

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ **Сергій СТИРЕНКО**
(підпис)

“__” _____ 2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

**за освітньо-професійною програмою “Комп’ютерні системи та мережі”
спеціальності 123 “Комп’ютерна інженерія”**

на тему: Система аналізу зображень з використанням мікроконтролера
STM-32

Виконав : студент 4 курсу, групи ІО-02
(шифр групи)

_____ **Корнієнко Дмитро Васильович**
(прізвище, ім’я, по батькові) (підпис)

Керівник _____ **ас. Гончаренко Олександр Олексійович**
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) с.в. Виноградов Юрій Миколайович
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ **к.т.н, Шимкович Володимир Миколайович**
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2024 р.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТРЕНКО

(підпис)

“ ” _____ 2024 р.

ЗАВДАННЯ

на бакалаврський дипломний проєкт студента

Корнієнко Дмитро Васильович

1. Тема проєкту Система аналізу зображень з використанням мікроконтролера STM-32
керівник проєкту ас. Гончаренко Олександр Олексійович
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 27 травня 2024 року №2112-с
2. Термін здачі студентом закінченого проєкту «02» червня 2024 р.
3. Вихідні дані до проєкту технічна документація, теоретичні дані для аналізу зображень.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
Огляд існуючих рішень аналізу зображень для мікроконтролерів STM-32.
Огляд алгоритмів та технологій при розробці системи.

Деталі розробки системи.

Тестування та аналіз розробленої системи.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) схема аналізу зображень (структурна схема), фільтрація зображення (функціональна схема), алгоритм взаємодії аналізу зображень (принципова схема)

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Виноградов Ю. М.		

7. Дата видачі завдання «6» жовтня 2023 р.

Календарний план

№ П/П	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	<i>Затвердження теми проєкту</i>	<i>10.12.2023-10.03.2024</i>	
2.	<i>Вивчення та аналіз завдання</i>	<i>15.12.2023-15.03.2024</i>	
3.	<i>Розробка архітектури та загальної структури системи</i>	<i>15.03.2024-25.03.2024</i>	
4.	<i>Розробка структур окремих підсистем</i>	<i>25.03.2024-05.05.2024</i>	
5.	<i>Програмна реалізація системи</i>	<i>05.04.2024-15.05.2024</i>	
6.	<i>Оформлення пояснювальної записки</i>	<i>15.04.2024-03.06.2024</i>	
7.	<i>Захист програмного продукту</i>	<i>03.06.2024</i>	
8.	<i>Передзахист</i>	<i>12.06.2024</i>	
9.	<i>Захист</i>	<i>17.06.2024</i>	

Студент-дипломник _____ Дмитро КОРНІЄНКО
(підпис)

Керівник проєкту _____ Олександр ГОНЧАРЕНКО
(підпис)

АНОТАЦІЯ

У данній дипломній роботі подвійного призначення, розглянемо працюючу модель мікроконтролера STM-32, що вміє аналізувати зображення. Піднімемо тему використання технологій штучного інтелекту, зокрема комп'ютерного зору, для пошуку об'єктів. Розберемось в поняттях глибиного аналізу та побудови нейронних мереж. Проговоримо галузі використання системи та інструменти, що будуть необхідні для створення. Проведемо порівняльну характеристику між існуючими рішеннями, передивимось релевантність обраних мною конфігурацій. Надам документацію та інструкцію по використанню даної моделі, для потенційної мілітаризації

Ключові слова: аналіз зображень, ШІ, розпізнавання об'єктів, мікроконтролер STM32, C.

ANNOTATION

I In this dual-purpose thesis, we will look at a working model of an STM-32 microcontroller capable of analyzing images. We will raise the topic of using artificial intelligence technologies, in particular computer vision, to search for objects. Let's understand the concepts of deep analysis and construction of neural networks. Let's talk about the areas of use of the system and the tools that will be needed for creation. Let's conduct a comparative analysis between the existing solutions, review the relevance of the configurations I have chosen. I will provide documentation and instructions for using this model for potential militarization.

Keywords: image analysis, AI, object recognition, STM32 microcontroller, C.

[illegible]

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «Система аналізу зображень з використанням мікроконтролера
STM-32»

Київ – 2024

ЗМІСТ

НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ	2
ПІДСТАВИ ДЛЯ РОЗРОБКИ.....	2
МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ	2
ДЖЕРЕЛА РОЗРОБКИ.....	2
ТЕХНІЧНІ ВИМОГИ.....	3
Вимоги до розробленого продукту	3
Вимоги до програмного забезпечення	3
Вимоги до апаратної частини	3
ЕТАПИ РОЗРОБКИ	3

					ІАЛЦ.467200.002 ТЗ								
		№ докум.	Підпис	Дата									
		Корнієнко Д. В.			Аналіз зображень за допомогою Мікроконтролера STM-32 Технічне завдання				Літ.	Аркуш	Аркушів		
Перевірив		Гончаренко О. О.									1	3	
Реценз.									КПІ ім. Ігоря Сікорського, ФІОТ, гр. ІО-02				
Н. Контр.		Виноградов Ю. М.											
Затвердив													

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Технічним завданням для мого проекту є покращення та імплементація штучного інтелекту для розпізнавання об'єктів на фото за допомогою мікроконтролеру STM32, на базі STM32F767VGT6

Областю застосування системи варіюється від призначення, оскільки вона може підійти, як і для військового призначення: аналіз позицій військових, аналіз піхоти, аналіз техніки, так і для цивільного використання: аналіз машин, аналіз людей, аналіз фруктів, тощо.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є покращення та оптимізації штучного інтелекту на базі мікроконтролерів STM-32, а також для виконання завдання бакалаврського дипломного проєкту, який затверджує факультетом "Інформатики та обчислювальної техніки" Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського» з написання дипломної роботи за темою подвійного призначення.

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Мета та призначенням даної роботи є розробка модельної працездатної мікроконтролерної системи, STM32 типу, для аналізу зображень Даний пристрій має подвійне призначення, тож може бути корисним у військовому та цивільному житті. Для тестування системи обрано, простіші об'єкти, але список об'єктів можна розширити та адаптувати за потребою.

4 ДЖЕРЕЛА РОЗРОБКИ

Як джерела розробки було використано усе що є у вільному доступі в мережі Internet, різні OpenSorce проєкти, наукові роботи, офіційні документації, наукова література.

					ІАЛЦ.467200.002 ТЗ	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги, для її тестування:

- Аналіз зображень, навіть тих, які вбудований ШІ ніколи не бачив;
- Імітація підключення файлів зображень до USB;
- Зчитування зображення та аналіз зображення, на які користувач не може впливати;
- Користувач системою, повинен мати змогу змінювати конфігурацію для об'єктів (список об'єктів, що потребує аналізу);
- Зберігання зображень, що були проаналізовані, але видалення їх з USB-носія для завантаження на них результатів аналізу;
- Документація для розробленого продукту, яка допоможе оптимізувати, покращити або зрозуміти принцип дії;

5.2. Вимоги до програмного забезпечення

- ОС Windows 11, Mac, або Linux.
- STM32CubeIDE 1.14.1 версії або вище.

5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel® Core (TM) i3-2100T.
- ЗП не менше ніж 64 ГБ.
- RAM не менше ніж 4 ГБ.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	10.12.2023-15.12.2023
Вивчення та аналіз завдання	15.12.2023-15.03.2024
Розробка архітектури та загальної структури системи	15.03.2024-25.03.2024
Розробка структур окремих частин системи	25.03.2024-5.04.2024
Програмна реалізація системи	5.04.2024-15.04.2024
Виправлення помилок	15.04.2024-20.05.2024
Оформлення пояснювальної записки	25.04.2024

					ІАЛЦ.467200.002 ТЗ	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

**ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ**

на тему: «Еволюційні алгоритми глобальної пошукової оптимізації»

Київ – 2024

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....	3
ВСТУП.....	4
РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ АНАЛІЗУ ЗОБРАЖЕНЬ ДЛЯ МІКРОКОНТРОЛЕРІВ STM-32	6
1.1 Огляд мікроконтролерів STM-32.....	6
1.1.1 Архітектура та основні характеристики мікроконтролерів STM32	6
1.1.2 Порівняння різних серій мікроконтролерів STM32	10
1.1.3 Вибір конкретного мікроконтролера	11
1.2 Огляд методів аналізу зображення.....	12
1.2.1 Фільтрація зображень.....	13
1.2.2 Бінаризація зображень.....	15
1.2.3 Сегментація зображень	16
1.2.4 Розпізнавання об'єктів.....	18
1.3 Огляд інших систем аналізу зображень.....	19
1.4.1 CMSIS-DSP	21
1.4.2 TensorFlow Lite for Microcontrollers	22
1.4.3 OpenMV	24
ВИСНОВОК ДО РОЗДІЛУ 1	26
РОЗДІЛ 2. ОГЛЯД АЛГОРИТМІВ ТА ТЕХНОЛОГІЙ ПРИ РОЗРОБЦІ СИСТЕМИ	29
2.1 Машинне навчання.....	29
2.1.1 Основні поняття	29
2.1.2 Використання DL у TensorFlow	30

					ІАЛЦ.467200.003 ПЗ						
		№ докум.	Підпис	Дата							
Розробив		Корнієнко Д. В			Аналіз зображень за допомогою Мікроконтролера STM-32 Пояснювальна записка			Літ.	Аркуш	Аркушів	
Перевірив		Гончаренко О.								1	76
Реценз.								НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-02			
Н. Контр.		Виноградов Ю. М.									
Затвердив											

2.2 Аналіз зображень	33
2.3 Технології та бібліотеки	37
2.4 STM32CubeMX та STM32CubeIDE.....	37
2.5 X-CUBE-AI	38
3.3 ST_SSD_MOBILENET_V1.....	39
ВИСНОВОК ДО РОЗДІЛУ 2	40
РОЗДІЛ 3. ДЕТАЛІ РОЗРОБКИ СИСТЕМИ.....	42
3.1 AI технологія комп'ютерного зору	42
3.1.1 Вибір інструменту для формування FTLite моделі	42
3.1.2 Інтеграція моделі FTLite у STM32CUBEMX.....	46
3.2 Мікроконтролер STM32 та код.....	48
3.2.1 Загальний опис та задачі	48
3.2.3 Функція соруDir	52
3.2.4 Функція processImage	54
3.2.5 Функція AnalyzeImageWithBoundingBox	56
3.2.5 Функція VisualizeResults	59
3.2.5 Функція main	60
ВИСНОВОК ДО РОЗДІЛУ 3	61
РОЗДІЛ 4. ТЕСТУВАННЯ ТА АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ.....	63
4.1 Тестування та аналіз main_model	63
4.2 Тестування розробленої системи.....	66
ВИСНОВОК ДО РОЗДІЛУ 4	69
ВИСНОВКИ.....	71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	72

ПЕРЕЛІК СКОРОЧЕНЬ

STM32	База мікроконтролерів, кожен з яких має документацію.
III (AI)	Система яка аналізує отримані данні, на базі свого досвіду.
ML	Машинне навчання
DL	Глибинне навчання
MCU	Microcontroller Unit. Мікроконтролер.
OpenSorce	Розробки з відкритим кодом та можливістю його модифікувати
DSP	Цифрова обробка сигналів
CNN	Конволюційні нейронні мережі
R-CNN	Регіональні CNN
YOLO	You Only Look Once. Нейромережа для аналізу об'єктів в реальному часі

ВСТУП

З популяризацією нейромереж, люди почали автоматизувати та спрощувати усе, що тільки можна. Без перебільшень, штучний інтелект, зараз – це нова революція, подібно до промислової у XVIII-XIX століттях. Невпинно, набираючи нових обертів, технології AI, отримують свої нові та революційні застосування. Комп'ютерний зір [1], про який буде іти мова у роботі, допомагає у багатьох сферах життя: автоматичний контроль за виробництвом, автономне водіння, діагностика захворювань, біометрична індикація, системи до наведення для воєнних дронів тощо.

Не надовго відходячи від теми штучного інтелекту, поговоримо про мікроконтролери. MCU – це така інтегрована мікросхема, які включають в себе процесор, пам'ять, виводи для підключення зовнішніх пристроїв, а також різноманітні периферійні пристрої такі як таймери, засоби зберігання даних, аналого-цифрові та цифро-аналогові конвертери, інтерфейси зв'язку тощо [2]. Цей винахід був створений ще у 80-х ХХ ст. але без того, постійно розвивається і набуває попит у різних сферах. Саме завдяки мікроконтролеру, в наше життя почались додаватись автоматизовані системи, незалежні від якоїсь центральної системи, які виконують прості задачі та виконують її самі по собі.

Поєднуючи прості MCU системи та AI технології створюють інноваційні, портативні та багатофункціональні розробки. Першочергова ціль – розробити мікроконтролерну систему, що реалізує аналіз зображень за допомогою комп'ютерного зору. Така система допоможе аналізувати різні об'єкти та зрештою автоматизувати взаємодії між мікроконтролерними системами у майбутньому, тому ця тема є актуальною, ба більше, користується великим попитом та залучає все більше ресурсів

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		4

Підтвердженням актуальності моєї теми є набуття популярності розробці безпілотних дронів, під час війни між Росією та Україною. Саме такі дрони базуються на мікроконтролерах, а доповнюючи їх комп'ютерним зором – можна вивести системи на новий рівень. Раніше дрони, в більшості, були потрібні лише, як інструмент створення красивих фото або відео, але тепер вони зарекомендували себе, як гарний військовий інструмент для розвідки, знищення техніки або сил супротивника. Зараз ІІІ в дронах збільшує ефективність їх використання у цій сфері, оскільки вміє донаводити себе до цілі.

Метою моєї роботи є автоматизація процесів аналізу зображень, за допомогою комп'ютерного зору. Дана система розробляється з метою популяризації систем пошуку об'єктів на фото, використовуючи спеціальні портативні засоби, зокрема мікроконтролери. Також акцентую увагу на темі подвійного призначення, оскільки популяризація розроблених систем STM-32, для аналізу зображень, наближає Україну до перемоги.

РОЗДІЛ 1

ОГЛЯД ІСНУЮЧИХ РІШЕНЬ АНАЛІЗУ

ЗОБРАЖЕНЬ ДЛЯ МІКРОКОНТРОЛЕРІВ STM-32

1.1 Огляд мікроконтролерів STM-32

1.1.1 Архітектура та основні характеристики мікроконтролерів STM32

Слід зазначити, що усі мікроконтролери поділяються за багатьма ознаками. Мікроконтролерів STM32 безліч, навіть не враховуючи ті, що вже не підтримуються компанією STMicroelectronics, офіційним розробником даних мікроконтролерів.

Особливістю обраного мною типу мікроконтролерів є ядро Cortex-M. Cortex-M - це сімейство 32-розрядних RISC процесорних ядер, які призначених для використання в мікроконтролерах STM32. Ядра Cortex-M відомі своєю високою продуктивністю, енергоефективністю та універсальністю, що робить їх популярним вибором для широкого спектра застосувань [3].

Слід трохи поясню про RISC (Reduced Instruction Set Computing) архітектурні системи, для повного розуміння переваги, наприклад, над CISC. RISC - архітектура процесорів зі скороченим набором команд. Її друга назва архітектура load-store, оскільки система команд цієї архітектури не включає арифметико-логічних операцій з операндами у пам'яті. Перед початком виконання операції, дані спочатку завантажуються в регістр, далі виконуються операції, а тоді відбувається процес збереження, назад, у пам'ять. Ця архітектура, використовує набір

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		6

інструкцій меншого розміру та більшої простоти, якщо порівнювати з CISC (Complex Instruction Set Computing). Це дозволяє заощаджувати час та бути більш енергоефективною [22].

Окрім RISC архітектури хочеться зазначити про різноманітність ядер. Ці ядра можуть слугувати для різних задач, для виконання певних потреб. Вони відрізняються за багатьма ознаками – функціональністю, продуктивністю, обчислювальною здатністю, тощо. Найпоширенішими ядрами є Cortex-M0, Cortex-M3, Cortex-M4 та Cortex-M7. Під час навчання в КПІ, розглядав ядра типу Cortex-M3 та Cortex-M4, але для виконання, зараз, взяв більш потужніше ядро Cortex-M7. Саме ядро M7 є найбільш потужнішим і підходящим по функціоналу, для виконання мною поставлених задач. Звичайно можна було обрати будь-яке ядро, з перелічених, але тоді продуктивність та швидкість обробки значно впаде, натомість, ціна зменшиться, а енергоспоживання знизиться із-за зменшення функціоналу [23].

Нижче хотів би поговорити про конкретні переваги та рішення Cortex-M, що надають переваги та змушують обрати цей вид мікроконтролерів поміж інших. Не менш вагомою ключовою особливістю ядер Cortex-M є забезпечення енергоефективності. Даний тип MCU мають ряд функцій енергозбереження, таких як режими сну або різкої зупинки, які дозволяють їм мінімізувати споживання енергії, при неактивні. Ця особливість ідеально підійде для портативних пристроїв та пристроїв з батарейним живленням, що за ідеєю може сприяти підняти, їх як дрони, в небо.

Ядра Cortex-M можуть забезпечити високу продуктивність для вбудованих систем. Вони здатні виконувати складні обчислення та обробляти дані в реальному часі, оскільки на це були і розраховані з самого початку створення.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		7

Важливо знати про особливості пам'яті у мікроконтролерах, адже при вивчені питання аналізу зображень помітив закономірність – чим більша пам'ять тим швидша обробка зображення. В прикладі “How to run OpenCV on STM32 MCU” [3] автора Дениса Дерюгіна можемо що об'єм пам'яті мікроконтролера STM32F7Discovery менший за модель STM32F769I-Discovery, але час підрахунку більший в 13.3 рази! (STM32F7Discovery – обробляє зображення за 40 секунд, а STM32F769I-Discovery за 3), але, напевно, про практику поговоримо далі, а зараз сконцентруємось на теорії

Мікроконтролери STM32 мають два основних типи пам'яті:

- Flash-пам'ять: використовується для зберігання програмного забезпечення та даних, які не змінюються часто. Flash-пам'ять є енергонезалежною, тобто дані зберігаються навіть після вимкнення живлення.
- SRAM: використовується для зберігання даних, які часто змінюються, таких як змінні та тимчасові буфери. SRAM є енергозалежною, тобто дані втрачаються після вимкнення живлення.

Легке підключення різних периферійних пристроїв та мікроконтролерів до інших пристроїв, дозволяє легко створювати цілі системи. З самого початку розробки були ідеї розробки камери на базі мікроконтролера, та окремого обробника даних, для реалізації “системи аналізу зображень”, але оскільки метою роботи є інша задача мою систему було спрощено та адаптовано під інші входи та виходити для інших пристроїв (Див. Розділ 3)

До того ж, розглянемо деякі з найпоширеніших інтерфейсів, для розуміння, що з себе вони представляють та які функції виконують

- UART: використовується для послідовного зв'язку з іншими пристроями, такими як комп'ютери та іншими мікроконтролерами.

- SPI: інструмент, який застосовується для синхронного послідовного зв'язку з іншими пристроями, такими як датчики та АЦП.
- I2C: потрібен для подвійно-направленого послідовного зв'язку з пристроями, які підтримують протокол I2C.
- CAN: інтерфейс з'єднання для високошвидкісного послідовного зв'язку у стільникових мережах.
- USB: дуже поширений та відомий інтерфейс обміну даних застосовується для універсального зв'язку з комп'ютерами та іншими пристроями.
- Ethernet: використовується для підключення до мереж Ethernet.

Зважаючи на задачі, треба чітко усвідомлювати, що саме потребує підключення, оскільки деякі підключення можуть конфліктувати між собою. В моєму випадку була ситуація де при обраних мною конфігураціях, ніяк не можна було обрахувати частоту оновлення даних на одному з порту USB, саме тому більше надаю перевагу залишити лише одне гніздо для підключення USB

ARM та інші виробники надають великий список інструментів розробки та бібліотек програмного забезпечення для ядер Cortex-M. Це полегшує розробникам створення програмного забезпечення для своїх застосувань.

А ще, дуже чітка документація та описи до кожного мікроконтролера. Активна підтримка, можливість отримання певних послуг за запитом змушують обирати саме цей тип мікроконтролерів.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		9

1.1.2 Порівняння різних серій мікроконтролерів STM32

В свою чергу мікроконтролери STM-32 поділяються на серії, які класифікують їх особливості, переваги та недоліки.

STMicroelectronics пропонує широкий спектр серій мікроконтролерів, саме тому треба підходити до вибору потрібної серії, усвідомлюючи усі особливості, усвідомлюючи обсяг виконання завдання.

Нижче наведена порівняльна характеристика до найпопулярніших серій мікроконтролерів [14]

Таблиця 1 - Серії мікроконтролерів STM32 та їх особливості

Серія	Ядро	Тактова частота	Пам'ять	Периферійні пристрої	Особливості
STM32F0	Cortex-M0	До 8 МГц	До 32 КБ Flash, 6 КБ SRAM	ADC, ШИМ, UART, SPI	Низька вартість, низьке споживання енергії
STM32F1	Cortex-M3	До 72 МГц	До 128 КБ Flash, 32 КБ SRAM	ADC, ШИМ, UART, SPI, CAN, USB	Баланс продуктивності та енергоефективності
STM32F2	Cortex-M3	До 120 МГц	До 2 МБ Flash, 256 КБ SRAM	ADC, ШИМ, UART, SPI, CAN, USB, Ethernet	Висока продуктивність, мультимедійні можливості
STM32F3	Cortex-M4	До 72 МГц	До 1 МБ Flash, 192 КБ SRAM	ADC, ШИМ, UART, SPI, CAN, USB	Низьке споживання енергії, висока продуктивність
STM32F4	Cortex-M4	До 180 МГц	До 2 МБ Flash, 512 КБ SRAM	ADC, ШИМ, UART, SPI, CAN, USB, Ethernet	Найвища продуктивність, мультимедійні можливості
STM32L0	Cortex-M0+	До 32 МГц	До 64 КБ Flash, 16 КБ SRAM	ADC, ШИМ, UART, SPI	Низька вартість, ультра низьке споживання енергії
STM32L1	Cortex-M3	До 32 МГц	До 512 КБ Flash, 128 КБ SRAM	ADC, ШИМ, UART, SPI, CAN	Низьке споживання енергії, висока продуктивність
STM32L4	Cortex-M4	До 80 МГц	До 2 МБ Flash, 512 КБ SRAM	ADC, ШИМ, UART, SPI, CAN, USB	Низьке споживання енергії, висока продуктивність
STM32H7	Cortex-M7	До 480 МГц	До 8 МБ Flash, 2 МБ SRAM	ADC, ШИМ, UART, SPI, CAN, USB, Ethernet, FPU	Найвища продуктивність, безпека, мультимедійні можливості

1.1.3 Вибір конкретного мікроконтролера

Між тим, перейдемо до більш прикладних ознак і зумовимо конкретикою вибір мого процесора, як думав та розмірковував, порівнюючи з іншими мікроконтролерами.

На що треба зважати при виборі? Після того як зрозумів, що саме мені потрібно для реалізації мого проекту, поступово почав шукати та аналізувати серії, що надавали мені необхідний функціонал, також хотів використати усі інтерфейси по повній, додаючи додаткову логіку, інтегруючи більше фіч, треба було знайти найкращий. Моє око впало на серію STM32F7xx

Як приклад, було обрано для аналізу переваг-недоліків STM32F7xx та STM32H7xx, що була згадана раніше.

Переваги STM32F7xx:

1. Дана модель має високу продуктивність. Мікроконтролери STM32F7xx мають високу частоту тактової частоти та потужність обчислення, що робить їх ідеальними для завдань, які вимагають великої продуктивності.

2. Ця лінійка має широкі можливості периферійних пристроїв, таких як USB, Ethernet, CAN, UART, SPI, I2C тощо.

3. Що є досить цікавим - наявність вбудованого LCD контролера. Вбудований контролер для роботи з LCD дисплеями, розширює ідею обробки зображень, а також може полегшити користування тих задач, які вимагають графічного інтерфейсу.

Переваги серії STM32H7xx:

1. Загалом, це найпродуктивніші процесори серед лінійки STM32. Дані мікроконтролери мають ще більшу швидкість тактової частоти і

більшу кількість кеш-пам'яті, що робить їх найпотужнішими мікроконтролерами в лінійці STM32.

2. Висока продуктивність в обробці сигналів дозволяє STM32H7xx мати обчислювальні блоки для обробки сигналів (DSP), а також апаратну підтримку векторизованих операцій (M7 Core), що робить їх ідеальними для застосувань в сфері обробки сигналів та машинного навчання.

3. Підтримка захисту пам'яті і безпеки. Мікроконтролери STM32H7xx мають розширені функції захисту пам'яті та безпеки, що робить їх ідеальними для застосувань, які вимагають високого рівня безпеки, таких як фінансові або медичні пристрої.

Оскільки мені потрібна висока продуктивність та широкі можливості периферійних пристроїв - STM32F7xx підходить найкраще і по ціні, так і по базовому набору. Хоча і основною перевагою H7 є машинне навчання, ставку на неї, я не став в силу більш дешевших моделей, в більшості робити акцент саме на аналіз зображень.

1.2 Огляд методів аналізу зображення

Розпізнавання зображень — це розділ сучасного штучного інтелекту, що дозволяє комп'ютерам ідентифікувати або розпізнавати візерунки, елементи, пікселі, об'єкти в цифрових зображеннях. Розпізнавання зображень дає змогу комп'ютерам ідентифікувати об'єкти, людей, місця та тексти на будь-якому зображенні.

Але існує багато методів індикації, що не пов'язані з штучним інтелектом, що базується на математичних та формують в подальшому розуміння логіки розпізнавання об'єктів на зображенні [25]

Обробляти зображення можна по різному, це не секрет, що штучний інтелект вміє покращувати зображення, визначати колір, знаходити

об'єкти, але варто розберімось, які поміж тим методи аналізу взагалі існують.

Спочатку, розглянемо більш математичні та загальні методи, що в своїй суті слугують більш детальнішому та важчому аналізу зображень за допомогою штучного інтелекту.

Це дуже головна тема для розуміння логіки аналізу зображень у дипломній роботі, саме в цьому пункті детально проговоримо про механізми аналізу зображення, розглянемо процеси, порівняємо логіку та з'ясуємо можливості усіх цих методів

1.2.1 Фільтрація зображень

Фільтрація зображень – це процес обробки зображень, що підвищує якість, шляхом видалення або заміни певних недоліків. Основна ідея використання цього методу заснована на математичних операціях, спробі прибрати зайвий шум, підсиленні насиченості та видаленні зайвих пікселів на зображенні.



Рисунок 1.1 – Фільтрація зображення на прикладі спеціальної програми.

Різні типи фільтрів виконують свої задачі, розберемось в кожному:

- Лінійна фільтрація використовує лінійні фільтри, такі як фільтр Гауса, фільтр Собеля, фільтр медіани та інші. Це необхідно для обробки кожного пікселя на зображенні. Саме за допомогою фільтрів відбувається процес згладжування зображення, видалення шуму та виділення зайвих елементів.
- Нелінійна фільтрація включає в себе застосування нелінійних алгоритмів, таких як медіанний фільтр. Цей тип фільтру також націлений на покращення зображення та видаленню шуму, але робить це використовуючи медіанне значення з певного оточення пікселя.
- Морфологічна фільтрація застосовується для обробки об'єктів на зображенні, операції збільшення або зменшення зображення гарно описують результат цього алгоритму фільтрації. Також алгоритм зосереджений на видаленні малих деталей або з'єднанні розділених об'єктів.
- Частотна фільтрація потрібна для аналізу та зміни частотної складової зображення. Перевагою такої фільтрації є методи фільтрації низьких, середніх та високих частот. За допомогою використання цього фільтру отримаємо підсилення тих чи інших елементів, набуття насиченості.

Робота фільтрації зображень працює за принципом обробки кожного пікселя, що, в свою чергу, дозволяє змінювати значення пікселя аналізуючи оточуючі пікселі до нього[17, 26].

1.2.2 Бінаризація зображень

Бінаризація - це процес перетворення кольорового на дві кольорові категорії (чорної та білої). Кожен піксель є чорним або білим, це залежить від межового значення для набуття того чи іншого кольору.

Основна мета цього математичного процесу слугує для виділення об'єктів на зображенні, виокремлення об'єктів від фону, або, навіть банальної підготувати зображення для подальшого аналізу.



Рисунок 1.2 – Бінаризація зображення. а) необроблене зображення. б) оброблене зображення.

Зараз розглянемо різні методи бінаризації зображень, розглянемо їх особливості:

- Глобальна бінаризація – це метод, що біналізує зображення, що підлаштовуючи поріг аналізуючи усе що є в зображенні. Даний метод ідеально підійти для зображень з однорідним фоном або контрастними об'єктами.
- Локальна бінаризація підлаштовує свій поріг бінаризації для кожного пікселя окремо, дивлячись і аналізуючи його оточення. Метод ефективний для зображень з нерівномірним освітленням або низьким контрастом.
- Адаптивна бінаризація комбінує глобальний та локальний підхід, аналізує та створює глобальний поріг для обробки основної частини зображення та додатково локальний – щоб відокремити об'єкти між собою, або різні області зображення.

- Статична бінаризація. Ніхто не забороняє виставити значення порогу в ручну, що може бути корисним для експертів по обробці зображень, цей поріг можна виставити за допомогою статистичних характеристик зображення, середнього значення або дисперсії пікселів.

Після застосування бінаризації зображення стає бінарним, що означає кожен піксель класифікується як чорний або білий, тобто 0 або 1. Цей процес часто застосовують в аналізі зображень для виділення об'єктів, шумів або для підготовки даних до іншого виду обробки[18].

1.2.3 Сегментація зображень

Сегментація, безумовно, є процесом, що розділяє зображення на частини або області, які мають подібні властивості. Мета такого аналізу досить проста – виділення необхідних об'єктів або зон, для подальшого аналізу. Також хочу зазначити, що існує багато методів сегментування, спробуємо розглянути найпопулярніші.

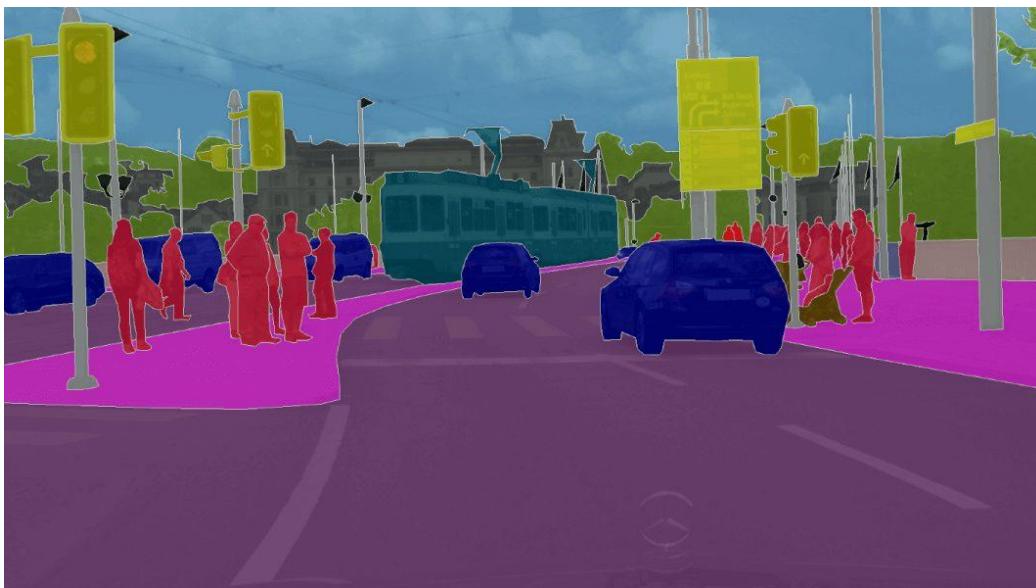


Рисунок 1.3 – Нейромережа сегментує зображення.

Дана обробка є більш складною та комплексною обробкою зображень, що включає методи, що були описані раніше.

- Вище згаданий алгоритм бінарзації дуже гарно справляється з пошуком та об'єднанням сегментів, тут такий самий принцип пошуку порогу, можна використовувати будь який з типів.
- Кластеризація – досить відомий та поширений алгоритм обробки, який базується на принципі групування пікселів за кольором, яскравістю, або іншою кольоровою властивістю. Вибір кількості кластерів робить цю модель більш гнучкою до багатозадачного аналізу прикладом якого можемо бачити на рис. 1.2.
- Алгоритм вододілу працює за принципом пошуку найнижчої яскравості на зображенні. Математично він робить з зображення тривимірну поверхню, де третьою координатою виступає висота. Логіка цих дій проста – пошук меж об'єктів. Особливістю такої сегментації можна вважати розбиття зображення на велику кількість сегментів.
- Метод Graph Cuts використовує логіку графів. Принцип дії досить математичний і пов'язаний з будівництвом графу між пікселями (вершинами графа) за допомогою подібності між ними (ребра графа), а далі сегментуємо завдяки розділенню графа на мінімально зв'язані елементи. Досить математичний метод, який при прямій обробці фото буде займати дуже багато часу, тому використовується в підходящих для нього умовах.
- Convolutional Neural Networks (CNN) метод використовує глибокі нейронні мережі для сегментації зображень. Мають високу точність, але потребують багато часу для реалізації повної моделі.

Є ще багато алгоритмів сегментування, але, у роботі, намагався обрати більш унікальні та цікаві методи.

На чому хотів би наголосити, так це на тому, що сегментація зображень є дійсно ключовим етапом аналізу зображень, оскільки воно дозволяє виділити значущі об'єкти або регіони. Вибір конкретного методу залежить від особливостей задачі та типу зображення, яке обробляється. Розуміння різних підходів до сегментації допоможе вибрати оптимальний метод для конкретних потреб проекту [19].

1.2.4 Розпізнавання об'єктів

Розпізнавання об'єктів на зображенні - процес виявлення та класифікації об'єктів, що дозволяє системам автоматично розуміти реагувати на надану візуальну інформацію. Прикладами використання таких систем є автономні транспортні засоби, медична діагностика, системи відеоспостереження, дрони розвідники з донаведенням, тощо [21].

Цього разу, не буду перераховувати алгоритми та принципи дії лише зауважу, що більшості усе зводиться до комп'ютерного зору.

Комп'ютерний зір — це інструмент штучного інтелекту, який автоматично отримує, обробляє та аналізує зображення та відео. За допомогою цього зору у комп'ютерів та інших системам (до прикладу, мікроконтролерних) відкривається здатність бачити та розуміти об'єкти та сукупність елементів на фото.

Не дарма, було акцентовано на основні методи обробки зображень, оскільки логіка комп'ютерного зору діє за принципом периферійної обробки, або навіть на сегментуванням. Зараз хотів би розвинути тему методів глибинного навчання, яка була трохи піднята вище, оскільки вже було сказано про CNN, які потрібні для автоматичного вилучення ознак та

класифікації об'єктів на зображеннях, але варто згадати про інші моделі такі як R-CNN, Fast R-CNN, Faster R-CNN, які заточенні для виявлення об'єктів на фото. Чого вартує обговорення нейронних систем YOLO, які є швидким методом для виявлення об'єктів у реальному часі. І як приклад імплементації аналізу сегментів за допомогою нейромереж може виступити – UNet.

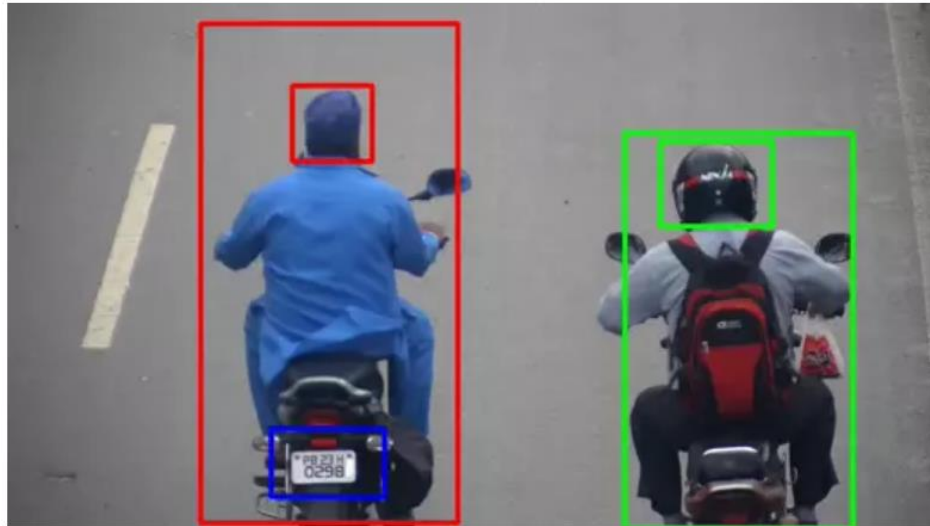


Рисунок 1.4 – Комп'ютерний зір розпізнає об'єкти на зображенні.

Усе це підкреслює, що використовуючи машинне навчання та наявні нам методи та алгоритми обробки зображення, комбінуючи їх, можемо отримувати дуже потужні системи, які можуть аналізувати зображення з дефектами, з поганим освітлення, а також чітко визначати об'єкти показавши їх нейронним мережам [20, 21].

1.3 Огляд інших систем аналізу зображень

Окрім автопілот Tesla з його можливістю розпізнавати об'єкти [28] поза межами машини, існує безліч інших систем аналізу зображень, які

застосовуються в різних галузях. Про це і буде йти мова в цій частині дипломної роботи.

1. Військова сфера. Тут такі системи застосовуються для згаданих вище, FPV дронів-камікадзе з системою авто-доведення. Це досить новітня розробка і звичайно приклад таких систем ніхто у відкритий доступ не надасть, оскільки це дозволить ворогу використовувати такі системи в своїх цілях.
2. Медичні сфера, а серед неї системи діагностики та системи аналізу зображень для діагностики захворювань на основі рентген, МРТ та КТ знімків. Ці системи можуть автоматично виявляти аномалії та допомагати лікарям ставити діагнози. Також гарним прикладом буде системи хірургічної навігації, які допомагають під час операцій. Такі системи можуть створювати 3D-зображення анатомії пацієнта, що допомагає хірургам та лікарям планувати та виконувати операції з більшою точністю, зменшуючи ризики.
3. Безпекова сфера. Гарним прикладом буде системи для розпізнавання обличчя які гарно підходять для ідентифікації людей для пошуку зловмисника, або відслідковування їх дій. Такі системи застосовуються у Китаї. Системи можуть використовуватися для контролю доступу, верифікації особистості та розслідування злочинів. Окрім цього можна відслідковувати і шукати іншу людину з камер, в певний момент часу, щоб зловмисник не пройшов. Подібні системи можуть бути застосовані для аналіз зображень для виявлення підозрілої поведінки та відстеження злочинців.
4. Промислова галузь: Тут подібні системи аналізу зображень гарно себе показують у питанні контролю якості продукції та виявлення візуальних дефектів. Подібні системи можуть допомогти

виробникам випускати більш якісну продукцію та знизити витрати на ручну перевірку

Поміж усім хочу наголосити, що таких систем дійсно безліч, вони корисні у своїх сферах, адже виконують свою основну роль та призначення. Порівнювати і казати, що краще, а що гірше немає сенсу, оскільки вони виконують свої функцію.

1.4 Огляд бібліотек та інструментів для аналізу зображень за допомогою мікроконтролера STM32

З чого б хотів почати, так це з пояснення чому не усі бібліотеки та інструменти можуть підійти для реалізації аналізу зображень на мікроконтролерних системах, усе дуже просто – обмеженість пам'яті. Внутрішня пам'ять мікроконтролера не перевищує 8мб, і то, це якщо враховувати, що беремо найдорожчі моделі. З серії та процесору, що було обрано у даній роботі, ми маємо 2мб на обробку та аналіз зображень, вже не говорячи про те, що саме за допомогою завантаження даних файлів у цю пам'ять ми будемо аналізувати зображення. Тому треба шукати та оптимізувати бібліотеки та інструменти під MCU, благо такі рішення є, розглянемо їх детальніше. Спробую дуже коротко.

1.4.1 CMSIS-DSP

CMSIS-DSP розшифровується, як Cortex Microcontroller Software Interface Standard - Digital Signal Processing. Ця бібліотека, розроблена ARM для роботи DSP на мікроконтролерах Cortex-M. Цей інструмент

містить оптимізовані функції для обробки зображень, такі як ми розглядали - фільтрація, трансформації та інші базові операції. Плюсом можна вважати підтримку широкого спектра функцій DSP. Висока продуктивність завдяки оптимізації під ARM Cortex-M ядра. Досить зручна інтеграція з іншими компонентами CMSIS. Як ми розглядали раніше, дані алгоритми виконують попередню обробку зображень (фільтрація, згладжування), тож можуть спокійно виконувати функцію розпізнавання простих шаблонів, ознак, елементів та пікселів [29].

CMSIS Version 3 Structure

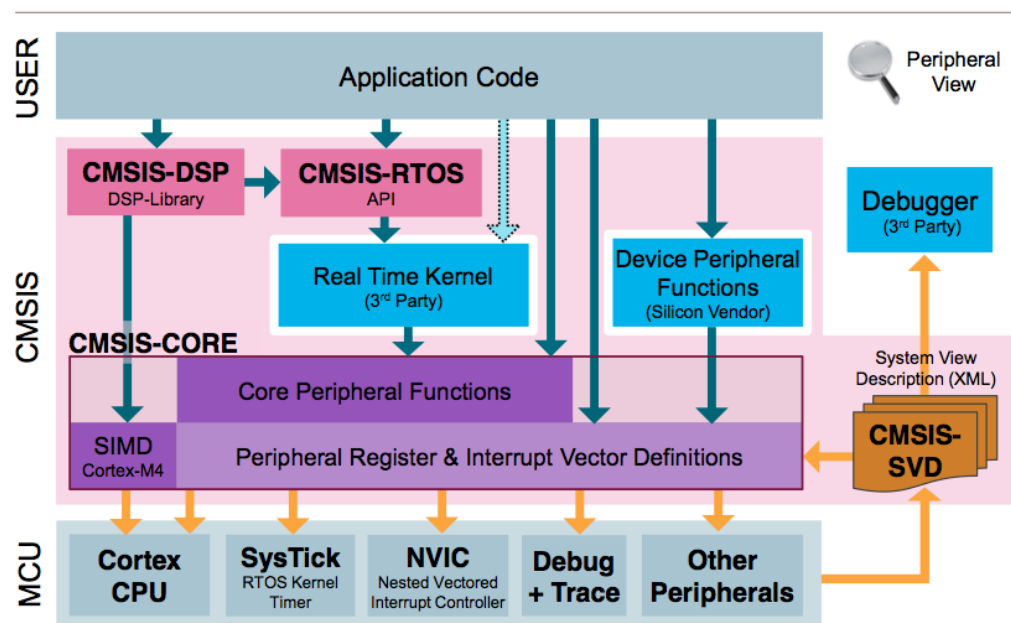


Рисунок 1.5 – Структурна схема застосування CMSIS v3.

1.4.2 TensorFlow Lite for Microcontrollers

Це варіація популярної бібліотеки TensorFlow, яка була оптимізована для роботи на мікроконтролерах, або інших пристроях з обмеженою кількістю пам'яті. Ця бібліотека дозволяє виконувати задачі машинного навчання, розпізнавання об'єктів на зображеннях, тощо. Особливостями

даного інструменту є: досить компактний розмір, підтримка моделей глибокого навчання, що можуть виконувати складні задачі аналізу зображень і роблять розпізнавання об'єктів більш точнішим. Також є вбудовані інструменти для квантованого навчання, які автоматично зменшують розмір моделей та підвищують ефективність системи. Але зрозуміло, що відмінністю від повної бібліотеки TensorFlow є урізання певних функцій, які не можливо реалізувати на даній спрощеній моделі. Але щоб підкреслити можливості даної системи попри свій зменшений функціонал дані системи спокійно можуть розпізнавати обличчя і об'єкти, а також гарно можуть підійти обробки зображень в режимі реального часу [30].

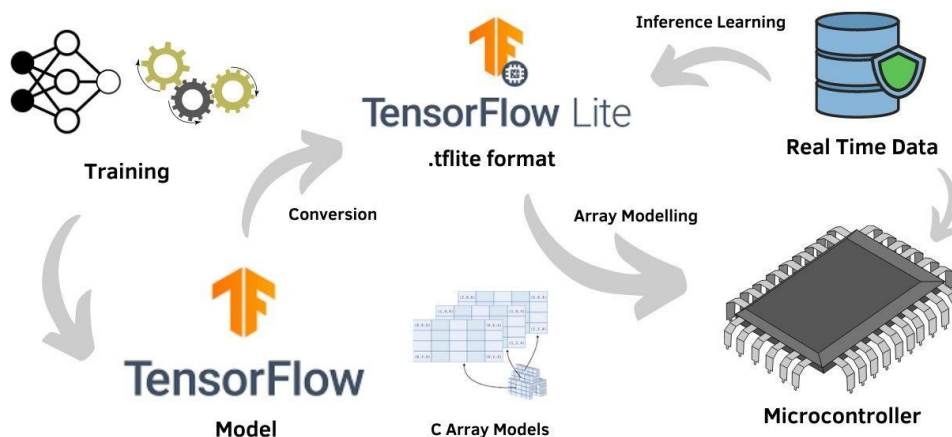


Рисунок 1.6 – Алгоритм дій від тренування системи до реалізації аналізу зображень на мікроконтролері для бібліотек TensorFlow

Як можна побачити на рис. 1.6, процес навчання реалізовується за допомогою загальної бібліотеки, далі вона конвертується у .tflite формат, що підтримується командами мікроконтролера. Єдиним моментом який

зустрічається на шляху розробки – це підтримка виключно мови C++, при реалізації та застосуванні бібліотеки TensorFlow Lite [4]

1.4.3 OpenMV

OpenMV — апаратно-програмний комплекс, який часто застосовується для обробки зображень в реальному часі, використовуючи камеру та мікроконтролер. Загалом платформа OpenMV дозволяє обробляти зображень та застосовувати комп'ютерний зір на мікроконтролері. Серед переваг можна зазначити безліч готових бібліотек для обробки зображення. Такі системи також застосовуються для виявлення руху та об'єктів, а також можуть виконувати більш простіші дії бінарізації та фільтрації, якщо потребуються такі рішення [32]. Легко відстежують кольорові об'єктів. Досить багато функціональна бібліотека з великою кількістю

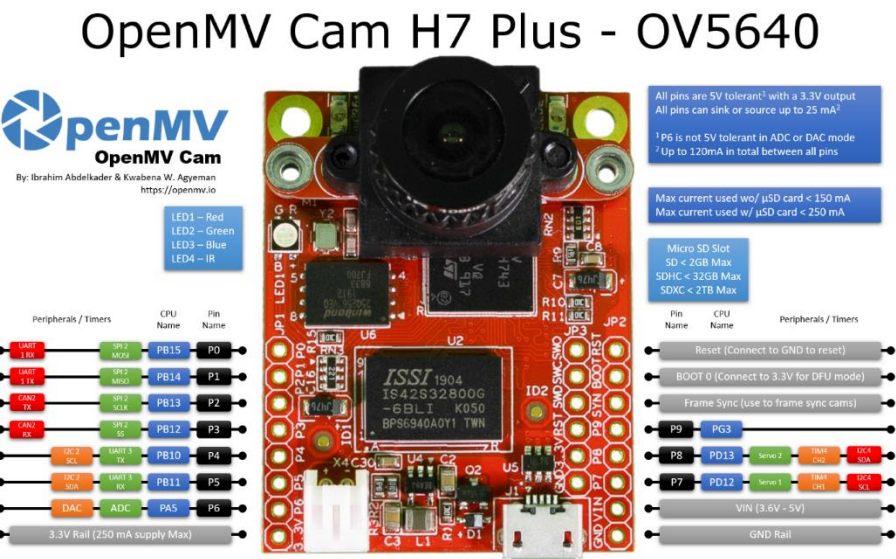


Рисунок 1.7 – Вигляд та характеристики готової OpenMV Cam системи для аналізу зображення

Унеможливлуючи фактором інтеграції бібліотеки MicroPython на MCU STM32 є не підтримка мови python STM32 мікроконтролерами[31]. Хоча у відкритому доступі, можна знайти рішення, які були засновані на спробі інтегрувати OpenMV до STM32 систем, що правда для обмеженої кількості процесорів

1.4.4 STMicroelectronics X-CUBE-AI

X-CUBE-AI — це програмний інструмент від розробників STMicroelectronics, який дозволяє імплементувати попередньо навчені нейронні мережі в код мікроконтролерів STM-32. Інструмент включає підтримку аналізу зображень за допомогою імплементції нейромережі, додавання “мозку” в середину мікроконтролера. Перевагою є велика документація та реалізація підтримка, навіть на платній основі. У STM32CubeMX вбудовану цю бібліотеку, тож не є проблемою скачати та додати її до потрібного вам мікроконтролера STM32, а далі згенерувати коду для виконання моделей нейронних мереж. Хочу зауважити що даний інструмент підтримує фреймворки машинного навчання, Keras, TensorFlow [1]. Також в бібліотеку вбудовано інструменти оптимізації та налагодження моделей, аналогічні з TensorFlow. Готові системи чудово показують себе у розпізнаванні та класифікації об'єктів на зображеннях. Приклади яким можна знайти на офіційному YouTube каналі компанії

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		25

ВИСНОВОК ДО РОЗДІЛУ 1

У цьому розділі ми оглянули існуючі мікроконтролери STM-32, готові рішення та алгоритми обробки зображень, приклади застосування подібних систем та обговорили інструменти та рішення для реалізації обробки зображень на мікроконтролерах. Підсумовуючи усе вище написане, хотів би зазначити, що для реалізації початкового задуму, а саме обробки зображення, зокрема визначення об'єктів на фото, нам могли знадобитись розглянути усе, що є пов'язано з цією темою, вивчити ці питання та зробити якісь аналітичні висновки. Маючи готові описові характеристики, ще більше дозволяють зрозуміти що цей проект, логіку обробки, систему аналізу та вибір тих чи інших алгоритмів, бібліотек, інструментів.

Головні та ключові висновки з кожного огляду, знаходяться нижче.

Для реалізації проекту було обрано мікроконтролер STM-32, а саме STM32F767VGT6 [33], оскільки він є гарним компромісом між ціною та вбудованими характеристиками, має необхідні підключення та достатньо пам'яті для якісної та не дуже довгої обробки даних.

Розглядаючи усі існуючі методи та алгоритми аналізу зображень, можна прийти до висновку, що кожен з них заточений для своєї задачі, чи то покращення зображення, чи то пошук елементів та пікселів, чи то сегментація, або виявлення об'єктів. Існує безліч задач, які можна поєднувати, чому б не використовувати декілька методів обробки для чіткішого розпізнавання елементів та сегментів на зображенні. Але тут можна прийти до точного висновку – використання нейронних мереж значно покращує аналіз та надає більше можливостей для обробки та аналізу зображень, не втомлюсь повторювати, як ті ж системи донаведення, що долітають до цілі усвідомлюючи об'єкт у просторі. Але зрозуміло, що

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		26

використання даними алгоритмами і системами спрощується та обмежується інструментами для системного аналізу зображень, писати дані системи немає сенсу, оскільки існує безліч готових рішень та інструментів, які ми не встигли розглянути, оскільки вони немає чітких документацій та є сумнівними для їх використання у дипломній роботі.

Огляд інших систем та їх сфер застосування, показав значемість та революційність даних систем, натякаючи – це майбутнє, за цим майбутнє. Ми могли бачити та розглядати різні системи, що правда без чітких кодів та значень. Не хотів наголошувати на конкретних прикладах, але хотів показати, що таких прикладів, дуже дуже багато, вони існують поємісно і кількість таких буде тільки зростати. Звичайно є більше сфер, які не були наведені, джерела яких я не можу докоріно підтвердити, оскільки немає конкретних прикладів підтверджуючих їх наявність. Для нашого прикладу, дуже важливо розуміти, що дані системи можуть легко адаптовуватись під різні умови – завантажуючи нейронну мережу, що аналізує машини, ми злегкістю можемо поміняти код, що аналізує яблука та сливи, або до більш складних задач, аналіз ворожої техніки, тощо.

Готові інструменти та бібліотеки показали нам наявність великого попиту на аналіз зображень, благо такі інструменти існують навіть вбудовані у STM-32, які звичайно підтримують нейронну мережу TensorFlow, яку ми далі будемо використовувати, для машинного навчання, для побудови штучного інтелекту. Що використовувати TensorFlow Lite for Microcontrollers чи STMicroelectronics X-CUBE-AI суттєвого значення немає, оскільки ці 2 системи базуються та гарно себе показують на мікроконтролерах STM-32, обидва мають гарну документацію, навальні уроки, та підтримку, просто кожен має різне використання, застосування та різні методи, я обрав вбудований X-CUBE-AI [1] та буду використовувати його, але паралельно збираюсь дивитись на

існуючі рішення TensorFlow Lite for Microcontrollers [6], щоб максимально спростити процес розробки загальної системи

Далі ми розглянемо, теоретичні дії для реалізації проекту, сам процес створення та документацію до проекту, а також поділюсь рекомендаціями, та загально підсумую проект.

					ІАЛЦ.467200.003 ПЗ	Арк.
						28
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2

ОГЛЯД АЛГОРИТМІВ ТА ТЕХНОЛОГІЙ ПРИ РОЗРОБЦІ СИСТЕМИ

2.1 Машинне навчання

З першого розділу ми розуміли, що найкращим для нас є аналіз зображень за допомогою нейронних мереж, тим паче, як ми з'ясували існують готові рішення машинного навчання по типу TensorFlow, що підтримується як і TensorFlow Lite for Microcontrollers [4] так і STMicroelectronics X-CUBE-AI, тож за реалізацію даного машинного навчання ми і візьмемось.

Деталі виконання машинного навчання, не буду, а лише розповім про основні моменти пов'язані зі створенням нейронних мереж.

Весь цей час вільно оперую поняттями Штучний інтелект, машинне навчання, та глибинне навчання, але варто розуміти, що це не одне і те саме, ба більше AI це більша область за ML та DL оскільки включає їх у себе [9]

2.1.1 Основні поняття

Розберімося в дефініціях.

1. Штучний інтелект або AI - це галузь комп'ютерних наук, яка фокусується на створенні систем, здатних виконувати завдання, що вимагають інтелекту, схожого на людський. Не хвилюйтесь, проговорю лише ті інструменти та технології, які стосується дипломної роботи, на інші фокусуватися не будемо [34].

- Машинне навчання (ML);
- Глибинне навчання (DL);
- Комп'ютерний зір;
- Обробка природної мови (NLP);
- Розпізнавання мови;

Та безліч інших.

Оскільки ми вже проговорили, що таке комп'ютерний зір, на ньому також не будемо фокусуватись, краще зосередимось на ML та DL.

2. Машинне навчання - це підгалузь AI, яка використовує статистичні методи та різні алгоритми для надання комп'ютерам здатності навчатися аналізуючи данні. ML навчаються на основі прикладів та досвіду, на основі будування правильних не правильних дій, з використання вчителя та без. Компонентами ML є моделі, в більшості це математичні структури, що представляють дані, певні алгоритми навчання, які необхідні для навчання моделей, та певні данні, на яких загальна модель навчається.

3. Глибинне навчання - є підгалузь ML, що базується на багатошарості нейронних мереж для аналізу даних. DL, безперечно, ефективним для обробки та аналізу великої кількості даних, а також для їх застосування у системах з комп'ютерним зором, до прикладу. DL послуговуються структурами нейронних мереж з багатьма прихованими шарами, які, в свою чергу, дозволяють моделі навчатися більш складним закономірностям в даних.

2.1.2 Використання DL у TensorFlow

Існує безліч курсів та туторіалів для створення своїх DL моделей. Зараз ми знову таки не будемо занурюватись в деталі, а краще поділюсь

інсайдами та особливостями, що обирав та що шукав. Не буду казати як не є, тут намагався максимально зосередитись на самому аналізі зображень в мікроконтролері STM-32, тож шукав спрощені приклади та системи для машинного навчання, це допоможе швидко формувати та потрібні списки об'єктів та змусить не винаходити велосипед на рівному місці, а також значно спростить роботу та логіку використання моєї моделі, тож в свої пошуках я відкрив багато OpenSorce готових проектів, для швидкого використання TensorFlow та початку глибокого навчання. Наша задача – навчити нейронну мережу розпізнавати об'єкти, для прикладу обрав об'єкт 1. Тож першим ділом нам треба знайти безліч зображень з такими автомобілями, вони потрібні для навчання мережі, та її тестування. Чим більше буде зображень, тим легше буде мережі запам'ятати та зрозуміти, що таке цей об'єкт під назвою об'єкт 1, звичайно, далі поетапно оброблю зображення, як це показано на рис. 2.1. Наступним етапом буде іти перевірка, а далі вже ми будемо реалізовувати сам аналіз за допомогою мікроконтролера

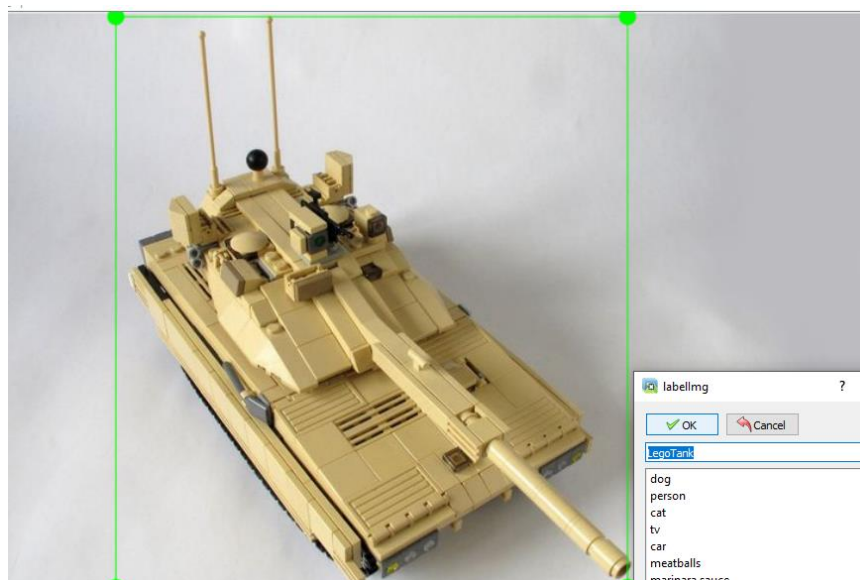


Рисунок 2.1 – Створення конфігурацій для розпізнавання об'єкту 1



Рисунок 2.2 – Виділення кількох об’єктів 1 на фото

Ретельніше розберемо процес створення моделей для навчання. Для тестування такого навчання обрав модель з довгою назвою - `ssd_mobilenet_v2_fpn-lite_320x320_coco17_tpu-8`, оскільки саме цю модель було застосовано у курсі TFODCourse [35], який я вивчав під час виконання дипломної роботи, щоб додатково розуміти принципи створення таких систем. Перевагою цієї моделі є її швидкість, але разом з тим, вона частіше помиляється. Ще така модель обробляє та працює з квадратним видом визначення об’єктів на зображенні, що якраз і необхідно для реалізації задуму створеної системи. Ну і звичайно, ми ж не просто так виділяли окремі елементи, щоб вчити систему (дивись рис. 2.1 та рис. 2.2). Звичайно, Моделі, що аналізують об’єкти на зображеннях, мають свою швидкість та результативність за допомогою чого роблять процеси пропрацювання та застосування нейронних мереж унікальними. Як виявилось такі моделі, як і було зазначено вище, базується на звичайних алгоритмах обробки зображень, які працюють по своєму. Якщо нейромережа вдало розуміє усі об’єкти та знаходить їх з легкістю, значить вона запам’ятовує цей

результат та називає цей об'єкт заданим чином. Дана специфіка дозволяє. Спочатку треба було протестувати та запустити готові існуючі файли, щоб перевірити на працездатність систему обробки об'єктів, після успішної перевірки запускаємо і нашу модель. Не вникаючи у деталі, принцип роботи простий, я відібрав з зображень вище 7 для тренування моделі, а для тестування 2 зображення, таким чином, модель зможе ефективніше розуміти та розпізнавати такі моделі, основним моментом тут є специфічність об'єкту і різність текстур, а також лего елементів, на які я перевіряти не буду, але як факт артикулюю. А також під час реалізації навчання дізнався про вже готові моделі з вже підготовленими системами, які якраз будуть потрібні для визначення, усього, що є на зображенні, а нашою кодовою особливістю і задачею буде – відмолювання тих об'єктів, які нам будуть дійсно необхідні.

Після того як у нас є готова модель, її треба імпортувати до мого мікроконтролера, тут усе просто, діємо інструкції, благо Tensorflow має гарну документацію. Отже, як ми бачили раніше, після того як ми натренували модель нам треба її спростити, оптимізувати, але простіше це називається – конвертувати. Прошу звернути свою увагу на рис. 1.6, там ми вже розглядали, а також ми вже проговорювали, в контексті використання даних систем, поетапність процесів.

Після конвертації та підготовки моделі, завантажуюмо її до інших файлів MCU STM-32, а також збираємо усе до купи, пишемо ми на мові C та C++, але про це детальніше у Розділі 3 та 4

2.2 Аналіз зображень

Поговоримо про основну частину – основна обробка зображень у самому мікроконтролері STM-32 зокрема STM32F767VGT6.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		33

Але не будемо поспішати і обговоримо увесь етап обробки зображень:

Сам по собі мікроконтролер має:

- a) USB підключення, для підключення камери або іншого USB-носія
- b) Quad-SPI підключення, для підключення зовнішніх накопичувачів, що реалізують запам'ятовування проаналізованих зображень (Інакше зображення просто зникне з носія)
- c) FATFS – для роботи з файлами, всередині мікроконтролера та у зовнішніх пристроях
- d) X-CUBE-AI – інструмент для реалізації аналізу об'єктів на зображенні, який ми розглядали вище

Етапи роботи системи аналізу зображень на мікроконтролері STM-32:

- 1) Ми підключаємо USB-носій, або камеру з USB підключенням, до нашого мікроконтролера
- 2) Мікроконтролер, після підключення, робить перенесення зображень до зовнішнього Quad-SPI накопичувача у папку, для аналізу зображень (work)
- 3) Мікроконтролер починає обробляти зображення, як тільки він закінчив переносити останній файл з накопичувача, що передає зображення
- 4) Вже проаналізоване зображення та вхідне зображення, як файли, переміщаються у папку store, цей процес повторюється, поки у папці work не залишиться жодного файлу
- 5) Як тільки MCU закінчив обробку та передав усі необхідні файли у папку store, він перевіряє:
 - a) Наявність USB-накопичувача. Якщо його немає – то, файли залишаються в цій папці

- б) Наявність файлів на USB-накопичуючи. Якщо було зроблено ще одне зображення, або завантажено на нього інший файл, тоді STM32 система, побачить це та виконає усі пункти ще раз
- 6) Як тільки виявиться, що USB-накопичувач є, а файлів на ньому немає – усі зображення, що містяться у папці store, потраплять до USB-накопичувача та у ще одну папку у зовнішній пам'яті мікроконтролера – archive

Для аналізу зображень використовуватимемо модель st_ssd_mobilenet_v1, оскільки вона швидко аналізує та легко інтегрується до мікроконтролера, що буде у нагоді під час використання її на практиці. А також у цій моделі є величезний список об'єктів, які вона знає та розуміє.

Найголовніше у системі – це аналіз, якщо чіткість падає, тоді корисність системи значно падає. Тоді, б здавалося, краще використовувати якісь базові інструменти, по типу алгоритмів аналізу зображень, які ми розглядали у першому розділі, можливо навіть їх поєднання з такою обробкою об'єктів.

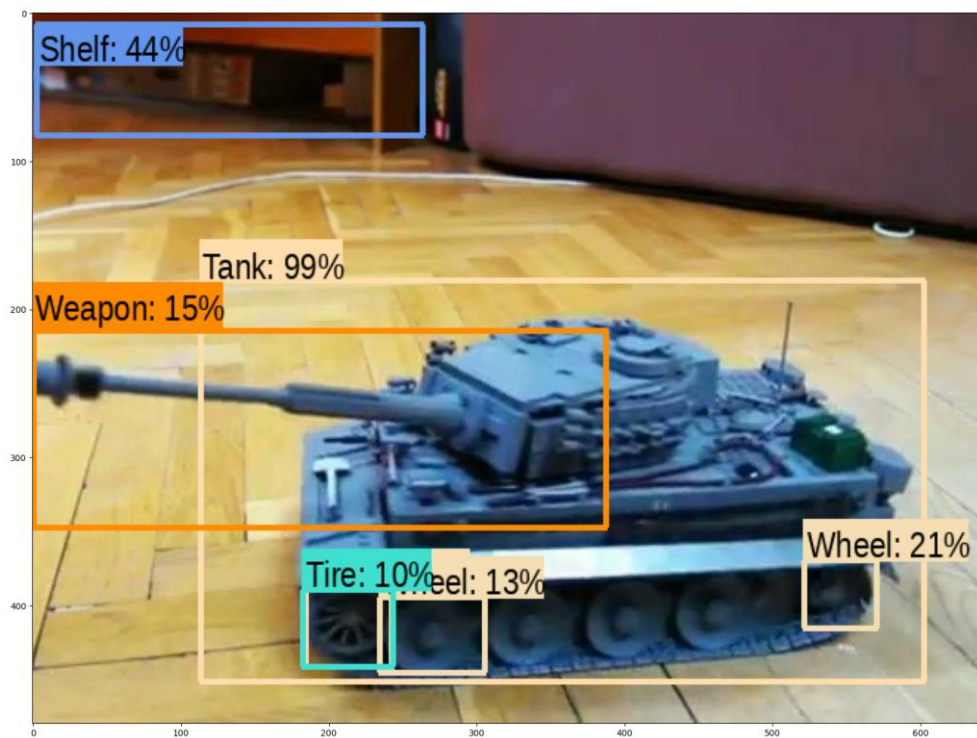


Рисунок 2.3 – Аналіз об’єкту 1, за допомогою системи з використанням FasterRCNN+InceptionResNet V2 моделі, без додаткового фільтру

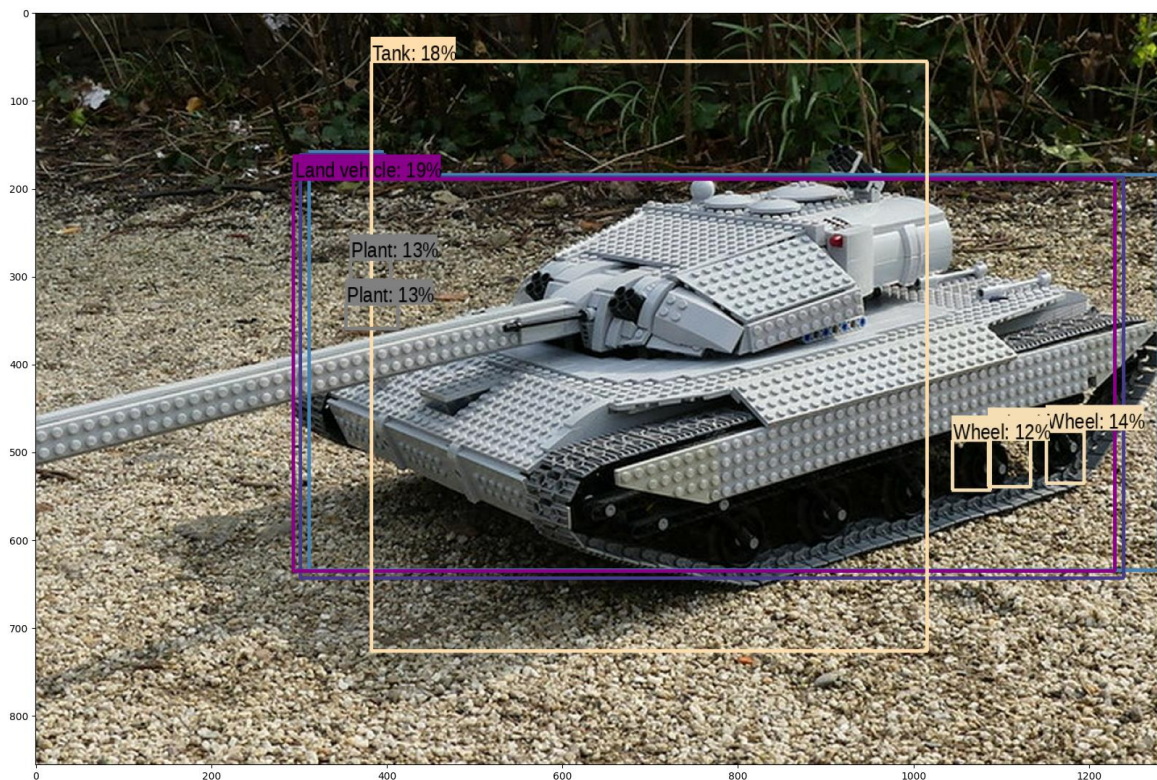


Рисунок 2.4 – Аналіз об’єкту 1, за допомогою системи з використанням ssd+mobilenet V2 моделі, без додаткового фільтру

Для тестування використав і іншу систему, яка показана трохи вище (рис. 2.4). Ця система базується на основі іншої моделі розпізнавання об'єктів `ssd + mobilenet V2`, чіткість визначення очікуємо впала, але значно зросла швидкість такого аналізу. Оскільки основною ідеєю є створення портативної системи аналізу зображень, я обираю більш точні інструменти визначення та і більший каталог для виявлення об'єктів стане у нагоді.

2.3 Технології та бібліотеки

Тепер можна перерахувати та описати основні інструменти, що зустрічатись у описі розробки системи, про що треба знати та що треба враховувати, якщо брати мою розробку як приклад для наслідування, для спроби повторити мій результат

Спробую досить коротко, для але спробую іноді дати поради та описати якісь цікаві особливості

2.4 STM32CubeMX та STM32CubeIDE

Ці інструменти дозволяють генерувати код для мікроконтролерів STM32, налаштовувати різноманітні параметри, підключати різні інструменти та готові рішення, дозволяють підключати периферійних пристроїв, налаштовувати тактові джерела й GPIO.

Різниця між STM32CubeMX та STM32CubeIDE полягає у тому, що перша заточена на вище описане, а друга на написання коду, компілювання, та інтеграції у різні програми та середовища, які ми реалізовували протягом навчання у КПІ.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		37

3.3 ST_SSD_MOBILENET_V1

Модель глибинного навчання `st_ssd_mobilenet_v1` призначена для об'єктного виявлення, але має інший підхід до вирішення цього завдання. У порівнянні з Faster R-CNN + InceptionResNet V2, вона є більш легкою та швидкою, що дозволяє використовувати її на пристроях з обмеженими ресурсами, таких як мобільні телефони або вбудовані системи. SSD (Single Shot Multibox Detector) виконує об'єктне виявлення за один прохід мережі, що значно зменшує час обробки зображень.

Mobilenet V1 [36] – це ефективна конволюційна нейронна мережа, яка оптимізована для мобільних пристроїв. Вона використовує глибинні сепарабельні згортки для зменшення кількості параметрів і обчислювальної складності, що робить її дуже швидкою, зберігаючи при цьому прийнятну точність.

`st_ssd_mobilenet_v1` добре підходить для реальних додатків, де необхідна швидка і відносно точна детекція об'єктів зображення, і де ресурси обмежені. Таким чином, ця модель дозволяє забезпечити швидке реагування та обробку зображень у режимі реального часу.

ВИСНОВОК ДО РОЗДІЛУ 2

У цьому розділі ми розглянули основні функції та етапи системи та нарешті перейшли до конкретики, до ідей та інструментів по реалізації системи, що виконує аналіз зображень, зокрема об'єктів на них, за допомогою штучного інтелекту, яка ще і базується на мікроконтролері STM32. Було розглянути, основні, принцип роботи, тестування окремих моделей, та принципи створення своїх моделей для аналізу зображень та пошуку об'єктів на них. Під час переходячи до етапу опису у коді, ми також застосуємо наші знання для використання інших математичних методів аналізу зображень, які ми розглядали у першій частині, дивіться далі. А також хотів би підсумувати і наголосити про основні особливості які були проговорено, відносно системи:

- В якості бази для навчальної моделі DL було обрано `st_ssd_mobilenet_v1`
- Розробка аналізу зображень здійснюватиметься завдяки інструменту X-CUBE-AI
- Мікроконтролер STM32F767VGT6 використовує різні підключення та вбудовані інструменти для реалізації усіх функцій
- Серед основних функцій мікроконтролерів є отримання зображень, обробка зображень, зберігання зображень, архівація та відправлення зображень
- Код буде написаний мовою C
- Тестування проходитиме у середовищі STM32CubeIDE 1.14.1
- Вибір необхідних об'єктів, що буде аналізувати MCU буде реалізовуватись за допомогою фільтра, який можна змінити у спеціальному файлі конфігурацій. У документації, буде розглянуто та

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		40

описано усі набори конфігурацій, для зручності використання дипломного проекту

У наступних розділах ми перейдемо вже до технічної та схематичної реалізації, опису документації, процесу тестування та підбиття загальних підсумків.

Завдяки тому, що вже розібрали основи роботи проекту, буде значно легше розуміти та реконструювати подальші розділи, в тому числі для створення подібної системи. Зауважу, що система не прив'язана до класів об'єктів, конфігурації можна змінювати, а мету підбирати для потрібних та поточних задач. Як буде розглянуто нижче, за допомогою використання інструментів STM32AI Model Zoo DL та знань отриманих з цього розділу можна створити власну систему аналізу, яку досить легко можна завантажити у код системи

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		41

РОЗДІЛ 3

ДЕТАЛІ РОЗРОБКИ СИСТЕМИ

3.1 AI технологія комп'ютерного зору

3.1.1 Вибір інструменту для формування FTLite моделі

Нижче буде описувано головні принципи, рекомендації та поради, а також власне виконання роботи.

Дивлячись та розбираючись у темі я відкрив для себе stm32ai-modelzoo. STM32AI Model Zoo [8] - це колекція попередньо навчених моделей машинного навчання, оптимізованих для роботи на мікроконтролерах STM32. Ці моделі можуть бути використані для широкого спектру завдань, таких як розпізнавання зображень, класифікація даних, прогнозування та багато іншого. Переваги використання STM32AI Model Zoo можна вважати швидкий початок роботи, де нам не потрібно витратити час на навчання власних моделей, ви можете використовувати готові моделі, які вже були навчені на великих наборах даних. До того ж, моделі STM32AI оптимізовані для роботи на мікроконтролерах, що робить їх ідеальними для вбудованих систем STM32 типу. Список моделей досить великий, їх можна навчати та покращувати. В моєму випадку, ці моделі STM32AI дуже легко інтегрувати у ваші програми за допомогою STM32Cube.AI, що нам буде значно покращувати життя

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		42

STM32AI Model Zoo це такий самий аналог TensorFlow 2 Detection Model Zoo та TensorFlow 1 Detection Model Zoo та подібних, але зрозуміло, що STM32AI заточений суто на використання його на мікроконтролерах, до того ж, є оптимізованим для використання на саме STM-32 типу, оскільки, як і зазначав, легко інтегруються в наше середовище [8].

Також в основному репозиторії Github проекту stm32ai-modelzoo, вже представлені готові інструменти для створення власної моделі, усе що для цього потрібно – слідувати вказівкам, що залишені у README.md файлі.

Прикладів для серії мого мікроконтролера немає, але це і добре, оскільки моя дипломна робота стає більш ціною і допоможе як посібник розібратись та розробити .FTLite модель, що у подальшому зможемо інтегрувати до мікроконтролера або інших систем

Для початку, нам треба обрати модель, це ми можемо зробити за допомогою методів підбору по пам'яті та відсотком виявлення

Models	Implementation	Dataset	Input Resolution	mAP*	MACCs (M)	Activation RAM (KiB)	Weights Flash (KiB)	STM32Cube.AI version	Source
ST SSD MobileNet v1 0.25	TensorFlow	COCO Person dataset	192x192x3	33.70%	40.54	249.54	438.28	8.1.0	link
ST SSD MobileNet v1 0.25	TensorFlow	COCO Person dataset	224x224x3	44.45%	60.091	366.15	595.66	8.1.0	link
ST SSD MobileNet v1 0.25	TensorFlow	COCO Person dataset	256x256x3	46.26%	72.67	439.34	595.66	8.1.0	link
SSD MobileNet v2 0.35 FPN-lite	TensorFlow	COCO Person dataset	192x192x3	40.73%	122.78	565.25	984.25	8.1.0	link
SSD MobileNet v2 0.35 FPN-lite	TensorFlow	COCO Person dataset	224x224x3	48.67%	167.47	769.12	993.05	8.1.0	link
SSD MobileNet v2 0.35 FPN-lite	TensorFlow	COCO Person dataset	256x256x3	55.56%	218.33	1003.7	1065.3	8.1.0	link
SSD MobileNet v2 0.35 FPN-lite	TensorFlow	COCO Person dataset	416x416x3	59.09%	577.66	2673.3	1109.3	8.1.0	link

Рисунок 3.1 – Візуалізація декількох моделей stm32ai-modelzoo

Ми обрали модель, st_ssd_mobilenet_v1, тепер обрали її підтип st_ssd_mobilenet_v1_025_256_int8.tflite, цю модель ми і будемо навчати того що нам потрібно. Оскільки мені не потрібно було розпізнавання додаткових об'єктів на фото, оскільки вона вже має можливості розпізнавати усього, що потрібно, додатково навчати модель не було сенсу, але якщо ви вирішите зробити власну систему, нижче опишу як це можна зробити

1. Завантажуємо репозиторій до себе на комп'ютер, код програм написаний на python. Виконуємо усі вказівки в кореневій папці
2. Нас цікавить object_detection, тож переходимо та ознайомлюємось зі структурою проекту. В нашому випадку в src розташовано усе що необхідно

3. Робимо свої datasets з зображеннями, як я показував у розділі 2. Тут треба створити папки training, validation, test, які ми заповнюємо фото та файлами YOLO Darknet TXT, що говорять про вміст об'єктів на фото, як приклад за розширенням object_detection/src/training, знаходяться посилання на датасети для завантаження, там є досить багато зображень, які і навчали моделі, які збираюсь використовувати у роботі (17 тисяч зображень)
4. Тепер працюємо з конфіг файлом, в якому виставляємо необхідні параметри, до прикладу мій код, який наведений у рис. 3.2. Обов'язково встановіть stm32ai вашої операційної системи, для роботи і навчання системи
5. Як тільки усе готово, запускаємо файл stm32ai_main.py, який і реалізує машинне навчання, треба буде чекати поки, комп'ютер обробить усе необхідне та сформує data нашої моделі, що говорить про пам'ять нашої моделі, в якій і зберігається логіка виявлення наших об'єктів. Візуалізація процесу навчання див. на рис. 3.3

```

1  general:
2      project_name: MyProject
3      model_type: ssd_mobilenet_v2_fpn-lite
4      model_path: ../pretrained_models/ssd_mobilenet_v2_fpn-lite/
5      logs_dir: logs
6      saved_models_dir: saved_models
7      gpu_memory_limit: 16
8      global_seed: 127
9
10 operation_mode: training
11
12 dataset:
13     name: COCO_2017_person
14     class_names: [ person ]
15     training_path: F:/GitProj/train
16     validation_path: F:/GitProj/valid
17     test_path: F:/GitProj/test
18
19 preprocessing:
20     rescaling: { scale: 1/127.5, offset: -1 }
21     resizing:
22         aspect_ratio: fit
23         interpolation: nearest
24         color_mode: rgb
25
26 data_augmentation:
27     rotation: 30
28     shearing: 15
29     translation: 0.1
30     vertical_flip: 0.5
31     horizontal_flip: 0.2
32     gaussian_blur: 3.0
33     linear_contrast: [ 0.75, 1.5 ]

```

Рисунок 3.2 – Візуалізація конфігурацій у файлі user_config.yaml

```
(st_zoo) F:\GitProj\stm32ai-modelzoo\object_detection\src>python stm32ai_main.py
[INFO] Setting upper limit of usable GPU memory to 16GBytes.
[INFO] : Running 'training' operation mode
2024-06-01 16:21:41.748735: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with
oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:
  AVX
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Loading Training dataset...
9% | ██████████ | 397/4292 [00:33<05:01, 12.91it/s]
```

Рисунок 3.3 – Запуск stm32ai_main.py

3.1.2 Інтеграція моделі FTLite у STM32CUBEMX

Тепер зосередимось на додаванні та інтеграції моделі у проект, та загалом у код. Усе реалізовується за допомогою інструменту X-CUBE-AI. Нам необхідно додати новий network який і реалізовує нашу модель, обрати формат та обрати нашу модель

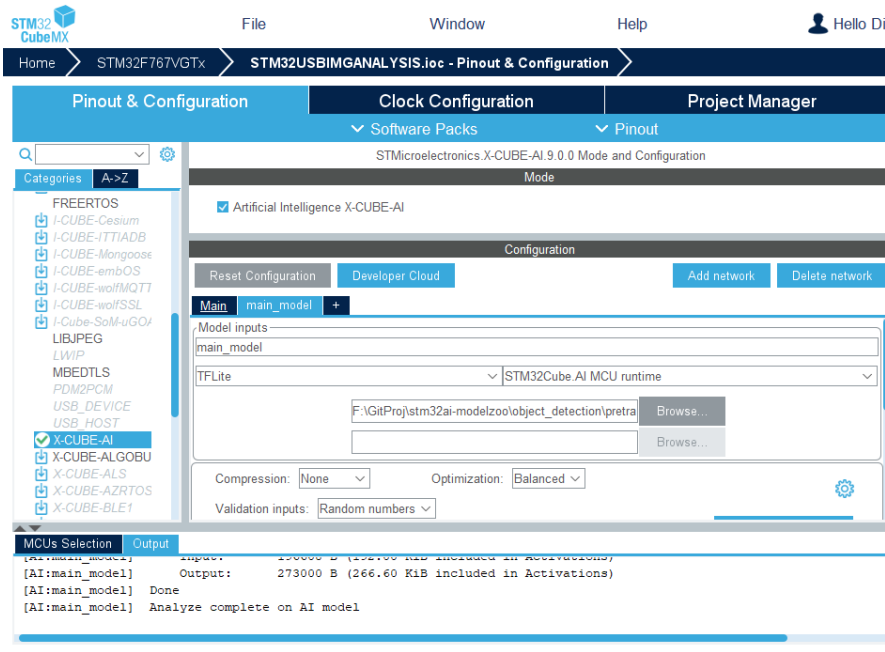


Рисунок 3.4 – Додавання мережі моделі до мікроконтролера за допомогою X-CUBE-AI

Після чого необхідно виконати аналіз, який підрахує та підставить модель до мікроконтролера та обрахує різні параметри, до прикладу пам'ять. Після чого покаже фінальний результат сумісності та дозволить переглядати граф, та робити тестування за допомогою генерації випадкових даних. Також, що не менш важливо, аналіз можна виконати на самому мікроконтролері, щоб звірити показники та більш реальний час роботи перевірки.

Нижче представлено приклади використання нейронних мереж у моїй роботі, будова, за допомогою графа, тощо.

Час роботи системи аналізу зображення становить – 63.248 секунд, в рамках тестування, даний результат можна побачити нижче. Треба запам'ятаємо, що обробка одного зображення триває 1-2 хвилини.

MCUs Selection	Output
[AI:main_model]	FLASH % RAM %
[AI:main_model]	-----
[AI:main_model]	RT total 110,632 15.4% 36,456 7.1%
[AI:main_model]	-----
[AI:main_model]	TOTAL 720,592 512,984
[AI:main_model]	-----
[AI:main_model]	Creating txt report file C:\Users\beano\.stm32cube\main_model_output\main_model_analyze_report.txt
[AI:main_model]	elapsed time (analyze): 30.107s
[AI:main_model]	Model file: st_ssd_mobilenet_v1_025_256_int8.tflite
[AI:main_model]	Total Flash: 720588 B (703.70 KiB)
[AI:main_model]	Weights: 609956 B (595.66 KiB)
[AI:main_model]	Library: 110632 B (108.04 KiB)
[AI:main_model]	Total Ram: 512984 B (500.96 KiB)
[AI:main_model]	Activations: 476528 B (465.36 KiB)
[AI:main_model]	Library: 36456 B (35.60 KiB)
[AI:main_model]	Input: 196608 B (192.00 KiB included in Activations)
[AI:main_model]	Output: 273000 B (266.60 KiB included in Activations)
[AI:main_model]	Done
[AI:main_model]	Analyze complete on AI model

Рисунок 3.5 – Показники забитості FLASH та RAM пам'яті

notes: - ACC metric is not computed ("--classifier" option can be used to force it)
 - the output of the reference model is used as ground truth/reference value
 - 10 samples (27300 items per sample)
 acc=n.a., rmse=0.000000000, mae=0.000000000, l2r=0.000000000, nse=1.000, cos=1.000
 Evaluation report (summary)

Output	acc	rmse	mae	l2r	mean	std	nse	cos	tensor
X-cross #1	n.a.	0.0005063	0.0000638	0.0007189	0.0000021	0.0005063	0.9999990	0.9999995	conversion_148, (6825,2), m_id=
X-cross #2	n.a.	0.0091444	0.0055244	0.0748443	0.0000212	0.0091444	0.9941458	0.9971990	conversion_155, (6825,4), m_id=
X-cross #3	n.a.	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	1.0000000	1.0000000	conversion_163, (6825,4), m_id=

acc : Classification accuracy (all classes)
 rmse : Root Mean Squared Error
 mae : Mean Absolute Error
 l2r : L2 relative error
 nse : Nash-Sutcliffe efficiency criteria, bigger is better, best=1, range=(-inf, 1]
 cos : COsine Similarity, bigger is better, best=1, range=(0, 1]
 Creating txt report file C:\Users\beano\.stm32cube\main_model_output\main_model_validate_report.txt
 elapsed time (validate): 63.248s
 Validation ended

Рисунок 3.6 – Результат тестування мережі на комп'ютері та фінальна статистика тестування.

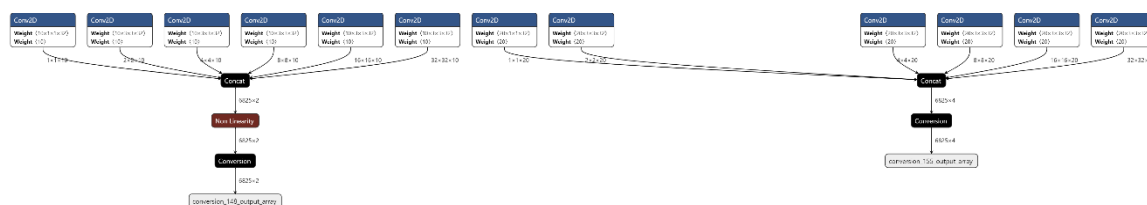


Рисунок 3.7 – Кінцівка графу, що представляє модель, що була використана у дипломній роботі

3.2 Мікроконтролер STM32 та код

3.2.1 Загальний опис та задачі

Мій мікроконтролер - STM32F767VGT6

Інструменти, що використовую - FATFS, X-CUBE-AI

Назва network моделі X-CUBE-AI main_model

Використовую Quad-SPI, для підключення зовнішньої пам'яті та USB HS для підключення USB-накопичувача (Або умовної камери)

Папки work, store, input_history та archive містяться на зовнішній пам'яті (Quad-SPI)

Загальна робота системи:

- 1) Ми підключаємо USB-носій до нашого мікроконтролера
- 2) Мікроконтролер, після підключення, робить перенесення зображень формату .jpg, з кореневої папки флешки, до зовнішнього Quad-SPI накопичувача у папку work
- 3) Мікроконтролер починає “обробляти” зображення з папки work. Оброблене зображення мікроконтролер створює у папці store, а зображення, що оброблялось переміщається у папку input_history
- 4) Якщо у папці work ще є файли .jpg, процес 3) повторюється
- 5) Після закінчення обробки усіх файлів мікроконтролер перевіряє наявність USB-накопичувача. Якщо він не підключений – то, файли залишаються в папці store. Якщо підключений то перевірка наявності файлів .jpg на USB-накопичуючи. Якщо файли .jpg є, то виконуються процеси 2) 3) та 4)
- 6) Якщо USB-накопичувач підключений до мікроконтролера, а файлів на ньому немає:
 - а) на USB-накопичувачі створюється папка FromSTM32, якщо її немає
 - б) файли .jpg з папки store переміщаються у папку FromSTM32 (USB) та archive (Quad-SPI). Дана дія виконується поки у папці store не закінчиться файли

Під обробкою зображень, мається на увазі, створення такого зображення, на якому обведено червоними квадратами результати аналізу системи, а також виписані назви об'єктів, а також ймовірність.

Оскільки наша модель не може повертати готові зображення, тож деякі дії треба прописати вручну.

Загалом, при розробці проекту зустрівся з проблемою, яка зумовлена відсутністю дисплею у мого мікроконтролера, а також підтримкою H7 моделей мікроконтролерів, оскільки більшість реалізацій були зроблені використовуючи цю модель. Тож, приймаючи виклик, який ставить мене на самостійну реалізацію мого проекту, я детальніше ознайомлююсь з документацією X-CUBE-AI, оскільки подібних проектів, що найменше на OpenSorce мною не було знайдено. Можливо це зумовлено новизною технології, а також загальною комерцією AI ринку, оскільки такі технології гарно продаються, а також реалізується великими компаніями

3.2.2 Підключення бібліотек, створення змінних та прототипів функції

Моєю цілю було використовувати, лише ті інструменти, які маю. Спрямовано це на мінімізацію код та використання пам'ять, оскільки вона досить обмежена. Та і загалом, більше інструментів непотрібно, оскільки усі задачі були вирішенні на моменті планування даної системи. До прикладу модуль fatfs.h значно покращив та оптимізував роботу з файлами, що буде видно при реалізації функції soruDir та опису основного коду у циклі функції main, що ми побачимо згодом

```

/* USER CODE END Header */
/* Includes -----
#include "main.h"
#include "fatfs.h"

/* Private includes -----
/* USER CODE BEGIN Includes */
#include "stm32f7xx_hal.h"
#include "../X-CUBE-AI/App/main_model.h"
/* USER CODE END Includes */

/* Private typedef -----
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define -----
/* USER CODE BEGIN PD */
#define WORK_FOLDER "work"
#define STORE_FOLDER "store"
#define INPUT_HISTORY_FOLDER "input_history"
#define ARCHIVE_FOLDER "archive"
#define FROM_STM32_FOLDER "FromSTM32"
/* USER CODE END PD */

```

Рисунок 3.8 – Підключення лише необхідних бібліотек та створення змінних з папками

Загалом, намагався досить зручно та читабельно назвати та використовувати зміни у коді. Хотілось презентувати та показувати цей код на загал, оскільки моя робота має освітню мету для бажаючих реалізувати подібні напрацювання, що були та будуть описані у даній роботі. Це і стосується назв функцій – кожна функція повинна розповідати, навіщо і як вона використовується. Але загалом, зосередився не лише на написанні функції, а на реалізації допоміжних функцій мого проекту у функції main(), що і побачите загалом, далі, ближче до опису даної функції. Опис змінних має сенс, оскільки це полегшало розробку, даний вигляд змінних, функцій, при розробці неоднократно змінювався, хоча і перевіряв усе на розуміння

та оптимізацію, проте ніхто не застрахований від банального людського фактору

```

/* Private variables -----
QSPI_HandleTypeDef hqspi;

/* USER CODE BEGIN PV */
FATFS fs;
DIR dir;
FRESULT fr;
FIL MyFile;
UINT bw;
/* USER CODE END PV */

/* Private function prototypes -----
void SystemClock_Config(void);
static void MPU_Config(void);
static void MX_GPIO_Init(void);
static void MX_QUADSPI_Init(void);
static void MX_USB_OTG_HS_USB_Init(void);
/* USER CODE BEGIN PFP */
void copyDir(char* srcDir, char* dstDir);
void processImage(char* imagePath);
void AnalyzeImageWithBoundingBox(Image* img);
void VisualizeResults(int predictedClass, float maxProbability);
/* USER CODE END PFP */

```

Рисунок 3.9 – Зміни та їх назви, а також прототипи функцій для їх використання

3.2.3 Функція copyDir

Функція copyDir, як можна було здогадатись, відповідає за копіювання. Потрібна вона для переміщення усіх файлів з папки srcDir до папки dstDir. У коді я її використовую, для переміщення з кореневої папки USB до моєї робочої папки work, що знаходиться на зовнішньому диску

```

75 void copyDir(char* srcDir, char* dstDir) {
76     char* path[255];
77     char* fileName;
78     fr = f_opendir(&dir, srcDir);
79     if (fr != FR_OK) {
80         return;
81     }
82     while ((fileName = f_readdir(&dir, path)) != NULL) {
83         if (*path[0] == '.') {
84             continue;
85         }
86         char* fullSrcPath = HAL_Malloc(strlen(srcDir) + strlen(path) + 2);
87         sprintf(fullSrcPath, "%s/%s", srcDir, path);
88
89         char* fullDstPath = HAL_Malloc(strlen(dstDir) + strlen(path) + 2);
90         sprintf(fullDstPath, "%s/%s", dstDir, path);
91
92         if (f_stat(fullSrcPath, &MyFile) != FR_OK) {
93             HAL_Free(fullSrcPath);
94             HAL_Free(fullDstPath);
95             continue;
96         }
97         if (MyFile.fsize == 0) {
98             if (f_mkdir(fullDstPath) != FR_OK) {
99                 HAL_Free(fullSrcPath);
100                HAL_Free(fullDstPath);
101                continue;
102            }
103            copyDir(fullSrcPath, fullDstPath);
104        } else {
105            f_open(&MyFile, fullSrcPath, FA_READ);
106            fr = f_open(&MyFile, fullDstPath, FA_CREATE_ALWAYS | FA_WRITE);
107
108            if (fr != FR_OK) {
109                f_close(&MyFile);
110                HAL_Free(fullSrcPath);
111                HAL_Free(fullDstPath);
112                continue;
113            }
114            while ((bw = f_read(&MyFile, MyFile.buf, MyFile.fsize, &bw)) == bw) {
115                f_write(&MyFile, MyFile.buf, bw, &bw);
116            }
117            f_close(&MyFile);
118            f_close(&MyFile);
119        }
120        HAL_Free(fullSrcPath);
121        HAL_Free(fullDstPath);
122    }
123    f_closedir(&dir);

```

Рисунок 3.10 – Код функції copyDir

Робота виконується функції вкинується таким чином:

- 1) Система перевіряє існування вихідного каталогу за допомогою функції f_stat, якщо каталог не існує, функція повертає помилку FR_NO_FILE.
- 2) Далі система використовує функцію f_mkdir, що створює каталог, якщо він не існує (в моєму випадку work). Якщо зовнішню пам'ять

не приєднано, або недостатньо прав, або інші причини неможливості створити папку, функція повертає помилку.

- 3) Після етапу підготовки, мікроконтролер використовує функція `f_opendir` для відкриття вихідного каталогу, для копіювання файлів з каталогу. Знову таки, помилки обробляються.
- 4) Далі іде функція, що використовує `f_readdir` для ітерації по всіх файлах та папках у вихідному каталозі. Вміст папок рекурсивно викликається `copuDir` з шляхами до поточної папки та до папки в цільовому каталозі з такою ж назвою, щоб охопити усі файли, для обробки. Відносно файлів усе простіше, прописав `f_open` та `f_read` для читання даних з файлу, а потім використовую `f_open` та `f_write` для запису даних у файл з таким же ім'ям у каталозі `dstDir`. Вкінці обов'язково закриваю файли за допомогою `f_close`.
- 5) Після того, як усі файли було скопійовано, використовую функцію `f_closedir` для закриття вихідного каталогу.
- 6) Ну і звичайно, якщо усе пройшло успішно повідомляю про це за допомогою функції `FR_OK`, якщо виник збій - повертаю помилку

3.2.4 Функція `processImage`

`processImage` – це головна функція, яка реалізовує наше виявлення об'єктів за допомогою нашої моделі, в ній містяться інші функції, про які ми поговоримо пізніше, саме тут ми запускаємо обробку моделі за допомогою `model.ReleaseModel()`.

```

126 void processImage(char* imagePath) {
127     MainModel model;
128
129     // Load model
130     if (model.LoadModel() != 0) {
131         Error_Handler();
132         return;
133     }
134
135     // Load image
136     Image img;
137     if (LoadImage(imagePath, &img) != 0) {
138         Error_Handler();
139         model.ReleaseModel();
140         return;
141     }
142
143     // Analyze image
144     AnalyzeImageWithBoundingBox(&img);
145
146     // Save processed image
147     SaveImage(&img, INPUT_HISTORY_FOLDER);
148
149     // Free resources
150     FreeImage(&img);
151     model.ReleaseModel();
152 }

```

Рисунок 3.11 – Код функції processImage

Час пішов. Пам'ятаєте було зацентовано на часі в 63 секунди, саме тут програма починає свій аналіз подібно до того, що ми робили у пункті 3.1.2, далі результати аналізу ми будемо використовувати і обговорювати нижче.

Як вхідний параметр функція приймає зображення, в нашому випадку, що знаходиться у папці work, ми будемо перебирати усі файли по черзі, виконуючи цю функцію

Загалом послідовність виконання виглядає таким чином:

- 1) Спочатку ми створюємо екземпляр класу MainModel, який використовується для роботи нашою попередньо навченою моделлю. Це буде потрібно для використання її у даній функції, як можна було здогадатись
- 2) За допомогою методу model.LoadModel() екземпляра нашої моделі MainModel завантажуюмо нашу модель з файлу ../../X-CUBE-AI/App/main_model.h.

- 3) За допомогою функцію `cv::imread` ми виконуємо завантаження зображення з файлу, що вказано в `imagePath`. Перетворити зображення у формат, який очікує модель. Іншими словами виконуємо первину обробку для подальшого аналізу
- 4) Запускаємо `AnalyzeImageWithBoundingBox`, що є екземпляром нашої `MainModel`, про нього ми поговоримо пізніше та окремо.
- 5) Далі ми виконуємо зберігання обробленого зображення у папку `INPUT_HISTORY_FOLDER`, ви можете виставити будь-яку іншу, якщо реалізовується або застосовуєте мою модель. Це збереження дозволяє зберегти зображення, що було проаналізовано, в іншому ж випадку, воно буде стерто та виведеться помилка.
- 6) Після того як функція підійшла до закінчення викликається `model.DestroyModel()`, що екземпляром `MainModel`, вона потрібна для звільнення ресурсів, зайнятих моделлю. Ця функція звільнить пам'ять, що була виділена для завантаженого зображення.

3.2.5 Функція `AnalyzeImageWithBoundingBox`

Саме у цій функції ми використовуємо модель, аналізуємо зображення, визначаємо шанс найбільш вірогіднішого об'єкту та виконуємо обробку, за допомогою створення квадратів та підпису зображень. Це основна функція, що виконує головну задачу мікроконтролера, його призначення, є головною частиною дипломної роботи, виконує застосовує і реалізує саму сутність даної системи. Саме ця система побудована на технології комп'ютерного зору і саме тут вона реалізовує себе, тож давайте детальніше поговоримо про неї та розберемо, майже, по елементне виконання та реалізацію даної частини мікроконтролерної системи

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		56

```

154 void AnalyzeImageWithBoundingBox(Image* img) {
155     MainModel model;
156     if (model.LoadModel() != 0) {
157         Error_Handler();
158         return;
159     }
160     if (img->data == NULL || img->width <= 0 || img->height <= 0) {
161         Error_Handler();
162         model.ReleaseModel();
163         return;
164     }
165     float* inputData = PrepareModelData(img);
166     if (inputData == NULL) {
167         Error_Handler();
168         model.ReleaseModel();
169         return;
170     }
171     float* predictions = model.Predict(inputData, img->width * img->height);
172     if (predictions == NULL) {
173         Error_Handler();
174         FreeImageData(inputData);
175         model.ReleaseModel();
176         return;
177     }
178     int numClasses = model.GetNumClasses();
179     float classProbabilities[numClasses];
180
181     for (int i = 0; i < numClasses; i++) {
182         classProbabilities[i] = 0.0f;
183         for (int j = 0; j < img->width * img->height; j++) {
184             classProbabilities[i] += predictions[j * numClasses + i];
185         }
186     }
187     int predictedClass = 0;
188     float maxProbability = 0.0f;
189
190     for (int i = 0; i < numClasses; i++) {
191         if (classProbabilities[i] > maxProbability) {
192             maxProbability = classProbabilities[i];
193             predictedClass = i;
194         }
195     }
196     // Visualize results (replace with your implementation)
197     VisualizeResults(predictedClass, maxProbability);
198     FreeImageData(inputData);
199     model.ReleaseModel();

```

Рисунок 3.12 – Код функції AnalyzeImageWithBoundingBox

Функція працює таким чином:

- 1) Спочатку ми завантажуюємо нашу модель MainModel model;:
Створюється екземпляр класу MainModel, який використовується для обробки з нейроною навченою моделлю. if (model.LoadModel() != 0) { Error_Handler(); return; } – виконує завантаження моделі. Якщо завантаження не вдається (а саме, якщо повертається

некоректне значення), система реалізує виклик викликається `Error_Handler`, що завершує функцію.

- 2) Далі ми перевіряємо зображення за допомогою додаткових умов, по типу, `if (img->data == NULL || img->width <= 0 || img->height <= 0) { Error_Handler(); model.ReleaseModel(); return; }`, саме тут відбувається перевірка, чи дійсний вказівник `img` (не `NULL`) або має правильні розміри зображення (ширину та висоту), які не є від'ємними, що захищає від взлому. Помилки обробляються
- 3) Далі ми працюємо з даними. За допомогою `float* inputData = PrepareModelData(img);` Попередньо обробляємо дані зображення (`img`) у формат, придатний для розуміння моделі. Отримаємо вказівник на підготовлені дані (`inputData`). Також обробляю помилку за допомогою `if (inputData == NULL) { Error_Handler(); model.ReleaseModel(); return; }`, та терміново зупиняю модель, оскільки дана помилка може свідчить про спробу взлому та поміщення не зображення у файл формату `.jpg`.
- 4) Далі модель прогнозує об'єкти на зображенні за допомогою `float* predictions = model.Predict(inputData, img->width * img->height);` У цій частині використовую `model` для прогнозування на підготовлених даних (`inputData`). Розмір зображення (`img->width * img->height`) є аргументом, який буде потрібен для реалізації подальших методів, в тому числі і цього. Підчас аналізу, функція повертає вказівник на прогнози (`predictions`), а також обробляє помилку, якщо така виникла.
- 5) Далі я отримую та використовую дані, отримані з моделювання, а саме отримую кількість класів, які модель може прогнозувати, з об'єкта `model`. `float classProbabilities[numClasses]` - створює масив `classProbabilities` для зберігання прогнозованих ймовірностей для

кожного класу. А далі використовую цикл для обчислення ймовірностей класу: `for (int i = 0; i < numClasses; i++) { ... }`: Цей цикл перебирає всі можливі класи й формує ймовірність, що виглядає таким чином `classProbabilities[i] = 0.0f;`. Цикл перебирає кожен елемент вихідних даних та формує ймовірність за допомогою `classProbabilities[i] += predictions[j * numClasses + i];` Далі функція шукає прогнозований клас. Вкінці порівнюється ймовірність поточного класу, чи вона більша за поточну максимальну. Якщо так, оновлює `maxProbability` та `predictedClass` поточними значеннями. Таким чином ми знаходимо найточніший об'єкт на фото

- 6) Далі я ініціалізую запуск функції `VisualizeResults(predictedClass, maxProbability);` Про неї ми поговоримо окремо
- 7) В кінці, відбувається очищення усіх результати за допомогою `FreeImageData(inputData);` завдяки чому, звільняється пам'ять, що була виділена для `inputData`. А також використовую `model.ReleaseModel();` для звільнення ресурсів моделі, оскільки цей цикл може бути останнім.

3.2.5 Функція `VisualizeResults`

Хоча і маленька, але досить потрібна функція, що візуалізує результати аналізу зображення, отриманих функцією `AnalyzeImageWithBoundingBox`. До неї передається значення `predictedClass`, що є цілим числом, яке представляє прогнозований клас зображення, а також `maxProbability`, що є плаваючим числом, яке представляє максимальну ймовірність прогнозованого класу. А ще передається саме зображення, для її повної обробки

Це зображення потім буде використовуватись у коді та переміститься у папку store

```
212 void VisualizeResults(Image* img, int predictedClass, float maxProbability) {
213     uint8_t* image_data = img.data;
214     int width = img.width;
215     int height = img.height;
216
217     for (int i = 0; i < predictedClass; i++) {
218         X_DrawRectangle(image_data, width, height, predictedClass, maxProbability);
219         X_DrawText(image_data, width, height, predictedClass, maxProbability);
220     }
221
222     return img;
223 }
```

Рисунок 3.12 – Код функції VisualizeResults

Як ви можете помітити візуалізація відбувається за допомогою 2 методів, першого, що малює квадрати, та другого, що візуалізує текст та ймовірності. Ці 2 методи є вбудованими у інструмент X-CUBE-AI та трохи адаптовані мною, для моїх цілей, а не вивід на LCD дисплей.

3.2.5 Функція main

У main, ми виконуємо усі дії, які я зазначав на початку переліку усіх функцій. Код буде представлений в окремому додатку, з ним ви можете ознайомитись детальніше там.

Загалом, функція main демонструє базову структуру для читання зображень з USB-накопичувача, їх обробки за допомогою спеціальної функції processImage та зберігання оброблених зображень в іншій папці на зовнішньому накопичувачі мікроконтролерної системи.

ВИСНОВОК ДО РОЗДІЛУ 3

При розробці, я стикався з складнощами, які вдалося подолати. Вони виникли у зв'язку з незвичним використанням аналізу зображень мікроконтролера, браком OpenSorce прикладів, браком досвіду з комп'ютерним зором. Але розібравшись, можу поділитись досвідом та практичною реалізацією проекту. Хочу підкреслити, що усі задачі, які я ставив перед своїм проектом були виконані. Перевагою мого проекту є портативність мого виду аналізу, маючи повну систему та USB накопичувач з фото – система з легкістю виконує операції закладені системою, копіює файли, переміщає, видаляє, аналізує, обробляє, візуалізує та відправляє готові результати назад на накопичувач

Була спокуса інтегрувати також змогу машинного навчання, вже під час інтеграції коду, але проблеми виникли при руйнуванні структури аналізу зображень. Для такої інтеграції, треба все одно, первино обробити багато зображень і так само підготувати їх до інтерпритації, а що буде якщо зробити щось не так? – система може зламати свої попередні конфігурації нейронних мереж.

У цьому розділі, було також показано детальний опис для створення власної модель за допомогою зовнішніх розробок STM32, що гарно інтегруються на ці мікроконтролери, мій проєкт є підтвердженням цьому. Як і обіцяв детально задокументував процес розробки та опис усіх функцій та інструментів, для реалізації подібного проекту. Саме ці технологічні розробки дійсно надихають та рухають перемогу України все більше та більше.

Що важливо, під час опису функцій я зосередився на основній сфері реалізації, а саме реалізації аналізу зображень, який і був основною метою

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		61

даної бакалаврської праці. Мною було озвучено основні механізми та ідеї, а також використання методів.

Під час написання коду, він поступово зменшувався та оптимізувався, неоднократно набував нових форм. Звичайно використовував свій досвід набутий в КПІ, зокрема роботу з процесорами на базі Cortex M та i STM32 системами

					ІАЛЦ.467200.003 ПЗ	Арк.
						62
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 4

ТЕСТУВАННЯ ТА АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

Оскільки основний код працює лише на справжній та повній збірці системи, що не можливо за моїх умов, та і нашою задачею усієї дипломної роботи є розробка та моделювання. Було вирішено розробити спеціальну програму, що імітує принципи роботи мікроконтролера, але робить це на комп'ютері, реалізовано дане рішення було за допомогою ознайомлення з документацією X-CUBE-AI та інструментами Tensorflow. Звичайно, самі тестування також будуть проходити у наших середовищах розробки мікроконтролера, зокрема STM32CubeIDE 1.14.1 та STM32CubeMX. Але ще раз наголошую, мені не вдалось знайти гарний емулятор, який реалізує підключення USB та зовнішнього накопичувача Quad-SPI підключення.

4.1 Тестування та аналіз main_model

Підчас запуску та аналізу моделі отримав такі результати зайняття пам'яті:

```
Complexity: 72664934 MACC
Used Flash: 595.66 KiB (595.66 KiB over 1024.00 KiB Internal)
Used Ram: 465.36 KiB (465.36 KiB over 512.00 KiB Internal)
Achieved compression: -
Analysis status: done
```

Рисунок 4.1 – Статистика пам'яті main_model після додавання та аналізу

За цими результатами можна побачити, що Ram пам'ять у нас залишається не так багато, а ось Flash пам'ять на половину вільна, давайте конкретно розберемо, по кожному типу пам'яті, в чому тут криється мінуси та плюси

Зокрема надлишок Flash пам'яті дозволяє нам використовувати більше місця для коду та даних, що у нашому проєкті. При виконання роботи це було вагомим, для реалізації, оскільки реалізовуємо досить складну програму, яка потребує багато місця для зберігання коду та даних .jpg формату. Також надлишок надає можливість оновлювати прошивку швидше – оскільки з'являється змога оновлювати прошивку мікроконтролера без необхідності видаляти існуючий код. Також зайву Flash пам'ять можна використати для зберігання даних аналізу, що я б не рекомендував робити, оскільки система націлена на велику обробку зображень. Звичайно що є і мінуси надлишку - збільшення споживання енергії, дорожча ціна для більшої пам'яті, та інші переваги. Але для нашого тестування та проєкту мікроконтролер проявив себе чудово

Нестача RAM пам'яті, ставить нас перед неможливістю виконувати певні дії паралельно. До прикладу, якщо програма потребує більше RAM пам'яті, для аналізу або малювання, ніж доступно, система працюватиме повільніше. Також ще одною причиною тотальної оптимізації коду є невеликий запас RAM. З коду прибрано усе зайве, саме для того, щоб система працювала правильно та швидше

Зараз проведемо тестування моделі, створимо порівняльну таблицю

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		64

Evaluation report (summary)									
Output	acc	rmse	mae	l2r	mean	std	nse	cos	tensor
X-cross #1	n.a.	0.0005063	0.0000638	0.0007189	0.0000021	0.0005063	0.9999990	0.9999995	conversion_148,
X-cross #2	n.a.	0.0091444	0.0055244	0.0748443	0.0000212	0.0091444	0.9941458	0.9971990	conversion_155,
X-cross #3	n.a.	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	1.0000000	1.0000000	conversion_163,
acc : Classification accuracy (all classes) rmse : Root Mean Squared Error mae : Mean Absolute Error l2r : L2 relative error nse : Nash-Sutcliffe efficiency criteria, bigger is better, best=1, range=(-inf, 1] cos : COsine Similarity, bigger is better, best=1, range=(0, 1]									
Creating txt report file C:\Users\beano\.stm32cubemx\main_model_output\main_model_validate_report.txt elapsed time (validate): 78.133s									

Рисунок 4.2 – Статистика пам'яті main_model після додавання та аналізу

Таблиця 2 – Перевірка мережі main_model, час аналізу моделі

Назва перевірки	Час (с)
Тестування 1	63.248
Тестування 2	78.133
Тестування 3	30.760
Тестування 4	33.968
Тестування 5	31.206
Середній час	47,463

Час різний, і на що я б хотів наголосити, що даний час пришвидшився завдяки додатковому хешуванню та накопиченню даних тестування, ніби запускаючи усі необхідні процеси, але не очищує їх. У мене було припущення, що з кожним запуском, об'єм пам'яті зростатиме, але на практиці я пересвідчився у тому, що після виконання одного або 5 тестувань, мережа не змінює об'єм пам'яті, це відбувається лише після першого запуску. Досі не впевнений, що не варто очищувати пам'ять, виділену для мережі, оскільки це може призвести до помилок в коді, за нестачі RAM пам'яті або, що гірше, до неправильних результатів аналізу, оскільки модель ігнорує нове зображення



Рисунок 4.3 – Статистика пам'яті main_model після проведення тестувань

4.2 Тестування розробленої системи

Трохи адаптувавши код для виконання тестування на комп'ютері та після підготовки усього необхідного, можна було розпочати загальне тестування. Хотів також урахувати час на переміщення файлів, але загалом він займає 1-3 секунди. Найдовше переміщення відбувається при архівації обробленого результату, а саме: переміщення з USB: у work, процес аналізу і обробки, створення файлів у store, копіювання на USB: та перенесення до archive. Будемо досліджувати час аналізу, а не загальну роботу системи

Підчас виконання, виникало багато помилок, що якраз свідчить про те, що обробка зроблена досить гарно, виправивши їх отримав такі результати аналізу

Згодом, після успішного запуску отримав таку картину, яка досить очевидна, реальний запуск далеко назбігається з випадковим аналізом, що був реалізований інструментом X-CUBE-AI. Кожен цикл, я очищую модель, тому і не дивно, що результат не вийшов меншим за 88.69. Велику кількість часу програми вона не тільки аналізує, а і обробляє

зображення. Для релевантної тестування виконував перший запуск тестування у зовнішній системі, що розташована на хмарі

Таблиця 3 – Реальний час аналізу та обробки зображень

Назва тестування	Час (с)
Зображення 1	105.25
Зображення 2	95.79
Зображення 3	80.76
Зображення 4	98.96
Зображення 5	88.69
Середній час	93,89

Таблиця 4 – Час аналізу цих зображень за допомогою інструменту object_detection tensorflow з 13 GB оперативної пам'яті, та GPU до Python

Назва тестування	Час (с)
Зображення 1	14.157492876052856
Зображення 2	9.363259315490723
Зображення 3	9.059394598007202
Зображення 4	9.643141031265259
Зображення 5	9.829904317855835
Середній час	10,4106359

Порівняно з потужними комп'ютерами, час на виконання моєї системи досить непоганий, при умові, що можна залишати мережу, а не перепідключати її, це може дати приріст у часі в 2 рази, як це було показано, тестуючи без оновлення моделі, на потужному комп'ютері, ми маємо аналіз, що менше 1 секунди. Результати аналізу збігаються,

оскільки модель була взята однакова, використовував той самий файл .tflite що і у моїй системі, та отримав такі результати:

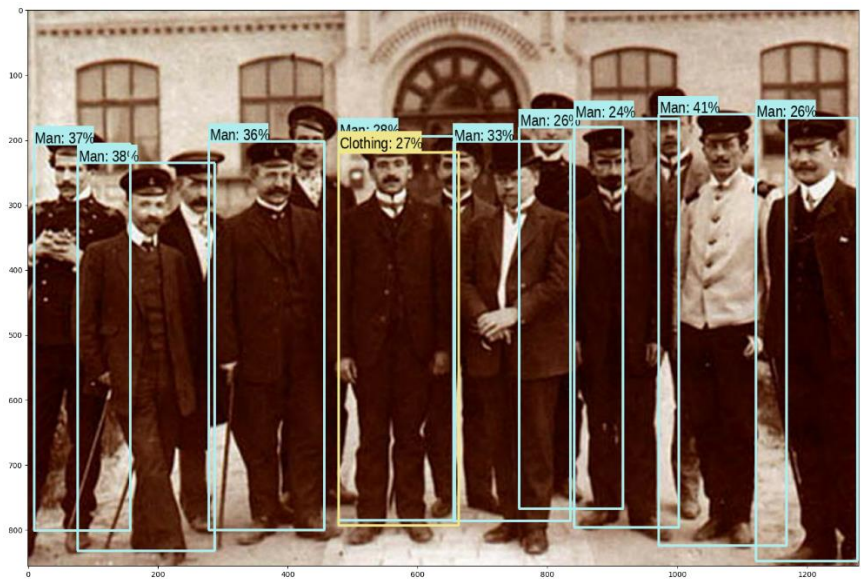


Рисунок 4.4 – Аналіз зображення 1 за допомогою мікроконтролера

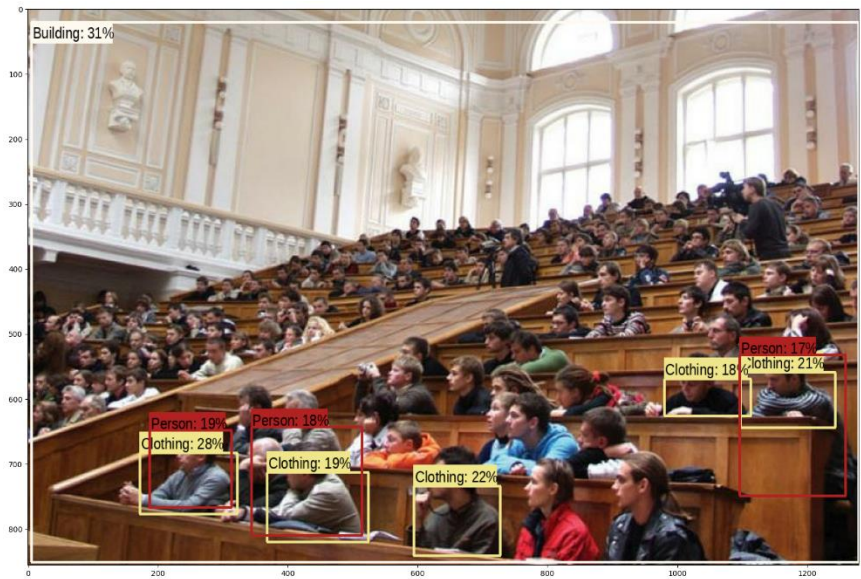


Рисунок 4.3 – Аналіз зображення 4 за допомогою комп'ютера

ВИСНОВОК ДО РОЗДІЛУ 4

Підбиваючи підсумки тестування та загалом останнього розділу у моїй роботі, хотів би підкреслити, що дані тестування, подекуди, здивували. Провів тестування та порівняння часу, що зараз розглянемо

Підчас тестування моделі ми побачили, що найменшим часом є 30.76 секунд, але ми враховуємо, що був вторинний запуск, без очищення даних попереднього тестування. Це лише спонукає на думку, що час моделювання може знизитись з 60-70 секунд до 30, але наскільки правильна буде аналіз, враховуючи, що ми маємо не найкращу з усіх модель, в силу не достатку пам'яті у мікроконтролері, тож ризикувати я не вирішив, та і загалом, лишається не так багато пам'яті, що ми бачили на рис. 4.3.

Загалом середній показник тестування дорівнює 47,463, на практиці, як ми бачимо у таблиці 3, середній час становить 93,89 секунд, що майже в 2 рази більше. Це пов'язано з тим що окрім аналізу ми виконуємо і інші операції, збереження зображення, додаткову обробку, накладання квадратів на об'єкти, класифікацію та ймовірносте, а також це може бути зумовлено браком RAM пам'яті, що змушують аналізувати та обробляти зображення сегментами

Порівнюючи час аналізу комп'ютерної моделі та мікроконтролерної системи, ми можемо побачити вагому різницю, готова tensorflow система аналізує зображення в 9 разів швидше, але тут варто та слід зауважити, що комп'ютерна модель виконує лише первину обробку та аналіз об'єктів, в той час як наша функція processImage виконує і збереження і створення зображення, та обробку та додавання. Таке порівняння нам говорить про те, що так комп'ютерні моделі швидше, але саме за

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		69

допомогою інструментів нашої системи можна аналізувати зображення портативна використовуючи невеликий обсяг пам'яті та ресурси. Модель може гарно себе проявити в польових умовах, підчас виконання певних розвідувальних місій та загалом гарно підійти для модернізації та автоматизації систем

Оскільки результати виявились майже однаковими, то система працює коректно, а часова різниця зумовлена нестачею ресурсу, або навіть виявлення чи не виявлення систем зумовлена різним підбором інструментів, так до прикладу X-CUBE-AI та Tensorflow використовують різні інструменти та підходи до однакових мереж

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		70

ВИСНОВКИ

Створення моделі системи аналізу, за допомогою комп'ютерного зору, на базі мікроконтролера STM-32 пройшло успішно, у чому можна переконатися вище. У всіх розділах ми детальніше вивчили принципи та роботу таких систем, порівняли усі наявні інструменти та технології. Розібрали процес створення нейронних моделей самостійно та зацентували увагу на аналізі зображень у мікроконтролері при його розробці. Проєкт можна вважати завершеним, оскільки в ньому було описано і задокументовано навіть процес розробки подібної системи, для бажаючих реалізувати подібний проєкт

Дана робота спрямована на популяризацію теми розробки та інтеграції AI у різні системах. Моя система ідеально може підійти для інтеграції її у будь-які сфери життя, від військових до цивільних. Зараз система аналізує прості об'єкти, людей, тварин, будівлі тощо, але за допомогою вказівок у розділі 2 та 3 ви можете самостійно розробити власну модель та легко інтегрувати її у систему маючи ті самі інструменти, що були описанні

Технічне завдання виконано у повному обсязі, у чому можна переконатися вище

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		71

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. STMicroelectronics. STM32Cube function pack for high performance STM32 with artificial intelligence (AI) application for Computer Vision [Електронний ресурс] / STMicroelectronics. – 2024. – Режим доступу до ресурсу: <https://www.st.com/en/embedded-software/fp-ai-vision1.html>.
2. How to use transfer learning to perform image classification on STM32 [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://wiki.st.com/stm32mcu/wiki/AI:How_to_use_transfer_learning_to_perform_image_classification_on_STM32?utm_source=wiki.st.com.
3. How to run OpenCV on STM32 MCU [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://medium.com/@deryugin.denis/how-to-run-opencv-on-stm32-mcu-b581f42b0766>.
4. TensorFlow Lite for Microcontrollers [Електронний ресурс] – Режим доступу до ресурсу: <https://www.tensorflow.org/lite/microcontrollers>.
5. Build and convert models [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/lite/microcontrollers/build_convert#model_conversion.
6. Running TensorFlow Lite for Microcontrollers on Contiki-NG [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://atise.medium.com/running-tensorflow-lite-for-microcontrollers-on-contiki-ng-d938f25193f5>.
7. Мікроконтролер [Електронний ресурс] // Вікіпедія – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/%D0%9C%D1%96%D0%BA%D1%80%>

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		72

[D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80.](#)

8. STMicroelectronics – STM32 model zoo [Електронний ресурс]. – 2024. – Режим доступу до ресурсу:
<https://github.com/STMicroelectronics/stm32ai-modelzoo?tab=readme-ov-file>.
9. Ian G. Deep Learning (Adaptive Computation and Machine Learning series) / G. Ian, Y. Bengio, A. Courville..
10. Getting Started with STM32 - Introduction to STM32CubeIDE [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.digikey.com/en/maker/projects/getting-started-with-stm32-introduction-to-stm32cubeide/6a6c60a670c447abb90fd0fd78008697>.
11. Getting Started with STM32 - How to Use SPI [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.digikey.com/en/maker/projects/getting-started-with-stm32-how-to-use-spi/09eab3dfe74c4d0391aaaa99b0a8ee17>.
12. Що таке комп'ютерний зір? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.unite.ai/uk/what-is-computer-vision/>.
13. ЗАГАЛЬНІ ВІДОМОСТІ ПРО КОМП'ЮТЕРНИЙ ЗІР [Електронний ресурс] – Режим доступу до ресурсу:
https://learn.ztu.edu.ua/pluginfile.php/272132/mod_resource/content/1/%D0%A8%D0%86_%D0%9B-11_%D0%9A%D0%97%D1%96%D1%80_2022.pdf.
14. STM32 32-bit Arm Cortex MCUs [Електронний ресурс] – Режим доступу до ресурсу: <https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html>.

15. Learn to Design Your Own Boards [Електронний ресурс]. – 2020. –
Режим доступу до ресурсу: <https://www.udemy.com/course/learn-to-design-your-own-boards/>.
16. Convolutional Neural Network (CNN) [Електронний ресурс] – Режим
доступу до ресурсу: <https://www.tensorflow.org/tutorials/images/cnn>.
17. Цифрова обробка сигналів [Електронний ресурс] – Режим доступу
до ресурсу:
https://web.posibnyky.vntu.edu.ua/fksa/2kvetnyj_komp'yuterne_modely_uvannya_system_procesiv/t2/1..htm.
18. Binarize 2-D grayscale image or 3-D volume by thresholding
[Електронний ресурс] – Режим доступу до ресурсу:
<https://www.mathworks.com/help/images/ref/imbinarize.html>.
19. Image segmentation [Електронний ресурс] – Режим доступу до
ресурсу: <https://www.tensorflow.org/tutorials/images/segmentation>.
20. You Only Look Once: Unified, Real-Time Object Detection / J.Redmon,
S. Divvala, R. Girshick, A. Farhadi..
21. Object detection [Електронний ресурс] – Режим доступу до ресурсу:
https://www.tensorflow.org/lite/examples/object_detection/overview.
22. Різниця між RISC і CISC [Електронний ресурс]. – 2024. – Режим
доступу до ресурсу: <https://www.guru99.com/uk/risc-vs-cisc-differences.html>.
23. Arm® Cortex®-M7 in a nutshell [Електронний ресурс] – Режим
доступу до ресурсу: https://www.st.com/content/st_com/en/arm-32-bit-microcontrollers/arm-cortex-m7.html.
24. ARM Cortex-M [Електронний ресурс] – Режим доступу до ресурсу:
https://en.wikipedia.org/wiki/ARM_Cortex-M.
25. Serra J. Image Analysis and Mathematical Morphology / Jean Serra.,
1983.

26. Фільтрація зображень [Електронний ресурс] – Режим доступу до ресурсу:
https://web.posibnyky.vntu.edu.ua/fksa/2kvetnyj_komp'yuterne_modely_uvannya_system_procesiv/t2/213..htm.
27. Комп'ютерний зір – це майбутнє автоматизації тестування програмного забезпечення – історія минулого, сьогодення та майбутнього [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.zaptest.com/uk/%D0%BA%D0%BE%D0%BC%D0%BF%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B8%D0%B9-%D0%B7%D1%96%D1%80-%D1%86%D0%B5-%D0%BC%D0%B0%D0%B9%D0%B1%D1%83%D1%82%D0%BD%D1%94-%D0%B0%D0%B2%D1%82%D0%BE%D0%BC>.
28. How Does Tesla Use AI in Cars [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://disruptionhub.com/how-does-tesla-use-ai-in-cars/>.
29. CMSIS-DSP [Електронний ресурс] – Режим доступу до ресурсу:
<https://github.com/ARM-software/CMSIS-DSP>.
30. TensorFlow [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.nvidia.com/en-us/glossary/tensorflow/>.
31. MicroPython libraries [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.openmv.io/library/index.html>.
32. OpenMV: Small - Affordable - Expandable [Електронний ресурс] – Режим доступу до ресурсу: <https://openmv.io/>.
33. STM32F767VG [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.st.com/en/microcontrollers-microprocessors/stm32f767vg.html>.

34. Що таке штучний інтелект (ШІ)? [Електронний ресурс] // TheTransmitted. – 2023. – Режим доступу до ресурсу: <https://thetransmitted.com/adlucem/shho-take-shtuchnij-intelekt-shi/>.
35. Tensorflow Object Detection Walkthrough [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://github.com/nicknochnack/TFODCourse>.
36. MobileNet V1 [Електронний ресурс] – Режим доступу до ресурсу: https://huggingface.co/docs/transformers/model_doc/mobilenet_v1.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		76

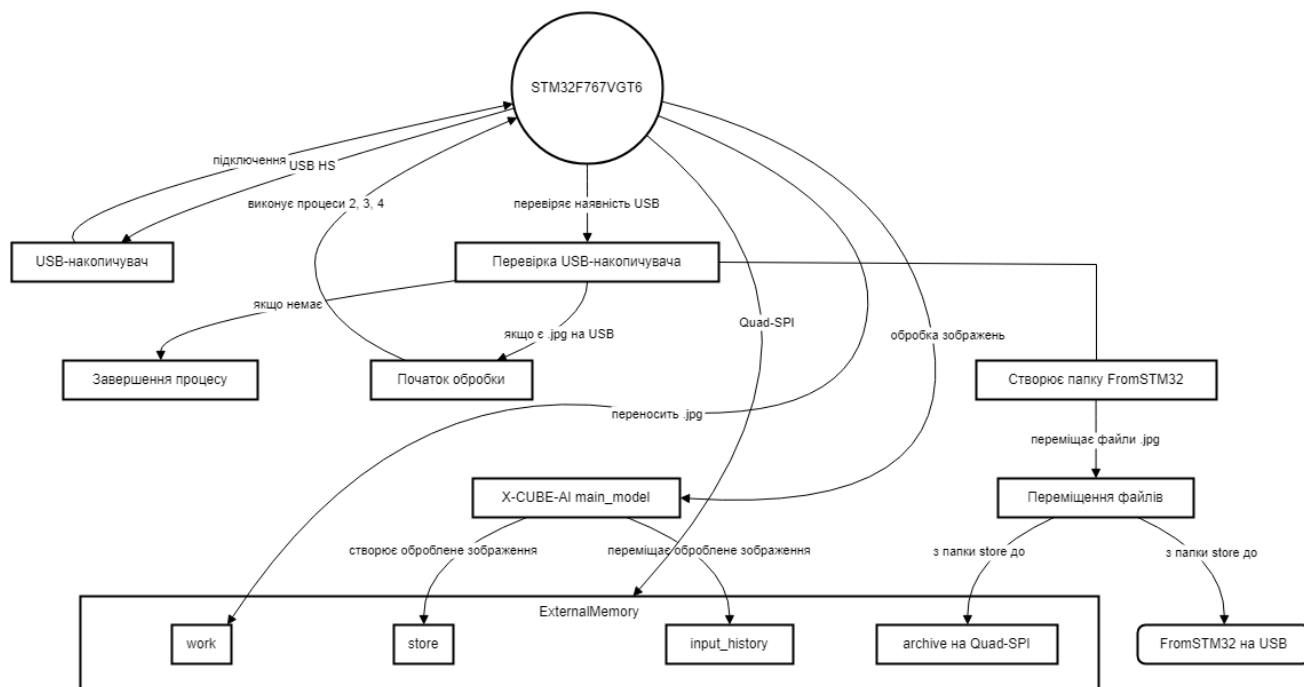
ДОДАТОК 1

**Система аналізу зображень з використанням мікроконтролера
STM-32**

**Схема аналізу зображень
ІАЛЦ.467200.004 Е1**

Аркушів 1

Київ 2024 р



ІАЛЦ.467200.004 Е1					Літ. Аркуш Аркшів 1 1 КПІ ім. Ігоря Сікорського, ФІОТ, гр. ІО-02		
		№ докум.	Підпис	Дата			
Розробив	Корнієнко Д.						
Перевірив	Гончаренко О. О.						
Н. Контр.	Виноградов Ю. М.				Аналіз зображень за допомогою Мікроконтролера STM-32 Схема аналізу зображень Схема структурна		
Затвердив							

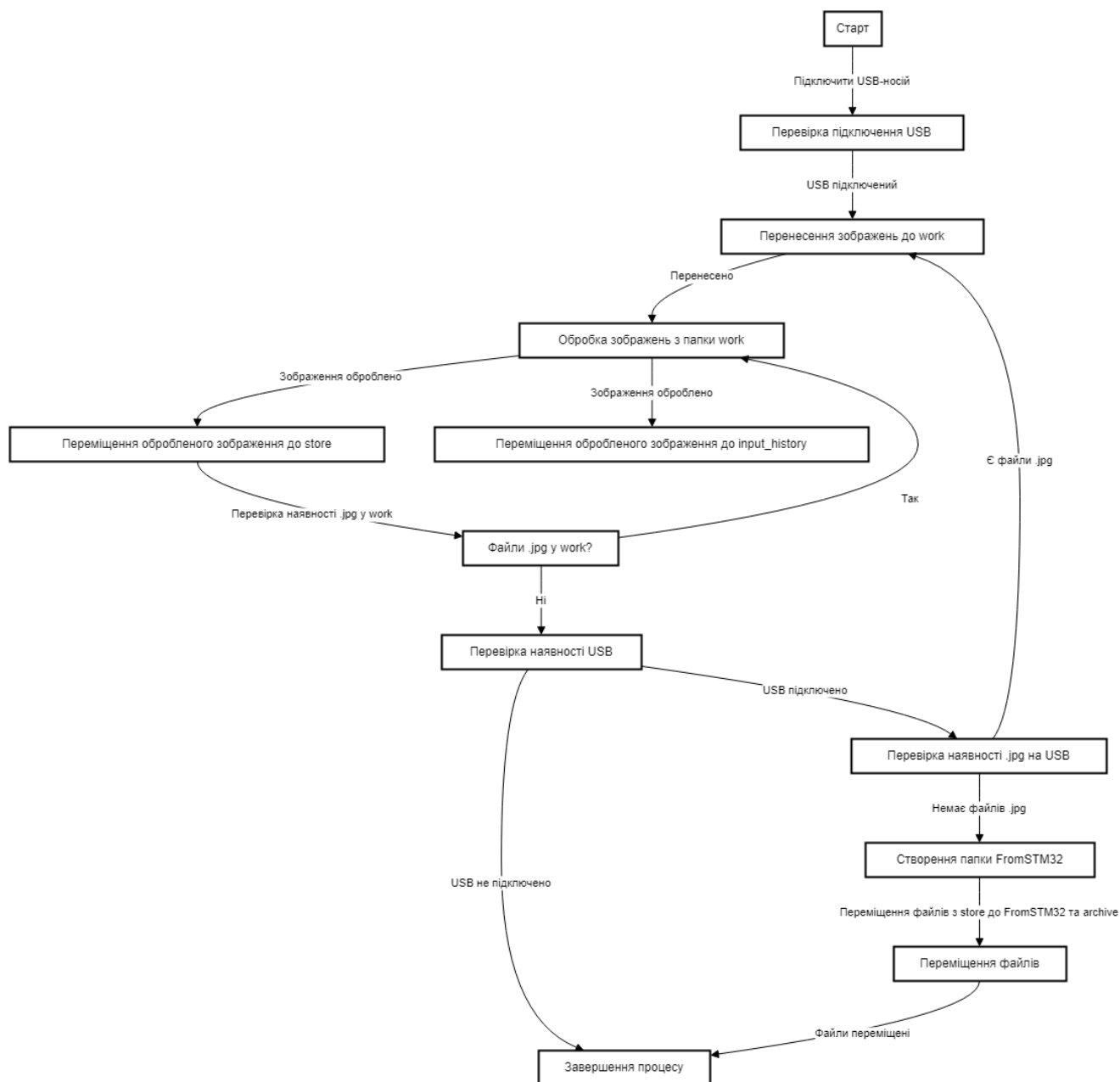
ДОДАТОК 2

**Система аналізу зображень з використанням мікроконтролера
STM-32**

**Фільтрація зображення
ІАЛЦ.467200.005 Д2**

Аркушів 1

Київ 2024 р



					ІАЛЦ.467200.005 Д2							
		№ докум.	Підпис	Дата								
Розробив		Корнієнко Д. В.			Аналіз зображень за допомогою Мікроконтролера STM-32 Фільтрація зображення Схема функціональна			Літ.	Аркуш	Аркушів		
Перевірив		Гончаренко О. О.								1	1	
Н. Контр.		Виноградов Ю. М.						КПІ ім. Ігоря Сікорського, ФІОТ, гр. ІО-02				
Затвердив												

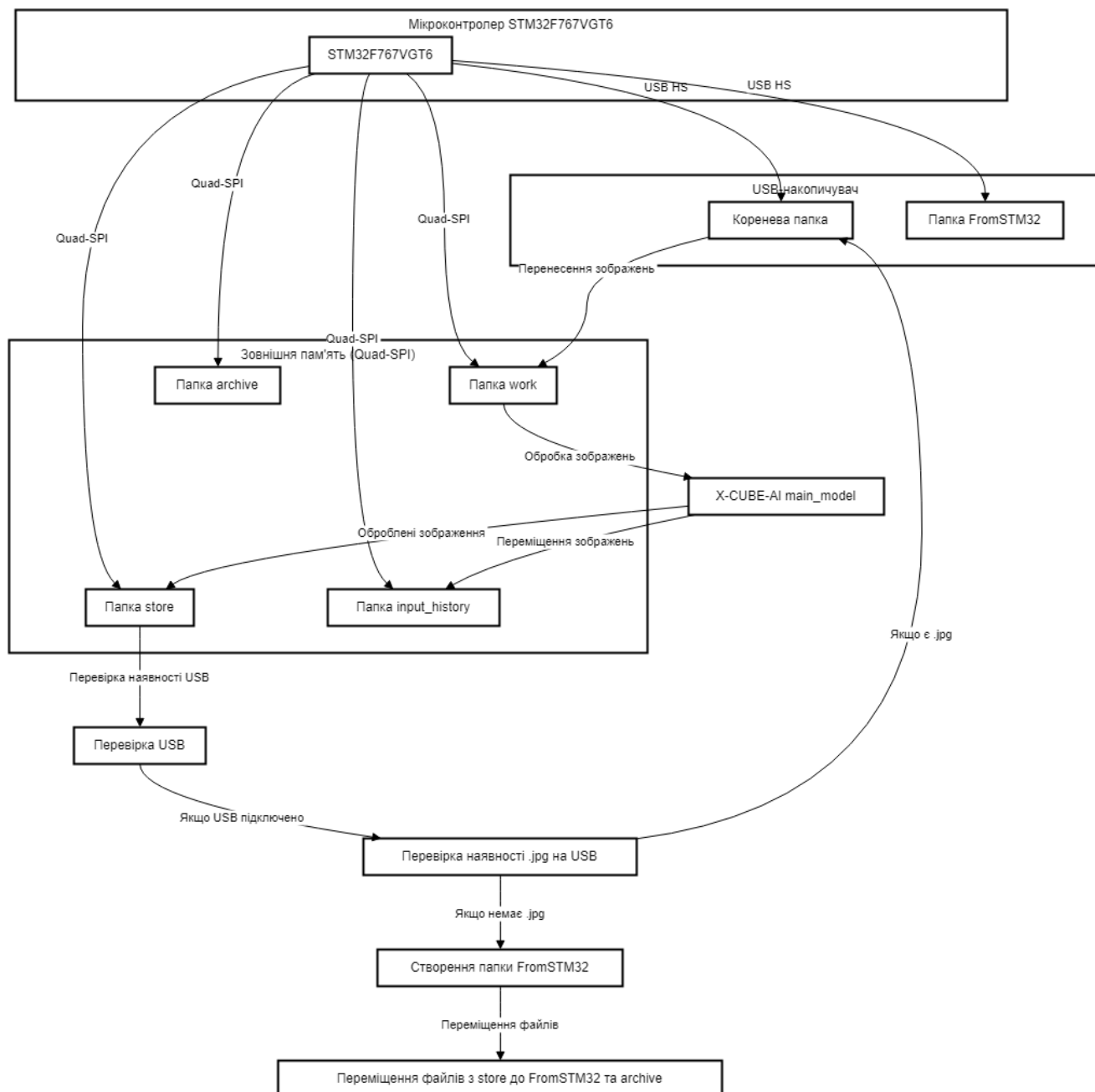
ДОДАТОК 3

Система аналізу зображень з використанням мікроконтролера
STM-32

Алгоритм взаємодії аналізу зображень
ІАЛЦ.467200.006 ДЗ

Аркушів 1

Київ 2024 р



ІАЛЦ.467200.006 ДЗ					Аналіз зображень за допомогою Мікроконтролера STM-32 Алгоритм взаємодії аналізу зображень Схема принципова			Літ. Аркуш Аркушів 1 1 КПІ ім. Ігоря Сікорського, ФІОТ, гр. ІО-02		
		№ докум.	Підпис	Дата						
Розробив	Корнієнко Д. В.									
Перевірив	Гончаренко О. О.									
Н. Контр.	Виноградов Ю. М.									
Затвердив										

ДОДАТОК 4

**Система аналізу зображень з використанням мікроконтролера
STM-32**

**Текст програмного коду
ІАЛЩ.467200.007 Д4**

Аркушів 14

Київ 2024 р

```
/* USER CODE BEGIN Header */

/**
 * ****
 *
 * @file      : main.c
 * @brief     : Main program body
 *
 * ****
 *
 * @attention
 *
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 * ****
 */

/* USER CODE END Header */

/* Includes ----- */

#include "main.h"
#include "fatfs.h"

/* Private includes ----- */

/* USER CODE BEGIN Includes */

#include "stm32f7xx_hal.h"
#include "../X-CUBE-AI/App/main_model.h"

/* USER CODE END Includes */

/* Private typedef ----- */

/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ----- */
```

					ІАЛЦ.467200.007 Д4						
		№ докум.	Підпис	Дата							
Розробив	Корнієнко Д. В.				Аналіз зображень за допомогою Мікроконтролера STM-32 Текст програмного коду			Літ.	Аркуш	Аркушів	
Перевірив	Гончаренко О. О.								1	14	
								КПІ ім. Ігоря Сікорського, ФІОТ, гр. ІО-02			
Н. Контр.	Виноградов Ю. М.										
Затвердив											

```

/* USER CODE BEGIN PD

#define WORK_FOLDER "work"
#define STORE_FOLDER "store"
#define INPUT_HISTORY_FOLDER "input_history"
#define ARCHIVE_FOLDER "archive"
#define FROM_STM32_FOLDER "FromSTM32"
/* USER CODE END PD */

/* Private macro -----*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables -----*/

QSPI_HandleTypeDef hqspi;

/* USER CODE BEGIN PV */
FATFS fs;
DIR dir;
FRESULT fr;
FIL MyFile;
UINT bw;
/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MPU_Config(void);
static void MX_GPIO_Init(void);
static void MX_QUADSPI_Init(void);
static void MX_USB_OTG_HS_USB_Init(void);
/* USER CODE BEGIN PFP */
void copyDir(char* srcDir, char* dstDir);
void processImage(char* imagePath);

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		2

```

void AnalyzeImageWithBoundingBox(Image* img);

void VisualizeResults(int predictedClass, float maxProbability);

/* USER CODE END PFP */

/* Private user code -----*/

/* USER CODE BEGIN 0 */

void copyDir(char* srcDir, char* dstDir) {
    char* path[255];
    char* fileName;
    fr = f_opendir(&dir, srcDir);
    if (fr != FR_OK) {
        return;
    }
    while ((fileName = f_readdir(&dir, path)) != NULL) {
        if (*path[0] == '.') {
            continue;
        }
        char* fullSrcPath = HAL_Malloc(strlen(srcDir) + strlen(path) + 2);
        sprintf(fullSrcPath, "%s/%s", srcDir, path);

        char* fullDstPath = HAL_Malloc(strlen(dstDir) + strlen(path) + 2);
        sprintf(fullDstPath, "%s/%s", dstDir, path);

        if (f_stat(fullSrcPath, &MyFile) != FR_OK) {
            HAL_Free(fullSrcPath);
            HAL_Free(fullDstPath);
            continue;
        }
        if (MyFile.fsize == 0) {
            if (f_mkdir(fullDstPath) != FR_OK) {
                HAL_Free(fullSrcPath);
                HAL_Free(fullDstPath);
                continue;
            }
        }
        copyDir(fullSrcPath, fullDstPath);
    }
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    } else {

        f_open(&MyFile, fullSrcPath, FA_READ);

        fr = f_open(&MyFile, fullDstPath, FA_CREATE_ALWAYS | FA_WRITE);

        if (fr != FR_OK) {

            f_close(&MyFile);

            HAL_Free(fullSrcPath);

            HAL_Free(fullDstPath);

            continue;

        }

        while ((bw = f_read(&MyFile, MyFile.buf, MyFile.fsize, &bw)) == bw) {

            f_write(&MyFile, MyFile.buf, bw, &bw);

        }

        f_close(&MyFile);

        f_close(&MyFile);

    }

    HAL_Free(fullSrcPath);

    HAL_Free(fullDstPath);

}

f_closedir(&dir);

}

```

```

void processImage(char* imagePath) {

    MainModel model;

    // Load model

    if (model.LoadModel() != 0) {

        Error_Handler();

        return;

    }

    // Load image

    Image img;

    if (LoadImage(imagePath, &img) != 0) {

        Error_Handler();

    }

}

```

					ІАЛЦ.467200.007 Д4	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        model.ReleaseModel();

        return;
    }

    SaveImage(&img, INPUT_HISTORY_FOLDER);

    AnalyzeImageWithBoundingBox(&img);

    FreeImage(&img);
    model.ReleaseModel();
}

void AnalyzeImageWithBoundingBox(Image* img) {
    MainModel model;

    // Load model
    if (model.LoadModel() != 0) {
        Error_Handler();
        return;
    }

    // Check image dimensions
    if (img->data == NULL || img->width <= 0 || img->height <= 0) {
        Error_Handler();
        model.ReleaseModel();
        return;
    }

    // Prepare data for model
    float* inputData = PrepareModelData(img);
    if (inputData == NULL) {
        Error_Handler();
        model.ReleaseModel();
        return;
    }
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

```

// Analyze image using model
float* predictions = model.Predict(inputData, img->width * img->height);
if (predictions == NULL) {
    Error_Handler();
    FreeImageData(inputData);
    model.ReleaseModel();
    return;
}

// Process prediction results
int numClasses = model.GetNumClasses();
float classProbabilities[numClasses];

for (int i = 0; i < numClasses; i++) {
    classProbabilities[i] = 0.0f;
    for (int j = 0; j < img->width * img->height; j++) {
        classProbabilities[i] += predictions[j * numClasses + i];
    }
}

int predictedClass = 0;
float maxProbability = 0.0f;

for (int i = 0; i < numClasses; i++) {
    if (classProbabilities[i] > maxProbability) {
        maxProbability = classProbabilities[i];
        predictedClass = i;
    }
}

// Visualize results (replace with your implementation)
VisualizeResults(img, predictedClass, maxProbability);

FreeImageData(inputData);

```

					ІАЛЦ.467200.007 Д4	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        model.ReleaseModel();
    }

void VisualizeResults(Image* img, int predictedClass, float maxProbability) {
    uint8_t* image_data = img.data;
    int width = img.width;
    int height = img.height;

    for (int i = 0; i < predictedClass; i++) {
        X_DrawRectangle(image_data, width, height, predictedClass, maxProbability);
        X_DrawText(image_data, width, height, predictedClass, maxProbability);
    }

    return img;
}

/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MPU Configuration-----*/
    MPU_Config();

    /* Enable the CPU Cache */

    /* Enable I-Cache-----*/

```

					ІАЛЦ.467200.007 Д4	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		

```

SCB_EnableICache();

/* Enable D-Cache-----*/
SCB_EnableDCache();

/* MCU Configuration-----*/

/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_QUADSPI_Init();
MX_USB_OTG_HS_USB_Init();
MX_FATFS_Init();
/* USER CODE BEGIN 2 */
// Initialize peripherals
MX_QSPI_Init();
MX_FATFS_Init();

// Check USB connection
if (HAL_USB_Host_Init(&USBH_Handle, HOST_CLASS_STORAGE) != HAL_OK) {
    Error_Handler();
}

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		8

```

while (1) {
    // Check USB connection
    if (HAL_USB_Host_GetState(&USBH_Handle) == HOST_OK) {
        if (f_mount(&fs, "USB:", 1) != FR_OK) {
            Error_Handler();
        }

        copyDir("USB:/", WORK_FOLDER);

        f_unmount("USB:");
    }

    // Process images in work folder
    fr = f_opendir(&dir, WORK_FOLDER);
    if (fr != FR_OK) {
        continue;
    }

    char path[255];
    char* fileName;
    while ((fileName = f_readdir(&dir, path)) != NULL) {
        if (*path == '.') {
            continue;
        }

        char* fullPath = HAL_Malloc(strlen(WORK_FOLDER) + strlen(path) + 2);
        sprintf(fullPath, "%s/%s", WORK_FOLDER, path);

        // Save raw image to input_history folder
        char* inputHistoryPath = HAL_Malloc(strlen(INPUT_HISTORY_FOLDER) + strlen(path) + 2);
        sprintf(inputHistoryPath, "%s/%s", INPUT_HISTORY_FOLDER, path);
        fr = f_rename(fullPath, inputHistoryPath);
        if (fr != FR_OK) {
            HAL_Free(fullPath);
        }
    }
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        HAL_Free(inputHistoryPath);

        continue;
    }

    // My Main Function

    processImage(inputHistoryPath);

    // Move processed image to store folder

    char* storePath = HAL_Malloc(strlen(STORE_FOLDER) + strlen(path) + 2);

    sprintf(storePath, "%s/%s", STORE_FOLDER, path);

    fr = f_rename(inputHistoryPath, storePath);

    if (fr != FR_OK) {

        HAL_Free(fullPath);

        HAL_Free(inputHistoryPath);

        HAL_Free(storePath);

        continue;

    }

    HAL_Free(fullPath);

    HAL_Free(inputHistoryPath);

    HAL_Free(storePath);

}

f_closedir(&dir);

// Wait before the next check

HAL_Delay(5000);

}

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

```

					ІАЛЦ.467200.007 Д4	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    /* USER CODE BEGIN 3 */

}

/* USER CODE END 3 */

}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
    */
    if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
    {
        Error_Handler();
    }

    /** Initializes the RCC Oscillators according to the specified parameters
    * in the RCC_OscInitTypeDef structure.
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = 4;
    RCC_OscInitStruct.PLL.PLLN = 216;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 2;
    RCC_OscInitStruct.PLL.PLLR = 2;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		11

```

{
    Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
*/

RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                               |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;

RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;

RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;

RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;

RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_7) != HAL_OK)
{
    Error_Handler();
}
}

/**
 * @brief QUADSPI Initialization Function
 * @param None
 * @retval None
 */
static void MX_QUADSPI_Init(void)
{

    /* USER CODE BEGIN QUADSPI_Init 0 */

    /* USER CODE END QUADSPI_Init 0 */

    /* USER CODE BEGIN QUADSPI_Init 1 */

    /* USER CODE END QUADSPI_Init 1 */

    /* QUADSPI parameter configuration*/

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		12

```

hqspi.Instance = QUADSPI;
hqspi.Init.ClockPrescaler = 1;
hqspi.Init.FifoThreshold = 4;
hqspi.Init.SampleShifting = QSPI_SAMPLE_SHIFTING_NONE;
hqspi.Init.FlashSize = 22;
hqspi.Init.ChipSelectHighTime = QSPI_CS_HIGH_TIME_1_CYCLE;
hqspi.Init.ClockMode = QSPI_CLOCK_MODE_0;
hqspi.Init.FlashID = QSPI_FLASH_ID_1;
hqspi.Init.DualFlash = QSPI_DUALFLASH_DISABLE;
if (HAL_QSPI_Init(&hqspi) != HAL_OK)
{
    Error_Handler();
}

/* USER CODE BEGIN QUADSPI_Init 2 */

/* USER CODE END QUADSPI_Init 2 */

}

/**
 * @brief USB_OTG_HS Initialization Function
 * @param None
 * @retval None
 */
static void MX_USB_OTG_HS_USB_Init(void)
{

/* USER CODE BEGIN USB_OTG_HS_Init 0 */

/* USER CODE END USB_OTG_HS_Init 0 */

/* USER CODE BEGIN USB_OTG_HS_Init 1 */

/* USER CODE END USB_OTG_HS_Init 1 */

/* USB_OTG_HS init function */

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		13

```

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOB_CLK_ENABLE();

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return state */

    /* USER CODE END Error_Handler_Debug */
}

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		14