НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри**
Інформаційних технологій	**Комп'ютерних систем, мереж та кібербезпеки**
_____ **Боблот І.М.**	_____ **Касаткін Д.Ю.**
(підпис) (ПІБ)	(підпис) (ПІБ)
"___" _____ 2025 р.	"___" _____ 2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «Дослідження та розробка ідентифікації нетипових об'єктів за допомогою методів комп'ютерного зору з використанням дрона »

Спеціальність 123 "Комп'ютерна інженерія"
(код і назва)

Освітня програма "Комп'ютерні системи і мережі "
(назва)

Гарант освітньої програми

_____ _____ _____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

к.т.н., доц. _____ **Смолій В.В.**
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ **Макодзей І.В.**
(підпис) (ПІБ)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних систем, мереж та кібербезпеки
_____ **Касаткін Д.Ю.**
(підпис) (ПІБ)
"___" _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Макодзей Іван Валерійович
(ПІБ)

Спеціальність 123 "Комп'ютерна інженерія"
(код і назва)

Освітня програма "Комп'ютерні системи і мережі "
(назва)

Тема магістерської кваліфікаційної роботи «Дослідження та розробка ідентифікації нетипових об'єктів за допомогою методів комп'ютерного зору з використанням дрона»

затверджена наказом від "29 " жовтня 2024 р. № 1941 "С"

Термін подання завершеної роботи на кафедру _____

Вихідні дані до випускної бакалаврської роботи (дипломного проекту бакалавра)_____

Перелік питань, які потрібно розробити:

1. Загальна характеристика роботи
2. Теоретичні основи роботи системи
3. Розробка та реалізація системи
4. Тестування розробленої системи

Дата видачі завдання "_____"_____20___р.

Керівник магістерської кваліфікаційної роботи _____ **Смолій В.В.**
(підпис) (ПІБ)

Завдання прийняв до виконання _____ **Макодзей І.В.**
(підпис) (ПІБ)

РЕФЕРАТ

Пояснювальна записка: 75 сторінок, 43 рисуноків, 1 таблиця, 1 додаткот, 24 джерел.

СИСТЕМА, АНАЛІЗ, РЕАЛІЗАЦІЯ, YOLOv4, YOLOLABEL, ORANGE PI, ДРОН

Об'єкт аналізу – процес ідентифікації об'єктів у відеопотоці, отриманому з камери

Метою роботи є розробити та дослідити систему ідентифікації нетипових об'єктів на платформі Orange Pi 5.

Предмет – Методи, алгоритми та програмні засоби, що забезпечують працездатність ідентифікації

Проект складається з трьох розділів.

Перший розділ присвячено теоретичним основам, аналізу та обґрунтуванням вибору системи.

Другий розділ присвячено розробці системи, її апаратної та програмної частині .

У третьому розділі показані результати та точність системи.

В рамках даного проекту було виконано детальний аналіз та розробку системи з використанням машинного навчання для обробки зображень, яка встановлена на дрон. Було проаналізовано платформу Orange Pi 5, оцінено її характеристики та можливості використання.

ЗМІСТ

ВСТУП

Технології активно розвиваються у напрямку використання технологій штучного інтелекту для вирішення складних завдань. Однією з актуальних проблем є ідентифікація нетипових об’єктів, що може мати важливе значення у різних галузях - від моніторингу територій до забезпечення безпеки чи контролю інфраструктурних об’єктів.

Традиційні методи виявлення таких об’єктів, засновані на візуальному огляді або ручній обробці даних, є неефективними, трудомісткими та обмеженими з точки зору точності та швидкості. Тому виникає потреба у створенні автоматизованої системи, здатної швидко та надійно розпізнавати нетипові об’єкти за допомогою сучасних технологій комп’ютерного зору.

У роботі проведено дослідження та розробку системи для ідентифікації нетипових об'єктів за допомогою платформи Orange Pi 5 та особистого дрону, з використанням операційної системи Linux, маркуванням об'єктів за допомогою YOLOv4, а також написанням програмного забезпечення мовою програмування C++.

Під час виконання роботи буде проаналізовано існуючі методи ідентифікації об’єктів, обґрунтовано вибір апаратної платформи та алгоритмічного підходу, розроблено архітектуру системи, реалізовано програмний модуль і проведено тестування для оцінки ефективності створеного рішення.

Очікується, що розроблена система забезпечить точну та оперативну ідентифікацію нетипових об’єктів у реальному часі, сприятиме підвищенню ефективності моніторингу територій та відкриє перспективи для подальшого вдосконалення технологій комп’ютерного зору на базі безпілотних літальних апаратів.

Важливо зазначити, що ця дипломна робота не ставить за мету розробку готового продукту.

Актуальність дослідженя

Сьогодні системи комп’ютерного зору швидко розвиваються та активно впроваджуються у різні сфери - від промисловості до безпілотних літальних апаратів. Дрони дедалі частіше використовують для моніторингу територій, спостереження, пошуку об’єктів та збору даних у важкодоступних місцях.

Отже, актуальність даного дослідження зумовлена необхідністю створення ефективної системи ідентифікації нетипових об’єктів, здатної працювати в умовах реального часу, змінного освітлення, шумів і руху камери, на базі компактного та енергоефективного апаратного рішення. Розробка даної системи сприятиме підвищенню точності автономного моніторингу, розширенню можливостей дронів у різних сферах діяльності

Мета та завдання роботи

Метою мого дослідження є розробка, дослідження та реалізація системи ідентифікації нетипових об’єктів з платформою Orange Pi 5, яка встановлена на мій дрон. Нетиповим об’єктом в цій темі я вирішив обрати танк.

Якщо розбити на більш конкретні підцілі, то можна виділити такі:

- Розробити програмну частину системи, яка здатна збирати зображення, обробляти їх, та ідентифікувати необхідний об’єкт

- Оцінити працездатність та ефективність розробленої системи за допомогою тестування на реальних даних

Для досягнення мети, я визначив конкретні завдання:

- Обрати платформу та камеру для системи

- Зібрати дрон, який може бути сумісним з нашою системою

- Розробити код для збору та обробки зображень.

- Провести тестування системи на реальних даних.

Предмет дослідження

Предмет - методи, алгоритми та програмні засоби, що забезпечують працездатність ідентифікації. Дослідження охоплює застосування архітектур нейронних мереж типу YOLO (You Only Look Once) для розпізнавання об’єктів у реальному часі, а також адаптацію цих моделей до умов, що виходять за межі стандартних сценаріїв навчання. Також інтеграція програмної частини системи комп’ютерного зору з апаратною платформою, що забезпечує стабільну роботу в реальному часі під управлінням операційної системи Linux. Застосування таких технологій дозволяє створити універсальний підхід до аналізу відеопотоку з дронів, здатний ідентифікувати нетипові або нові об’єкти в різних умовах середовища.

Практичне значення роботи.

Практичне значення магістерської роботи полягає у створенні прототипу системи автоматизованої ідентифікації об’єктів, здатної виявляти нетипові або невідомі об’єкти з використанням дрона та вбудованої платформи Orange Pi 5. Розроблена система поєднує алгоритми глибокого навчання з оптимізованими засобами обробки відеопотоку, що дає змогу здійснювати реальний аналіз зображень без підключення до потужних серверів або хмарних рішень.

Практичні результати можуть бути використані у таких напрямах:

- Системи моніторингу та безпеки - виявлення нетипових об’єктів, транспортних засобів або потенційно небезпечних предметів у зоні спостереження.
- Військове та технічне спостереження - виявлення нетипових об’єктів техніки або аномальних змін на місцевості.

1. ТЕОРЕТИЧНІ ОСНОВИ ІДЕНТИФІКАЦІЇ ОБ’ЄКТІВ ТА РОБОТИ СИСТЕМИ

1.1 Аналіз проблеми ідентифікації нетипових об’єктів

1.1.1. Особливості роботи з об'єктами, нетиповими для навчання нейронних мереж

Виявлення об'єктів - це завдання комп'ютерного зору, яке допомагає ідентифікувати та знаходити об'єкти на зображеннях за допомогою обмежувальних рамок. Хоча обмежувальні рамки надають корисну інформацію, вони надають лише приблизну оцінку положення об'єкта та не можуть фіксувати його точну форму чи межі. Це робить їх менш ефективними в додатках, які потребують точної ідентифікації.

Нетипові об’єкти - це зразки, характеристики яких суттєво відрізняються від тих, що були представлені в навчальному наборі даних. До таких можуть належати нові типи техніки, об’єкти з деформаціями, іншими кольорами або зйомкою під нестандартними кутами [1].

Одна з головних причин таких обмежень полягає у природі самої нейронної мережі як математичного об’єкта. Якщо розглядати найпростіший випадок - нейронну мережу без прихованих шарів то вона здатна відтворювати лише лінійно роздільні функції. Одна з простих - це функція XOR. Через те, що XOR не є лінійно роздільною, її не можна представити нейронною мережею без прихованих шарів [1].

Одна з найбільших причин обмежень при використанні нейронних мереж полягає у складності інтерпретації того, що робить мережа. Мережа поступово нарощує розуміння функції, переходячи від вхідного шару до вихідного, але нам дуже важко зрозуміти цей процес нарощування та

інтерпретувати, що намагається зробити нейронна мережа. Це дуже ускладнює, якщо не робить неможливим, ручне налаштування вашої нейронної мережі змістовним чином [1].

Ще одним обмеженням є необхідність навчання (у великих обсягах), щоб отримати змістовне представлення ваших даних. Нейронні мережі, як правило, потребують великих обсягів даних, перш ніж зійтися до змістовного простору гіпотез. Це призвело до появи розумних алгоритмів для генерації навчальних даних без необхідності втручання людини, таких як генеративно-змагальні мережі, але основна проблема залишається [1].

Отже, робота з нетиповими об’єктами вимагає спеціальних підходів до навчання та тестування нейронних мереж, що включають адаптацію до нових умов, підвищення узагальнюючої здатності та вдосконалення архітектур. У цьому контексті важливим напрямом є розробка методів відкритого світу (open-world recognition) та самонавчальних систем, які здатні адаптуватися до нових типів об’єктів без необхідності повного перенавчання моделі.

1.1.2. Проблеми розпізнавання у складних або нових умовах

У процесі розпізнавання об’єктів нейронні мережі стикаються з низкою труднощів, коли умови спостереження відрізняються від тих, за яких здійснювалося навчання. До таких належать: зміна освітлення, поява шумів, часткове перекриття, низька роздільність або динамічний рух камери. У результаті точність детекції та класифікації суттєво знижується, оскільки модель не здатна коректно відновити просторові взаємозв’язки між частинами об’єкта [2].

Згідно з дослідженням Wang et al. (2020) «Robust Object Detection under Occlusion with Context-Aware CompositionalNets», більшість сучасних моделей, таких як Faster R-CNN, демонструють різке падіння точності при частковій або повній вкритості об’єкта. Автори вказують, що традиційні CNN-

архітектури мають обмежену здатність до просторового узагальнення, адже формують ознаки переважно з видимих частин зображення, і тому навіть незначне перекриття може призвести до неправильної класифікації або пропуску об’єкта [2].

Для подолання цієї проблеми у роботі Wang et al. було запропоновано архітектуру Context-Aware CompositionalNets, яка розділяє процеси аналізу контексту сцени та ознак самого об’єкта. Її головна особливість - використання частково-часткової схеми (part-based voting), де кожен фрагмент об’єкта робить внесок у фінальне передбачення навіть у разі часткового перекриття. Такий підхід дозволяє моделі враховувати контекстні зв’язки між видимими частинами об’єкта та фоном, що підвищує стійкість до втрати інформації [2].

Результати експериментів, наведених у публікації, показують, що CompositionalNets досягають покращення точності детекції перекритих об’єктів на 41 % у наборі даних PASCAL3D+ та на 35 % у COCO, порівняно з базовою архітектурою Faster R-CNN [2].

Таким чином, одним із ключових викликів для сучасних систем комп’ютерного зору залишається стійкість до непередбачуваних факторів середовища, зокрема перекриття, шумів і варіацій у масштабі або куті спостереження. Подальший розвиток підходів, подібних до Context-Aware CompositionalNets, спрямований на покращення узагальнюючих властивостей моделей, що є критично важливим для автономних систем, таких як дрони, де умови зйомки часто виходять за межі стандартних сценаріїв [2].

1.2 Аналіз сучасних алгоритмів комп’ютерного зору

1.2.1 Порівняння YOLOv4, YOLOv5 та YOLOv8

Алгоритм YOLO зазнав значних удосконалень протягом багатьох років, що призвело до появи кількох версій з підвищеною ефективністю. Реліз YOLOv4 2020 року включає модуль просторового пірамідального пулінгу, який дозволяє мережі виявляти ознаки в різних масштабах, що покращує її здатність виявляти об'єкти різних розмірів. Крім того, YOLOv4 використовує нову функцію активації, відому як mish, яка, як було продемонстровано, перевершує традиційні функції активації, такі як ReLU. Він використовує нову функцію втрат, яка покращує процес навчання, підкреслюючи складні приклади. [3]

Моделі серії YOLO (You Only Look Once) є одними з найпоширеніших архітектур у сфері комп’ютерного зору для задач виявлення об’єктів у реальному часі. Кожна нова версія покращує швидкодію, точність і адаптивність до різних умов, однак рівень цих покращень залежить від середовища використання та характеристик даних. [3]

У дослідженні Gašparović et al. (2023) [3] проведено експериментальне порівняння моделей YOLOv4, YOLOv5, YOLOv6, YOLOv7 та YOLOv8 у складних умовах - зокрема підводному середовищі, де зображення мають низький контраст, неоднорідне освітлення, шум і часткові перекриття об’єктів. Такий тип середовища можна вважати аналогом нетипових або «польових» умов, у яких часто працюють дрони під час моніторингу чи розвідки.

Результати порівняння показали, що:

YOLOv4 демонструє стабільну роботу з середньою точністю (mAP@0.5 ≈ 94,21 %), але поступається новішим версіям за швидкістю та ефективністю обчислень. [3]

YOLOv5 показала найкращий результат серед усіх протестованих моделей - mAP@0.5 ≈ 97,10 %. Вона відзначається високою стабільністю при зміні умов освітлення та хорошим балансом між точністю й швидкістю роботи [3]. Крім того, YOLOv5 має більш зручні інструменти для навчання та

розгортання, що дозволяє ефективно використовувати її на обмежених апаратних платформах, таких як Orange Pi 5.

YOLOv8, хоча й має новішу архітектуру, не продемонструвала істотного покращення точності (mAP@0.5 ≈ 95,10 %) [3].

Таким чином, хоча YOLOv8 вважається технологічно сучаснішою, вона не завжди перевершує попередні версії у складних умовах зйомки. Для систем, які працюють у реальному часі на борту дрона, YOLOv4 залишається найоптимальнішим варіантом завдяки збалансованому поєднанню точності, швидкодії та стабільності [3].

Дослідження також підкреслює, що зростання номера версії YOLO не гарантує автоматичного покращення результатів [3]. Ефективність сильно залежить від умов зйомки, якості даних і типу апаратної платформи. Для завдань виявлення нетипових об’єктів із дронів, де важлива автономна обробка зображень, YOLOv5 є найдоцільнішим вибором.

1.2.2 Інші методи: Mask R-CNN, EfficientDet

Mask R-CNN, що розшифровується як Mask Region-based Convolutional Neural Network (згорткова нейронна мережа на основі маскових областей), — це модель глибокого навчання, розроблена для завдань комп'ютерного зору, таких як виявлення об'єктів та сегментація екземплярів. Представлена у 2017 році компанією Facebook AI Research (FAIR) [4].

Сегментація екземплярів виходить за рамки традиційного виявлення об'єктів, не лише ідентифікуючи об'єкти на зображенні, але й точно окреслюючи кожен з них. Вона призначає унікальну мітку кожному виявленому об'єкту та фіксує його точну форму на рівні пікселів. Цей детальний підхід дозволяє чітко розрізняти об'єкти, що перекриваються, та точно обробляти складні форми.

Mask R-CNN базується на Faster R-CNN, який виявляє та позначає об'єкти, але не визначає їх точну форму. Mask R-CNN покращує це, ідентифікуючи точні пікселі, що складають кожен об'єкт, що дозволяє проводити набагато детальніший та точніший аналіз зображень [4].

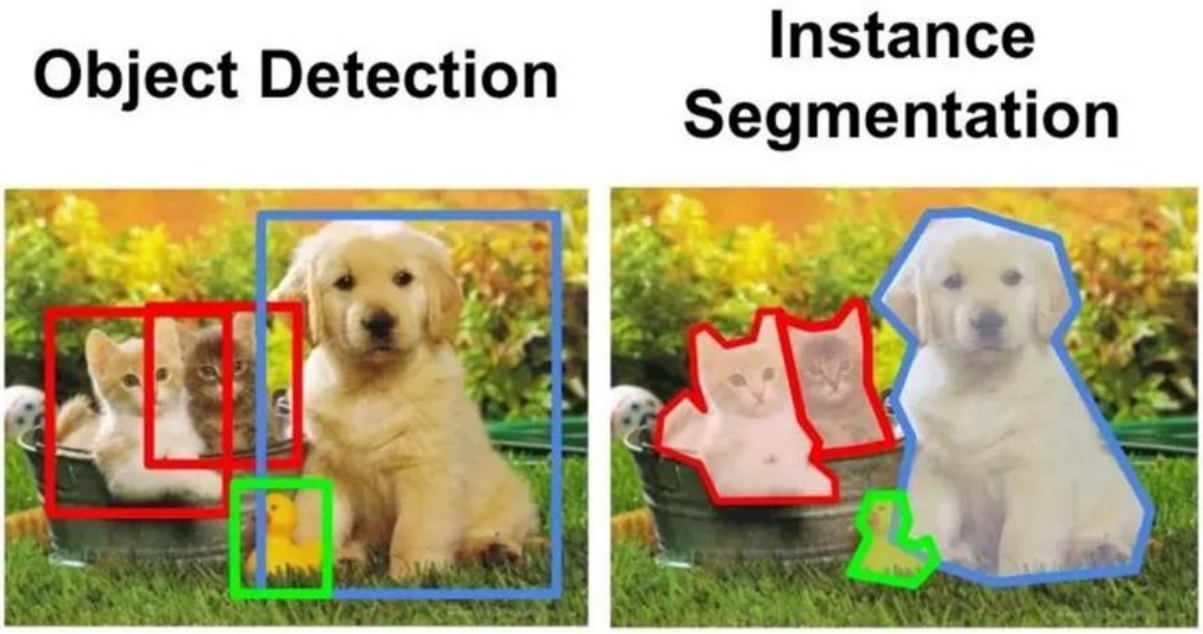


Рис. 1.1 Порівняння виявлення об'єктів та сегментації екземплярів [4]

Озираючись на історію комп'ютерного зору, можна сказати, що Mask R-CNN визнається значним проривом у виявленні та сегментації об'єктів. Він забезпечує дуже точні результати навіть у складних умовах завдяки своєму детальному багатоетапному процесу [4].

Однак, цей самий процес робить його повільнішим порівняно з моделями реального часу, такими як YOLO. Зі зростанням потреби в швидкості та ефективності, багато програм зараз використовують одноетапні моделі, такі як Ultralytics YOLO11, які пропонують швидке та точне виявлення об'єктів. Хоча Mask R-CNN важливий для розуміння еволюції комп'ютерного зору, тенденція до рішень реального часу підкреслює зростаючий попит на швидші та ефективніші рішення комп'ютерного зору [4].

EfficientDet — це сучасна модель виявлення об'єктів у реальному часі, спочатку написана на Tensorflow та Keras, але тепер реалізована на PyTorch. Цей блокнот використовує реалізацію EfficientDet на PyTorch. Він має основу

EfficientNet та власну мережу виявлення та класифікації. Завдяки цій основі EffcientDet розроблений для ефективного масштабування від найменшого розміру моделі. Найменший EfficientDet, EfficientDet-D0, має 4 мільйони вагових параметрів — він справді крихітний. EfficientDet робить висновки за 30 мс у цьому розподілі та враховується, і може зберігатися лише з 17 мегабайтами сховища, що робить його одночасно невеликою та швидкою моделлю [5].

EfficientDet показав найсучасніші результати на COCO на момент свого випуску та працює трохи краще, ніж YOLOv3.

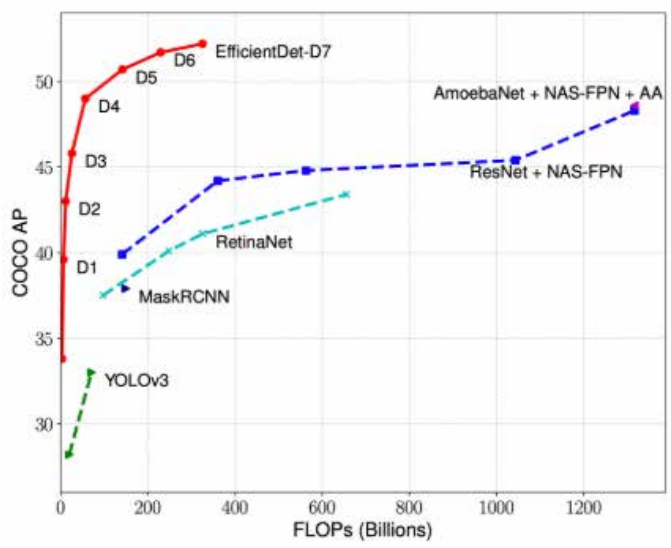


Рис. 1.2 Порівняння швидкості та точності [5]

1.3 Обґрунтування вибору системи Orange Pi 5

1.3.1 Актуальність використання дронів з системою Orange Pi 5

Використання дронів з системою Orange Pi 5 є актуальним рішенням через високу ефективність та технологічні можливості. Дрони дозволяють швидко та безпечно обстежувати великі площі, виявляючи нетипові об'єкти,

такі як уламки зброї або інше сміття, яке залишилося після бойових дій. Система Orange Pi 5, яка є потужною та водночас компактною платформою, забезпечує необхідну обчислювальну потужність для обробки даних у реальному часі, що дозволяє оперативно реагувати на виявлені загрози.

Завдяки високій роздільній здатності камер та можливостям штучного інтелекту, дрони з системою Orange Pi 5 можуть автоматично розпізнавати різні типи предметів, класифікувати їх та визначати їх точне розташування. Це значно зменшує ризики, адже дозволяє уникати фізичного контакту з потенційно небезпечними об'єктами до їх нейтралізації.

Дрони можуть здійснювати моніторинг територій та виявляти ділянки, що потребують негайної уваги, наприклад, через наявність нетипових або небезпечних об’єктів. Це сприяє ефективнішому використанню ресурсів, підвищенню рівня безпеки та оптимізації процесів аналізу даних. Використання дронів із системою Orange Pi 5 дозволяє знизити витрати на ручну працю та технічні засоби, які зазвичай залучаються для виконання аналогічних завдань, що є особливо важливим в умовах обмежених ресурсів.

Застосування дронів із системою Orange Pi 5 є актуальним та ефективним рішенням для виявлення нетипових об’єктів у складних або важкодоступних умовах. Такий підхід забезпечує безпечність, економічну доцільність та високу точність ідентифікації, сприяючи розвитку сучасних технологій моніторингу та комп’ютерного зору.

1.3.2 Аналіз існуючих платформ для дронів, таких як Raspberry Pi 5, Radxa Rock 5C

Raspberry Pi 5 оснащений чотирьохядерним процесором Broadcom BCM2712 Arm Cortex A76 з частотою 2,4 ГГц, що робить його до трьох разів швидше, ніж попереднє покоління (рис. 1.3). З варіантами оперативної пам'яті до 8 ГБ це сама швидка і плавна робота Raspberry Pi [6].

Raspberry Pi 5 побудований з використанням контролера вводу-виводу RP1, пакета, що містить мікросхему, розроблену власними силами Raspberry Pi. USB 3 має найбільшу загальну пропускну здатність, що забезпечує набагато більшу швидкість передачі даних (рис. 1.4.) [6].

Роз'єми камери і дисплея DSI взаємозамінні, тому ви можете використовувати на одному роз'ємі кожен з них або два однакових [6].

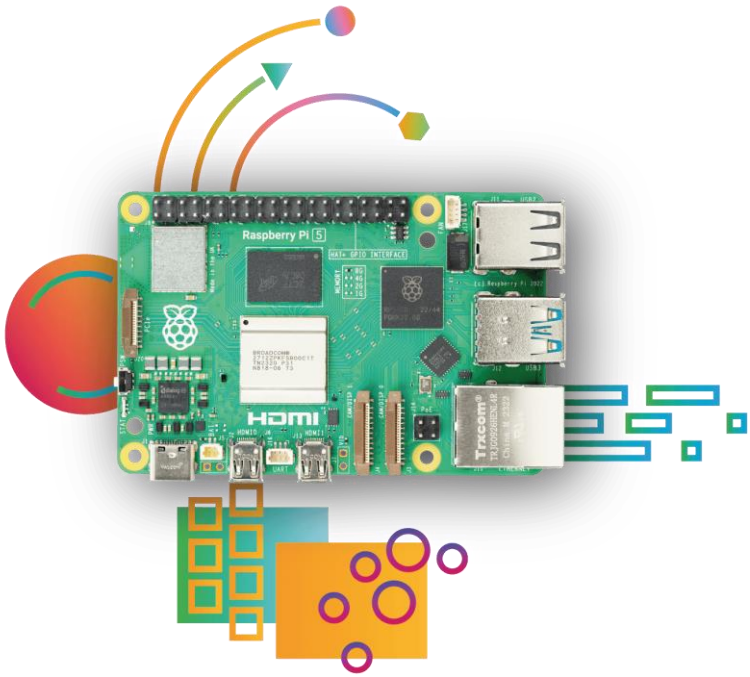


Рис. 1.3 Мінікомп’ютер Raspberry Pi 5 [6]

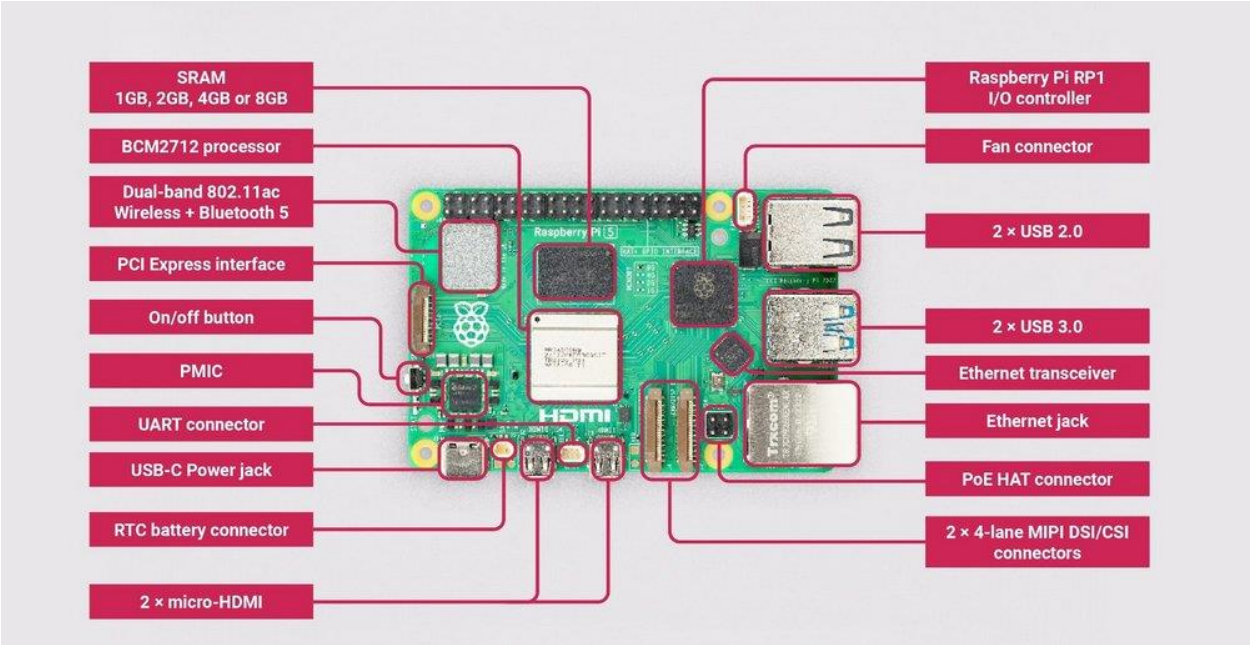
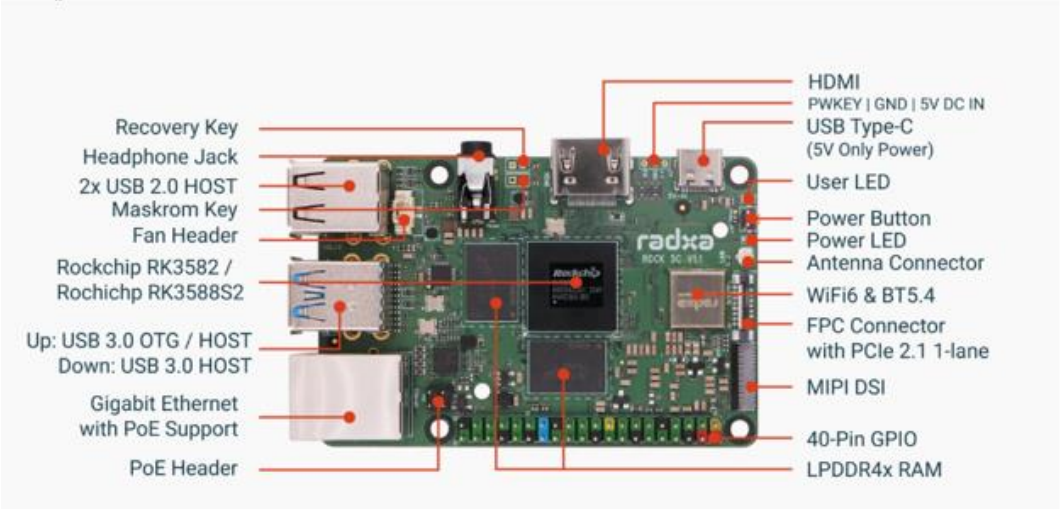


Рис. 1.4 Схема портів та можливих підключень Raspberry Pi 5 [6]

Radxa ROCK 5C — це компактний одноплатний комп'ютер (SBC), який пропонує низку передових функцій, характеристик та можливостей розширення. Це ідеальний вибір для розробників, ентузіастів Інтернету речей, любителів, геймерів, користувачів ПК та всіх, кому потрібна високопродуктивна платформа з відмінною продуктивністю та надійністю. Radxa ROCK 5C випускається у двох версіях: стандартна версія на базі RK3588S2 та Lite версія на базі RK3582. Стандартна та Lite версії мають більшість спільних функцій, єдина відмінність полягає в SoC (системі на кристалі). У наступній документації, якщо не зазначено інше, «ROCK 5C» стосується як стандартної версії ROCK 5C, так і версії 5C Lite [7].

ROCK 5C можна використовувати як :

- Персональний настільний комп'ютер
- Персональний приватний сервер
- Відео- та аудіоплеєр Android
- Контролер робота
- Вузол блокчейну



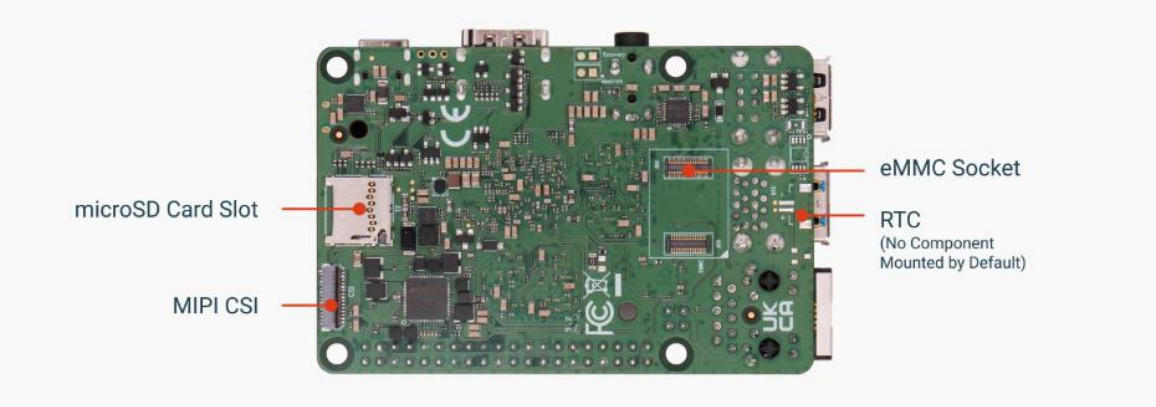


Рис. 1.5 Схема портів та можливих підключень Radxa Rock 5C [8]

1.3.3 Порівняння характеристик та можливостей Orange Pi 5 з іншими платформами (табл. 1.1.)

Таблиця 1.1 – Порівняння характеристик трьох платформ

Характеристика	Orange Pi 5	Raspberry Pi 5	Radxa ROCK 5C
Процесор	Rockchip RK3588, 8 ядер (4x Cortex-A76, 4x Cortex-A55)	ARM Cortex-A76, 4 ядра	Arm Quad Cortex®-A76
Графіка	ARM Mali-G610 MP4	VideoCore VII	Arm Mali™ G610MP4
ОЗП	До 16 ГБ LPDDR4/4x	До 16 ГБ LPDDR4	LPDDR4x до 32 ГБ
Пам'ять	microSD, eMMC до 128 ГБ, USB 3.0	microSD, підтримка USB 3.0	MicroSD для зберігання журналів
Порти	HDMI 2.1, 2x USB 3.0, USB	2x HDMI, 2x USB 3.0, 2x USB 2.0,	2 хост-порти USB 2.0 типу A

	Type-C, Ethernet 1 Гбіт/с, GPIO	Ethernet 1 Гбіт/с, GPIO	1 хост-порт USB 3.0 типу A 1 порт USB 3.0 Type-A OTG / HOST
Підтримка ОС	Android, Ubuntu, Debian	Raspberry Pi OS, Ubuntu, Windows IoT Core	ОС Radxa / Android
Додаткові можливості	Вбудоване Wi-Fi та Bluetooth, підтримка 4K відео, AI прискорювач	Вбудоване Wi-Fi та Bluetooth, підтримка двох дисплеїв, 4K відео	WiFi 6 та BT 5.4 із роз'ємом для зовнішньої антени

Основні відмінності та можливості платформ

Orange Pi 5, Raspberry Pi 5 та Raxda Rock 5C [5,6,7,8]:

- Усі три платформи є універсальними одноплатними комп’ютерами, призначеними для широкого кола завдань, включаючи навчання програмуванню, створення вбудованих систем, домашніх серверів, медіа-центрів, систем комп’ютерного зору та інших DIY-проєктів.

- Raspberry Pi 5 має більш широке співтовариство користувачів та кращу підтримку програмного забезпечення, що робить його привабливим для початківців і тих, хто шукає надійну підтримку.

- Orange Pi 5 пропонує більш потужний процесор і більший обсяг оперативної пам'яті, що робить його кращим для ресурсомістких задач, таких як обробка зображень або відео, штучний інтелект та інші вимогливі програми.

- RADXA Rock 5C є потужною альтернативою, побудованою на базі процесора Rockchip RK3588S, і підтримує до 16 ГБ оперативної пам’яті типу LPDDR4/LPDDR5. Завдяки високій продуктивності графічного процесора та підтримці сучасних інтерфейсів (USB 3.0, HDMI 2.1, PCIe), ця плата добре підходить для застосувань у сфері комп’ютерного зору, обробки потокового відео, робототехніки та автономних систем, зокрема дронів.

1.3.4 Orange Pi 5, як оптимальний вибір для проекту

Orange Pi 5 є оптимальним вибором для проекту зі створення системи ідентифікації нетипових об'єктів, таких як танки, завдяки своїм технічним характеристикам і можливостям. Orange Pi 5 має потужний процесор, який забезпечує високу обчислювальну потужність, необхідну для обробки великої кількості даних у реальному часі. Це дозволяє системі швидко аналізувати зображення, отримані з дронів або камер, і точно розпізнавати нетипові об'єкти на полях.

Платформа Orange Pi 5 підтримує різноманітні модулі та периферійні пристрої, що робить її дуже гнучкою для інтеграції з різними сенсорами і камерами. Це важливо для створення комплексної системи, яка може ефективно працювати в різних умовах і на різних типах місцевості. Система може бути налаштована під специфічні вимоги проекту, що забезпечує високу точність і надійність ідентифікації об'єктів.

Orange Pi 5 є доступним за ціною, що робить його економічно вигідним рішенням для впровадження. З огляду на обмежені ресурси, це дозволяє створити ефективну систему без значних фінансових витрат. Енергетична ефективність Orange Pi 5 також сприяє зниженню експлуатаційних витрат, що є додатковою перевагою.

Завдяки підтримці різних операційних систем і програмного забезпечення, Orange Pi 5 дозволяє використовувати сучасні алгоритми

машинного навчання і штучного інтелекту для покращення точності розпізнавання об'єктів. Це значно підвищує можливості системи виявлення та класифікації танків на полях, забезпечуючи високу швидкість і точність ідентифікації.

Таким чином, Orange Pi 5 є оптимальним вибором для мого проекту завдяки своїй потужності, гнучкості, економічності та підтримці сучасних технологій. Це забезпечує ефективне і надійне рішення для виявлення танків та інших нетипових об'єктів.

1.4 Архітектури нейронних мереж для комп'ютерного зору

1.4.1. Згорткові нейронні мережі

Згорткові нейронні мережі, або Convolutional Neural Networks (далі CNN), є різновидом нейронних мереж для обробки зображень. CNN використовуються в завданнях комп’ютерного зору: генерації і класифікації зображень, розпізнавання об’єктів і поз тощо. Ці завдання раніше намагались вирішити як за допомогою класичних нейронних мереж, тобто багатошарового персептрона Multilayer Perceptron (MLP), так і різними евристичними методами [9].

Згорткові нейронні мережі поділяють за типом виміру на 1D, 2D та 3D, і за типом задач, які вони вирішують, їх можна розділити на такі:

1D CNN. Обробка сигналів: 1D CNN часто використовуються для обробки часових рядів або аудіосигналів, коли потрібно виявити важливі події або зміни в сигналі [9].

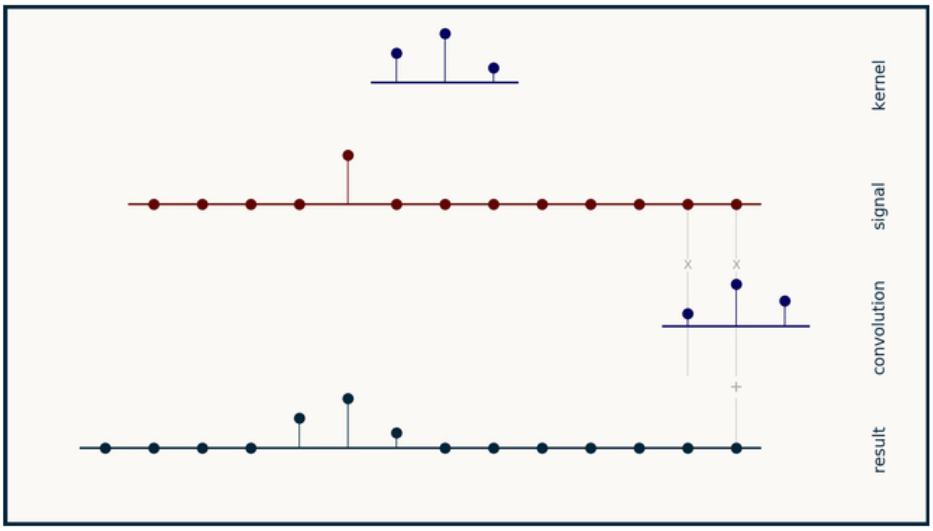


Рис 1.6 Приклад згортки одновимірного сигналу [9]

2D CNN. Класифікація зображень: 2D CNN є стандартним вибором для задач класифікації зображень, вони вміють виявляти ключові особливості на зображенні та використовувати їх для визначення класу зображення [9].

Розпізнавання образів: CNN використовуються для розпізнавання конкретних об’єктів на зображеннях, від облич до автомобілів.

Сегментація зображень: CNN можуть визначати, до якого сегмента належить кожен піксель зображення, що дозволяє виділяти конкретні об’єкти та області.

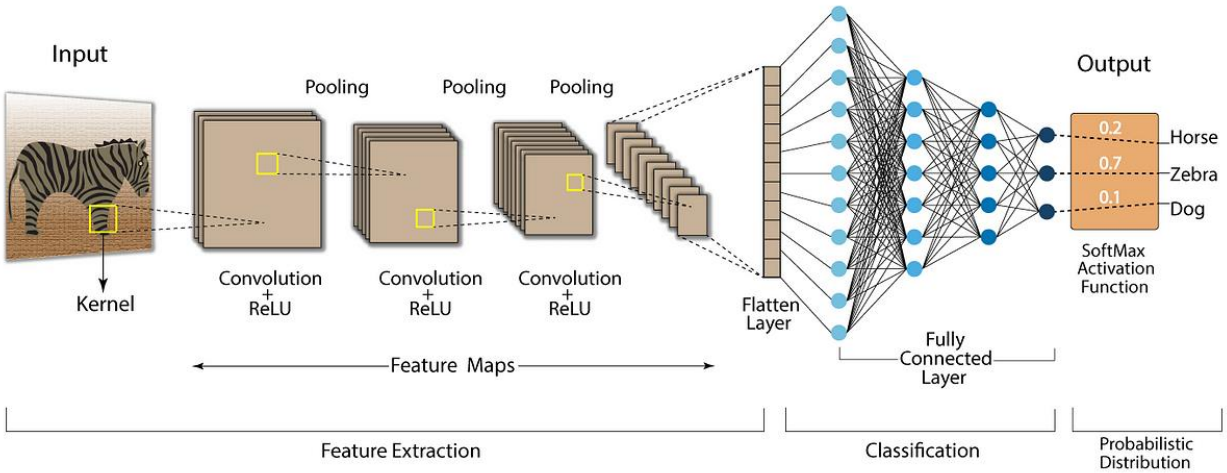


Рис. 1.7 Приклад архітектури згорткової нейронної мережі, для класифікації зображень, яка складається зі згорткових шарів та повнозв’язних шарів класичних штучних нейронів [9].

3D CNN. Класифікація відео: 3D CNN використовуються для аналізу відеоданих, вони можуть враховувати як просторову, так і часову інформацію.

Нейронні мережі на основі CNN відіграють ключову роль у семантичній сегментації. Вони, як і в попередніх завданнях, використовуються для визначення ознак на зображенні, що допомагають в ідентифікації та сегментації об’єктів. Згорткові шари в CNN ефективно виявляють локальні та глобальні ознаки на зображенні, що допомагає у точному визначенні об’єктів та їх меж [9].

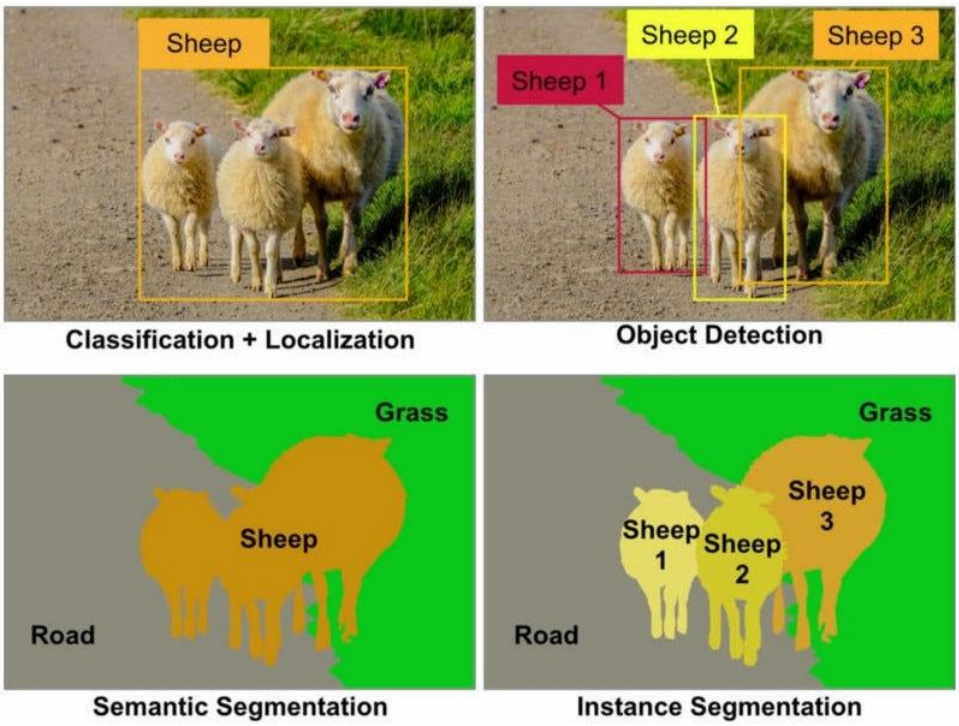


Рис. 1.8 Приклад того, як відрізняються між собою задачі сегментації від задач класифікації та детекції [9].

1.4.2. Трансформери у комп'ютерному зорі (ViT, DETR)

Трансформери - це потужна архітектура глибокого навчання, спочатку розроблена для завдань обробки природної мови (NLP), таких як машинний переклад, реферування тексту та аналіз настроїв. По суті, трансформери використовують структуру кодера-декодера, де кодер обробляє вхідну

послідовність (наприклад, речення) та створює насичене контекстуальне представлення, тоді як декодер генерує вихідну послідовність на основі цього представлення [10].

Ключовою інновацією трансформаторів є механізм самоуваги, який дозволяє моделі зважувати важливість кожного елемента в послідовності відносно інших, фіксуючи як локальні, так і довгострокові залежності, не покладаючись на рекурентні або згорткові операції. Така конструкція дозволяє трансформаторам обробляти цілі послідовності паралельно, що призводить до значного підвищення ефективності та продуктивності порівняно з попередніми архітектурами, такими як RNN та LSTM [10].

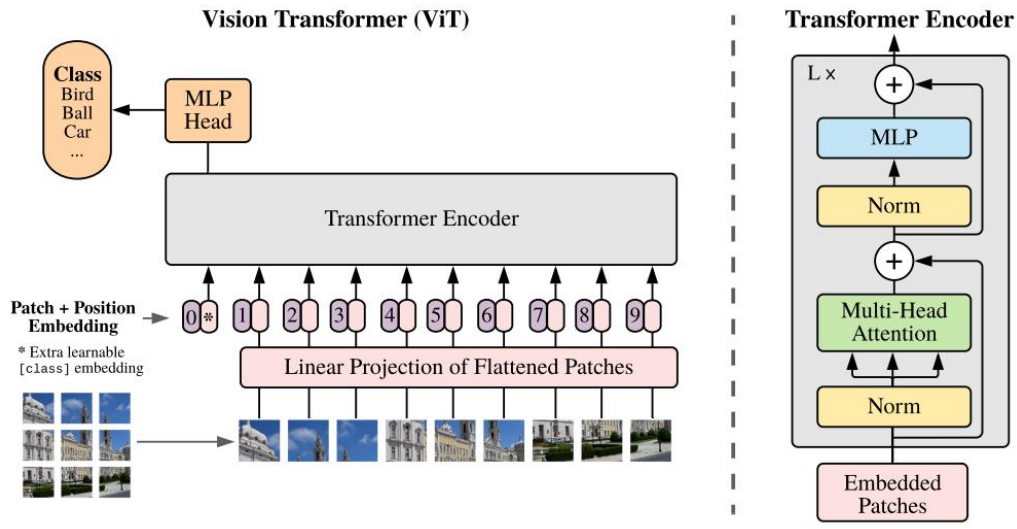


Рис. 1.9 Огляд моделі кодера [10]

ViT-технології привносять потужність трансформаторних архітектур у світ комп'ютерного зору, переосмислюючи процес обробки та розуміння зображень. Замість того, щоб покладатися на згортки для вилучення ознак, ViT-технології обробляють зображення як послідовності ділянок та використовують механізми самоаналізу для фіксації зв'язків по всьому зображенню. Такий підхід дозволяє їм моделювати як локальні, так і глобальні залежності, що призводить до вражаючої продуктивності в різноманітних візуальних завданнях [10].

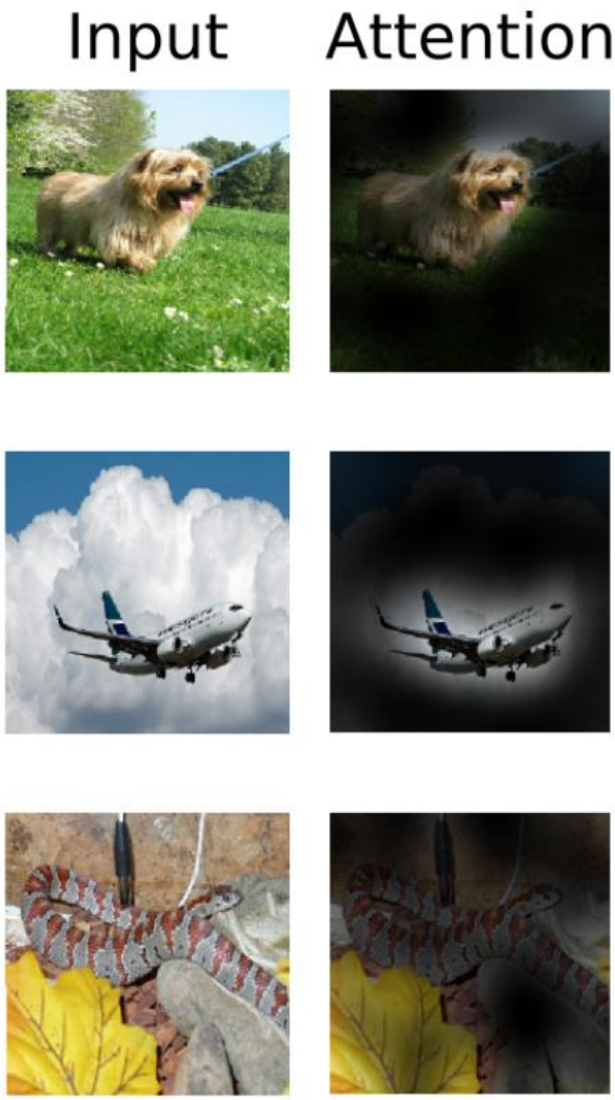


Рис. 1.10 Візуалізація механізму уваги моделей ViT [10]

DETR в основному використовується для завдань виявлення об'єктів, тобто процесу виявлення об'єктів на зображенні. Наприклад, вхідними даними моделі буде зображення дороги, а вихідними даними моделі можуть бути [('car',X1,Y1,W1,H1),('pedestrian',X2,Y2,W2,H2)], де X, Y, W, H представляють координати x, y, що позначають розташування обмежувальної рамки, а також ширину та висоту рамки [11]. Традиційна модель виявлення об'єктів, така як YOLO, складається з ручно створених ознак, таких як апріорні значення обмежувальної рамки, що вимагає початкових припущень щодо розташування та форми об'єктів, що впливає на подальше навчання. Потім використовуються кроки постобробки для видалення обмежувальних рамок, що перекриваються,

що вимагає ретельного вибору евристики фільтрації. DEtection TRAnsformer, скорочено DETR, спрощує детектор, використовуючи трансформатор кодера-декодера після магістралі вилучення ознак для безпосереднього прогнозування обмежувальних рамок паралельно, що вимагає мінімальної постобробки [11].

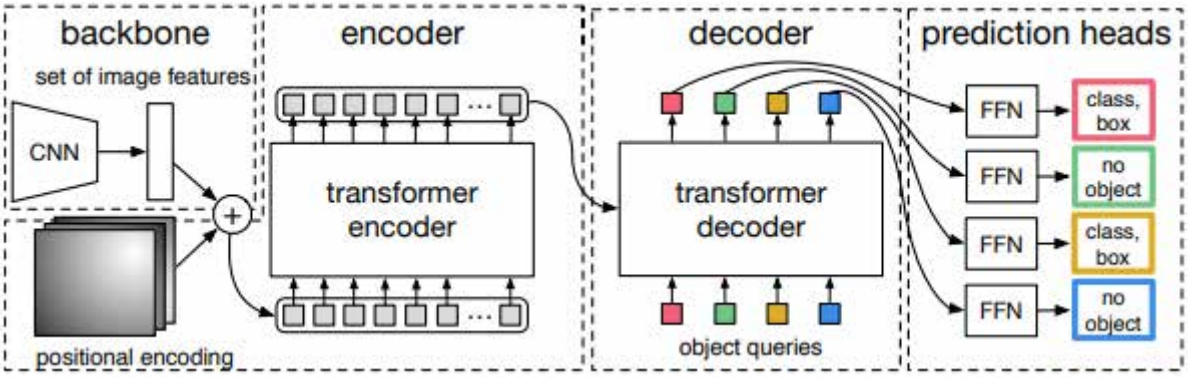


Рис. 1.11 Діаграма архітектури DETR [11]

Архітектура моделі DETR починається з магістралі CNN, подібної до інших мереж на основі зображень, вихідні дані якої обробляються та подаються в трансформатор-кодер, що призводить до N вбудовувань. Вбудовування кодера додаються до вивчених позиційних вбудовувань (які називаються об'єктними запитами) та використовуються в трансформаторному декодері, генеруючи ще N вбудовувань. Як останній крок, кожне з N вбудовувань проходить через окремі шари прямої подачі для прогнозування ширини, висоти, координат обмежувальної рамки, а також класу об'єкта (або наявності об'єкта) [11].

Продуктивність новітніх детекторів об'єктів з точки зору середньої точності (AP) та затримки. Детально. Текст на графіку вказує на розмір моделі магістралі. Затримку вимірювали з розміром пакету 1 з роздільною здатністю 800 × 1333 на графічному процесорі NVIDIA A100. AP та затримка (мілісекунди) 3 [12].

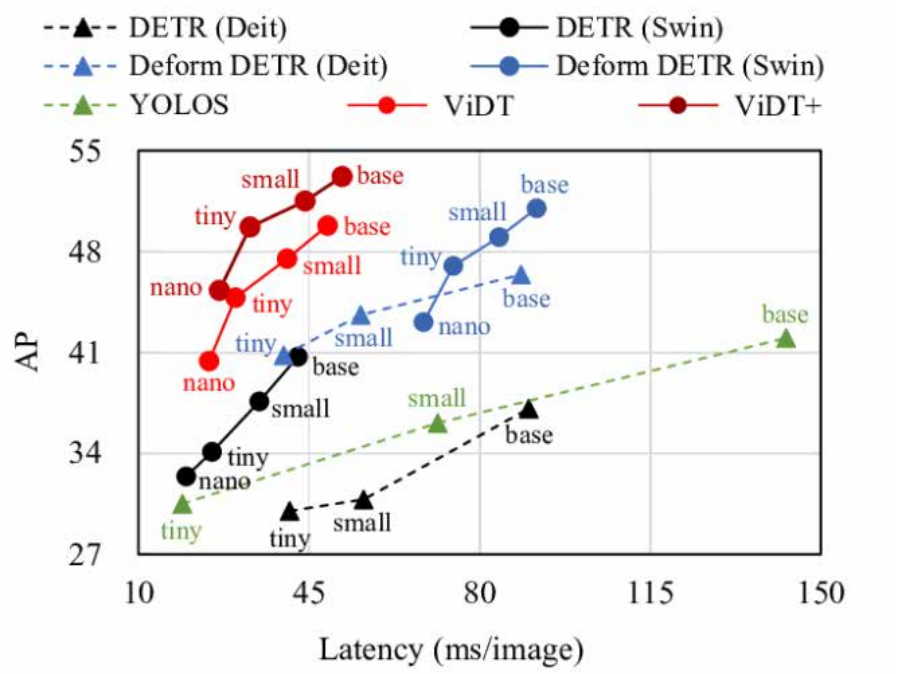


Рис. 1.12 Середня точність (AP) та затримка [12]

1.5 Принцип роботи дронів

1.5.1 Основні типи дронів та їх характеристики

Дрон – Безпілотний літальний апарат (БПЛА), військового чи цивільного призначення, різновид військового робота; в ширшому сенсі - мобільний, автономний апарат, запрограмований на виконання якихось завдань (наприклад, автономні системи, створені для польоту, розроблені для виконання місій, потенційно небезпечних для людини). [13]

Дрони літають за рахунок принципу тяги. Пропелери, що обертаються, створюють тягу, яка штовхає дрон вперед, вгору, вниз або назад. Контролер польоту використовує дані з датчиків, щоб регулювати швидкість обертання пропелерів і таким чином керувати напрямком і швидкістю польоту дрона.

Основні характеристики дронів:

Розмір – дрони бувають різних розмірів, від маленьких і портативних до великих і потужних.

Вага – важливий фактор, оскільки вона впливає на його вантажопідйомність і час польоту.

Час польоту – це час, протягом якого він може літати на одному заряді акумулятора.

Дальність польоту - це максимальна відстань, на яку він може пролетіти.

Вантажопідйомність – це максимальна вага, яку він може нести.

Камера – деякі дрони оснащені камерами, які можуть робити фотографії та відео. Якість камери може варіюватися від типу.

Застосування:

- контроль переміщення людей, техніки, ідентифікація тварин (тегованих);

- логістика: переміщення вантажів, постачання (у тому числі медичне) до важкодоступних місцевостей;

- сільське господарство: обробка культур пестицидами, висівання;

- дика природа: відстеження браконьєрів;

- розвідка: оптична, лазерна, телевізійна, радіаційна.

Типи дронів:

Коптери - дрон з мультироторним двигуном, що складається з трьох або більше гвинтів (пропелерів) (рис. 1.13). Їх класифікують за кількістю гвинтів: квадрокоптери, гексакоптери , вісьмокоптери.

Рис. 1.13 Гексакоптер

Дрони з фіксованим крилом - мають крила, як літак, і пропелер, який використовується для зльоту та посадки (рис. 1.14). Вони можуть літати набагато далі та швидше, ніж дрони з мультиротором, але вони менш маневрені.

Рис. 1.14 Крило (БПЛА)

FPV-дрони ("First Person View") - оснащені камерою, яка передає відео в режимі реального часу на окуляри або дисплей пілота (рис. 1.15). Це дозволяє пілоту літати дроном так, ніби він сидить у ньому.

Рис. 1.15 FPV-дрон

1.5.2 Управління дроном та його стабілізація

Дрони управляються і стабілізуються в повітрі завдяки комбінації електронних компонентів, програмного забезпечення і фізичних принципів. Основою управління є контролер, який виконує роль мозку дрона, незалежно від типу та моделі. Він збирає дані від різних датчиків, таких як гіроскопи, акселерометри, GPS, магнітометри та барометри, щоб визначити поточне положення та орієнтацію дрона в просторі.

Якщо ми говоримо про FPV-дрони та коптери, то в них гіроскопи вимірюють кутові швидкості, тобто як швидко дрон обертається навколо своїх осей, а акселерометри визначають прискорення, яке дрон відчуває у всіх трьох напрямках. GPS дає інформацію про координати дрона на місцевості, магнітометри допомагають визначити напрямок відносно магнітних полюсів Землі, а барометри вимірюють висоту над рівнем моря, використовуючи атмосферний тиск. Контролер використовує алгоритми, такі як PID-регулятори (пропорційно-інтегрально-диференціальні регулятори), щоб обробляти ці дані та відправляти команди на мотори дрона. Кожен мотор з'єднаний з пропелером і може змінювати свою швидкість обертання.

Змінюючи швидкість окремих моторів, контролер може керувати нахилом, креном, обертанням і висотою дрона.

Для стабілізації дрона в повітрі важливо підтримувати баланс сил, які діють на нього. Коли всі пропелери обертаються з однаковою швидкістю, створюється рівномірна підйомна сила, яка протидіє силі тяжіння і утримує дрон на місці. Якщо потрібно, наприклад, нахилити дрон вперед, контролер збільшує швидкість обертання задніх пропелерів і зменшує швидкість передніх, створюючи необхідний нахил. Подібним чином дрон може повертатися навколо своєї вертикальної осі шляхом зміни швидкості протилежних пар пропелерів.

Програмне забезпечення також відіграє важливу роль в управлінні дроном. Спеціальні програми дозволяють користувачам задавати маршрути польоту, висоту, швидкість та інші параметри, а також можуть автоматично виконувати завдання, такі як зліт, посадка, обхід перешкод і повернення додому. Завдяки поєднанню точних датчиків, швидких алгоритмів управління і потужних моторів, дрони здатні літати стабільно і точно виконувати команди навіть у складних умовах.

Для дронів з фіксованим крилом принцип роботи всіх компонентів такий-же, за винятком двигунів, оскільки в ньому двигун за чисту один. Керування здійснюється за допомогою сервоприводів, які задають висоту та кути повороту дрона. Контролер потрібно виставити в горизонт, після можна здійснювати керування вручну з пульта або налаштувати автопілот, який буде регулювати положення всіх компонентів автоматично.

1.6 Алгоритми та методи обробки зображень

1.6.1 Маркування цілей в Yololabel

YoloLabel – це інструмент, спеціально розроблений для оптимізації процесу маркування. Замість використання так званого методу «перетягування», який реалізований багатьма інструментами маркування, YoloLabel надає перевагу підходу «подвійного клацання», коли ви клацаєте, щоб розпочати створення обмежувальної рамки, і клацаєте ще раз, щоб зупинити це [14].

Сімейство детекторів об'єктів YOLO, де YOLO розшифровується як «Ви тільки шукаєте один раз» (You Only Look Once) – один із найвідоміших та найпоширеніших типів детекторів об'єктів. Вже у п'ятій версії на початку 2021 року YOLO можна відносно легко навчити та оптимізувати для швидкості – отже, перегляд один раз [14].

Навчання власного детектора об'єктів YOLO вимагає надання позначеного набору даних. Використовуючи інструмент під назвою YoloLabel, який працює на Windows та macOS, можна створювати обмежувальні рамки для власної моделі виявлення об'єктів YOLO [14].

1.6.2 YOLOv4: Оптимальна швидкість і точність виявлення об'єктів

Більшість детекторів об'єктів на основі CNN в основному застосовуються лише для рекомендаційних систем. Наприклад, пошук вільних паркувальних місць за допомогою міських відеокамер виконується повільними точними моделями, тоді як попередження про зіткнення автомобілів відноситься до швидких неточних моделей. Підвищення точності детектора об’єктів у реальному часі дозволяє використовувати їх не лише для систем рекомендацій, що генерують підказки, але й для автономного керування процесами та зменшення людського втручання [15]. Робота детектора об’єктів у режимі реального часу на звичайних графічних процесорах (GPU) дозволяє їх масово використовувати за доступною ціною. Найточніші сучасні нейронні мережі не працюють у режимі реального часу й

потребують великої кількості графічних процесорів для навчання з великим міні-пакетним розміром. Ми вирішуємо такі проблеми, створюючи CNN, який працює в режимі реального часу на звичайному графічному процесорі, для навчання якого потрібен лише один звичайний графічний процесор [15].

Порівняння різних нейронних мереж (рис. 1.16) :

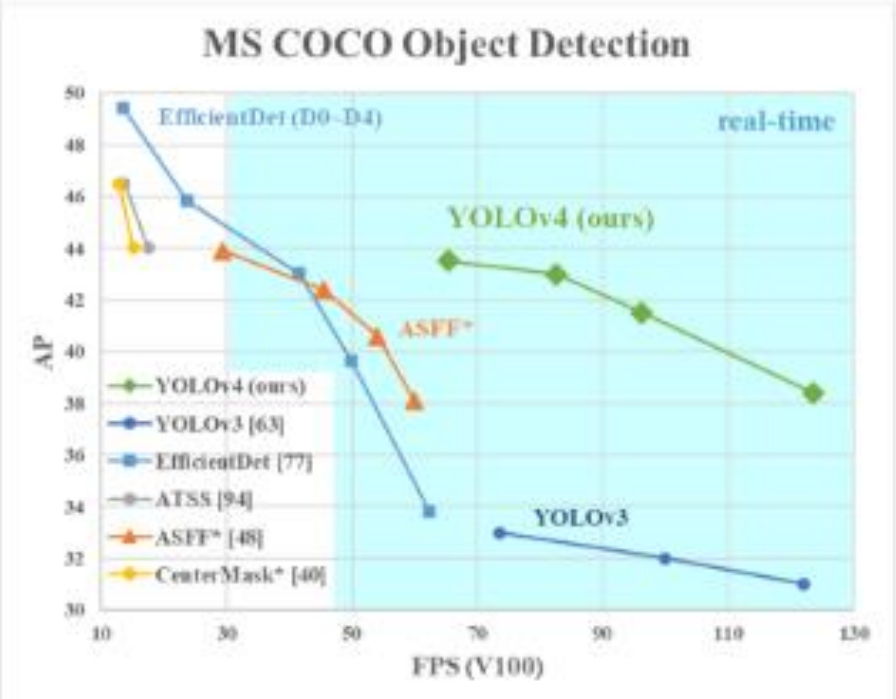


Рис. 1.16 Порівняння запропонованого YOLOv4 та інших найсучасніших детекторів об’єктів [15]

YOLOv4 працює вдвічі швидше, ніж EfficientDet, із порівнянною продуктивністю (рис. 1.17). Покращує AP і FPS YOLOv3 на 10% і 12% відповідно.

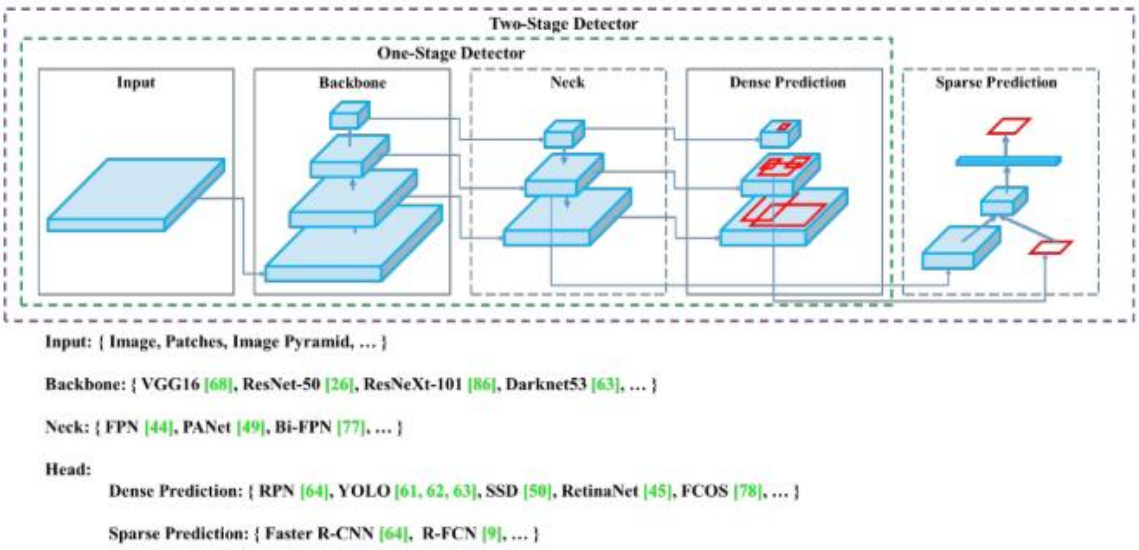


Рис. 1.17 Детектор предметів [16]

Нейронна мережа YOLOv4 краще за швидкістю FPS і точності AP50:95 і AP50 на датасеті Microsoft COCO, ніж DL-фреймворки та нейронні мережі: Google TensorFlow EfficientDet, FaceBook Detec /MaskRCNN, PyTorch Yolov3-ASFF, і багато інших… YOLOv4 досягає точності 43.5% AP / 65.7% AP50 на тесті Microsoft COCO при швидкості 62 FPS TitanV або 34 FPS RTX 2070. На відміну від інших сучасних детекторів, YOLO кого є ігрова відеокарта nVidia з 8-16 GB VRAM Тепер не тільки великі компанії можуть навчати нейронну мережу на сотнях GPU/TPU для використання великих розмірів mini-batch для досягнення вищої точності, тому ми повертаємо контроль над штучним інтелектом звичайним користувачам, тому що для YOLOv4 велика міні-партія не потрібна, можна обмежитися розміром 2 – 8 [15].

YOLOV4- оптимальна від використання real-time, т.к. мережа лежить на кривій оптимальності за Парето на графіку AP(accuracy)/FPS(speed) (рис. 1.18) [15].

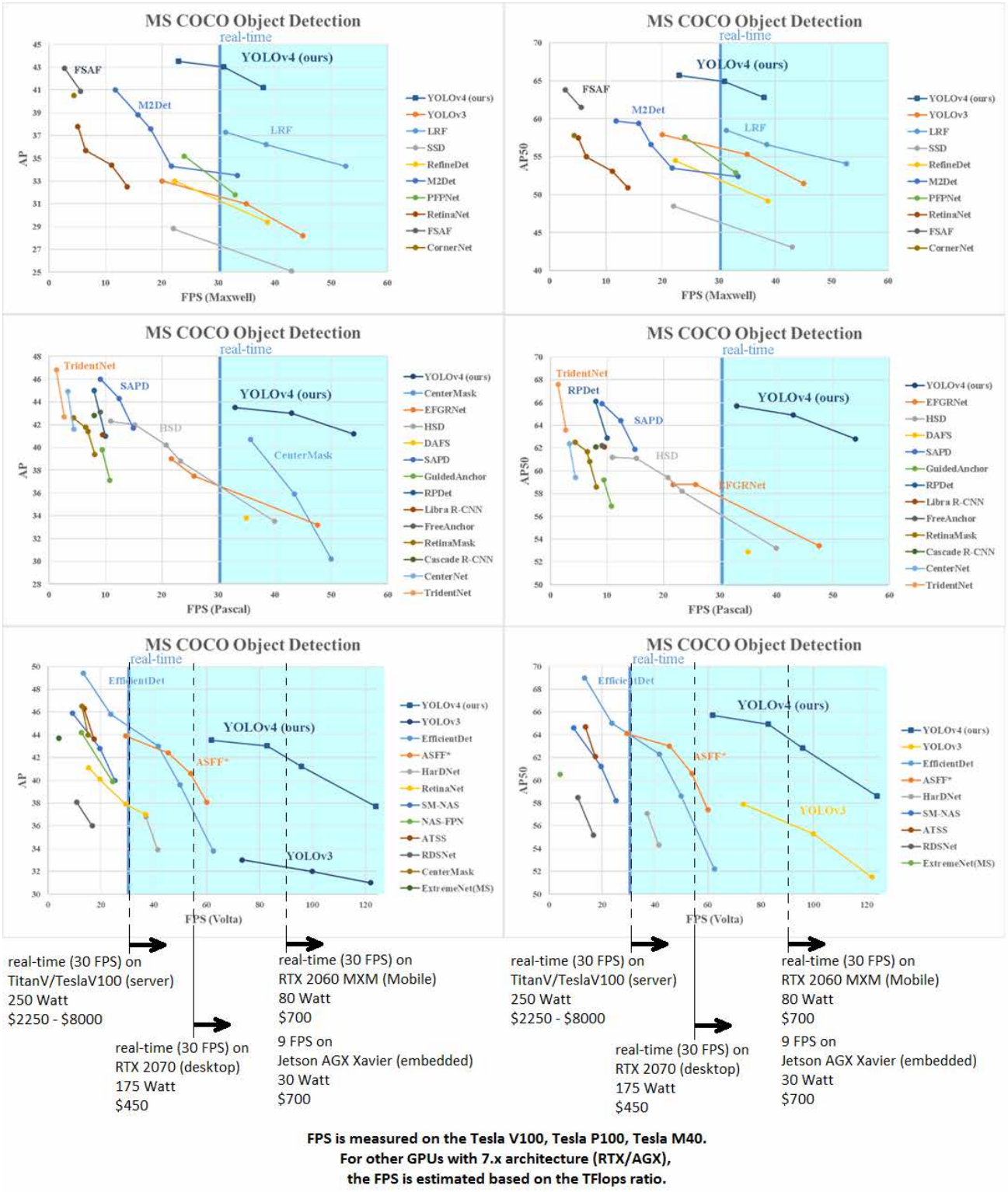


Рис. 1.18 Порівняння швидкості та точності різних детекторів об'єктів [15]

Графіки точності (AP) та швидкості (FPS) безлічі нейронних мереж для виявлення об'єктів заміряних на відеокартах GPU TitanV/TeslaV100,

TitanXP/TeslaP100, TitanX/TeslaM40 для двох основних індикаторів точності AP50:95 та AP50 [15].

Більшість практичних завдань мають мінімально необхідні вимоги до детекторів - це мінімально допустимі точність і швидкість. Зазвичай мінімально допустима швидкість від 30 FPS (кадрів за секунду) і вище для реальних систем.

Як видно з графіків, у Real-time системах з FPS 30 і більше:

для YOLOv4-608 необхідно використовувати RTX 2070 за 450$ (34 FPS) з точністю 43.5% AP / 65.7% AP50

для EfficientDet-D2 необхідно використовувати TitanV за 2250 $ (42 FPS) з точністю 43.0% AP / 62.3% AP50

для EfficientDet-D0 необхідно використовувати RTX 2070 за 450$ (34 FPS) з точністю 33.8% AP / 52.2% AP50

Тобто YOLOv4 вимагає в 5 разів дешевшого обладнання і при цьому точніше, ніж EfficientDet-D2 (Google-TensorFlow). Можна використовувати EfficientDet-D0 (Google-TensorFlow), тоді вартість обладнання буде однакова, але точність буде на 10% AP нижче.

Крім того, деякі промислові системи мають обмеження щодо тепловиділення або використання пасивної системи охолодження – у цьому випадку ви не зможете використовувати TitanV навіть маючи гроші.

При використанні YOLOv4 (416x416) на GPU RTX 2080 Ti з використанням TensorRT+tkDNN ми досягаємо швидкість у 2x рази більше, а при використанні batch=4 у 3x-4x рази більше (рис. 1.19) .

Size	Darknet FPS (avg)	tkDNN TensorRT FP32 FPS	tkDNN TensorRT FP16 FPS	OpenCV FP16 FPS	tkDNN TensorRT FP16 batch=4 FPS	OpenCV FP16 batch=4 FPS	tkDNN Speedup
320	100	116	**202**	171	**423**	384	**4.2x**
416	82	103	**162**	146	**284**	260	**3.5x**
512	69	91	**134**	125	**206**	190	**2.9x**
608	53	62	**103**	100	**150**	133	**2.8x**

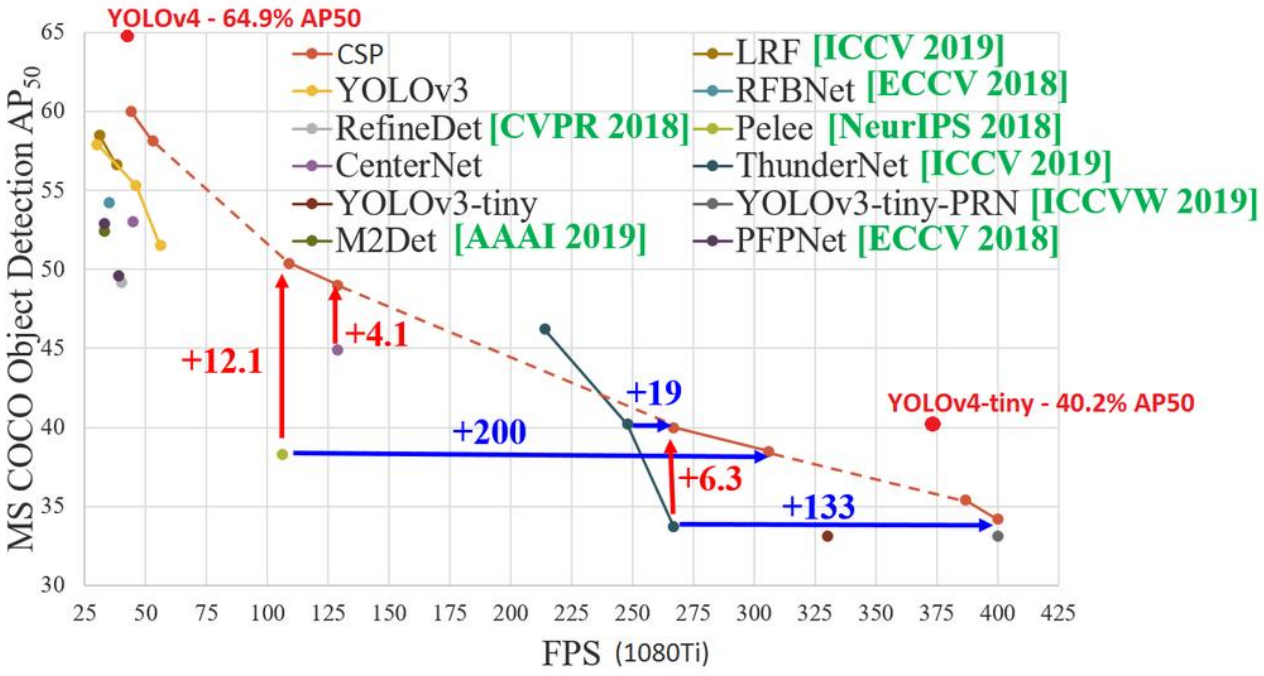
nVidia GPU GeForce RTX 2080 Ti

Рис. 1.19 Порівняння при різних бібліотеках [15]

Нейронна мережа YOLOv4 (416x416) FP16 (Tensor Cores) batch=1 досягає 32 FPS на обчислювачі nVidia Jetson AGX Xavier під час використання бібліотеки TensorRT+tkDNN [16].

Трохи меншу швидкість дає бібліотека OpenCV-dnn, скомпільована з CUDA [19].

Також вийшла модель YOLOv4-tiny з екстремально високою швидкістю 370 FPS на GPU 1080ti, або 16 FPS на Jetson Nano (max_N, 416x416, batch=1, Darknet-framework). Зі швидкістю до 1000 FPS використовуючи OpenCV/tkDNN-TensorRT (рис. 1.20) [17].



- **1770 FPS** - on GPU RTX 2080Ti - (416x416, fp16, batch=4) tkDNN/TensorRT

- **1353 FPS** - on GPU RTX 2080Ti - (416x416, fp16, batch=4) OpenCV 4.4.0

- **39 FPS** - 25ms latency - on Jetson Nano - (416x416, fp16, batch=1) tkDNN/TensorRT

- **290 FPS** - 3.5ms latency - on Jetson AGX - (416x416, fp16, batch=1) tkDNN/TensorRT

- **20 FPS** on CPU ARM Kirin 990 Smartphone Huawei P40 (416x416, GPU-disabled, batch=1) Tencent/NCNN

- **42 FPS** - on CPU i7 7700HQ Laptop - (416x416, fp16, batch=1) OpenCV-dnn (OpenVINO backend)

Рис. 1.20 Виявлення об'єкта YOLOv4-tiny [17]

Особливості розробки YOLOv4

Під час розробки YOLOv4 доводилося розробляти і нейронну мережу YOLOv4 та фреймворк Darknet на C/C++/CUDA. Т.к. в Darknet немає автоматичного диференціювання та автоматичного виконання chain-rule, всі градієнти доводиться реалізовувати вручну на CUDA C++. З іншого боку, ми можемо відходити від суворого проходження chain-rule, змінювати backpropagation і пробувати дуже нетривіальні речі для підвищення стабільності навчання та точності [17].

Додаткові висновки під час створення нейронних мереж

- Не завжди найкраща мережа для класифікації об'єктів буде кращою як backbone для мережі, яка використовується для виявлення об'єктів

-Використання weights навчених із фічами, які збільшили точність у класифікації, може негативно позначитися на точності детектора (на деяких мережах)

- Не всі фічі, заявлені в різних дослідженнях, покращують точність мережі.

- Не всі фічі сумісні між собою та деякі комбінації фіч часто зменшують точність мережі при використанні спільно.

- Не завжди мережа з нижчим BFLOPS буде швидше, навіть якщо BFLOPS менше у десятки разів

- Мережі для виявлення об'єктів вимагають більш високої роздільної здатності мережі для виявлення безлічі об'єктів різного розміру та їх точного розташування, це вимагає більш високого receptive field для покриття збільшеної роздільної здатності мережі, а значить потрібно більше шарів зі stride=2 та/або conv3x3, і більший розмір weights (filters) для запам'ятовування більшої кількості деталей об'єктів.

Використання та навчання YOLOv4

Виявлення об'єктів за допомогою навчених моделей YOLOv4 вбудовано в бібліотеку OpenCV-dnn [18] так що ви можете використовувати YOLOv4 безпосередньо з OpenCV без використання фреймворку Darknet. Бібліотека OpenCV підтримує виконання нейронних мереж CPU, GPU (nVidia GPU), VPU (Intel Myriad X) [19].

2. РОЗРОБКА СИСТЕМИ ІДЕНТИФІКАЦІЇ ОБ'ЄКТІВ

2.1 Апаратна частина системи

2.1.1 Апаратна частина платформи та компоненти

В Orange Pi 5 використовується процесор Rockchip RK3588S ARM нового покоління, який складається з чотирьох ядер A76 і чотирьох ядер A55. Він оснащений 8-нанометровою технологією LP від Samsung, основною частотою великого ядра до 2,4 ГГц і вбудованим графічним процесором ARM Mali-G610 MP4 для високопродуктивного прискорення 3D і 2D зображень. Крім того, він оснащений прискорювачем AI NPU, який може обробляти до 6 найвищих обчислювальних потужностей. Пристрій також має 4 ГБ/8 ГБ/16 ГБ/32 ГБ (LPDDR4/4x) пам'яті та підтримує можливості обробки відображення до 8K [20].

Orange Pi 5 пропонує широкий спектр інтерфейсів, таких як вихід HDMI, Type-C, M.2 pcie 2.0, порт Gigabit Ethernet, інтерфейс USB 2.0, USB 3.0 і 26-контактний роз'єм розширення. його можна широко використовувати в планшетах високого класу, периферійних обчисленнях, штучному інтелекті, хмарних обчисленнях, ar/vr, інтелектуальній безпеці, розумному домі та інших галузях, що охоплюють різні галузі aiot [20].

Orange Pi 5 сумісний з кількома операційними системами, включаючи офіційну Orange Pi OS. крім того, він підтримує Android 12.1, Debian 11, Ubuntu 20.04, Ubuntu 22.04 та інші системи [20].

Використання Orange Pi 5

- Настільний комп'ютер Linux
- Мережевий сервер Linux
- Планшет Android

Загальний вид та можливі підключення Orange Pi 5 задані на рис. 2.1-2.4 [20].

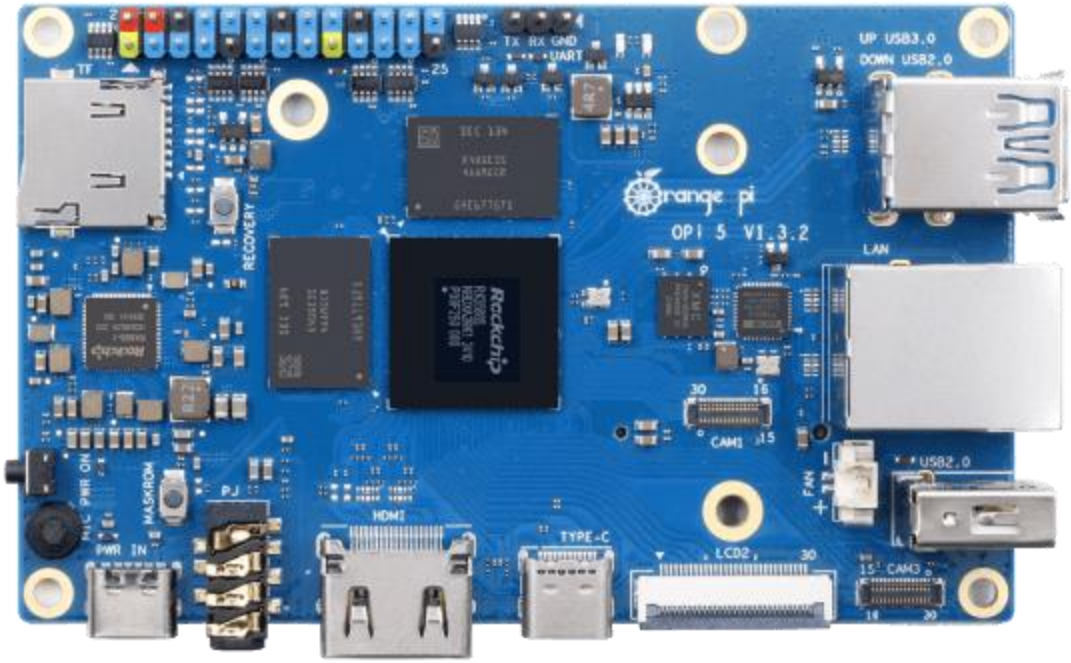


Рис. 2.1 Orange Pi 5 (вид зверху)

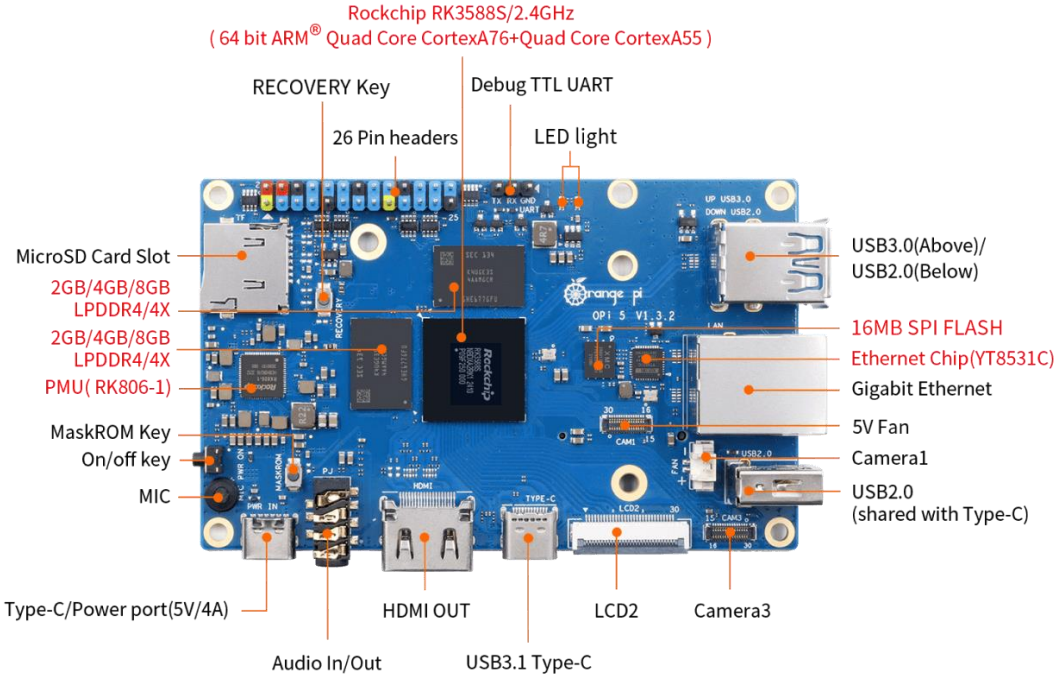


Рис. 2.2 Orange Pi 5 підключення зверху

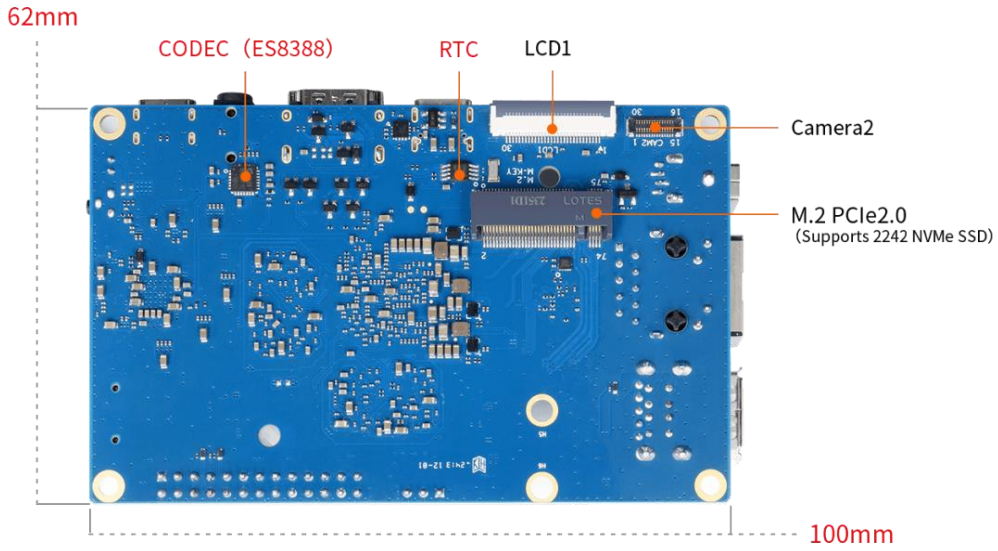


Рис. 2.3 Orange Pi 5 підключення знизу

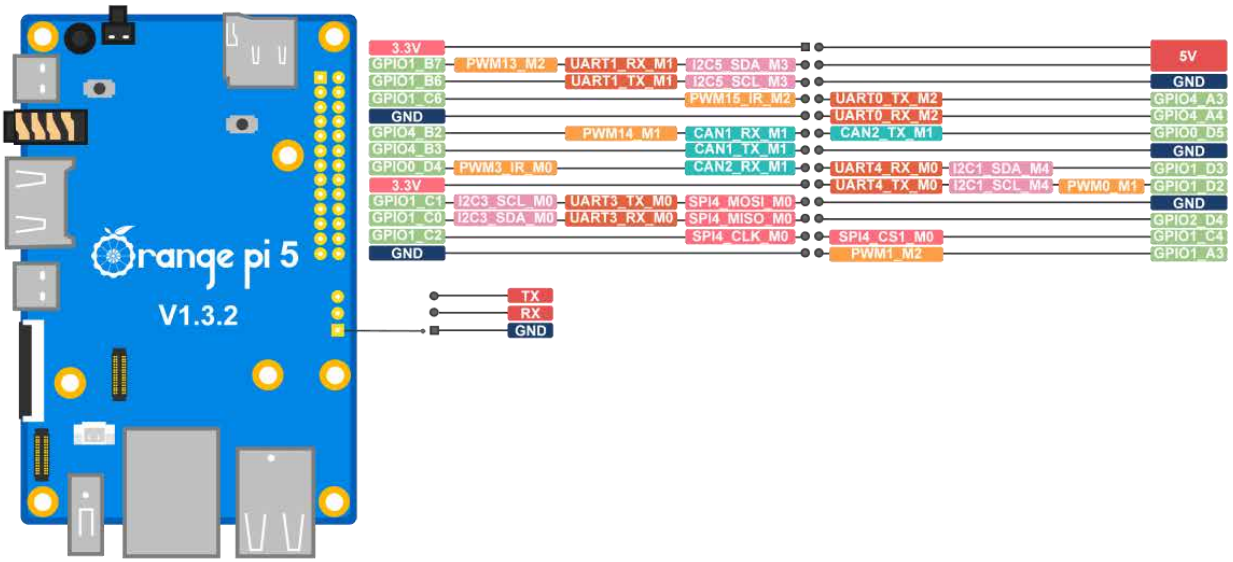


Рис. 2.4 Схема виходу живлення з контактних роз'ємів плати

Arducam 12MP USB Camera Module (рис. 2.5). Висока роздільна здатність 4K HD: на основі 12-мегапіксельної CMOS-матриці з об'єктивом

M12; Максимальна роздільна здатність: 3840(Г)×3032(В). 4k Ultra HD дозволяє зафіксувати найбільш захоплюючі деталі зі швидкістю 4k при 30 кадрах на секунду. Формат стиснення: MJPG/YUY2 [21].

Ширококутний об'єктив M12: оснащений ширококутним об'єктивом M12 із кутом огляду 110° (HFOV). Ширококутні об'єктиви дозволяють охопити більше переднього плану, а більша глибина різкості покращує загальну композицію та цілісність об'єкта у кадрі. Крім того, кріплення об'єктива M12 дозволяє замінювати інші об'єктиви, сумісні з кріпленням об'єктива M12 (примітка: оптичний розмір об'єктива повинен відповідати оптичному розміру сенсора) [21].

Сумісність з UVC: підтримується OTG, просто підключіть USB-кабель до USB-порту вашого ПК, і запуск програмного забезпечення дозволить захоплювати відео високого рівня та вести безперервний запис без потреби у драйверах [21].

Широка сумісність: USB-камери широко доступні для Raspberry Pi, Windows, Android, Linux та Mac OS із протоколом UVC. [21]

Більше додатків: Ідеально підходить для вбудованих програм, відеосистем, 3D-сканерів, VR-камер, електронних мікроскопів та медичного обладнання [21].

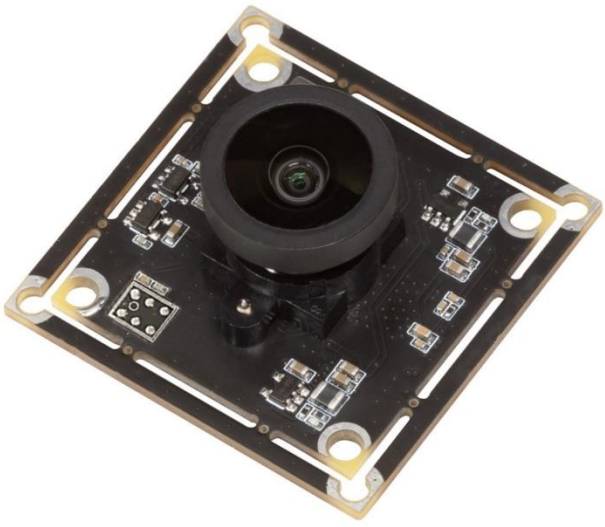


Рисунок 2.5 Arducam 12MP USB Camera Module [21]

Адаптер AWUS036ACH - це флагман компанії Alfa Network. Він є потужним та надійним пристроєм для прийому/передачі Wi-Fi сигналу в діапазоні 2.4 та 5 ГГц. Адаптер працює за стандартами 802.11 a/b/g/n/ac та має дві дводіапазонні антени з посиленням 5dBi. Антени при бажанні можна замінити на потужніші всеспрямовані або панельні. Підключається до USB-порту за допомогою кабелю USB 3.0 довжиною 1 м [22].

Пристрій розроблений на основі чіпсету Realtek RTL8812AU, який відрізняється надвисокою чутливістю та працює навіть зі слабким сигналом. Висока чутливість прийому в поєднанні з внутрішнім підсилювачем потужності також дозволяє йому отримувати сигнали Wi-Fi з відстані, яка до 3 разів перевищує відстань сигналу від внутрішніх карт Wi-Fi 802.11ac, які постачаються в сучасних ноутбуках [22].

Бездротовий адаптер досягає наступної швидкості передачі даних:

на частоті 2,4 ГГц - до 300 Мбіт/с

на частоті 5 ГГц - до 867 Мбіт/с у стандарті AC!

Така швидкість дозволяє дивитися онлайн відео у високій якості та грати в онлайн ігри без підключення додаткових проводів.

Адаптер сумісний з усіма відомими операційними системами, до яких належить і Kali Linux.

Рис. 2.6 Адаптер AWUS036ACH [22]

2.1.2 Конструкція дрона

Основні компоненти апаратної частини дрону:

- Політний контролер для дронів Matek F405 miniTE - компактний і потужний контролер для безпілотних літальних апаратів (рис. 3.6). Він оснащений процесором STM32F405, який забезпечує високу продуктивність, а також гіроскопом та акселерометром MPU6000, відомим своєю надійністю та точністю. Контролер підтримує різноманітні порти, включаючи UART, I2C та SPI, що дозволяє підключати додаткові модулі. Він сумісний з популярними прошивками для дронів, такими як Betaflight. Завдяки своїм компактним розмірам, контролер ідеально підходить для невеликих дронів. Вбудований BEC 5V/2A забезпечує стабільне живлення компонентів. Крім того, він підтримує функції OSD, вібраційний демпфер та інші розширення, що покращують польотні характеристики.

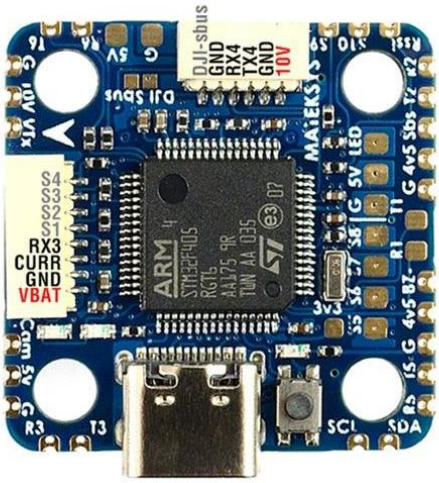


Рис. 2.7 Політний контролер Matek F405 miniTE

- Регулятор ходу T-Motors V45A LITE 6S 4IN1 (рис. 3.7)

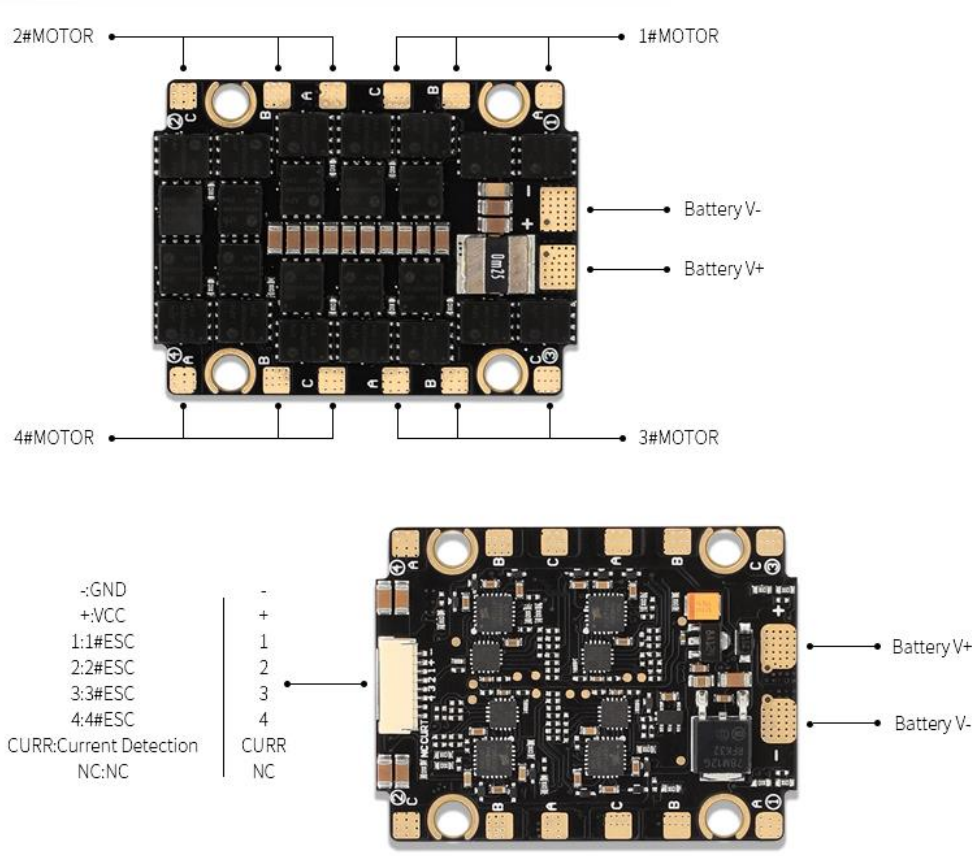


Рис. 2.8 Діаграма підключень

- Приймач GPS Matek M10Q-3100 з компасом

Приймач GPS Matek M10Q-3100 з компасом заснований на прошивці AP_Periph від ArudPilots (рис. 3.8). Цей периферійний пристрій GNSS + COMPASS з інтерфейсами CAN/DroneCAN та UART/MSP.

M10Q-3100 використовує багатозірковий GNSS на базі серії u-blox M10. Це одночасний приймач GNSS, який може приймати та відстежувати декілька систем GNSS. Завдяки багатодіапазонній архітектурі RF-інтерфейсу всі чотири основні угруповання GNSS, GPS, Galileo, GLONASS та BeiDou можуть прийматися одночасно. Патч-антена з високим коефіцієнтом посилення 15 x 15 мм2 забезпечує найкращий баланс між продуктивністю та невеликими розмірами. Всеспрямована діаграма спрямованості антени підвищує гнучкість установки пристрою.

M10Q-3100 оснащений компасом промислового класу PNI RM3100, який забезпечує високу роздільну здатність, низьке енергоспоживання, відсутність гістерезису, великий динамічний діапазон та високу частоту дискретизації.

Рис. 2.9 Приймач GPS Matek M10Q-3100

Іншу компоненти можуть бути змінені в залежності потреб, тому в їх описі немає необхідності

- Камера для FPV дрону RunCam Phoenix 2 SE
- Приймач для FPV дронів (Wifi module)
- Мотори Flymod Gravity E2306.5 V2 1799KV
- Акумулятор Fullymax 22.2V 5000mAh Li-Po 6S 70C XT60

На рисунках 2.10-2.12 зображено мій дрон з реалізованою системою.

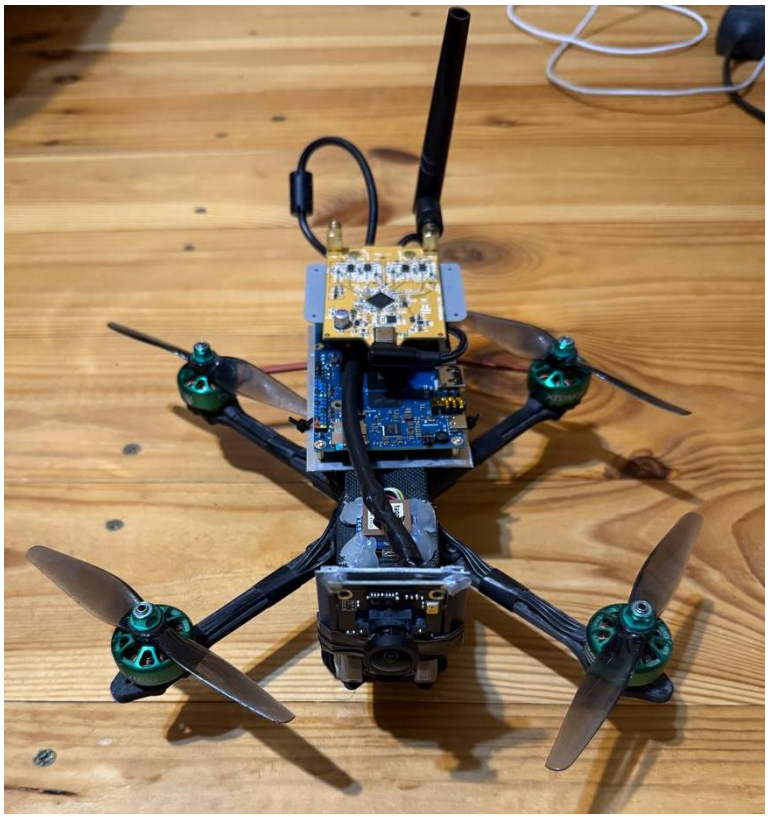


Рис. 2.10 Дрон з реалізованою системою (1)

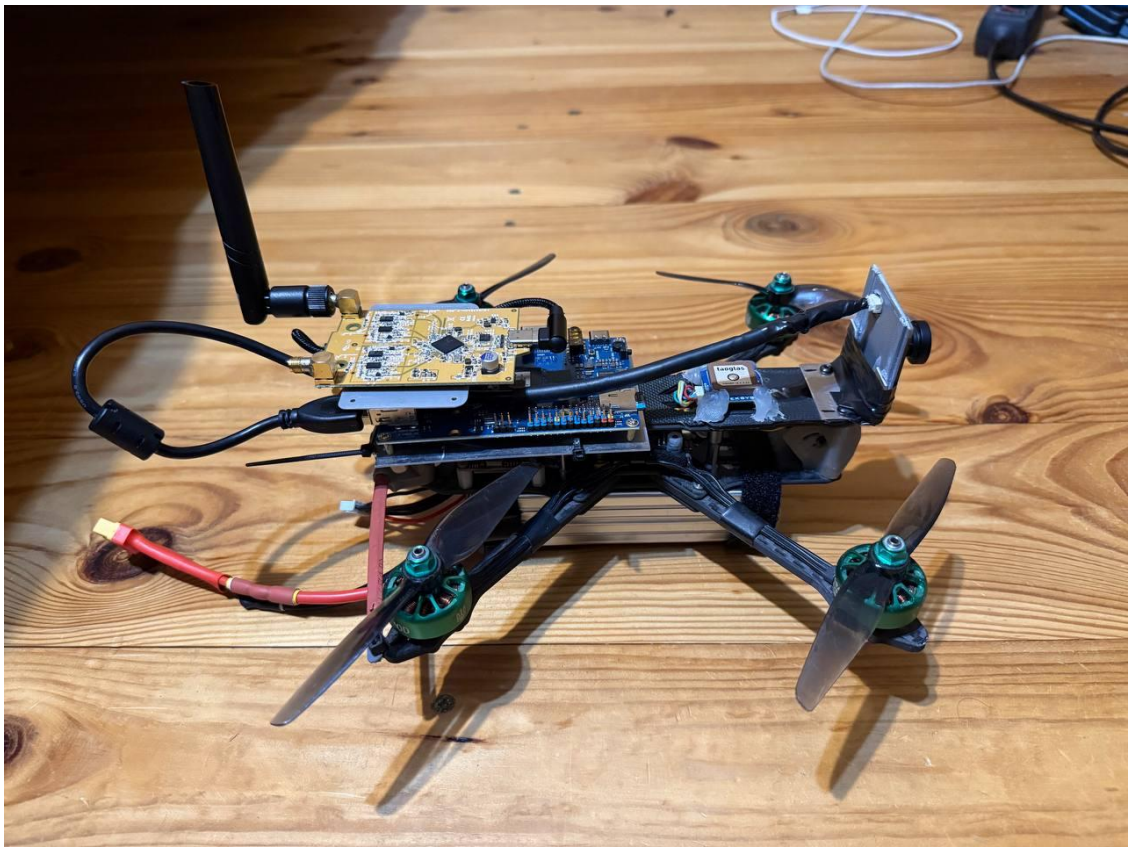


Рис. 2.11 Дрон з реалізованою системою (2)

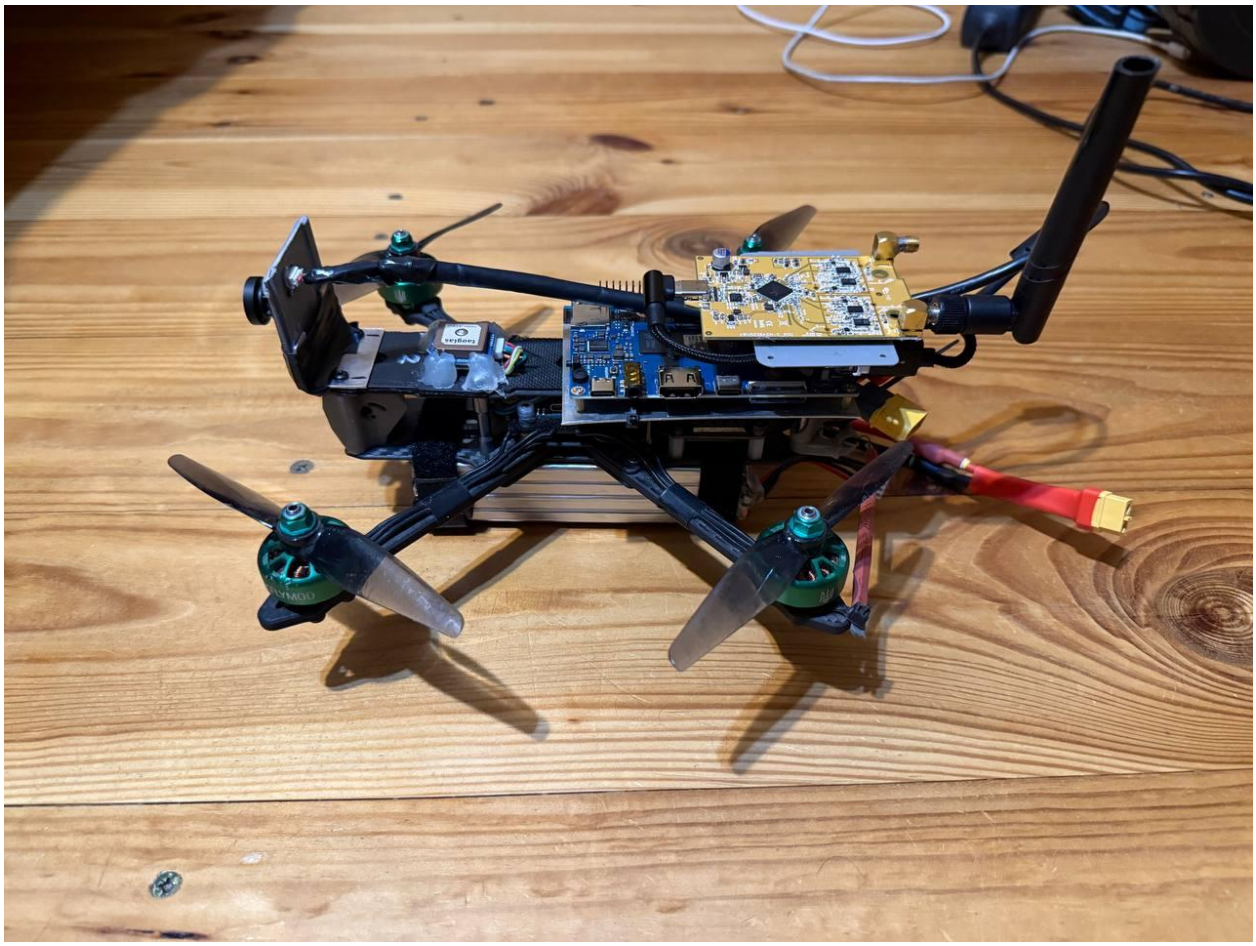


Рис. 2.12 Дрон з реалізованою системою (3)

2.1.3 Взаємодія компонентів один з одним

В системі камера підключена до плати Orange Pi 5 через USB, alfa підключена теж до USB , встановлена картка TF, подача енергії відбувається через ubec 5v, вивід здійснюється на монітор через HDMI для початкового налаштування айпі альфи, якщо підключення відбувається через нову мережу Wi-Fi (рис. 3.13). Отримуємо відео в онлайні з системи на прийстій по мережі , в моєму випадку звичайний ПК отримує зображення на VLC.

Компоненти, які потрібні для запуску системи:

- Картка TF, високошвидкісна карта SanDisk класу 10 або вище з мінімальною ємністю 8 ГБ (рекомендовано 32 ГБ або вище)

- Дисплей з інтерфейсом HDMI

- Кабель HDMI-HDMI, який використовується для підключення плати розробки до монітора HDMI або телевізора для відображення

- Адаптер живлення, Orange Pi 5 рекомендується використовувати джерело живлення 5 В/4 А Type-C для живлення

- Миша та клавіатура інтерфейсу USB, якщо миша та клавіатура стандартного інтерфейсу USB прийнятні, миша та клавіатура можуть використовуватися для керування платою розробки Orange Pi

- Камера з інтерфейсом MIPI

- Ноутбук або ПК на який будемо в онлайні отримувати зображення

2.2 Програмна частина системи

2.2.1 Програмне забезпечення, яке використано на Orange Pi 5

Використовується операційна система Linux

Основний файл для завантаження, Orangepi5_1.1.8_debian_linux 5.10.160. [23]

Ця збірка OrangePi 5 поставляється з операційною системою Debian 11 ("Bullseye") та ядром Linux 5.10.160. Вона включає в себе всі необхідні драйвери та програмне забезпечення для роботи OrangePi 5.

Debian - популярний дистрибутив Linux, який відомий своєю стабільністю та надійністю. Він є хорошим вибором для досвідчених користувачів, які хочуть мати більше контролю над своєю системою.

2.2.2 Підготовка та маркування даних

Модель натренована на домашньому ПК з операційною системою Windows 10 в Yolo V4 [24]

Зображення для навчання нейронної мережі було взято з відкритих джерел, в моєму випадку це google.

Зроблено маркування цілей на dataset за допомогою застосунка Yololabel. [24]

Операція маркування цілей на dataset за допомогою застосунка Yololabel має на меті підготовку даних для навчання моделей комп'ютерного зору, зокрема для завдань, пов'язаних із виявленням об'єктів. [24]

За допомогою Yololabel користувачі вручну зазначають області на зображеннях, де знаходяться об'єкти, які потрібно виявити.

На рисунках 2.13-2.19 зображено послідовне виконання маркування

..			Папка с файлами		
translations	4 701 668	1 384 301	Папка с файлами	12.01.2023 18:09	
styles	178 384	78 990	Папка с файлами	12.01.2023 18:09	
platforms	981 712	408 393	Папка с файлами	12.01.2023 18:09	
imageformats	530 752	187 743	Папка с файлами	12.01.2023 18:09	
iconengines	73 424	31 189	Папка с файлами	12.01.2023 18:09	
YoloLabel.exe	111 616	46 757	Приложение	12.01.2023 18:06	4E8F700B
Qt6Widgets.dll	6 296 264	2 665 755	Расширение при...	24.02.2021 19:16	C9CC6882
Qt6Svg.dll	350 920	142 770	Расширение при...	25.02.2021 2:07	F9A11834
Qt6Gui.dll	8 892 616	3 824 672	Расширение при...	24.02.2021 19:16	FAFC1CD2
Qt6Core.dll	6 066 392	2 650 733	Расширение при...	24.02.2021 19:16	A44497B3
opengl32sw.dll	20 633 208	7 677 020	Расширение при...	04.06.2020 10:50	92F954CD
libwinpthread-1.dll	52 224	23 801	Расширение при...	12.05.2018 9:11	7E085F05
libstdc++-6.dll	1 417 216	424 783	Расширение при...	12.05.2018 9:11	B067A34A
libgcc_s_seh-1.dll	76 288	36 332	Расширение при...	12.05.2018 9:11	9D9D2920
D3Dcompiler_47.dll	4 173 928	1 755 841	Расширение при...	11.03.2014 12:54	36D3FED2

Рисунок 2.13 Файли програми, запуск через YoloLabel.exe

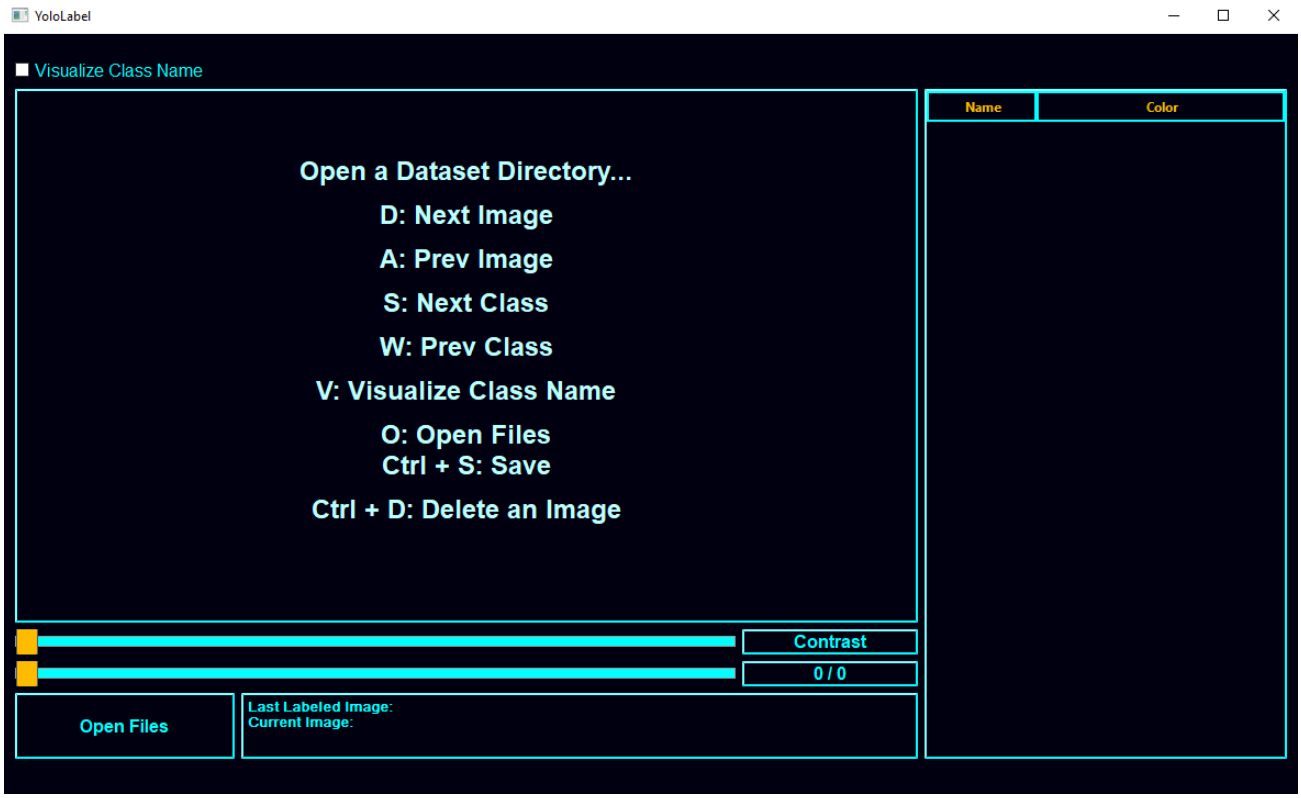


Рисунок 2.14 Запущений YoloLabel

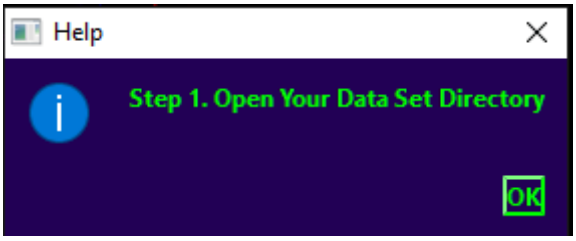


Рисунок 2.15 Вікно вибору папки

Рисунок 2.16 Вибір текстового файлу з назвою

Рисунок 2.17 Виділення першої моделі\

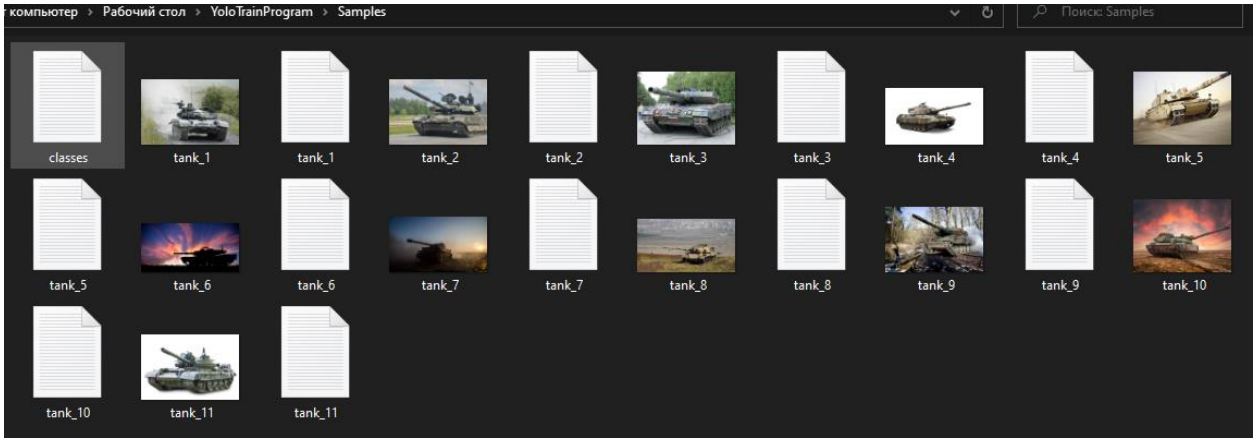


Рисунок 2.18 Виконане маркування

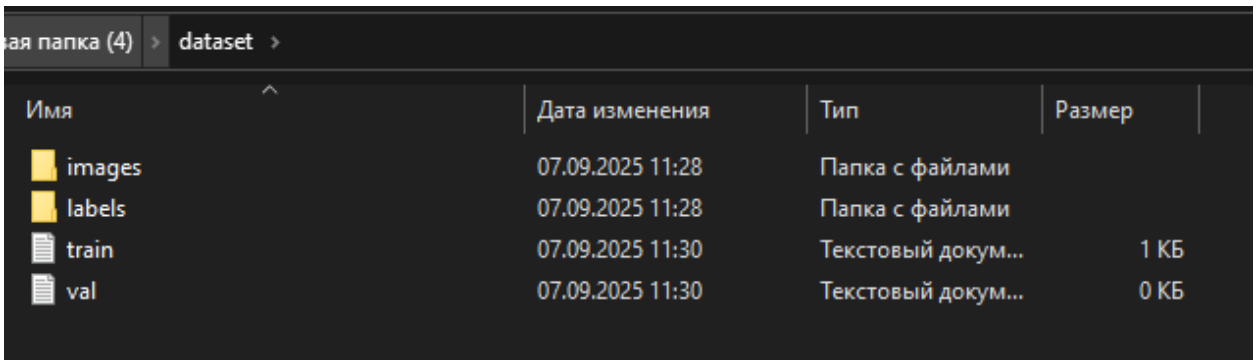


Рисунок 2.19 Структура папки datset перед навчанням моделі

dataset/
├── images/
│ ├── train/
│ └── val/
├── labels/
│ ├── train/
│ └── val/

Навчання на Windows

1. Завантаж базові ваги (пре-треновані на COCO):
2. wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/yolov4.conv.137
3. Запусти тренування:
4. darknet.exe detector train data/obj.data cfg/yolov4-tank.cfg yolov4.conv.137 -map
5. У процесі з’являться файли:
6. backup/yolov4-tank_last.weights
7. backup/yolov4-tank_best.weights

Маркування цілей на датасеті є невід'ємним кроком у створенні та навчанні ефективних моделей детекції об'єктів.

2.2.3 Опис етапів складання та налаштування системи Orange Pi 5

- Встановлення Qt на Orange Pi 5

1. Оновити систему:

```
sudo apt update && sudo apt upgrade -y
```

2. Встановити Qt і Qt Creator (це спрощує роботу з GUI):

```
sudo apt install qt5-default qtcreator -y
```

3. Встановити OpenCV:

sudo apt install libopencv-dev –y

- Структура проекту Qt

my_yolo_project/

├── main.cpp

├── mainwindow.h

├── mainwindow.cpp

├── mainwindow.ui

├── yolo4-tiny.cfg

├── yolov4-tiny_final.weights

└── classes.names

mainwindow.ui — створюємо через Qt Designer

ui_mainwindow.h з'являється автоматично під час компіляції з mainwindow.ui

Компіляція:

- qmake

- make

- ./yolo_qt

В результаті отримуємо вікно програми з відеопотоком і обробкою YOLOv4-tiny.

2.2.4 Код та дії які виконує програма

Повний код основної програми знаходиться в Додатку А

Етапи та дії програми:

1. Підключення бібліотек
2. Малюємо прямокутник із зображенням обмежувальної рамки

3. Отримати мітку для назви класу та його надійність (наше маркування моделей)

4. Відображення мітки у верхній частині обмежувальної рамки

5. Отримуємо назви вихідних шарів

6. Створення головного вікна

7. Ініціалізація змінних, підготовка до роботи з відео або зображенням

8. Ініціалізація відеозаписувача для збереження вихідного відео

9. Створення 4D краплі з кадру.

10. Просканувати всі обмежувальні рамки, виведені з мережі, зберегти лишез високими оцінками достовірності. Призначте мітку класу коробки як клас з найвищим балом.

11. Виконати придушення, щоб усунути зайві блоки, що перекриваються

12. Визначаємо коефіцієнт різниці положення та розміру

13. Задаємо умови збігу та модифікації, коефіцієнт змінюється

14. Вивід на екран

3. ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

3.1 Результати тестування

Програма виконує захват об’єкту та виводить вірогідність того, що це танк від 0 до 1 (рис. 3.1-3.4).

Рисунок 3.1 Результат тестування системи (1.1)

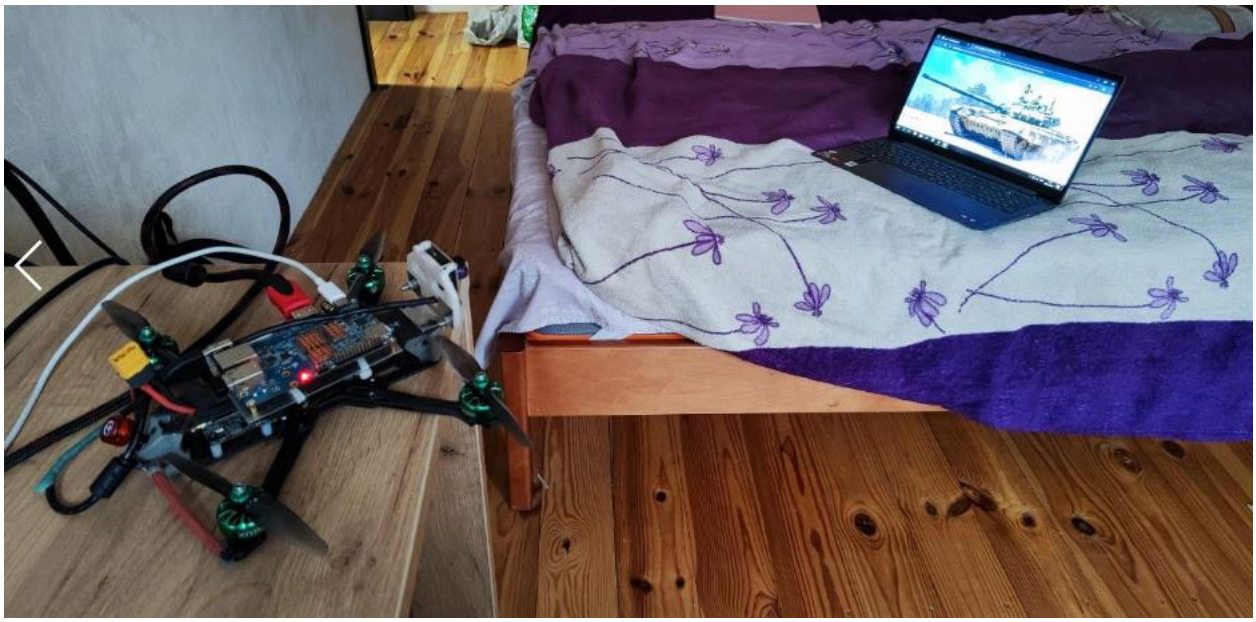


Рисунок 3.2 Результат тестування системи (1.2)

Рисунок 3.3 Результат тестування системи (2)

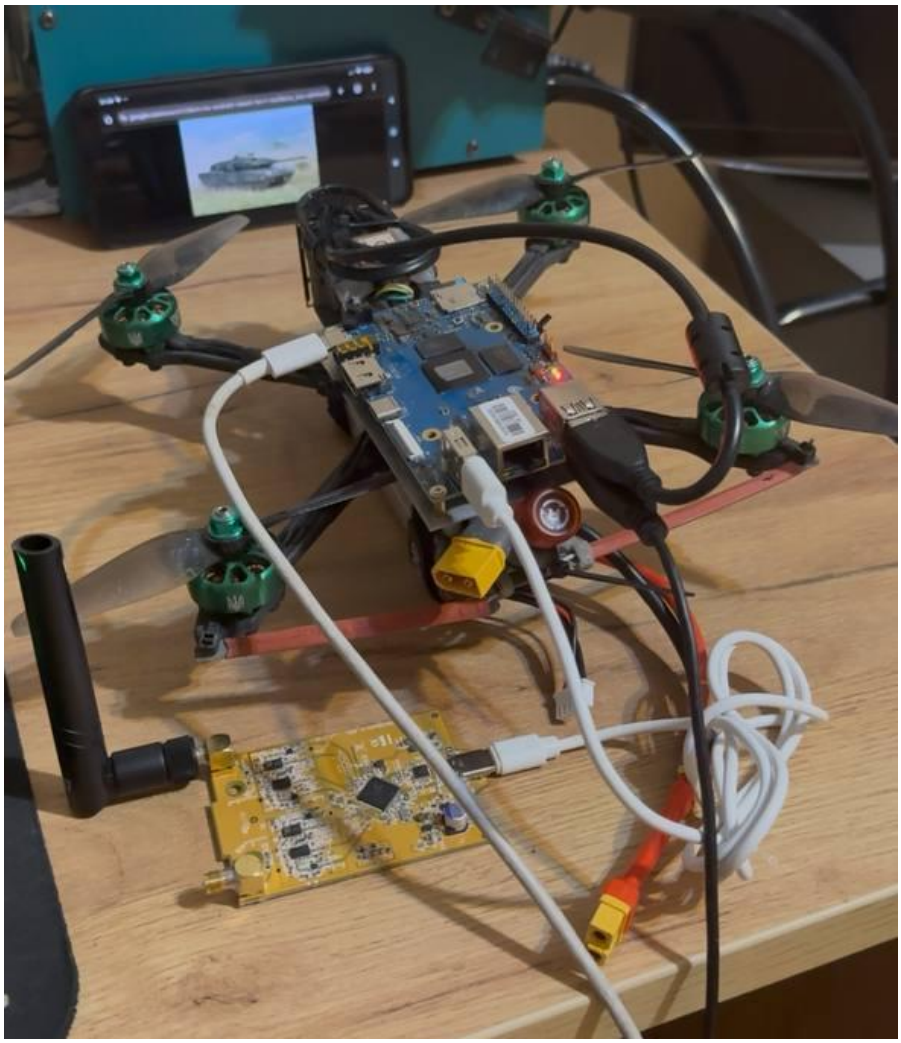


Рисунок 3.4 Результат тестування системи (3.1)

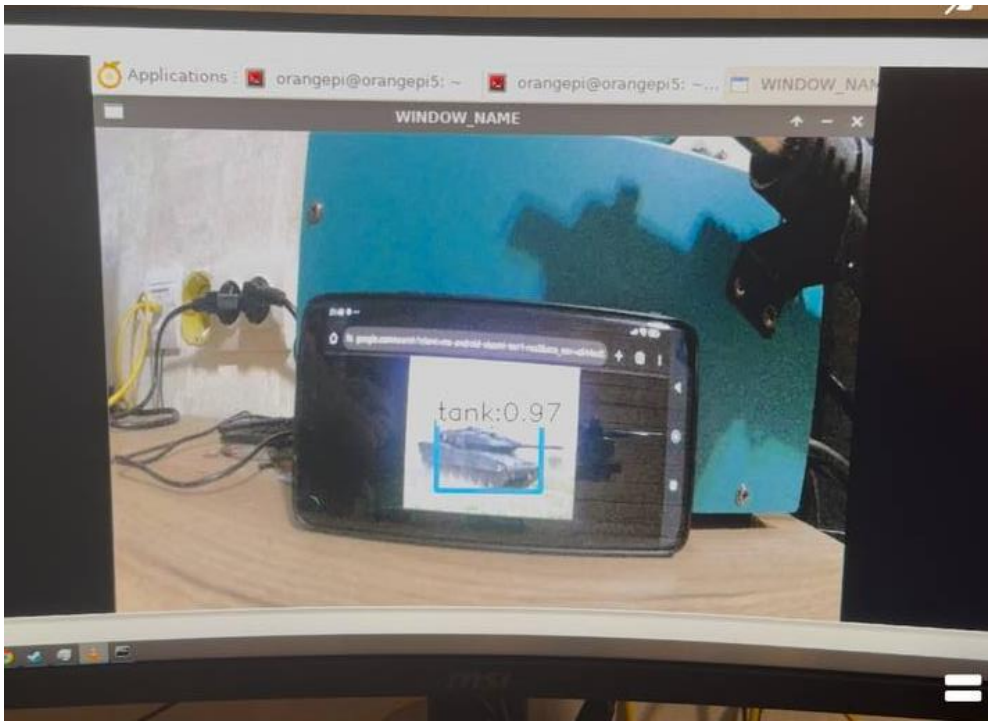


Рисунок 3.4 Результат тестування системи (3.2)

3.2 Методи і засоби тестування

Для оцінки працездатності та ефективності системи ідентифікації нетипових об'єктів на полях можна використовувати комплексний підхід, який включає наступні методи та засоби тестування:

- Модульне тестування – чи коректно працює кожен окремий компонент системи

- Інтеграційне тестування – як компоненти взаємодіють між собою. Наприклад, чи коректно працює дрон

- Системне тестування – правильність роботи товару загалом.

- Приймальний тест – чи відповідає продукт первісному технічному завданню

3.3 точність та ефективність системи

Система по собі є доволі ефективною, але є деякі нюанси. Використання моделі, яка натренована за допомогою зображень, які знаходяться в вільному доступі інтернету, не є точними. Таким чином ми зіткнемося з не ефективністю в поганих погодних умовах, або в тій же відстані до об’єкту, також наш об’єкт може бути накритий маскувальними сітками і не зможе бути розпізнаним.

3.4 Вдосконалення системи

Для першочергового вдосконалення системи потрібно вибрати максимальну комплектацію обладнання, далі взяти всі можливі зображення в

різних умовах для маркування, потім натренувати модель з урахуванням всіх нюансів, допрацювати програму для всіх можливих випадків.

Наступним етапом вдосконалення є повна інтеграція системи в польотний контролер, який керує дроном, В цьому випадку потрібен додатковий код, та обладнання яке зможе це реалізувати.

Заключним етапом можна буде вважати налаштування системи, яка зможе працювати на ефективну відстань та робити захват об’єктів не залежно від умов.

ВИСНОВКИ

В рамках даного проекту було виконано дослідження та розробку системи з використанням машинного навчання для обробки зображень, яка встановлена на дрон.

Було проаналізовано платформу Orange Pi 5, оцінено її характеристики та можливості використання.

Використано Yololabel для анотування даних, який застосовується для підготовки датасетів, та YOLOv4, яка є однією з найпопулярніших моделей для детекції об'єктів. Yololabel використовується для підготовки даних, які потім використовуються для тренування моделі YOLOv4, яка, у свою чергу, виконує детекцію об'єктів на нових зображеннях.

Розроблену систему на платформі Orange Pi 5, яка ідентифікує нетиповий об’єкт.

Розроблено FPV-дрон, який успішно взаємодіє зі створеною системою.

Покращення системи можливе за рахунок розширення та урізноманітнення датасету, додавши більше типів об'єктів для тренування моделі, а також забезпечивши повну інтеграцію системи з політним контролером дрона.

Розроблена система може бути використана і для інших напрямків, в залежності від використаної моделі, також розроблено рекомендації щодо її модернізації для підвищення продуктивності та точності. Розроблені рекомендації можуть бути використані для подальшого вдосконалення системи та її адаптації до різних умов експлуатації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Which functions can't neural networks learn efficiently [Електронний ресурс]. URL: https://ai.stackexchange.com/questions/5539/which-functions-cant-neural-networks-learn-efficiently

2. Wang, F., He, X., Gao, J., & Cheng, X. Robust Object Detection Under Occlusion With Context-Aware CompositionalNets. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, WA, USA, 2020. P. 13247–13256. URL:https://openaccess.thecvf.com/content_CVPR_2020/papers/Wang_Robust_Object_Detection_Under_Occlusion_With_Context-Aware_CompositionalNets_CVPR_2020_paper.pdf

3. Gašparović, M., Gašparović, L., & Topolnjak, I. (2023). Evaluating YOLOv5, YOLOv6, YOLOv7 and YOLOv8 in Underwater Environment: Is There Real Improvement ? ResearchGate. URL:https://www.researchgate.net/publication/372841442_Evaluating_YOLOV5_YOLOV6_YOLOV7_and_YOLOV8_in_Underwater_Environment_Is_There_Real_Improvement

4. What is Mask R-CNN and How Does It Work [Електронний ресурс] URL: https://www.ultralytics.com/blog/what-is-mask-r-cnn-and-how-does-it-work

5. EfficientDet [Електронний ресурс]. URL: https://roboflow.com/model/efficientdet

6. Raspberry pi 5 [Електронний ресурс]. URL: https://www.raspberrypi.com/products/raspberry-pi-5/

7. Radxa Team. Getting Started with ROCK 5C: Introduction. [Електронний ресурс]. URL: https://docs.radxa.com/en/rock5/rock5c/getting-started/introduction

8. Radxa Official Website. ROCK 5C Product Page: Specifications and Features [Електронний ресурс]. URL: https://radxa.com/products/rock5/5c#techspec

9. DOU.ua. Форум: Обговорення нейронних мереж та сучасних AI-технологій [Електронний ресурс]. URL: https://dou.ua/forums/topic/48368/

10. Roboflow Blog Vision Transformers: The Ultimate Guide [Електронний ресурс]. URL: https://blog.roboflow.com/vision-transformers/

11. DEtection TRansformer (DETR) [Електронний ресурс]. URL: https://huggingface.co/learn/computer-vision-course/unit3/vision-transformers/detr

12. Song H., Sun D., Chun S., Yang M. *An Extendable, Efficient and Effective Transformer-based Object Detector* URL:https://www.researchgate.net/figure/Performance-of-recent-object-detectors-in-terms-of-average-precision-AP-and-latency_fig1_360031031

13. Безпілотний апарат [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/%D0%94%D1%80%D0%BE%D0%BD

14. How to label your dataset for YOLO Object Detection [Електронний ресурс]. URL: https://machinecurve.com/index.php/2021/03/30/how-to-label-your-dataset-for-yolo-object-detection

15. YOLOv4: Optimal Speed and Accuracy of Object Detection [Електронний ресурс]. URL:https://arxiv.org/abs/2004.10934

16. Deep neural network library and toolkit to do high performace inference on NVIDIA jetson platforms (Yolo v4 порівняння) [Електронний ресурс]. URL: github.com/ceccocats/tkDNN

17. AlexeyAB: Yolo v4 – tiny [Електронний ресурс]. Режим доступу: github.com/AlexeyAB/darknet/issues/6067

18. State-of-art Yolo v4 Detector [Електронний ресурс]. URL: https://github.com/opencv/opencv/issues/17148

19. Oliveira Faria, A., Shtanchaev, A.: Running pre-trained YOLO model in OpenCV [Електронний ресурс]. URL: docs.opencv.org/master/da/d9d/tutorial_dnn_yolo.html

20. Orange pi 5 User Manual [Електронний ресурс]. URL: https://drive.google.com/file/d/1-6IwPNMOhWDvbLe2-PvHUocHStwavGGi/view

21. Camera Module 12MP USB [Електронний ресурс]. URL: https://www.arducam.com/product/12mp-usb-camera-module-with-m12-lens-1-2-3-3840hx3032v-4k30fps-for-windows-linux-macos-and-android/

22. Wi-Fi USB адаптер Alfa Network AWUS036ACH v2 [Електронний ресурс]. URL: https://www.alfa-network.com.ua/wi-fi-adaptery/wi-fi-usb-adapter-alfa-network-awus036ach-v2

23. Orange pi 5 Hardware [Електронний ресурс]. URL: http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-pi-5.html

24. Yong Kwon: GUI for marking bounded boxes of objects in images for training neural network YOLO [Електронний ресурс]. URL: https://github.com/developer0hye/Yolo_Label

ДОДАТОК А

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <fstream>
#include <sstream>
#include <iostream>
#include <math.h>
#include <QObject>
#include <opencv2/dnn.hpp>


#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/videoio.hpp>
#include <ctime>

#include "QDebug"
#pragma warning(disable : 4996)
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace dnn;
using namespace std;
    std::vector<String> classes;

    void drawPred(int classId, float conf, int left, int top, int right, int
  bottom, Mat& frame)
    {
        //Draw a rectangle displaying the bounding box
        rectangle(frame, Point(left, top), Point(right, bottom), Scalar(255, 178,
  50), 3);

        //Get the label for the class name and its confidence
        string label = format("%.2f", conf);
        if (!classes.empty())
        {
            CV_Assert(classId < (int)classes.size());
            label = classes[classId] + ":" + label;
        }

        //Display the label at the top of the bounding box
        int baseLine;
        Size labelSize = getTextSize(label, FONT_HERSHEY_SIMPLEX, 0.5, 1,
  &baseLine);
        top = max(top, labelSize.height);
        rectangle(frame, Point(left, top - round(1.5 * labelSize.height)),
  Point(left + round(1.5 * labelSize.width), top + baseLine), Scalar(255, 255,
  255), FILLED);
        putText(frame, label, Point(left, top), FONT_HERSHEY_SIMPLEX, 0.75,
  Scalar(0, 0, 0), 1);
    }

    // Get the names of the output layers
    std::vector<String> getOutputsNames(const Net& net)
    {
        static vector<String> names;
        if (names.empty())
        {
```

```
            //Get the indices of the output layers, i.e. the layers with
  unconnected outputs
            vector<int> outLayers = net.getUnconnectedOutLayers();

            //get the names of all the layers in the network
            vector<String> layersNames = net.getLayerNames();

            // Get the names of the output layers in names
            names.resize(outLayers.size());
            for (size_t i = 0; i < outLayers.size(); ++i)
                names[i] = layersNames[outLayers[i] - 1];
        }
        return names;
    }


MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    float confThreshold = 0.5f; // Confidence threshold
    float nmsThreshold = 0.4f;  // Non-maximum suppression threshold
    bool inertialAlgorithm = false;
    int inpWidth = 640;  // Width of network's input image
    int inpHeight = 480; // Height of network's input image

    int xpos, ypos, dx = 48, dy = 48; //Размер окна относительно центра
    float alpha = 0.75f;
    float cedge = 10.0f;
    float kedge = 0.15f;
    bool track = false;
    bool tracking = false;
    bool saveVideo = false;
    bool detection = true;
    void detectionTracking();
    Mat signature, frame1;
    int aimH, aimW;
    signature = Mat::zeros(Size(2 * dy, 2 * dx), CV_32FC1);
    Mat hanwin;

    createHanningWindow(hanwin, Size(2 * dy, 2 * dx), CV_32FC1);

    double imagesize = sqrt(inpWidth*inpWidth + inpHeight * inpHeight);
    // Load names of classes
    string classesFile = "classes.names";
    ifstream ifs(classesFile.c_str());
    string line;
    while (getline(ifs, line)) classes.push_back(line);

    // Give the configuration and weight files for the model
    String modelConfiguration = "yolov4-tiny.cfg";
    String modelWeights = "yolov4-tiny_final.weights";

    // Load the network
    Net net = readNetFromDarknet(modelConfiguration, modelWeights);
    net.setPreferableBackend(DNN_BACKEND_OPENCV);
    net.setPreferableTarget(DNN_TARGET_CPU);

    // Open a video file or an image file or a camera stream.
    string str, outputFile;
```

```
    VideoCapture cap;
    VideoWriter video;
    Mat frame, blob;



    cap.open(0);
    /*str.replace(str.end() - 4, str.end(), "_yolo_out_cpp.avi");
    std::time_t result = std::time(nullptr);
    std::cout << std::asctime(std::localtime(&result));
    outputFile = "D:/Work/data/bayr/";
    outputFile.append(std::asctime(std::localtime(&result)));
    outputFile.erase(outputFile.size() - 1, 1);
    outputFile.append(".avi");
    replace(outputFile.begin(), outputFile.end(), ' ', '_');
    replace(outputFile.begin()+3, outputFile.end(), ':', '_');
    std::cout << outputFile;

    // Open the webcaom

    video.open(outputFile, VideoWriter::fourcc('M', 'J', 'P', 'G'), 10,
  Size(inpWidth, inpHeight));
*/


// Get the video writer initialized to save the output video


// Create a window
    cv::Mat main_frame = cv::Mat(inpHeight, inpWidth + 240, CV_8UC3);
    cv::Mat camera = cv::Mat(inpHeight, inpWidth, CV_8UC3);
    float count = 0;

    // Init a OpenCV window and tell cvui to use it.

    vector<int> main_classIds;
    vector<float> main_confidences;
    vector<Rect> main_boxes;
    vector<bool> modificated;
    Rect2d bbox(250, 100, 200, 225);
    cv::Point  min_loc, max_loc;
    bool init = false;
    bool photo = false;
    double Rc = 0;

    // Process frames.
    while (true)
    {

        double t;
        // Show a button at position (110, 80)
        /*if (cvui::button(main_frame, inpWidth + 20, 50, "Hello, world!")) {
            // The button was clicked, so let's increment our counter.
            count++;
        }*/


        // Show how many times the button has been clicked.
        // Text at position (250, 90), sized 0.4, in red.
```

```
        // get frame from the video
        cap >> frame;
        //cv::cvtColor(frame, frame, COLOR_BGR2GRAY);
        //cv::cvtColor(frame, frame, COLOR_GRAY2BGR);
        // Stop the program if reached end of video
        if (frame.empty()) {
            cout << "Done processing !!!" << endl;
            cout << "Output file is stored as " << outputFile << endl;
            waitKey(3000);
            break;
        }
        cv::rotate(frame,frame, ROTATE_180);
        cv::resize(frame, frame, Size(inpWidth, inpHeight));
        frame1 = frame.clone();
        cv::cvtColor(frame1, frame1, COLOR_BGR2GRAY);
        cv::filter2D(frame1, frame1, -1, Mat::ones(Size(7, 7), CV_32FC1) / 49);
        cv::Canny(frame1, frame1, cedge*1.5, cedge);

        cv::filter2D(frame1, frame1, -1, Mat::ones(Size(1, 1), CV_32FC1));
        frame1.convertTo(frame1, CV_32FC1);


        Mat aimImage;


        if (detection)
        {


            // Create a 4D blob from a frame.
            blobFromImage(frame, blob, 1 / 255.0, cv::Size(inpWidth, inpHeight),
Scalar(0, 0, 0), true, false);

            //Sets the input to the network
            net.setInput(blob);

            // Runs the forward pass to get output of the output layers
            vector<Mat> outs;
            net.forward(outs, getOutputsNames(net));

            // Remove the bounding boxes with low confidence
            //postprocess(frame, outs);
            vector<int> classIds;

            vector<float> confidences;
            vector<Rect> boxes;

            for (size_t i = 0; i < outs.size(); ++i)
            {
                // Scan through all the bounding boxes output from the network
and keep only the
                // ones with high confidence scores. Assign the box's class label
as the class
                // with the highest score for the box.
                float* data = (float*)outs[i].data;
                for (int j = 0; j < outs[i].rows; ++j, data += outs[i].cols)
                {
                    Mat scores = outs[i].row(j).colRange(5, outs[i].cols);
                    Point classIdPoint;
                    double confidence;
                    // Get the value and location of the maximum score
```

```
                    minMaxLoc(scores, 0, &confidence, 0, &classIdPoint);
                    if (confidence > confThreshold)
                    {
                        int centerX = (int)(data[0] * frame.cols);
                        int centerY = (int)(data[1] * frame.rows);
                        int width = (int)(data[2] * frame.cols);
                        int height = (int)(data[3] * frame.rows);
                        int left = centerX - width / 2;
                        int top = centerY - height / 2;

                        classIds.push_back(classIdPoint.x);
                        confidences.push_back((float)confidence);
                        boxes.push_back(Rect(left, top, width, height));
                    }
                }
            }

            // Perform non maximum suppression to eliminate redundant overlapping
boxes with
            // lower confidences
            vector<int> indices;

            NMSBoxes(boxes, confidences, confThreshold, nmsThreshold, indices);
            if (inertialAlgorithm)
            {
                if (main_boxes.size() < 1)
                {
                    for (size_t i = 0; i < indices.size(); ++i)
                    {
                        int idx = indices[i];
                        Rect box = boxes[idx];
                        drawPred(classIds[idx], confidences[idx], box.x, box.y,
                            box.x + box.width, box.y + box.height, frame);
                        main_boxes.push_back(boxes[idx]);
                        main_confidences.push_back(confidences[idx]);
                        main_classIds.push_back(classIds[idx]);
                        modificated.push_back(true);
                    }
                }
                Else {
                    for (size_t j = 0; j < indices.size(); ++j)
                    {
                        bool founded = false;
                        int idx = indices[j];
                        Rect box = boxes[idx];
                        double sizebox = sqrt(boxes[j].height*boxes[j].height +
boxes[j].width *boxes[j].width);

                        for (size_t i = 0; i < main_boxes.size(); i++)
                        {
                            if (main_classIds[i] != classIds[idx] &&
modificated[i] == true)
                                break;
                            double rangeCoef = sqrt(pow(boxes[j].x -
main_boxes[i].x, 2) + pow(boxes[j].y - main_boxes[i].y, 2)) / imagesize;
                            ;
                            double sizeCoef = abs(1 - sizebox /
sqrt(main_boxes[i].height*main_boxes[i].height + main_boxes[i].width *
main_boxes[i].width));
                            if (rangeCoef < 0.2 && sizeCoef < 0.2)
                            {
```

```
                                main_confidences[i] = (main_confidences[i] +
  confidences[j]) / 2;
                                modificated[i] = true;
                                founded = true;
                                main_boxes[i] = boxes[j];
                            }

//Коефіцієнт різниці положення та розміру
//Умови збігу та модифікації
//Якщо так то вважаємо новий коефіцієнт
//і відзначаємо, що він модифікований                          }
                            if (!founded)
                            {
                                main_boxes.push_back(boxes[idx]);
                                main_confidences.push_back(confidences[idx]);
                                main_classIds.push_back(classIds[idx]);
                                modificated.push_back(true);
                            }


                        }



                    }

                    for (int i = 0; i < main_boxes.size(); ++i)
                    {
                        if (modificated[i] == false)
                            main_confidences[i] = (main_confidences[i] +
  confThreshold * 0.8) / 2;
                        if (main_confidences[i] < confThreshold)
                        {
                            main_boxes.erase(main_boxes.begin() + i);
                            main_confidences.erase(main_confidences.begin() + i);
                            main_classIds.erase(main_classIds.begin() + i);
                            modificated.erase(modificated.begin() + i);
                            i--;
                            break;
                        }
                        Rect box = main_boxes[i];
                        drawPred(main_classIds[i], main_confidences[i], box.x, box.y,
                            box.x + box.width, box.y + box.height, frame);
                    }
                    for (size_t i = 0; i < modificated.size(); i++)
                        modificated[i] = false;
                }
                else
                {
                    for (size_t i = 0; i < indices.size(); ++i)
                    {
                        int idx = indices[i];
                        Rect box = boxes[idx];
                        drawPred(classIds[idx], confidences[idx], box.x, box.y,
                            box.x + box.width, box.y + box.height, frame);
                    }
                }
                vector<double> layersTimes;
                double freq = getTickFrequency() / 1000;
                t = net.getPerfProfile(layersTimes) / freq;
```

```
            //cvui::printf(main_frame, inpWidth + 20, 200, 0.4, 0xff0000, "Time
    for a frame %.2f ms", t);
        }



        // vec.erase(vec.begin() + index);
        // Put efficiency information. The function getPerfProfile returns the
    overall time for inference(t) and the timings for each of the layers(in
    layersTimes)

        //string label = format("Inference time for a frame : %.2f ms", t);
        //putText(frame, label, Point(0, 15), FONT_HERSHEY_SIMPLEX, 0.5,
    Scalar(0, 0, 255));

        // Write the frame with the detection boxes
       /* if (saveVideo)
        {
            Mat detectedFrame;
            frame.convertTo(detectedFrame, CV_8U);
            video.write(detectedFrame);
        }*/

        //if (parser.has("image")) imwrite(outputFile, detectedFrame);
        //else



      //  cvui::printf(main_frame, inpWidth + 20, 220, 0.4, 0xff0000, "Detected
    objects %1d", main_boxes.size());
      //  cvui::printf(main_frame, inpWidth + 20, 300, 0.4, 0xff0000, "Rc %1f",
    Rc);
      //  cvui::image(main_frame, 0, 0, frame);
       // cvui::update();

         //Show everything on the screen
        cv::imshow("WINDOW_NAME", frame);
       waitKey(20);

    }

    cap.release();

}

MainWindow::~MainWindow()
{
    delete ui;
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
```

```
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget* parent = nullptr);
    ~MainWindow();

private:
    Ui::MainWindow* ui;
};
#endif // MAINWINDOW_H
```

Qmake.txt

```
QMAKE_CXX.QT_COMPILER_STDCXX = 201402L
QMAKE_CXX.QMAKE_GCC_MAJOR_VERSION = 10
QMAKE_CXX.QMAKE_GCC_MINOR_VERSION = 2
QMAKE_CXX.QMAKE_GCC_PATCH_VERSION = 1
QMAKE_CXX.COMPILER_MACROS = \
    QT_COMPILER_STDCXX \
    QMAKE_GCC_MAJOR_VERSION \
    QMAKE_GCC_MINOR_VERSION \
    QMAKE_GCC_PATCH_VERSION
QMAKE_CXX.INCDIRS = \
    /usr/include/c++/10 \
    /usr/include/aarch64-linux-gnu/c++/10 \
    /usr/include/c++/10/backward \
    /usr/lib/gcc/aarch64-linux-gnu/10/include \
    /usr/local/include \
    /usr/include/aarch64-linux-gnu \
    /usr/include
QMAKE_CXX.LIBDIRS = \
    /usr/lib/gcc/aarch64-linux-gnu/10 \
    /usr/lib/aarch64-linux-gnu \
    /usr/lib \
    /lib/aarch64-linux-gnu \
    /lib
```