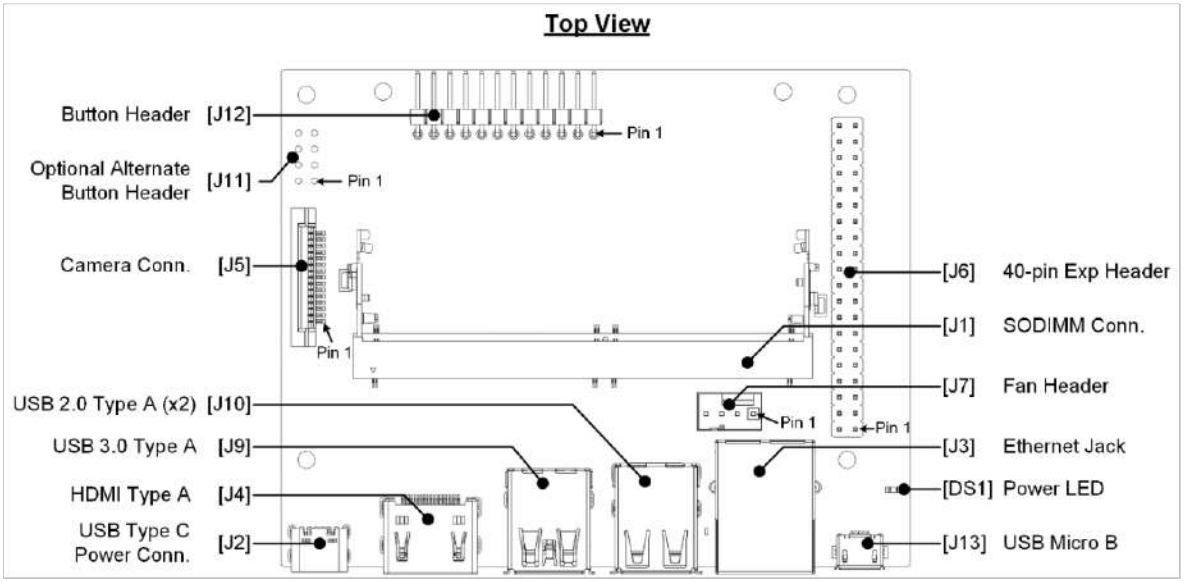

Рисунок 2.4 – Розміщення елементів платформи NVIDIA Jetson Nano [11]

Платформа оснащена чотириядерним ARM Cortex-A57 процесором та графічним процесором Maxwell з 128 CUDA-ядер. Це дозволяє використовувати

її для виконання складних обчислень, таких як інференс глибоких нейронних мереж, обробка відео високої роздільної здатності та інтерактивні системи комп'ютерного зору. Порівняно з іншими одноплатними комп'ютерами, такими як Raspberry Pi, Jetson Nano надає суттєво більшу потужність для задач глибокого навчання завдяки спеціалізованим апаратним ресурсам, що підтримують технології паралельних обчислень.

Завдяки програмному забезпеченню, яке базується на Linux, і використанню таких фреймворків як TensorFlow, PyTorch, OpenCV, Jetson Nano є потужним інструментом для розробки систем, орієнтованих на обробку зображень, розпізнавання об'єктів та автономні системи. Платформа підтримує безліч додаткових бібліотек і інструментів для машинного навчання, що дозволяє розробникам швидко розпочати роботу без необхідності створення складної інфраструктури.

Однак, одним із обмежень Jetson Nano є його порівняно маленька спільнота користувачів у порівнянні з іншими платформами, такими як Raspberry Pi. Хоча на офіційному форумі NVIDIA доступні документи та ресурси, для деяких специфічних версій операційної системи чи фреймворків на допомогу іноді приходять ентузіасти, які підтримують оновлення та створюють патчі на таких платформах, як GitHub. Це може створювати певні труднощі, адже не всі версії Jetson Nano отримують регулярні офіційні оновлення.

NVIDIA Jetson Nano має вбудований Wi-Fi та Bluetooth (через додаткові модулі), порти для підключення камер, таких як Raspberry Pi Camera Module або сторонні камери через інтерфейс MIPI CSI. Також присутні порти USB для підключення інших периферійних пристроїв, таких як мишки, клавіатури або навіть додаткові камери.

Для більшого рівня інтерфейсу та підключення зовнішніх пристроїв Jetson Nano підтримує стандартні порти GPIO, I2C, SPI, що робить його універсальним інструментом для розробки роботизованих систем, автономних транспортних засобів, а також систем для розпізнавання зображень.

ESP32-CAM є компактною платою з вбудованою камерою, яка використовує мікроконтролер ESP32 для обробки даних та управління камерою. Ця плата має багато можливостей для розпізнавання зображень, передачі даних через Wi-Fi або Bluetooth, а також для виконання простих задач, пов'язаних з комп'ютерним зором. ESP32-CAM ідеально підходить для створення бездротових систем відеоспостереження, розпізнавання осіб, а також для автономних проектів з малою потужністю.

Основною перевагою ESP32-CAM є її вартість та зручність у використанні. Завдяки вбудованому чіпу ESP32, ця плата має підтримку Wi-Fi та Bluetooth, що дозволяє бездротово передавати зображення або отримувати керування камерою з віддаленого пристрою. Оскільки ESP32 підтримує роботу з OpenCV та іншими фреймворками для обробки зображень, вона може виконувати базові операції комп'ютерного зору, такі як розпізнавання простих фігур або відстеження руху.

Центральним елементом ESP32-CAM є її вбудована камера OV2640, що забезпечує роздільну здатність до 2 Мп. Для простих задач, таких як розпізнавання облич або виявлення руху, цього достатньо. Однак, для більш складних алгоритмів розпізнавання зображень, ESP32-CAM може бути обмежена через відсутність потужних графічних процесорів або апаратних засобів для обробки великих обсягів даних у реальному часі.

На рисунку 2.5 наведено зовнішній вигляд й розміщення елементів ESP32-CAM.

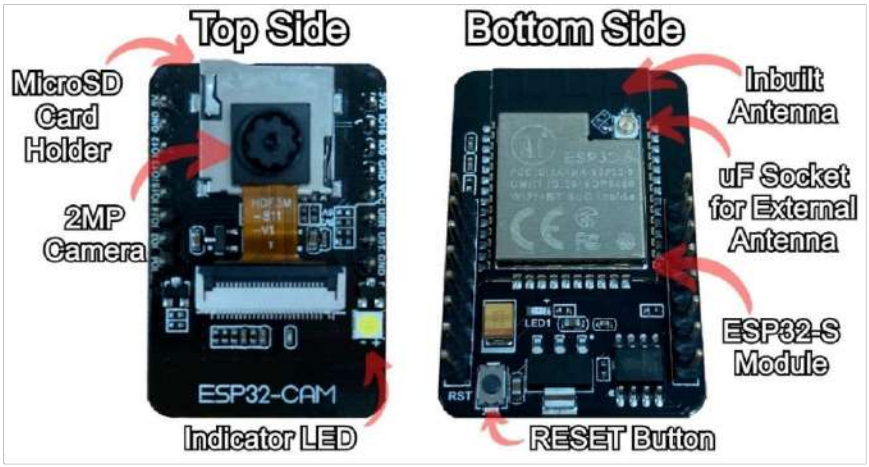

Рисунок 2.5 – Розміщення елементів ESP32-CAM [21]

ESP32-CAM можна інтегрувати з іншими мікроконтролерами, такими як Arduino Uno, для розширення її функціональності. Arduino Uno, зокрема версія R3, є одним з найпоширеніших мікроконтролерів, який часто використовується в робототехніці, автоматизації та для навчальних проектів. Він оснащений чіпом ATmega328P, який має 14 цифрових вхідних/вихідних пінів, 6 аналогових входів, а також можливість підключення до зовнішніх пристроїв через інтерфейси UART, SPI та I2C [3].

Розміщення елементів та зовнішній вигляд Arduino Uno R3 наведено на рисунку 2.6.

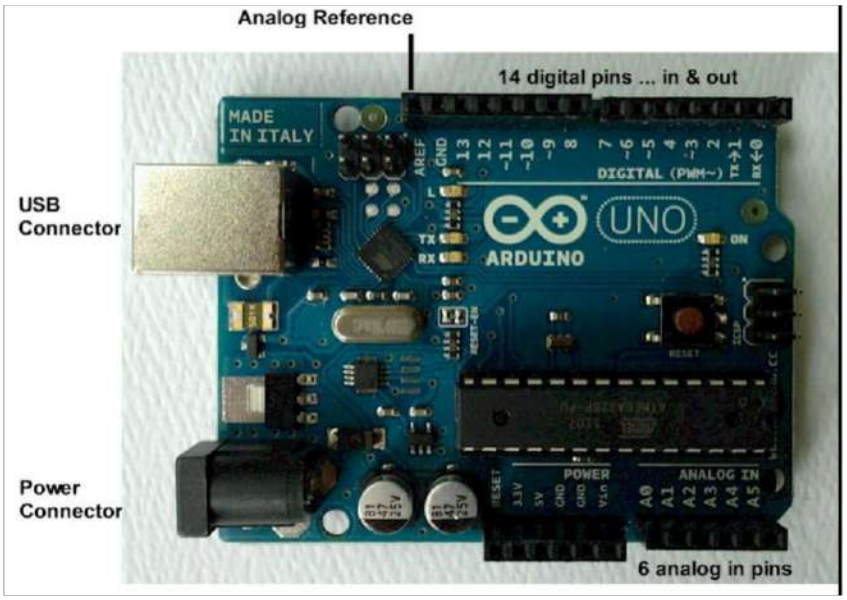

Рисунок 2.6 – Розміщення елементів Arduino Uno R3 [2]

Arduino Uno R3 відрізняється своєю простотою у використанні, а також широкою спільнотою розробників, що дозволяє швидко знаходити рішення для різноманітних задач. Для використання ESP32-CAM в парі з Arduino Uno, використовується стандартний USB-кабель для підключення плат до комп'ютера для програмування та відлагодження.

Інтеграція ESP32-CAM з Arduino Uno дозволяє використовувати обидва пристрої для виконання складних завдань, де ESP32-CAM відповідає за захоплення та передачу зображень, а Arduino Uno виконує управління

зовнішніми компонентами або контролює певні сенсори для автоматизації процесу.

Однією з особливостей такого з'єднання є необхідність взаємодії між двома пристроями через серійний інтерфейс. Arduino Uno може бути використаний для обробки даних з сенсорів, таких як датчики температури, вологості або ультразвукові датчики, для взаємодії з іншими компонентами системи, а ESP32-CAM – для передачі або обробки зображень, а також для здійснення бездротової передачі даних.

Платформа ESP32-CAM підтримує програмування через Arduino IDE, що робить її дуже зручною для розробників, знайомих з цією середовищем. Для налаштування ESP32-CAM можна використовувати бібліотеки для роботи з камерою, Wi-Fi або Bluetooth, а також готові бібліотеки для роботи з комп'ютерним зором, такі як ESP32-CAM Video Streaming або OpenCV для роботи з Raspberry Pi чи іншими пристроями.

Оскільки ESP32-CAM працює на мікроконтролері з 240 МГц тактовою частотою та 520 КБ оперативної пам'яті [7], вона здатна виконувати обмежені задачі розпізнавання зображень у реальному часі. Однак для більш складних завдань, таких як глибоке навчання або обробка великих обсягів даних, доцільно використовувати більш потужні платформи, як NVIDIA Jetson Nano або Raspberry Pi, що мають більші обчислювальні ресурси.

2.3 Вибір оптимальних інструментів та технологій для реалізації системи

Вибір ESP32-CAM з Arduino Uno як основних компонентів для реалізації системи розпізнавання зображень обґрунтовано низкою факторів. ESP32-CAM має достатню обчислювальну потужність для виконання базових операцій з обробки зображень, а також можливість підключення до бездротових мереж через Wi-Fi або Bluetooth. Це дозволяє здійснювати передавання даних або отримання оновлень без потреби в додаткових дротових з'єднаннях. Додатково,

камера плати ESP32-CAM підтримує зйомку з роздільною здатністю 640x480 пікселів [16], що є достатнім для багатьох задач на етапах тестування та розробки.

Для обробки зображень, отриманих за допомогою камери, планується використання бібліотеки numpy, яка є стандартом для числових обчислень в Python. Вона дозволяє працювати з масивами даних, зокрема, для обробки пікселів зображення та проведення математичних операцій, що є необхідними на етапі попередньої обробки зображення, таких як фільтрація та нормалізація. Завдяки широким можливостям numpy, система зможе здійснювати маніпуляції з даними без додаткових витрат на стороннє програмне забезпечення.

Для задач розпізнавання зображень буде використано модель YOLOv3, яка виконує детекцію об'єктів за один прохід через нейронну мережу. YOLO (You Only Look Once) – це архітектура згорткової нейронної мережі (CNN), розроблена Джозефом Редмоном у 2015 році. Вона відрізняється від традиційних методів тим, що прогнозує межові рамки та класифікацію об'єктів одночасно, без попереднього генерування регіонів.

Архітектура YOLO поділяє вхідне зображення на сітку, де кожна комірка відповідає за виявлення об'єктів, розташованих у її межах. Мережа складається із згорткових шарів для вилучення ознак і повнозв'язних шарів, що прогнозують координати прямокутників, клас об'єкта та рівень довіри (confidence score). Завдяки такому підходу досягається висока швидкість обробки, що робить YOLO придатною для роботи в реальному часі.

YOLOv3 є вдосконаленою версією цієї архітектури, яка використовує багаторівневі ознаки для детекції об'єктів різного розміру. Вона базується на нейромережі Darknet-53, що складається з 53 згорткових шарів із залишковими зв'язками (residual connections), що покращує стабільність навчання. Передбачення виконується на трьох рівнях деталізації (multi–scale detection), що підвищує точність роботи з дрібними об'єктами.

Інференс YOLOv3 реалізується через відповідні бібліотеки та оптимізовані фреймворки, що дає змогу застосовувати її на пристроях з обмеженими

обчислювальними ресурсами. Серед них TensorRT для платформ NVIDIA, OpenVINO для процесорів Intel, а також TensorFlow Lite та NCNN для ARM-архітектури. Це дозволяє інтегрувати модель у системи з низькою продуктивністю, зокрема ESP32-CAM [25].

Використання ESP32-CAM доцільне для збору зображень, після чого отримані дані передаватимуться на обробку в систему, де numpy та YOLOv3 проводитимуть необхідні операції з виявленням об’єктів та класифікацією. Це дозволить реалізувати систему для різноманітних задач розпізнавання зображень, таких як виявлення певних об’єктів.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ НА ОСНОВІ ARDUINO

3.1 Проєктування апаратної частини системи

В основі апаратної частини системи лежить модуль ESP32-CAM та плата Arduino Uno R3. Взаємодія цих пристроїв забезпечує обробку отриманих зображень та керування зовнішніми пристроями.

Модуль ESP32-CAM містить 32-бітний мікроконтролер ESP32 з тактовою частотою до 240 МГц, вбудованими модулями Wi-Fi та Bluetooth, а також інтерфейсом для підключення камери OV2640. Він оснащений 10-бітним АЦП, підтримує інтерфейси SPI, I²C, I²S та UART [8]. Виводи модуля розташовані по краях плати, що дозволяє легко підключати його до інших пристроїв. Розташування виводів ESP32-CAM представлено на рисунку 3.1.

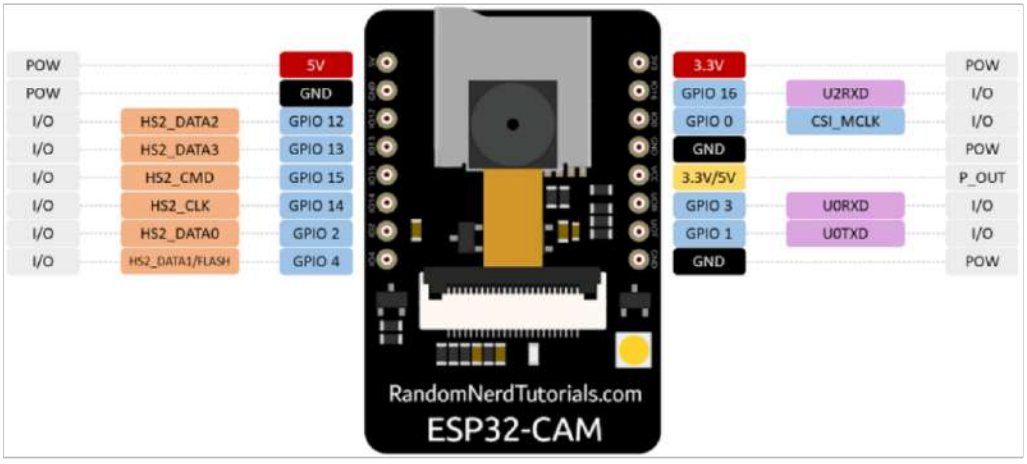

Рисунок 3.1 – Розташування виводів ESP32-CAM [15]

Плата Arduino Uno R3 побудована на мікроконтролері ATmega328P, що працює на частоті 16 МГц. Вона містить 14 цифрових виводів, з яких 6 підтримують ШІМ, 6 аналогових входів, а також інтерфейси UART, I²C та SPI. Живлення здійснюється через USB або зовнішній блок живлення 7-12 В. Розташування виводів Arduino Uno R3 показано на рисунку 3.2.

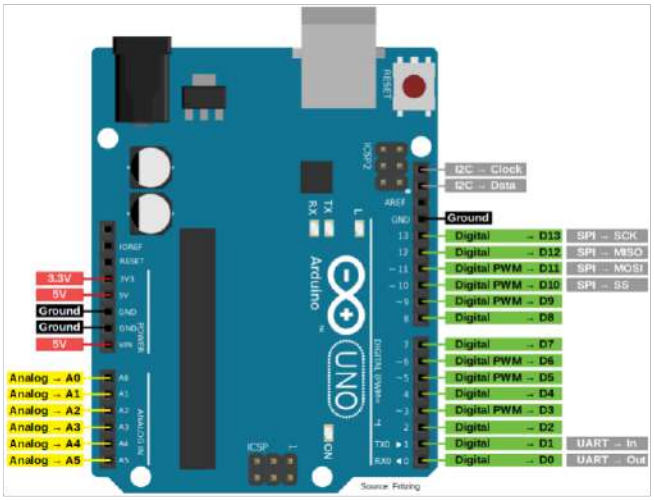

Рисунок 3.2 – Розташування виводів Arduino Uno R3 [22]

Взаємодія між ESP32-CAM та Arduino Uno R3 реалізується через послідовний інтерфейс UART. Arduino Uno виступає як керуючий контролер, який отримує зображення з ESP32-CAM та виконує обробку даних. Структурна схема взаємодії компонентів системи представлена на рисунку 3.3.

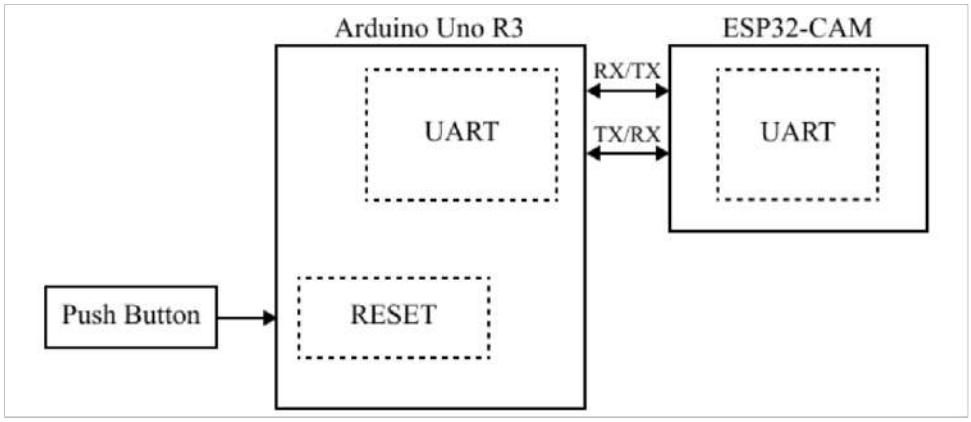

Рисунок 3.3 – Структурна схема приладу

Для підключення ESP32-CAM до Arduino Uno R3 використовуються наступні з’єднання: TX ESP32-CAM підключається до RX Arduino Uno, RX ESP32-CAM – до TX Arduino Uno, а також загальний провід GND. Живлення ESP32-CAM здійснюється від стабілізованого джерела 5 В через Arduino Uno. Принципова схема підключення елементів системи представлена на рисунку 3.4.

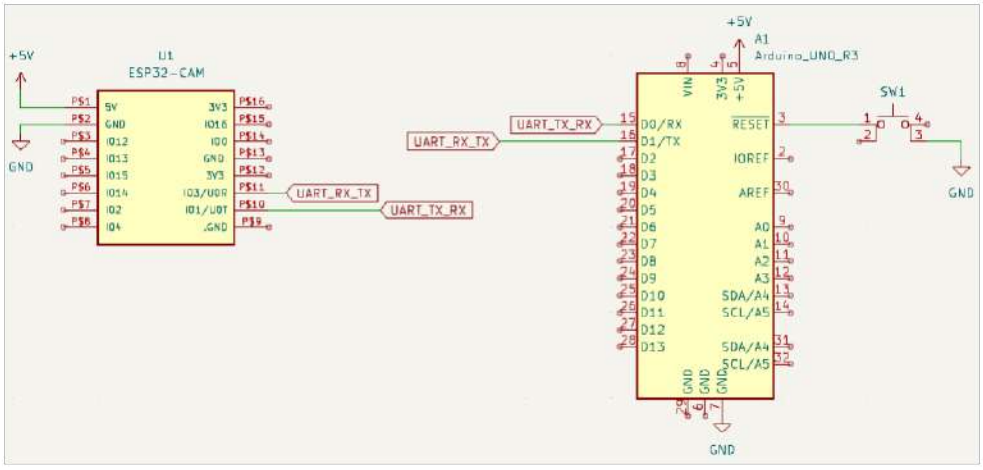

Рисунок 3.4 – Принципова схема приладу

На основі проєктування було зібрано фізичний функціональний прототип пристрою. Прототип складається з плати ESP32-CAM, яка відповідає за зйомку зображень, та Arduino Uno R3, що забезпечує керування процесами обробки даних та передачі їх на інші пристрої. Зовнішній вигляд зібраного прототипу наведено на рисунку 3.5.

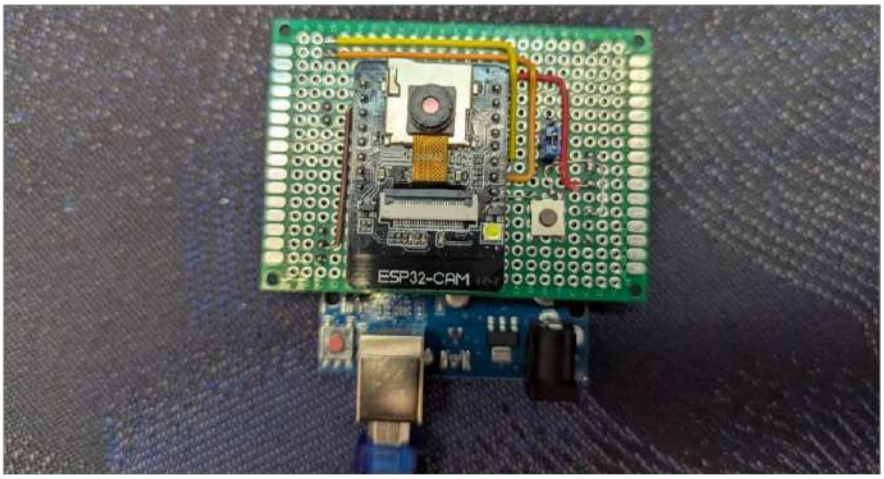

Рисунок 3.5 – Вигляд спроєктованої системи

3.2 Розробка програмного забезпечення

Код для плати відповідає за ініціалізацію камери, налаштування бездротової мережі та обробку запитів користувача для отримання зображень.

В коді для ESP32-CAM спочатку налаштовується бездротова точка доступу (AP), яка дозволяє підключатися до плати з інших пристроїв через Wi-Fi. Сервер налаштовується на обслуговування HTTP-запитів, що дозволяє отримати зображення з камери при зверненні до відповідного URL. Камера налаштовується на зйомку з роздільною здатністю 800x600 пікселів, використовуючи бібліотеки для роботи з ESP32 та камерою.

Основними етапами роботи програми є:

– ініціалізація камери;

– підключення плати до бездротової мережі;

– встановлення веб–сервера, який обслуговує запити на отримання зображення;

– зйомка зображень за допомогою камери при отриманні запиту від клієнта;

– передача зображень клієнту через Wi-Fi.

Блок-схему алгоритму коду для сервера наведено на рисунку 3.6.

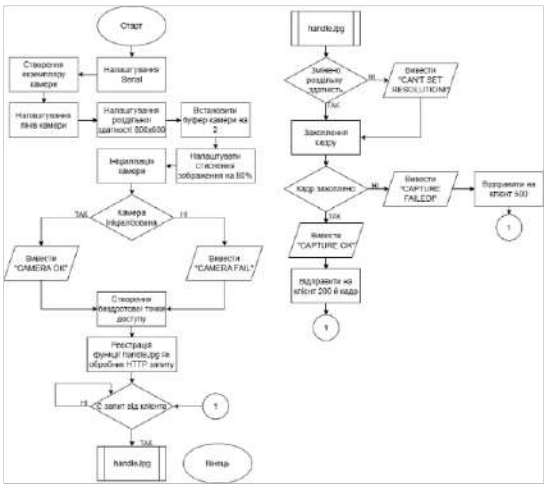

Рисунок 3.6 – Блок-схема алгоритму роботи сервера

Після налаштування камери, програма ініціалізує бездротову точку доступу через функцію initAP(). Це дозволяє користувачеві підключатися до плати через Wi-Fi, а також визначати IP-адресу пристрою, на якій доступний веб-сервер. Для цього виводиться повідомлення на серійну консоль. Код цієї функції продемонстровано на рисунку 3.7.

```
void initAP() {
    WiFi.softAP(AP_SSID, AP_PASS);
    IPAddress IP = WiFi.softAPIP();
    Serial.printf("Access Point IP address: %s\n", IP.toString().c_str());
    Serial.printf("Open http://%s%s in your browser\n", IP.toString().c_str(), URL);
}
```

Рисунок 3.7 – Код функції initAP()

Далі в коді налаштовується веб–сервер через функцію initServer(), яка визначає URL для доступу до зображень, а також вказує обробник запитів – handleJpg(). Ця функція виконує перевірку на можливість зміни роздільної здатності камери та викликає функцію serveJpg(), яка захоплює кадр і передає його клієнту через веб-сервер. Код для реалізації цих двох функцій наведено на рисунку 3.8.

```
void serveJpg() {
    auto frame = esp32cam::capture();
    if (frame == nullptr) {
        Serial.println("CAPTURE FAILED!");
        server.send(503, "", "");
        return;
    }
    Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(), static_cast<int>(frame->size()));
    server.setContentLength(frame->size());
    server.send(200, "image/jpeg");
    WiFiClient client = server.client();
    frame->writeTo(client);
}

void handleJpg() {
    if (!esp32cam::Camera.changeResolution(RES)) {
        Serial.println("CAN'T SET RESOLUTION!");
    }
    serveJpg();
}
```

Рисунок 3.8 – Код функціq handleJpg() та serveJpg()

При кожному зверненні до сервера, у функції loop() викликається server.handleClient(), що обробляє запити користувачів. Якщо захоплення кадру успішне, сервер передає зображення клієнту через Wi-Fi, використовуючи протокол HTTP. Якщо ж захопити кадр не вдалося, клієнту відправляється код помилки 503. Повний код програми для ESP32-CAM наведено в лістингу А.1.

Програмне забезпечення для клієнта розроблено для обробки зображень, отриманих з камери ESP32-CAM, за допомогою моделі YOLO для розпізнавання об’єктів на зображеннях. Клієнтська частина програми використовує бібліотеки OpenCV та NumPy для обробки зображень та взаємодії з веб-сервером ESP32-CAM. Зображення отримуються через HTTP-запит та передаються на подальшу обробку для детекції об’єктів. Блок - схему алгоритму коду для клієнта наведено на рисунку 3.9. Код програми наведено на лістингу А.2.

У програмі спочатку визначається URL для отримання зображень з камери ESP32-CAM. Зображення будуть надходити за адресою «http://192.168.4.1/cam.jpg». Далі завантажуються файли моделі YOLO, що містять ваги моделі, конфігурацію мережі та імена класів об’єктів, які можуть бути виявлені. Для цього використовуються файли «yolov3.weights», «yolov3.cfg» та «coco.names». Після завантаження моделей, ініціалізується нейронна мережа за допомогою функції cv2.dnn.readNet().

Цей підхід дозволяє ефективно використовувати ресурси ESP32-CAM, делегуючи обчислювально складну обробку нейронній мережі на стороні клієнта. В результаті досягається висока швидкість обробки та точність розпізнавання об'єктів у реальному часі. Передбачено також обробку випадків, коли камера тимчасово недоступна або мережеве з’єднання переривається, що підвищує надійність системи. Подібна архітектура ідеально підходить для задач відеоспостереження, моніторингу навколишнього середовища або розумного будинку.

Програма також може бути легко адаптована для інших моделей нейронних мереж або змін у форматі зображень. Це забезпечує гнучкість розширення функціоналу без суттєвої модифікації архітектури. При потребі можливе збереження отриманих зображень або результатів обробки у базу даних. Такий підхід дозволяє створювати масштабовані системи, орієнтовані на довготривалу роботу з великими обсягами візуальних даних.

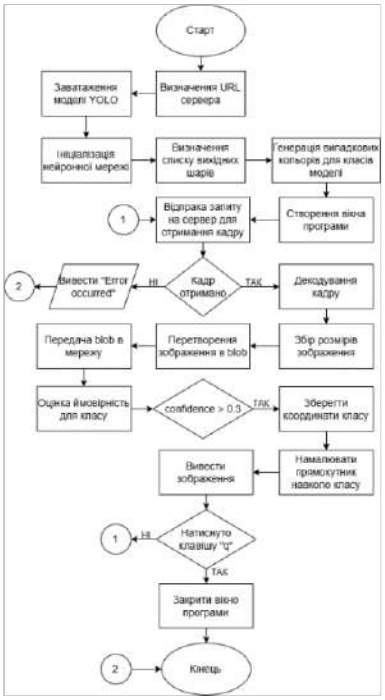

Рисунок 3.9 – Блок-схема алгоритму роботи клієнта

Після завантаження мережі з налаштованими шарами та класами об’єктів, визначається список вихідних шарів мережі, які використовуються для отримання результатів детекції. Це здійснюється за допомогою функцій getLayerNames() та getUnconnectedOutLayers(). Вихідні шари дозволяють отримати детекції об’єктів після обробки зображень через мережу.

Далі генеруються випадкові кольори для кожного класу об’єктів, щоб можна було зручно позначати об’єкти різними кольорами на зображеннях. Це робиться за допомогою бібліотеки NumPy, використовуючи функцію np.random.uniform(), що генерує масив випадкових значень.

Код для підготовки до розпізнавання об’єктів наведено на рисунку 3.10.

Після генерації кольорів зображення надсилається до нейронної мережі для виявлення об’єктів. Результати детекції включають координати об'єктів, рівень впевненості та відповідні класи, які візуалізуються на зображенні у вигляді прямокутників з підписами.

```
url = "http://192.168.4.1/cam.jpg"

weights_path = r"./YOLO/yolov3.weights"
config_path = r"./YOLO/yolov3.cfg"
names_path = r"./YOLO/coco.names"

net = cv2.dnn.readNet(weights_path, config_path)
with open(names_path, "r") as f:
    classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()

out_layers = net.getUnconnectedOutLayers()
if isinstance(out_layers[0], list):
    output_layers = [layer_names[i[0] - 1] for i in out_layers]
else:
    output_layers = [layer_names[i - 1] for i in out_layers]

colors = np.random.uniform(0, 255, size=(len(classes), 3))
```

Рисунок 3.10 – Створення глобальних змінних та об’єктів

Функція detect_objects() виконує основну роботу з обробки зображень. Спочатку вона приймає зображення, отримане з камери, і за допомогою функції cv2.dnn.blobFromImage() готує його для обробки нейронною мережею. Всі пікселі зображення нормалізуються, змінюється його розмір, та зображення конвертується в формат, сумісний з входом нейронної мережі. Після цього мережа здійснює розпізнавання об’єктів на зображенні, і функція net.forward() передає оброблене зображення через вихідні шари для отримання результатів.

Отримані результати детекції включають координати меж об’єктів, рівень їхньої ймовірності та ідентифікатори класів. Якщо ймовірність виявленого об’єкта перевищує поріг (0.3), то координати об’єкта додаються до списку, і він буде позначений на зображенні. Для кожного об’єкта малюється прямокутник та виводиться текст з іменем класу та ймовірністю. Для цього використовуються функції cv2.rectangle() та cv2.putText(). Після цього зображення повертається до основної програми для відображення.

Цей підхід дозволяє не лише візуалізувати результати, а й використовувати їх для прийняття рішень у реальному часі. За потреби функцію можна

модифікувати для фільтрації конкретних класів об’єктів або інтеграції з іншими модулями системи.

У основній функції main() запускається безперервний цикл, який отримує зображення через HTTP-запит з камери ESP32-CAM, як це наведено на рисунку 3.11.

```
def main():
    cv2.namedWindow("Object Detection", cv2.WINDOW_AUTOSIZE)

    while True:
        try:
            img_resp = urllib.request.urlopen(url)
            imgnp = np.array(bytearray(img_resp.read()), dtype=np.uint8)
            frame = cv2.imdecode(imgnp, -1)
            frame = detect_objects(frame)

            cv2.imshow("Object Detection", frame)

            if cv2.waitKey(1) & 0xFF == ord("q"):
                break
        except Exception as e:
            print(f"Error occurred: {e}")
            break

    cv2.destroyAllWindows()
```

Рисунок 3.11 – Реалізація функції main

Зображення передаються до функції detect_objects() для обробки. Весь процес триває до тих пір, поки не буде натиснута клавіша "q", щоб завершити роботу програми. Якщо виникає помилка під час отримання зображення, програма виводить повідомлення про помилку та припиняє виконання.

3.3 Тестування системи розпізнавання зображень та аналіз результатів

Підключимо камеру до джерела живлення, налаштуємо підключення до Wi-Fi мережі та перевіримо роботу системи розпізнавання зображень.

На рисунку 3.12 наведено результат розпізнавання класів, що містяться в кадрі. У консоль виводяться назви класів, а також значення довіри до кожного об’єкта, що було визначено. Наприклад, були виявлені об’єкти класів: plant (0.90), teddy bear (0.97), laptop (1.0), vase (0.92), clock (0.47), cell phone (0.99), remote (0.99), cup (1.0), bottle (0.82), book (0.75). Як видно з результатів, найменша впевненість системи була у класі "clock", що ймовірно пов’язано з його квадратною формою, та у класі "book", оскільки об’єкт не був повністю видимим у кадрі.

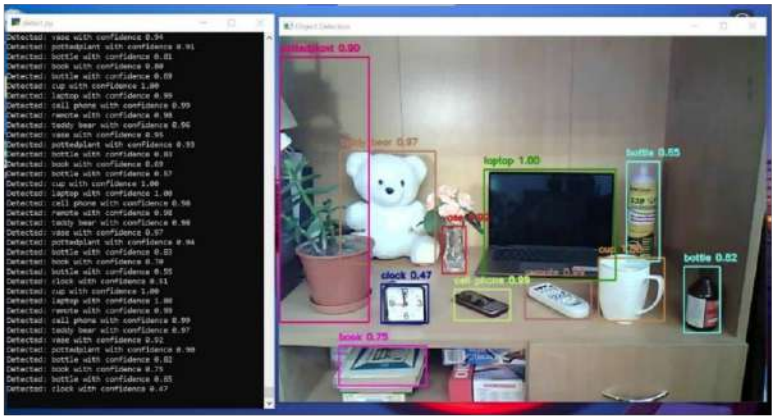

Рисунок 3.12 – Результат розпізнавання класів

Також на рисунку 3.13 наведено кадр, на якому система визначила класи cat (1.0) та chair (1.0). У цьому випадку довіра до визначених класів була максимальною. Результати визначення дублюються в консолі, де можна побачити виведення назв класів та їх довіри.

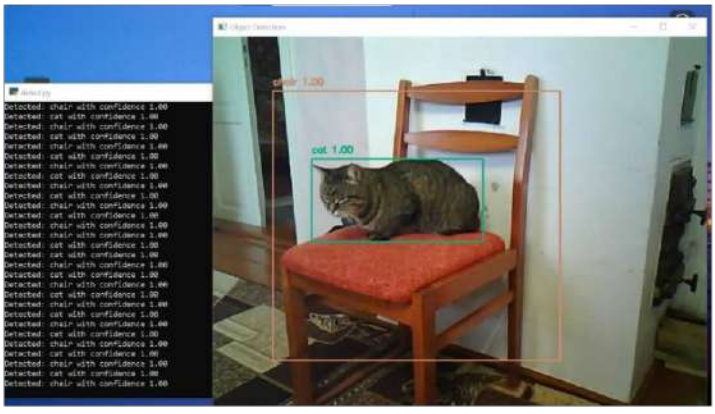

Рисунок 3.13 – Результат розпізнавання класів cat та chair

Варто зазначити, що система не виводить ніяких логів у консоль та не визначає об’єкти, коли на кадрі відсутні класи, які можуть бути розпізнані моделлю. Прикладом цього є ситуація, показана на рисунку 3.14, де на зображенні відсутні визначені об’єкти, і консоль не містить жодного виведення.

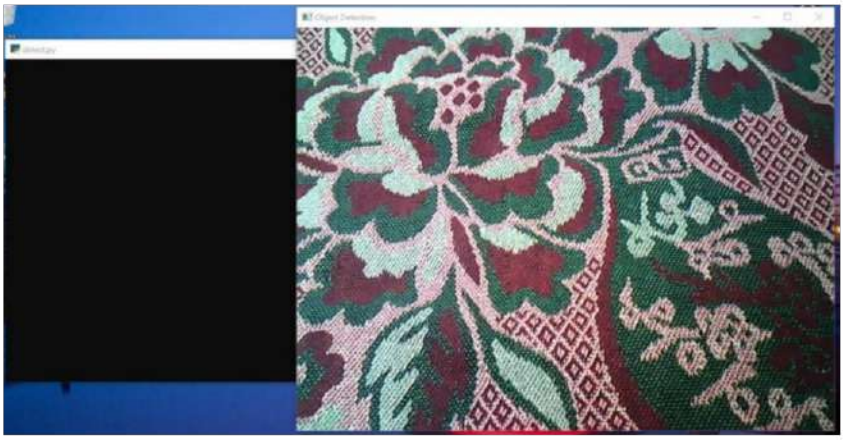

Рисунок 3.14 – Відсутність визначених класів на зображенні

ВИСНОВКИ

У ході виконання роботи було досліджено і розробленно систему розпізнавання зображень за допомогою штучного інтелекту на основі платформи Arduino. Аналіз апаратних обмежень мікроконтролера показав, що його обчислювальні ресурси недостатні для виконання складних операцій обробки зображень. З огляду на це, прийнято рішення використовувати Arduino для збору даних, а їх подальшу обробку виконувати на клієнтській частині системи.

У процесі виконання роботи була реалізована система, здатна розпізнавати об’єкти на зображеннях, отриманих через камеру ESP32-CAM. Для цього було використано штучне інтелектуальне середовище для обробки зображень. Технічне рішення, що включає інтеграцію мікроконтролера Arduino та камери ESP32-CAM, дозволило створити компактну систему для розпізнавання та індетифікації об’єктів у реальному часі.

Спроектовано систему, що включає інтеграцію ESP32-CAM з Arduino для розпізнавання зображень.

Розроблений алгоритм для мікроконтролеру для отримання зображень і взаємодії з клієннською частиною системи, комплекс включає серверний модуль для обробки даних та візуалізації результатів. На основі отриманих даних здійснюється побудова маркірованих зображень, що містять позначки розпізнаних об’єктів. Вивід інформації у текстовому та графічному вигляді дає змогу аналізувати роботу алгоритму та оцінювати точність розпізнавання.

Для розпізнавання об’єктів реалізовано алгоритм обробки зображень за допомгою штучного інтелекту на основі нейронної мережі YOLO. Дана модель дозволяє визначати об’єкти на зображеннях та класифікувати їх з певним рівнем достовірності. У ході тестування алгоритму було отримано результати, що підтверджують можливість використання даного підходу для ідентифікації об’єктів у режимі реального часу.

Проведено аналіз методів розпізнавання, включаючи класичні алгоритми (метод Кенні, SIFT, SURF) та методи на основі згорткових нейронних мереж.

Вибір YOLO обґрунтований його здатністю працювати з потоковими зображеннями та виконувати виявлення об’єктів з високою швидкістю. Реалізована система дозволяє отримувати список виявлених об’єктів із зазначенням їх класів та рівня достовірності.

Виконано тестування працездатності прототипу в умовах обмежених апаратних ресурсів. Отримані результати підтверджують можливість використання розробленої системи для розпізнавання об’єктів у реальному часі. Пристрій може застосовуватись у задачах автоматизованого моніторингу, контролю доступу та обробки відеопотоків.

Результати дослідження демонструють, що запропонований підхід дозволяє реалізувати систему розпізнавання зображень на основі бюджетних апаратних засобів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abdallah M. Object detection with arduino and ultrasonic sensor. *GitHub*. URL: https://salo.li/29B9c09 (дата звернення: 01.05.2025).

2. Arduino UNO R3 Wikimedia project participants – Вікіпедія. *Вікіпедія*. URL: https://salo.li/beCcd13 (дата звернення: 14.05.2025).

3. Arduino UNO R3 Datasheet. *Docs Arduino*. URL: https://salo.li/9fDf4a3 (дата звернення: 26.01.2025).

4. Automatic number plate recognition. *GitHub*. URL: https://salo.li/F293538 (date of access: 11.02.2025).

5. Budiawan J. C. Comprehensive overview of convolutional neural networks. *Medium*. URL: https://salo.li/bbFBe26 (date of access: 09.02.2025).

6. Convolutional Neural Networks. *Medium*. URL: https://salo.li/582E01d (дата звернення: 22.01.2025).

7. ESP32 datasheet. Electronic Parts Datasheet Search. URL: https://salo.li/66096Fa (дата звернення: 17.02.2025).

8. ESP32-CAM ai-thinker pinout guide. *Random Nerd Tutorials*. URL: https://salo.li/0FF08ed (дата звернення: 17.02.2025).

9. Harmaya P. L., Suryanata M. G., Ibnutama K. Comparison Canny and Sobel Methods Detecting Edges of Dental Caries. 2023. P. 647– 654. URL: https://salo.li/8181044 (дата звернення: 11.02.2025).

10. Iftekher A. Fundamental OMachine Learning. *HashNode*. URL: https://salo.li/c8D1978 (date of access: 12.02.2025).

11. Jetson Nano 2GB Developer Kit User Guide. *NVIDIA Developer*. URL: https://salo.li/b64A237 (дата звернення: 14.02.2025).

12. Lee T. Arduino AI with tensorflow lite. *Instructables*. URL: https://salo.li/9bc0f24 (дата звернення: 01.05.2025).

13. Liu L., et al. Research on Pedestrian Detection Algorithm Based on MobileNet-YoLo. *Computational Intelligence and Neuroscience*. 2022. Vol. 2022. P. 8924027. URL: https://salo.li/2789679 (дата звернення: 17.02.2025).

14. Loreto S. Multi-criteria approach for the energy and environmental impact evaluation in urban districts in the central Mediterranean area. *ScienceDirect*. URL: https://salo.li/D067349 (date of access: 14.02.2025).

15. Modulo WIFI ESP32 Cámara. *BIGTRONICA*. URL: https://salo.li/c8A43b2 (date of access: 13.03.2025).

16. OV2640 datasheet. *Electronic Parts Datasheet Search*. URL: https://salo.li/2Cf1237 (дата звернення: 12.02.2025).

17. Poda X., Qirici O. Shape Detection and Classification Using OpenCV and Arduino Uno. *RTA-CSIT*. 2019. Vol. 2280. P. 128-136. URL: https://salo.li/86523c2 (дата звернення: 29.01.2025).

18. Raspberry pi. Kubii. URL: https://salo.li/C5045Bd (date of access: 13.02.2025).

19. Raspberry Pi 5 Datasheets. *Raspberry*. URL: https://salo.li/00192d2 (дата звернення: 15.02.2025).

20. SVM Python. *Hands-On Cloud*. URL: https://salo.li/d750b63 (дата звернення: 15.02.2025).

21. Use a ESP32-CAM module to stream HD video over local network. *Electronics*. URL: https://salo.li/c45c3D3 (date of access: 10.02.2025).

22. Venkatasamy R., Dhanraj J. A. Contrivance of ssd–mobilenets algorithm-based smart door lock system. *SpringerLink*. URL: https://salo.li/EcCEde2 (date of access: 14.02.2025).

23. Warden P., Situnayake D. TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. *O'Reilly Media*, 2019. URL: https://salo.li/3A0357b (дата звернення: 08.02.2025).

24. West P. C. Image Processing for Machine Vision. *Quality Magazine*. URL: https://salo.li/1126c7F (дата звернення: 17.02.2025).

25. Гапонов Д. О. Автоматизована система для розпізнавання об'єктів. URL: https://salo.li/a6fc41e(дата звернення: 02.02.2025).

26. Зарічковий О. А. Алгоритмічне забезпечення для розмітки надвеликих об’ємів даних для задачі детекції об’єктів методами комп’ютерного зору. URL: https://salo.li/97Ee2Aa (дата звернення: 25.01.2025).

27. Крайнюк М. Ю., Полярус О. В. Глибоке навчання для вимірювань: застосування нейронних мереж у рішенні реальних задач. URL: https://salo.li/57691f0 (дата звернення: 14.02.2025).

28. Ладуба М. Що таке нейронна мережа, як вона працює та які завдання може вирішувати. *MC Today*. URL: https://salo.li/D44E792 (дата звернення: 17.02.2025).

29. Мельник В., Мельник К., Коптюк Ю. Дослідження методів розпізнавання зображень на основі нейронних мереж. 2019. №35. С. 161-165.

30. Морква С. О. Методи розпізнавання образів у робототехніці з використанням платформи Arduino. URL: https://salo.li/660EC18 (дата звернення: 16.01.2025).

31. Синєглазов В. М. Прикладні системи штучного інтелекту: комп’ютерний зір. *Вісник Національної академії наук України*. 2024. №6. С. 43-48. URL: https://salo.li/d88a7e8(дата звернення: 14.02.2025).

32. Старчиков І. Розпізнавання зображень із впровадженням моделі YOLO. URL: https://salo.li/Bdf8C3f (дата звернення: 17.02.2025).

33. Таран Д. А. Система комп’ютерного зору на базі нейронної мережі. URL: https://salo.li/C44Defc (дата звернення: 17.02.2025).

ДОДАТКИ

Додаток А

Коди програм

Лістинг А.1 – Код програми для сервера

```
#include "WebServer.h"
#include "WiFi.h"
#include "esp32cam.h"

const char* AP_SSID = "ESP32-CAM";
const char* AP_PASS = "12345678";

const char* URL = "/cam.jpg";
static auto RES = esp32cam::Resolution::find(800, 600);

WebServer server(80);

void serveJpg() {
    auto frame = esp32cam::capture();
    if (frame == nullptr) {
        Serial.println("CAPTURE FAILED!");
        server.send(503, "", "");
        return;
    }
    Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(),
frame->getHeight(), static_cast<int>(frame->size()));
    server.setContentLength(frame->size());
    server.send(200, "image/jpeg");
    WiFiClient client = server.client();
    frame->writeTo(client);
}

void handleJpg() {
    if (!esp32cam::Camera.changeResolution(RES)) {
        Serial.println("CAN'T SET RESOLUTION!");
    }
    serveJpg();
```

```
}

void initCamera() {
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(RES);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);
    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
}

void initAP() {
    WiFi.softAP(AP_SSID, AP_PASS);
    IPAddress IP = WiFi.softAPIP();
        Serial.printf("Access Point IP address: %s\n",
IP.toString().c_str());
      Serial.printf("Open http://%s%s in your browser\n",
IP.toString().c_str(), URL);
}
void initServer() {
    server.on(URL, handleJpg);
    server.begin();
}
void setup() {
    Serial.begin(115200);
    initCamera();
    initAP();
    initServer();
}
void loop() {
    server.handleClient();
}
```

Кінець лістингу А.1

Лістинг А.2 – Код програми для клієнта

```
import cv2
import numpy as np
import urllib.request

# Camera URL
url = "http://192.168.4.1/cam.jpg"

# YOLO model files
weights_path = r"./YOLO/yolov3.weights"
config_path = r"./YOLO/yolov3.cfg"
names_path = r"./YOLO/coco.names"

# Load the YOLO model and COCO class names
net = cv2.dnn.readNet(weights_path, config_path)
with open(names_path, "r") as f:
    classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()

# Handling the return value of getUnconnectedOutLayers()
out_layers = net.getUnconnectedOutLayers()
if isinstance(out_layers[0], list):
    output_layers = [layer_names[i[0] - 1] for i in out_layers]
else:
    output_layers = [layer_names[i - 1] for i in out_layers]

# Generate random colors for each class
colors = np.random.uniform(0, 255, size=(len(classes), 3))

def detect_objects(frame):
    height, width, _ = frame.shape
    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
swapRB=True, crop=False)
    net.setInput(blob)
```

```
layer_outputs = net.forward(output_layers)

boxes = []
confidences = []
class_ids = []

for output in layer_outputs:
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.3:
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)

            x = int(center_x - w / 2)
            y = int(center_y - h / 2)

            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.3, 0.4)

# Draw detections on the frame
if len(indexes) > 0 and isinstance(indexes, np.ndarray):
    indexes = indexes.flatten()
    for i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        confidence = confidences[i]
        color = colors[class_ids[i]]
```

```
                    print(f"Detected: {label} with confidence
{confidence:.2f}")

                cv2.rectangle(frame, (x, y), (x + w, y + h), color,
2)
                cv2.putText(
                    frame,
                    f"{label} {confidence:.2f}",
                    (x, y - 10),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    0.5,
                    color,
                    2,
                )

        return frame

    def main():
        cv2.namedWindow("Object Detection", cv2.WINDOW_AUTOSIZE)

        while True:
            try:
                img_resp = urllib.request.urlopen(url)
                    imgnp = np.array(bytearray(img_resp.read()),
dtype=np.uint8)
                frame = cv2.imdecode(imgnp, -1)
                frame = detect_objects(frame)

                cv2.imshow("Object Detection", frame)

                if cv2.waitKey(1) & 0xFF == ord("q"):
                    break
            except Exception as e:
                print(f"Error occurred: {e}")
                break
```

```
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

Кінець лістингу А.2