КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 – Комп'ютерні науки,
освітня програма «Інформаційна аналітика та впливи»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

"Методи комп'ютерного зору для розпізнавання безпілотних літальних апаратів**"**

Студента 2-го курсу групи ІАВ-21

Мірошника Андрія Петровича
(прізвище, ім'я, по батькові)

_____
(підпис студента)

Науковий керівник:

доцент кафедри технологій управління, доктор технічних наук з інформаційних технологій
(науковий ступінь, вчене звання)

Заріцький Олег Володимирович
(прізвище, ім'я, по батькові)

_____ _____
(дата) (підпис)

Попередній захист:

_____
(Висновок: «До захисту в Екзаменаційній комісії»)

Завідувач кафедри
технологій управління _____ _____ _____
(підпис) (прізвище, ініціали) (дата)

Київ – 2026 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління
Освітньо-кваліфікаційний рівень Магістр
Спеціальність 122 - Комп'ютерні науки
Освітня програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Морозов В.В.
_____
«____» _____20__року

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент Мірошник Андрій Петрович

Група ІАВ-21

1. Тема кваліфікаційної роботи

Методи комп'ютерного зору для розпізнавання безпілотних літальних апаратів

Затверджена наказом по від «___» _____20___р. № ___.

2. Строк подання студентом готової роботи – "06" травня 2026 р.

3. Цільова установка та вихідні дані до роботи

Розробка та дослідження методів комп'ютерного зору для автоматичного розпізнавання безпілотних літальних апаратів з розгортанням на малопотужному мікроконтролері ESP32-S3. Вихідні дані: набір даних зображень БПЛА з трьома класами об'єктів (Other_UAV, Shahed, Bird); мікроконтролер ESP32-S3 як цільова апаратна платформа для розгортання; архітектура FOMO на основі MobileNetV2 як базова архітектура нейронної мережі; фреймворки TensorFlow та TensorFlow Lite Micro для навчання та розгортання моделей відповідно.

4. Зміст роботи

Аналіз сучасних загроз від БПЛА та методів їх розпізнавання; огляд архітектур нейронних мереж для розпізнавання об’єктів на крайових пристроях; обґрунтування вибору архітектури FOMO на основі MobileNetV2; розробка та навчання п’яти архітектурних варіантів моделі з різними параметрами роздільної здатності зображення та конфігурацією; дослідження впливу розміру вихідної сітки grid на деградацію точності розпізнавання класу Shahed після INT8 квантизації; конвертація моделей у формат TFLite INT8 та оцінювання деградації по класах; розгортання квантизованих моделей на мікроконтролері ESP32-S3 з реалізацією веб-інтерфейсу; тестування дворівневої системи розпізнавання БПЛА та техніко-економічне обґрунтування системи.

5. Перелік графічного матеріалу (слайдів)

Актуальність теми, мета роботи та наукова новизна; Аналіз загроз від БПЛА; Огляд методів комп’ютерного зору для розпізнавання БПЛА; Архітектура повноцінної моделі FOMO; Архітектура спрощеної моделі FOMO; Результати навчання моделей; Дослідження деградації при квантизації; Розгортання моделей на ESP32-S3; Демонстрація роботи системи; Порівняння моделей; Техніко-економічне обґрунтування; Напрями подальших досліджень; Висновки.

6. Календарний план виконання роботи:

№ з/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1.	Вибір теми дипломної роботи	3	15.09.25	15.09.25
2.	Протокол кафедри ТУ про затвердження тем дипломних робіт та призначення наукових керівників	2	27.12.25	27.12.25
3.	Формування переліку нормативних матеріалів, літератури з проблематики дипломної роботи	10	09.01.26	09.01.26
4.	Складання розгорнутого плану кваліфікаційної роботи	5	19.01.26	19.01.26

5.	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін.	5	21.01.26	21.01.26
6.	Підготовка розділу 1 «Аналітичний огляд проблеми розпізнавання БПЛА»	10	17.02.26	17.02.26
7.	Підготовка розділу 2 «Модель нейронної мережі для розпізнавання БПЛА»	14	09.03.26	09.03.26
8.	Підготовка розділу 3 «Моделювання системи розпізнавання БПЛА»	14	06.04.26	06.04.26
9.	Підготовка розділу 4 «Розгортання та практичне застосування системи розпізнавання БПЛА»	14	21.04.26	21.04.26
10.	Оформлення кваліфікаційної роботи. Підготовка висновків і пропозицій	14	01.05.26	01.05.26
11.	Передача кваліфікаційної роботи науковому керівникові	2	06.05.26	06.05.26
12.	Передача кваліфікаційної роботи рецензенту для рецензування	2	07.05.26	07.05.26
13.	Попередній захист кваліфікаційної роботи	5	11.05.26	11.05.26

Дата видачі завдання «____» _____ 20___р.

Керівник роботи доцент кафедри технологій управління, доктор технічних наук з інформаційних технологій
Заріцький Олег Володимирович
(посада, прізвище, ім’я, по батькові)

_____
(підпис)

Завдання прийняв до виконання студент групи ІАВ-21

Мірошник Андрій Петрович
(прізвище, ім’я, по батькові)

_____
(підпис)

ЗМІСТ

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра технологій управління
Спеціальність 122 - Комп'ютерні науки,
освітня програма "Інформаційна аналітика та впливи"

Дипломна робота магістра Мірошника Андрія Петровича.

Тема роботи – «Методи комп'ютерного зору для розпізнавання безпілотних літальних апаратів».

Мета дипломної роботи магістра – розробити та дослідити методи розпізнавання БПЛА на основі архітектури FOMO, реалізувати повний цикл від навчання моделі до її розгортання на мікроконтролері ESP32-S3.

Об'єкт дослідження – процес автоматичного розпізнавання та класифікації безпілотних літальних апаратів на зображеннях з використанням методів комп'ютерного зору.

Предмет дослідження – методи та моделі комп'ютерного зору на основі згорткових нейронних мереж для виявлення та розпізнавання БПЛА з їх розгортанням на крайових пристроях з обмеженими апаратними ресурсами.

Наукова новизна роботи – удосконалено підхід до квантизації моделей FOMO шляхом виявлення залежності між розміром вихідної сітки теплової карти та точністю розпізнавання специфічних класів БПЛА. Також удосконалено метод навчання моделі FOMO для розпізнавання БПЛА шляхом підбору оптимальної роздільної здатності вхідного зображення, що забезпечує збереження точності розпізнавання специфічних класів БПЛА після квантизації при розгортанні на крайовому пристрої.

У роботі досліджено наявні архітектури нейронних мереж для розпізнавання об’єктів та проведено їх порівняльний аналіз для задачі розпізнавання БПЛА. Було навчено декілька варіантів моделі FOMO на основі MobileNetV2 з різними параметрами та роздільною здатністю зображень. Було виявлено ефект диференційованої деградації класів при INT8 квантизації та визначено оптимальні параметри розгортання на крайовий пристрій. Реалізовано прототип системи на мікроконтролері ESP32-S3 з веб-інтерфейсом для дистанційного розпізнавання. Вдалося отримати точність розпізнавання Shahed та Other_UAV на рівні 85-95% на тестових зображеннях.

Дипломна робота складається зі вступу, основної частини, яка містить чотири розділи, висновків та списку використаних інформаційних джерел. Всього налічує 107 сторінок та перелік посилань з 29 джерел на 4 сторінках.

Ключові слова: безпілотний літальний апарат, комп’ютерний зір, FOMO, MobileNetV2, INT8 квантизація, TensorFlow Lite Micro, ESP32-S3, Edge AI, розпізнавання БПЛА.

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

БПЛА – Безпілотний Літальний Апарат

FOMO – Faster Objects, More Objects

HOG – Histogram of Oriented Gradients

SVM – Support Vector Machine

CNN – Convolutional Neural Network

YOLO – You Only Look Once

GPU – Graphics Processing Unit

SSD – Single Shot Detector

FPS – Frames Per Second

RAM – Random-Access Memory

ARM – Advanced RISC Machine

SRAM – Static Random-Access Memory

SIMD – Single Instruction, Multiple Data

CPU – Central Processing Unit

RGB – Red, Green, Blue

JSON – JavaScript Object Notation

IoU – Intersection over Union

MAC – Multiply-Accumulate operation

PSRAM – Pseudo-Static Random-Access Memory

IDE – Integrated Development Environment

SDK – Software Development Kit

HTML – HyperText Markup Language

NMS – Non-Maximum Suppression

HTTP – Hypertext Transfer Protocol

NPU – Neural Processing Unit

ВСТУП

Досить широке застосування безпілотних літальних апаратів (БПЛА) у сучасних збройних конфліктах стало стандартною тактикою ведення військових дій. Маючи відносно низьку собівартість виробництва, масове застосування таких засобів дозволяє доволі ефективно виснажувати систему протиповітряної оборони країни, витрачаючи незначні ресурси порівняно з вартістю оборонної військової техніки. При виконанні бойових завдань БПЛА атакують переважно об’єкти критичної інфраструктури з метою підриву функціонування державних систем. Через недосконаліть навігаційних систем такі засоби доволі часто вражають цивільну інфраструктуру, що призводить до значних руйнувань та жертв серед звичайного мирного населення [1].

Особливу загрозу становлять ударні БПЛА типу Shahed, відомі за класифікацією НАТО як тактичні БПЛА другого класу. Такі засоби систематично застосовуються проти об’єктів критичної енергетичної та цивільної інфраструктури України, а їх відносно невелика радіолокаційна сигнатура ускладнює виявлення та розпізнавання традиційними радарними засобами [2]. Окрім того, великого поширення набули розвідувальні БПЛА, які вирішують завдання пошуку зенітно-ракетних комплексів малої дальності дії на лінії фронту для їх подальшого ураження.

Саме тому в таких умовах масового застосування БПЛА виникає необхідність у дешевих автономних системах їх виявлення та розпізнавання. Дуже перспективним напрямком є застосування методів комп’ютерного зору безпосередньо на малопотужних крайових пристроях (Edge AI). Це дозволяє реалізувати розподілену архітектуру розпізнавання БПЛА з мінімальними апаратними витратами. Незважаючи на наявність окремих публікацій з тематики Edge AI для задач виявлення та розпізнавання об’єктів, питання оптимізації

невеликих квантизованих моделей для розпізнавання специфічних класів БПЛА саме безпосередньо на мікроконтролерах залишається недостатньо дослідженим.

Відповідно встановлено мету, завдання, об’єкт і предмет дослідження.

Мета дослідження – розробити та дослідити метод розпізнавання БПЛА на основі архітектури FOMO з backbone MobileNetV2, реалізувати повний цикл від навчання моделі до її розгортання на мікроконтролері ESP32-S3, та визначити оптимальні параметри квантизації для забезпечення балансу між точністю розпізнавання та апаратними обмеженнями крайового пристрою.

Для досягнення поставленої мети **сформульовано такі завдання**:

- проаналізувати наявні методи та архітектури нейронних мереж для виявлення та розпізнавання БПЛА на зображеннях;
- проаналізувати наявні платформи для розгортання моделей комп’ютерного зору та обрати цільову платформу;
- розробити та навчити FOMO модель на основі MobileNetV2 для класифікації БПЛА трьох категорій: інші БПЛА (Other_UAV), ударний БПЛА типу Shahed та птах (Bird);
- провести порівняльний аналіз архітектурних варіантів моделі за метриками F1-score та mAP@0.5;
- дослідити вплив роздільної здатності вхідного зображення на якість INT8 квантизації;
- розгорнути квантизовану модель на мікроконтролері ESP32-S3 та реалізувати веб-інтерфейс для тестування;
- оцінити практичну ефективність системи та визначити напрямки подальших досліджень.

Об’єкт дослідження – процес автоматичного розпізнавання та класифікації безпілотних літальних апаратів на зображеннях з використанням методів комп’ютерного зору.

Предмет дослідження – методи та моделі комп'ютерного зору на основі згорткових нейронних мереж для виявлення та розпізнавання БПЛА з їх розгортанням на крайових пристроях з обмеженими апаратними ресурсами.

У даній роботі застосовано комплексний підхід, що поєднує теоретичні та емпіричні методи. Для аналізу предметної області використано методи системного аналізу та огляду наукових джерел. Для розробки та оцінки моделей було застосовано методи машинного навчання: трансферне навчання на основі ImageNet-pretrained MobileNetV2, застосування focal loss для навчання на дисбалансованих класах, післятренувальна INT8 квантизація. Для оцінки якості моделей використано метрики F1-score (з centre-point tolerance) та mAP@0.5. Для розгортання системи було застосовано методи апаратного моделювання на платформі ESP32-S3 з використанням TensorFlow Lite Micro. Для статистичної оцінки ефективності квантизації було використано порівняльний аналіз точності розпізнавання по класах при різних роздільних здатностях вхідного зображення.

Наукова новизна одержаних результатів. Удосконалено підхід до квантизації моделей FOMO шляхом виявлення залежності між розміром вихідної сітки теплової карти та точністю розпізнавання специфічних класів БПЛА. Також удосконалено метод навчання моделі FOMO для розпізнавання БПЛА шляхом підбору оптимальної роздільної здатності вхідного зображення, що забезпечує збереження точності розпізнавання специфічних класів БПЛА після квантизації при розгортанні на крайовому пристрої.

Практичне значення одержаних результатів. Розроблену систему реалізовано у вигляді прототипу на базі мікроконтролера ESP32-S3 (вартість ~7 USD). Розпізнавання виконується безпосередньо на крайовому пристрої без підключення до інших зовнішніх допоміжних систем. Реалізовано веб-інтерфейс для дистанційного завантаження зображень та отримання результатів розпізнавання, що підтверджує правильну роботу системи з рівнем розпізнавання

Shahed та Other_UAV 85-95% на тестових зображеннях. Результати щодо залежності деградації квантизації від розміру сітки можуть бути використані розробниками систем Edge AI для вибору оптимальних параметрів квантизації моделей розпізнавання об’єктів. Прототип системи може бути застосований як складова частина комплексної системи розпізнавання БПЛА в задачах протиповітряної оборони країни.

Апробація результатів роботи. Результати досліджень, які безпосередньо пов’язані з кваліфікаційною роботою, оприлюднено на двох міжнародних науково-практичних конференціях:

- O. Zaritskyi, A. Miroshnyk. Machine learning in Counter-Unmanned Aircraft Systems. Information Technology and Implementation (Satellite) (IT&Is-2024). Kyiv, Ukraine, 2024. 100 p.
- O. Zaritskyi, A. Miroshnyk. A combined method for recognizing and intercepting unmanned aerial vehicles using machine learning methods. Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: XII Міжнародна науково-практична конференція. м. Запоріжжя, Україна, 2024. 127-131 с.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРОБЛЕМИ РОЗПІЗНАВАННЯ БПЛА

1.1. Аналіз сучасних загроз від БПЛА та необхідність автоматизованого розпізнавання

Застосування безпілотних літальних апаратів у сучасних збройних конфліктах набуло безпрецедентних масштабів. Відносно невисока собівартість виробництва та простота масового запуску перетворили БПЛА на один із ключових інструментів сучасної війни. БПЛА здатні ефективно виснажувати системи протиповітряної оборони противника. Масоване застосування БПЛА дозволяє атакувати об’єкти критичної та цивільної інфраструктури з мінімальними витратами порівняно з вартістю традиційних засобів ураження.

Досить широке застосування безпілотних літальних апаратів різних класів і типів в останніх збройних конфліктах призвело до значних матеріальних і людських втрат як серед військових, так і серед цивільного населення. Досвід військових дій в Україні в умовах застосування агресором досить великої кількості БПЛА різних модифікацій для нанесення шкоди критичній інфраструктурі та виснаження систем ППО дає підстави стверджувати про неефективність застосування зенітних керованих ракет для їх знищення з економічної точки зору, а в перспективі це може призвести до повного дефіциту зенітних керованих ракет для боротьби з крилатими та балістичними ракетами.

Особливе місце серед сучасних ударних БПЛА займає БПЛА типу Shahed. За класифікацією НАТО даний апарат відноситься до тактичних БПЛА другого класу (злітна маса 150-600 кг). На рис. 1.1 показано динаміку запусків БПЛА типу Shahed проти об’єктів критичної та цивільної інфраструктури під час повномасштабної війни в Україні. Аналіз базується на наборі даних “Massive Missile Attacks on Ukraine”, який містить інформацію про запущені та збиті ракети і безпілотники під час масованих ударів по інфраструктурі України з жовтня 2022 року [3].

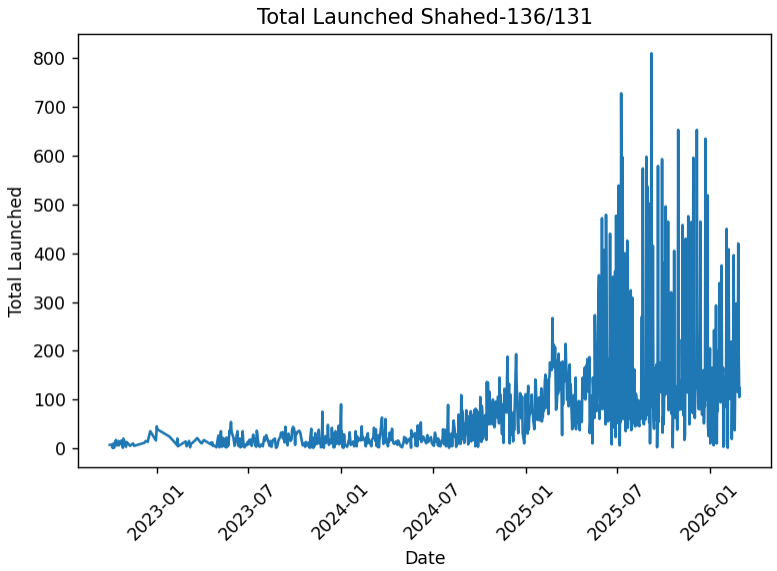

Рисунок 1.1 – Динаміка запусків БПЛА “Shahed” з жовтня 2022 року

З початку повномасштабного вторгнення в Україну було запущено близько 79000 БПЛА цього типу, як показано на рис. 1.1. Загальні тенденції дозволяють припустити, що кількість атак з використанням цього типу БПЛА продовжить зростати. Варто зазначити, що БПЛА також використовуються в комбінованих атаках з різними типами ракет як повітряного так і наземного базування для перевантаження систем протиповітряної оборони.

Окрім БПЛА типу Shahed існують й інші варіанти таких засобів. За класифікацією НАТО БПЛА поділяються на три основні класи залежно від злітної маси: Клас I (до 150 кг), Клас II (150-600 кг) та Клас III (понад 600 кг). Shahed-136 з масою 200 кг відноситься до Класу II [4]. Разом з ударними БПЛА широкого застосування набули розвідувальні БПЛА Класу I, які коригують вогонь та ведуть пошук зенітно-ракетних комплексів малої дальності на лінії зіткнення. Все це формує багаторівневу загрозу, що вимагає різноманітних засобів протидії БПЛА.

Традиційні методи виявлення та розпізнавання БПЛА мають суттєві обмеження. Радарні системи стикаються з малою ефективною площею розсіювання БПЛА Shahed-136 – дельтоподібна конструкція з переважно пластиковими

елементами значно ускладнює виявлення та розпізнавання. Акустичні системи ефективні лише на обмеженій відстані та залежать від рівня фонового шуму. Теплові сигнатури БПЛА з поршневими двигунами є відносно слабкими, що знижує ефективність інфрачервоних сенсорів [5]. Більшість атак ударними БПЛА відбуваються вночі, що суттєво ускладнює візуальне спостереження цих засобів. Таким чином, жоден з наявних методів не є повністю достатнім, що обумовлює необхідність розробки комплексних рішень виявлення та розпізнавання БПЛА.

В умовах описаних вище загроз методи комп'ютерного зору набувають особливої актуальності як доповнення до традиційних систем виявлення та розпізнавання. Візуальне розпізнавання БПЛА на зображеннях дозволяє використовувати досить недорогі камери як сенсори та виконувати класифікацію типу БПЛА, що неможливо для більшості радарів. Методи глибокого навчання демонструють найвищу ефективність серед усіх підходів до автоматичного виявлення БПЛА за візуальними даними, досягаючи точності понад 90% на стандартних тестових вибірках [6]. Розглядається чотири типи розпізнавання БПЛА: на основі радіочастотного сигналу, візуальних даних, акустичних даних та даних з радару – і робиться висновок про перспективність саме візуальних методів розпізнавання з огляду на їх масштабованість та вартість.

Проте більшість наявних систем розпізнавання БПЛА на основі глибокого навчання потребують використання потужних віддалених серверів обчислення або спеціалізованих відеокарт, що унеможливлює їх автономне польове застосування. Задача розгортання моделей розпізнавання на малопотужних крайових пристроях є актуальною і невирішеною дослідницькою проблемою [7, 8]. Більшість наявних методів розпізнавання БПЛА були розроблені без урахування жорстких апаратних обмежень реальних крайових пристроїв, таких як мікроконтролери [9]. Таким чином, існує дуже актуальна потреба у розробці систем розпізнавання БПЛА,

здатних функціонувати безпосередньо на досить дешевих пристроях без підключення до зовнішньої апаратної інфраструктури для обчислень.

Тому, виходячи з викладеного вище аналізу, було сформульовано основні вимоги до системи розпізнавання БПЛА, що розробляється в даній роботі:

- здатність класифікувати БПЛА типу Shahed та інші типи БПЛА з точністю не менше 85% після квантизації моделі;
- розгортання на мікроконтролері;
- вартість апаратної платформи для розгортання не більше 10 доларів США;
- автономна робота системи без підключення до зовнішньої апаратної обчислювальної інфраструктури.

1.2. Огляд наявних методів комп'ютерного зору для розпізнавання БПЛА

Методи комп'ютерного зору для розпізнавання БПЛА можна поділити на два основних напрямки: класичні методи обробки зображень та методи на основі глибокого навчання. Класичні методи, які засновані на ручному конструюванні ознак, поступово витісняються більш ефективними рішеннями з використанням глибокого навчання, хоча в певних сценаріях вони можуть зберігати свою актуальність [7].

До класичних підходів розпізнавання БПЛА відносяться методи виявлення рухомих об'єктів на основі різниці кадрів (frame differencing) та алгоритми виокремлення фону (background subtraction). Такі підходи ефективні для виявлення та розпізнавання БПЛА на статичному фоні, проте демонструють суттєве зниження якості при роботі з рухомою камерою або складним динамічним фоном. Метод гістограм орієнтованих градієнтів (HOG) у поєднанні з класифікатором SVM дозволяє виявляти та розпізнавати БПЛА за їх силуетом, однак цей метод вимагає ретельного підбору ознак та не узагальнюється на нові типи БПЛА [8].

Метод оптичного потоку (optical flow) застосовується для відстеження траєкторії руху БПЛА між відеокадрами. Алгоритм Lucas-Kanade та його модифікації забезпечують відстеження та розпізнавання в умовах незначного зміщення між кадрами, проте обчислювальна складність зростає зі збільшенням кількості об'єктів у кадрі. Незважаючи на відносну ефективність, класичні методи мають один спільний недолік – це особлива чутливість до умов освітлення та неможливість класифікації типу БПЛА без додаткового модуля розпізнавання [9].

Поява згорткових нейронних мереж (CNN) кардинально змінила підхід до виявлення та розпізнавання об'єктів на зображеннях. Сучасні архітектури розпізнавання поділяються на одноступеневі та двоступеневі. Двоступеневі архітектури, такі як Faster R-CNN, спочатку генерують регіони-кандидати, а вже потім класифікують їх. Такий підхід забезпечує високу точність, проте характеризується значною обчислювальною складністю, що обмежує їх застосування на малопотужних мікроконтролерах [7]. Одноступеневі архітектури розпізнавання вирішують задачу виявлення, розпізнавання та класифікації застосовуючи один єдиний прохід нейронної мережі. Серед них найбільшого поширення набула серія YOLO (You Only Look Once). Архітектура YOLOv8n – це найменша версія восьмого покоління може досягати mAP@0.5=53.9% на наборі даних COCO при 80 FPS на GPU [10]. Проте навіть компактні версії YOLO (YOLOv8n: 3.2M параметрів, 8.7 GFLOPs) вимагають ресурсів, що значно перевищують можливості типових мікроконтролерів. Зокрема, одна з найменших версій YOLO використовує близько 516 MB оперативної пам'яті [10], що неприйнятно для пристроїв з обмеженнями обчислювальних ресурсів у кілька мегабайт.

Архітектура SSD (Single Shot MultiBox Detector) у комбінації з MobileNet backbone забезпечує кращий баланс між точністю та швидкодією порівняно з архітектурою YOLO [11]. З MobileNet SSD FPN-Lite 320×320 можливо отримати

близько 3 FPS на Raspberry Pi 4, що недостатньо для роботи системи у реальному часі. EfficientDet-D0 демонструє точність mAP=33.8% при 10.2M параметрах, однак також орієнтований на пристрої з GPU [12].

У 2022 році компанія Edge Impulse представила принципово новий підхід до виявлення та розпізнавання об'єктів на мікроконтролерах – FOMO (Faster Objects, More Objects) [13]. Ключова ідея FOMO полягає у відмові від використання класичних bounding boxes на користь виявлення центроїдів об'єктів на основі теплових карт (heatmap). Замість використання регресії координат рамки модель формує повністю згортковий класифікатор, що застосовується до кожної комірки регулярної сітки, на яку поділяється вхідне зображення [13]. Принцип роботи FOMO базується на використанні MobileNetV2 як backbone мережі. За замовчуванням вхідне зображення зменшується у 8 разів при переході до вихідної теплової карти, тобто із зображення 96×96 отримуємо 12×12. Це реалізується відсіканням MobileNetV2 на шарі block_6_expand_relu. Вибір точки відсікання визначає компроміс між просторовою роздільною здатністю heatmap та обчислювальною складністю моделі [13]. FOMO вирішує лише задачу локалізації центроїдів об'єктів, відмовляючись від регресії розмірів, що суттєво знижує обчислювальну складність моделі. Після визначення центроїду об'єкта, рамки об'єктів можливо потім досить просто додати програмно.

Продуктивність FOMO на різних платформах є вражаючою: 60 FPS на Raspberry Pi 4 (160×160, MobileNetV2, alpha=0.1), 30 FPS на Arduino Nicla Vision (Cortex-M7, 96×96, 245 KB RAM), 14 FPS на Himax WE-I Plus DSP (96×96, до 150 KB RAM), 10 FPS на Cortex-M4F при 80 MHz (96×96, менше 100 KB RAM, alpha=0.05) [14]. Приклад для порівняння: використовуючи MobileNet SSD на Raspberry Pi 4 можливо досягти лише 2-3 FPS. Таким чином, FOMO забезпечує майже тридцятикратне прискорення обчислень у порівнянні з MobileNet SSD при значно менших вимогах до оперативної пам'яті [14].

Таблиця 1.1 – Порівняльний аналіз методів розпізнавання БПЛА

Метод/архітектура	Параметри, M	mAP@0.5	FPS (RPi4)	Мін. RAM
YOLOv8n	3.2	53.9% (COCO)	~15	>512 MB
MobileNet SSD FPN-Lite 320×320	3.4	~45% (COCO)	3	>256 MB
EfficientDet-D0	3.9	33.8% (COCO)	~5	>256 MB
FOMO (96×96, α=0.35)	~0.3	Залежить від набору даних	60	<1 MB
HOG + SVM	–	~70–80%	>30	<10 MB

Згідно таблиці 1.1, сучасні методи розпізнавання на основі нейронних мереж демонструють найвищу точність, однак їх вимоги до оперативної пам’яті унеможливлюють розгортання на мікроконтролерах з дуже малою кількістю оперативної пам’яті. FOMO є єдиним варіантом серед всіх розглянутих рішень, що здатний функціонувати в межах апаратних обмежень мікроконтролера ESP32-S3 за умови правильного підбору параметрів квантизації моделі комп’ютерного зору. Саме тому для подальшої роботи пропонується використання саме цієї архітектури.

Разом перевагами, FOMO має ряд принципових обмежень, які необхідно враховувати при проєктуванні системи. По-перше, відсутність bounding boxes означає неможливість визначення розміру об’єкта. По-друге, FOMO не здатний виявляти два об’єкти, центроїди яких потрапляють в одну комірку сітки. По-третє, якість виявлення та розпізнавання суттєво залежить від розміру вхідного зображення: чим менша роздільна здатність, тим грубішою є сітка і тим вища ймовірність суміщення центроїдів кількох об’єктів [14]. Зрештою, оригінальна архітектура FOMO від Edge Impulse не передбачає регресії координат центроїда та розміру об’єкта всередині комірки, що обмежує точність локалізації об’єкта.

Для задачі розпізнавання БПЛА у даній роботі обмеження щодо відсутності bounding boxes є некритичним: система призначена для виявлення наявності БПЛА в кадрі, розпізнавання та класифікації типу БПЛА та визначення їх приблизного положення в кадрі. Обмеження щодо суміщення центроїдів також є прийнятним, оскільки в типовому сценарії спостереження за БПЛА кількість БПЛА в одному кадрі є досить часто невеликою. Водночас, у даній роботі архітектура FOMO розширена трьома виходами: тепловою картою (heatmap), зміщенням центроїда всередині комірки (offset) та розміром об'єкта (size), що частково усуває зазначені вище обмеження та забезпечує точнішу локалізацію та розпізнавання БПЛА. Окрім того, у роботі досліджено спрощену архітектуру FOMO з одним виходом (heatmap), що дозволяє досягти максимальної швидкодії роботи на мікроконтролері.

Таким чином, аналіз наявних вже відомих методів виявлення та розпізнавання БПЛА за допомогою методів комп'ютерного зору показує, що архітектура FOMO на основі MobileNetV2 є оптимальним вибором для виконання задач даної роботи з огляду на обмеження цільової платформи мікроконтролера ESP32-S3.

1.3. Аналіз платформ для розгортання моделей на крайових пристроях

Вибір апаратної платформи для розгортання моделі комп'ютерного зору є критично важливим рішенням, що визначає архітектуру всієї системи. Для задачі виявлення та розпізнавання БПЛА у польових умовах платформа повинна відповідати вимогам, сформульованим у підрозділі 1.1: низька вартість, автономність, підтримка TensorFlow Lite Micro та наявність інтерфейсу для підключення камери.

TensorFlow Lite Micro (TFLM) – це менша версія TensorFlow Lite, розроблена Google для запуску моделей машинного навчання на мікроконтролерах з обсягом RAM від 16 KB [15]. На відміну від повної версії TFLite, TFLM не вимагає операційної системи, динамічного виділення пам'яті та файлової системи. Модель

завантажується як масив байтів (C header file) безпосередньо у Flash-пам’ять мікроконтролера, а для проміжних обчислень використовується статично виділений буфер – Tensor Arena [16].

TFLM підтримує обмежений набір операторів (ops), що відповідають типовим операціям CNN: Conv2D, DepthwiseConv2D, Add, Mul, Logistic та ін. Відсутність оператора у стандартній бібліотеці вимагає його ручної реалізації або відмови від відповідного шару при проєктуванні моделі. Компанія Espressif Systems розробила оптимізовану бібліотеку ESP-NN, що прискорює базові операції TFLM у 4-8 разів за рахунок SIMD-інструкцій процесора Xtensa LX7 [16].

Для обґрунтованого вибору цільової платформи проведено порівняльний аналіз основних варіантів: ESP32-S3, Raspberry Pi Zero 2W та Arduino Nano 33 BLE Sense. Характеристики платформ наведено у таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз платформ для розгортання моделей CV

Характеристика	**ESP32-S3 N16R8**	**RPi Zero 2W**	**Arduino Nano 33 BLE**	**RPi 5 (2GB)**	**Arduino Nicla Vision**
Процесор	Xtensa LX7 240 MHz dual-core	ARM Cortex-A53 1 GHz quad-core	ARM Cortex-M4 64 MHz	ARM Cortex-A76 2.4 GHz	ARM Cortex-M7 480 MHz
RAM	512 KB SRAM + 8 MB PSRAM	512 MB LPDDR2	256 KB SRAM	2 GB LPDDR4	1 MB SRAM
Flash / сховище даних	16 MB Flash	MicroSD	1 MB Flash	MicroSD	2 MB Flash

NPU / прискорювач	Hi (SIMD LX7)	Hi	Hi	Hailo-8L (опційно)	Hi
Підтримка камери	DVP / MIPI CSI	MIPI CSI	Hi (без модуля)	MIPI CSI	Вбудована
Споживання	~200 мА	120-160 мА	~20 мА	~3000 мА	~100 мА
Вартість (орієнт.)	~7 USD	~15 USD	~30 USD	~60 USD	~50 USD
ОС / середовище	FreeRTOS / Arduino	Linux (RPi OS)	Mbed OS / Arduino	Linux (RPi OS)	Arduino / OpenMV
Підтримка TFLM	+ (esp-tflite-micro)	+ (TFLite)	+ (обмежена)	+ (TFLite)	+ (Edge Impulse)

Arduino Nano 33 BLE Sense. Платформа на базі ARM Cortex-M4 з 256 KB SRAM є найменш придатною для задачі даної роботи. Обсяг RAM не дозволяє розмістити Tensor Arena розміром, необхідним для моделі з вхідним зображенням 128×128 пікселів. Відсутність вбудованого інтерфейсу камери та Wi-Fi ускладнює інтеграцію в реальну бойову систему розпізнавання.

Raspberry Pi Zero 2W. Платформа на базі quad-core ARM Cortex-A53 з 512 MB RAM забезпечує достатні ресурси для запуску повної версії TFLite у середовищі Linux. Завдяки наявності MIPI CSI інтерфейсу підтримується підключення камер серії Raspberry Pi Camera Module. Споживання під час операції розпізнавання (inference) становить 120-160 мА [17], що є прийнятним показником для стаціонарних застосувань з живленням від мережі. Проте вартість (~15 USD) вдвічі перевищує вартість ESP32-S3, а відсутність апаратної підтримки режимів енергозбереження ускладнює автономне живлення від батареї.

Raspberry Pi 5 з Hailo-8L NPU. Найпотужніша з розглянутих платформ (ARM Cortex-A76 з 2 GB RAM та встановленням опційного NPU Hailo-8L (13 TOPS)) здатна виконувати задачу розпізнавання в реальному часі при роздільній здатності відеопотоку 1080p. Однак висока вартість (~60 USD лише для плати без NPU) та споживання близько 3 Вт роблять цю платформу надлишковою для задачі виявлення та розпізнавання БПЛА на досить дешевих автономних системах.

ESP32-S3 як цільова платформа. ESP32-S3 від Espressif Systems є мікроконтролером на базі двоядерного процесора Xtensa LX7 з тактовою частотою 240 МГц. Ключовою перевагою для задач Edge AI є підтримка розширення векторних інструкцій (SIMD), що прискорює операції CNN у 4-8 разів порівняно зі стандартними мікроконтролерами без такої підтримки [16]. Версія N16R8 оснащена 16 MB Flash та 8 MB PSRAM, що дозволяє зберігати модель розміром до ~3 MB у Flash та виділяти достатній Tensor Arena у PSRAM. Espressif надає офіційну бібліотеку esp-tflite-micro з оптимізованими ядрами для Xtensa LX7, що суттєво спрощує розгортання TFLM моделей. Підтримка DVP та MIPI CSI інтерфейсів дозволяє підключати широкий спектр OV-сумісних камерних модулів. Інтегровані Wi-Fi 802.11 b/g/n та Bluetooth 5.0 забезпечують можливість передачі результатів розпізнавання через веб-інтерфейс без застосування зовнішніх комунікаційних модулів. Вартість модуля ESP32-S3 N16R8 WROOM в Україні становить близько 7 USD, що робить цей пристрій найдоступнішим з розглянутих варіантів.

Разом з перевагами ESP32-S3 має суттєві обмеження, які необхідно враховувати при проєктуванні системи. Відсутність виділеного NPU означає, що задача розпізнавання виконується виключно на CPU без апаратного прискорення операцій згортки. Внутрішня SRAM (512 KB) є недостатньою для великих моделей – Tensor Arena необхідно розміщувати у PSRAM, доступ до якої здійснюється через SPI-шину і цей варіант є повільнішим порівняно з використанням внутрішньої пам’яті [16]. За теоретичними оцінками, для квантизованих INT8 моделей класу

FOMO на Xtensa LX7 без NPU очікувана швидкодія становить порядку 1-2 секунди на кадр, що недостатньо для розпізнавання БПЛА у режимі реального часу [16]. Незважаючи на це обмеження, така швидкодія є прийнятною для задачі виявлення та розпізнавання БПЛА типу Shahed та інших типів. При швидкості БПЛА ~185 км/год та дальності спостереження 500 метрів він знаходиться у полі зору камери протягом декількох секунд, що залишає достатньо часу для виявлення та розпізнавання навіть при частоті розпізнавання менше 1 FPS. Точне вимірювання швидкодії та аналіз її впливу на якість виявлення та розпізнавання наводяться у наступних розділах.

Таким чином, за сукупністю критеріїв, а саме вартість, підтримка TFLM, наявний інтерфейс підключення камери та наявність вбудованого Wi-Fi, ESP32-S3 було обрано як цільову платформу для розгортання системи розпізнавання БПЛА.

1.4. Постановка задачі дослідження

На основі проведеного аналізу предметної області, наявних методів виявлення та розпізнавання БПЛА та можливостей цільової апаратної платформи сформульовано формальну постановку задачі дослідження.

Задача дослідження полягає у розробці та навчанні моделі нейронної мережі для автоматичного розпізнавання БПЛА на зображеннях, її конвертації у формат TFLite INT8 та розгортанні на мікроконтролері ESP32-S3 у вигляді автономної системи розпізнавання з веб-інтерфейсом.

Наукова новизна роботи полягає у виявленні залежності між розміром вихідної сітки теплової карти та точністю розпізнавання специфічних класів БПЛА; підборі оптимальної роздільної здатності вхідного зображення, що забезпечує збереження точності розпізнавання специфічних класів БПЛА після квантизації при розгортанні на крайовому пристрої.

Вхідними даними для розроблених моделей є RGB-зображення розміром 96×96 або 128×128 пікселів, які завантажуються через веб-інтерфейс. Зображення можуть містити один або декілька об’єктів наступних класів на фоні неба:

- Other_UAV – інші типи БПЛА (різні ударні та розвідувальні БПЛА);
- Shahed – ударні БПЛА типу Shahed-136/131 та їх аналоги (Герань-2);
- Bird – птахи як типовий клас хибнопозитивних спрацювань.

Для кожного вхідного зображення система повертає структуровану відповідь у форматі JSON, що містить: час розпізнавання (inference) на мікроконтролері (мс); список виявлених об’єктів з полями: клас (string), впевненість розпізнавання (float, 0-1); нормалізовані координати центроїда cx, cy та розміри bounding box w, h.

Для навчання та оцінки моделі використовується набір даних із зібраними зображеннями better_drones_dataset, що містить анотовані зображення БПЛА різних типів. Набір даних поділено на три підвибірки: train (80%), val (10%), test (10%). Набір даних містить три класи об’єктів: Other_UAV, Shahed та Bird. Зображення анотовані у форматі bounding box (PASCAL VOC), з яких при навчанні моделі FOMO обчислюються центроїди розпізнаних об’єктів.

На основі аналізу предметної області та апаратних обмежень цільової платформи сформульовано такі технічні вимоги:

1) Точність розпізнавання класу Shahed після INT8 квантизації – не менше 85% на тестовій вибірці (для моделі точного розпізнавання).
2) Загальний F1-score моделі на тестовій вибірці (Float32) – не менше 0.80 (для моделі точного розпізнавання) та не менше 0.60 (для моделі швидкого розпізнавання).
3) Розмір TFLite INT8 моделі – не більше 3 MB (обмеження розміру APP-розділу Flash пам’яті ESP32-S3).
4) Розмір Tensor Arena – не більше 3 MB (з урахуванням доступного обсягу PSRAM).

5) Функціональність веб-інтерфейсу: завантаження зображення через браузер, відображення результатів розпізнавання у форматі JSON та візуалізація bounding box.
6) Вартість апаратної частини прототипу – не більше 10 USD.

Для оцінки якості розробленої моделі використовуються такі метрики:

- F1-score з допуском centre-point tolerance (radius=1 комірка grid) – основна метрика для порівняння архітектурних варіантів;
- mAP@0.5 (mean Average Precision при IoU=0.5, 11-точкова інтерполяція PASCAL VOC) – для порівняння з результатами;
- точність розпізнавання по класах після INT8 квантизації – для оцінки деградації квантизації;
- час розпізнавання (inference) на ESP32-S3 (мс) – для оцінки практичної придатності системи.

Відповідно до загальної постановки задачі та технічних вимог визначено такі конкретні завдання дослідження:

1) Розробити архітектуру моделі FOMO на основі MobileNetV2 з multi-scale feature fusion та трьома виходами (heatmap, offset, size), а також спрощену архітектуру з одним виходом (heatmap) для досягнення максимальної швидкодії роботи на мікроконтролері.
2) Навчити та порівняти декілька архітектурних варіантів моделі з різними роздільними здатностями вхідного зображення (96×96, 128×128, 224×224) та значеннями alpha (0.35, 1.0).
3) Дослідити вплив роздільної здатності вхідного зображення на якість INT8 квантизації, зокрема на точність розпізнавання класу Shahed.
4) Реалізувати pipeline конвертації моделі у формат TFLite INT8 з balanced calibration dataset та перевірити відповідність розміру моделі вимогам Flash ESP32-S3.

5) Розгорнути квантизовану модель на мікроконтролері ESP32-S3 N16R8 з реалізацією веб-сервера для тестування розпізнавання БПЛА.
6) Провести порівняльне тестування всіх варіантів моделей та визначити оптимальний варіант для двох сценаріїв: сценарій високої точності розпізнавання та сценарій максимальної швидкодії розпізнавання на мікроконтролері.

Висновки до розділу 1

Отже, у першому розділі проведено комплексний аналіз предметної області задачі розпізнавання БПЛА методами комп'ютерного зору. Встановлено, що БПЛА типу Shahed становлять особливу загрозу для критичної інфраструктури України, а традиційні методи їх виявлення та розпізнавання мають суттєві обмеження, що обумовлює актуальність розробки систем візуального розпізнавання.

Огляд наявних методів розпізнавання показав, що архітектура FOMO на основі MobileNetV2 є єдиним практично придатним рішенням для розгортання на мікроконтролері ESP32-S3 з огляду на її мінімальні вимоги до оперативної пам'яті (менше 1 MB) та підтримку TensorFlow Lite Micro. За допомогою порівняльного аналізу платформ було обґрунтовано вибір ESP32-S3 N16R8 як цільової платформи завдяки оптимальному співвідношенню вартості (~7 USD), продуктивності та підтримки використання камери.

Було сформульовано формальну постановку задачі дослідження з визначенням вхідних і вихідних даних, технічних вимог та метрик оцінювання. Архітектура моделі та методика її навчання розглядаються у наступному розділі.

РОЗДІЛ 2. МОДЕЛЬ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ РОЗПІЗНАВАННЯ БПЛА

2.1. Архітектура MobileNetV2 як основа нейронної мережі

Вибір основи (backbone) архітектури є ключовим рішенням при проєктуванні моделі розпізнавання БПЛА. Backbone відповідає за екстракцію ознак з вхідного зображення і визначає більшу частину обчислювальної складності всієї нейронної мережі. Для задачі розпізнавання БПЛА на мікроконтролері ESP32-S3 у даній роботі обрано MobileNetV2 – архітектуру, спеціально розроблену для застосувань у моделях для розгортання на пристроях з обмеженими обчислювальними ресурсами.

Фундаментальним будівельним блоком MobileNet є згортка (depthwise separable convolution). Ідея полягає у розкладанні стандартної згортки на два послідовних кроки: depthwise convolution (просторова фільтрація кожного каналу незалежно від інших) та pointwise convolution (1×1 згортка для комбінування каналів між собою). Таке розкладання теоретично зменшує кількість операцій множення-накопичення (MAC) приблизно у k^2-разів, де k – це розмір ядра. Для ядра 3×3 це дає зниження обчислювальної складності у 8-9 разів порівняно зі стандартною згорткою при незначній втраті точності [18].

Архітектура MobileNetV2 вводить дві ключові інновації. Перша – це inverted residual block: залишкове з’єднання між вузькими bottleneck шарами, тоді як широке представлення використовується лише всередині блоку для нелінійних перетворень. Це дозволяє зменшити використання оперативної пам’яті під час операції розпізнавання, оскільки не потрібно одночасно зберігати широкі активаційні карти. Друга – це linear bottleneck: активаційна функція ReLU6, яка не застосовується після фінальної 1×1 згортки, оскільки нелінійність у низьковимірному просторі призводить до втрати інформації [19].

Схематично inverted residual block складається з трьох кроків: 1×1 pointwise згортка з expansion factor *t*, що розширює кількість каналів у *t* разів; 3×3 depthwise

згортка, що обробляє кожен канал незалежно; 1×1 pointwise згортка, що стискає канали назад до вихідної розмірності. Skip connection додається лише якщо вхідна і вихідна розмірності збігаються (рис. 2.1).

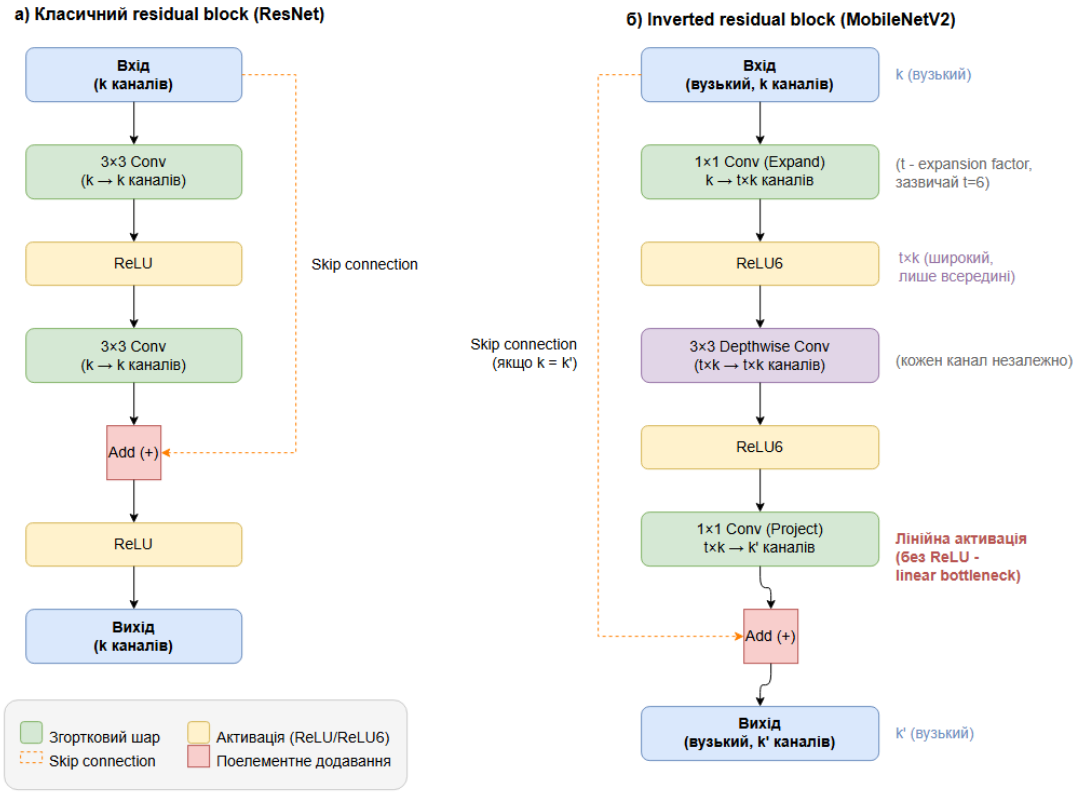

Рисунок 2.1 – Порівняння двох архітектур: а) класичний residual block (ResNet); б) inverted residual block (MobileNetV2)

MobileNetV2 підтримує масштабування ширини мережі через параметр alpha ∈ (0, 1]. При alpha < 1 кількість каналів у кожному шарі пропорційно зменшується: замість k каналів використовується ⌊alpha×k⌋ каналів. Кількість MAC-операцій та кількість параметрів масштабуються як $alpha^2$. Зменшення alpha дозволяє плавно регулювати різницю між обчислювальною складністю та виразністю ознак без зміни глибини мережі. Порівняння теоретичних характеристик backbone при різних значеннях alpha наведено у таблиці 2.1.

Таблиця 2.1 – Теоретичне порівняння MobileNetV2 при різних alpha

Alpha	**Кількість параметрів**	**Очікуваний розмір INT8 моделі**	**Відносна обчислювальна складність**
1.0	~2.5M	~2.5 MB	100%
0.35	~0.3M	~0.3-0.5 MB	~12%
0.1	~0.025M	~0.025-0.05 MB	~1%

Архітектура MobileNetV2 складається з послідовності inverted residual блоків, що поступово зменшують просторові розміри тензору. У таблиці 2.2 наведено ключові параметри архітектури для вхідного зображення 128×128 пікселів з alpha=1.0 разом з розмірами feature maps на кожному рівні.

Таблиця 2.2 – Архітектура MobileNetV2 для вхідного зображення 128×128 (alpha=1.0)

Блок	**Вхід (H×W×C)**	**Вихід (H×W×C)**	**Stride**	**MAC (млн)**	**Param (K)**
Conv2D 3×3	128×128×3	64×64×32	2	1.8	0.9
block_1 (×1, t=1)	64×64×32	64×64×16	1	2.4	0.9
block_2-3 (×2, t=6)	64×64×16	32×32×24	2 → 1	13.2	8.0
block_4-6 (×3, t=6)	32×32×24	16×16×32	2 → 1	15.2	15.1
block_7-10 (×4, t=6)	16×16×32	16×16×64	1	23.1	54.0
block_11-13 (×3, t=6)	16×16×64	16×16×96	1	29.4	117.0

block_14-16 (×3, t=6)	16×16×96	8×8×160	2 → 1	22.0	314.0
block_17 (×1, t=6)	8×8×160	8×8×320	1	10.8	473.0

При використанні MobileNetV2 як основи (backbone) для розпізнавання об’єктів мережа відсікається на певному проміжному шарі, а не використовується повністю. Вибір точки відсікання визначає розмір вихідної feature map та просторову роздільну здатність теплової карти розпізнавання. Шари глибшого рівня (block_10, block_13) містять більш абстрактні семантичні ознаки та мають менший просторовий розмір, тоді як шари раннього рівня (block_3, block_6) зберігають більше просторової інформації та менше абстрактних ознак. Детальний опис використання цих точок відсікання в архітектурах моделей наведено у наступних підрозділах 2.2 та 2.3.

2.2. Основна архітектура моделі FOMO для розпізнавання БПЛА

У даному підрозділі описано архітектуру моделі FOMO з multi-scale feature fusion. Використання даної архітектури розширює стандартний підхід до архітектури FOMO від Edge Impulse трьома виходами та окремим модулем об’єднання ознак різних масштабів. Дана архітектура є основою для чотирьох з п’яти розроблених варіантів моделі у даній роботі (моделі 1-4), що відрізняються роздільною здатністю вхідного зображення та значенням параметра alpha.

Архітектуру розроблених моделей описано як послідовність трьох етапів. На першому етапі вхідне RGB-зображення розміром H×H×3 подається на backbone MobileNetV2, який формує чотири проміжні feature maps у точках block_3, block_6, block_10 та block_13. На другому етапі feature maps об’єднуються модулем multi-scale fusion в один єдиний тензор. На третьому етапі три окремі згорткові голови

формують вихідні теплову карту (heatmap), зміщення центроїда (offset) та розмір об’єкта (size). Загальну схему архітектури моделі наведено на рисунку 2.2. Програмний код реалізації даної архітектури наведено у Додатку А.

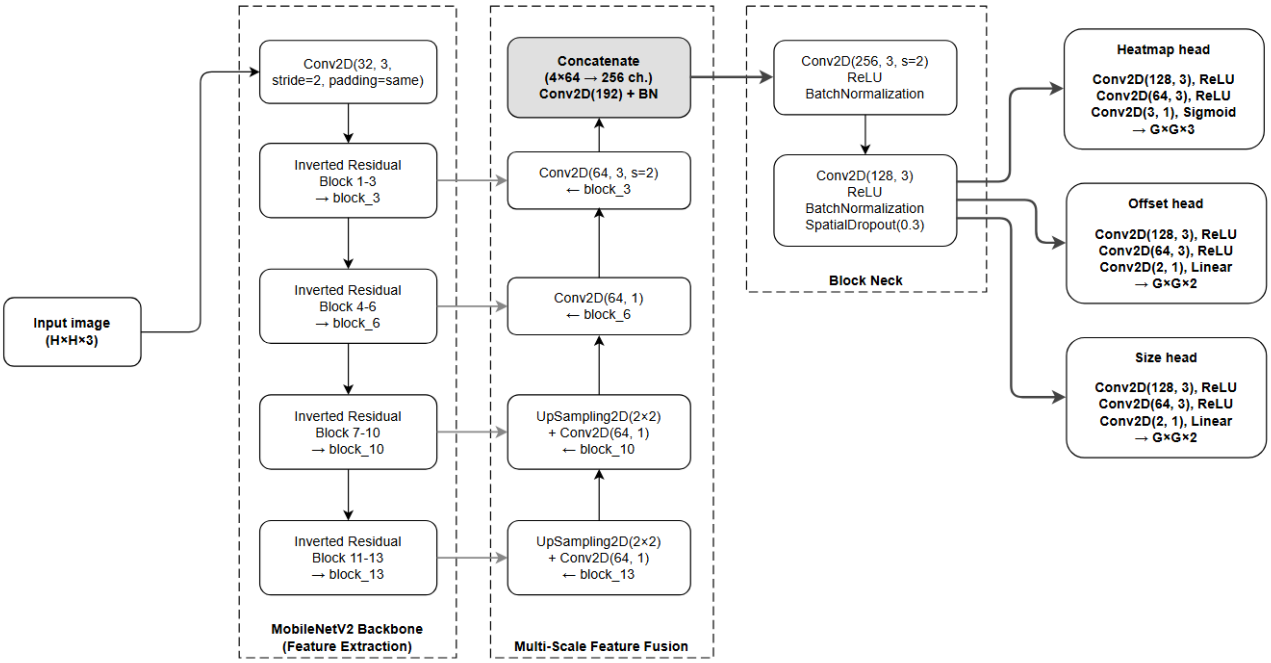

Рисунок 2.2 – Загальна архітектура моделі

Ключовою відмінністю розробленої архітектури від стандартної архітектури FOMO є використання multi-scale fusion замість відсікання на одному шарі. Чотири feature maps з різних рівнів backbone мають різні просторові розміри та кількість каналів, тому перед об’єднанням необхідне їх вирівнювання до спільного формату.

Детальніше опишемо процес fusion для розроблених моделей 1-4. Feature map з block_3 має найбільший просторовий розмір і стискається до цільового розміру операцією Conv2D зі stride=2 та 64 каналами. Feature map з block_6 вже має цільовий просторовий розмір і проходить через Conv2D 1×1 для приведення до 64 каналів. Feature maps з block_10 та block_13 мають менший просторовий розмір і розширюються операцією UpSampling2D(2×2) з подальшим Conv2D 1×1 до 64 каналів. Після вирівнювання всі чотири feature maps об’єднуються операцією

Concatenate по осі каналів, формуючи тензор з 256 каналами. Результуючий тензор обробляється Conv2D(192) з BatchNormalization для змішування отриманих ознак від різних масштабів [20]. Схему модуля multi-scale feature fusion наведено на рисунку 2.3.

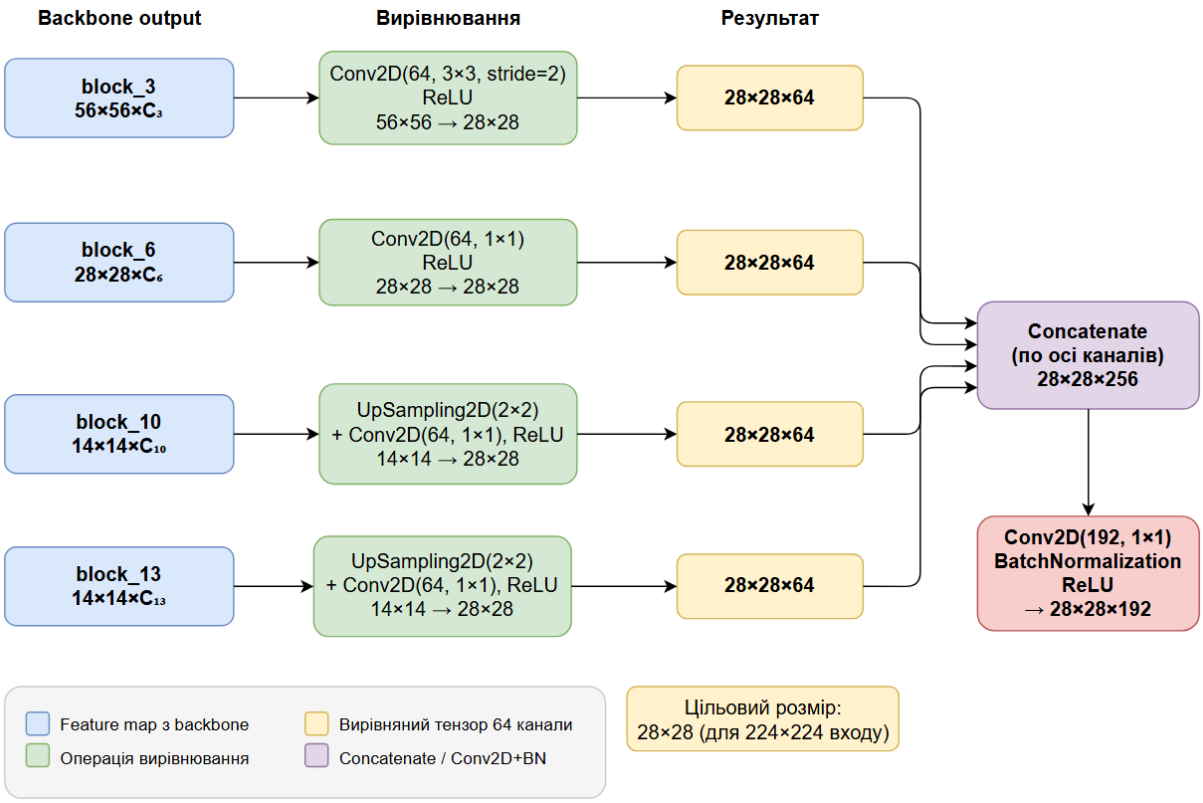

Рисунок 2.3 – Схема модуля multi-scale feature fusion (для вхідного зображення 224×224 пікселів)

Після операції Concatenate об’єднаний тензор обробляється двома послідовними згортковими шарами (neck), які формують спільне представлення для всіх трьох виходів. Перший шар Conv2D(256, kernel=3×3, stride=2) зменшує просторові розміри до розміру вихідної сітки Grid та встановлює глибину 256 каналів. Другий шар Conv2D(128, kernel=3×3, stride=1) зменшує глибину до 128 каналів без зміни просторових розмірів. Між шарами застосовується SpatialDropout2D(0.3) для регуляризації під час навчання [21]. Вихідний тензор neck

має розмір G×G×128, де G – розмір вихідної сітки (Grid), і є спільним входом для всіх трьох голів.

Перший вихід моделі – це теплова карта (heatmap) розміром G×G×NUM_CLASSES, де NUM_CLASSES=3 (Other_UAV, Shahed, Bird). Кожен елемент heatmap у позиції (i, j) для класу c містить значення від 0 до 1, що відображає ймовірність наявності центроїда об’єкта класу c у відповідній комірці вхідного зображення. Голова складається з двох Conv2D шарів (128 та 64 канали) з активацією ReLU, фінального Conv2D(NUM_CLASSES) та функції активації Sigmoid. Зміщення порогу bias_initializer=Constant(-4.0) застосовується для компенсації сильного дисбалансу між позитивними та негативними комірками на початку навчання.

Стандартна архітектура FOMO від Edge Impulse визначає лише факт наявності об’єкта у комірці, приписуючи центроїд до геометричного центру комірки. Розроблена модель розширює цей підхід виходом offset розміром G×G×2, що містить субпіксельне зміщення (Δx, Δy) центроїда відносно центру комірки у нормалізованих координатах у діапазоні [0, 1). Це дозволяє точніше локалізувати об’єкт всередині комірки сітки та є аналогом підходу CenterNet [22]. Голова складається з двох Conv2D шарів (128 та 64 канали) та фінального Conv2D(2) без функції активації.

Третій вихід (size розміром G×G×2) містить нормалізовані значення ширини (w) та висоти (h) bounding box об’єкта. Значення нормалізуються відносно розміру Grid, тобто w=1.0 відповідає ширині рівній одному розміру Grid. Цей вихід дозволяє реконструювати повний bounding box об’єкта для візуалізації у веб-інтерфейсі. Голова має таку саму структуру як і offset head.

Після отримання трьох вихідних тензорів виконується post-processing для формування фінального списку об’єктів, які було розпізнано. Спочатку до heatmap застосовується операція локального максимуму (local max pooling з вікном 3×3) для

виявлення піків, тобто комірок, що мають найвище значення серед своїх сусідів. Потім відфільтровуються комірки, де значення heatmap перевищує заданий Detection Threshold. Для відфільтрованих комірок (i, j, c) координати центроїда обчислюються як (формула 2.1):

$$cx = \frac{(j + offset[i, j, 0])}{G}, \qquad cy = \frac{(i + offset[i, j, 1])}{G} \tag{2.1}$$

де G – це розмір Grid, offset[i, j, 0] та offset[i, j, 1] – це субпіксельні зміщення по осях x та y відповідно.

Розмір bounding box визначається як (формула 2.2):

$$w = \frac{size[i, j, 0]}{G}, \qquad h = \frac{size[i, j, 1]}{G} \tag{2.2}$$

Результат обчислення повертається у форматі JSON з полями class, score, cx, cy, w, h для кожного розпізнавання БПЛА.

Детальну пошарову структуру архітектури розробленої моделі FOMO (моделі 1-4) наведено у таблиці 2.3 для прикладу для вхідного зображення роздільною здатністю 128×128 та значенням alpha=0.35.

Таблиця 2.3 – Пошарова архітектура моделі (вхід 128×128×3, alpha=0.35)

Компонент / шар	**Вхідний тензор**	**Вихідний тензор**	**Параметри**	**Призначення**
Input	–	128×128×3	–	Вхідне RGB зображення
MobileNetV2 block_1-3	128×128×3	32×32×C_3	alpha=0.35	Низькорівневі просторові ознаки

MobileNetV2 block_4-6	32×32×C_3	16×16×C_6	alpha=0.35	Проміжні ознаки
MobileNetV2 block_7-10	16×16×C_6	16×16×C_{10}	alpha=0.35	Семантичні ознаки середнього рівня
MobileNetV2 block_11-13	16×16×C_{10}	16×16×C_{13}	alpha=0.35	Високорівневі семантичні ознаки
Fusion alignment	Різні тензори	16×16×64	Conv2D + UpSampling	Вирівнювання до спільного розміру
Concatenate	4×(16×16×64)	16×16×256	axis=channel	Об’єднання multi-scale ознак
Fusion Conv2D + BN	16×16×256	16×16×192	kernel=1×1	Змішування ознак різних масштабів
Neck Conv2D + SpatialDropout	16×16×192	8×8×256	kernel=3×3, stride=2	Зменшення до розміру Grid
Neck Conv2D	8×8×256	8×8×128	kernel=3×3, stride=1	Формування спільного представлення
Head: heatmap (Conv2D×2 + Sigmoid)	8×8×128	8×8×3	bias_init=-4.0	Ймовірність наявності об’єкта класу c у комірці (i, j)
Head: offset (Conv2D×2)	8×8×128	8×8×2	без функції активації	Субпіксельне зміщення центроїда (Δx, Δy)
Head: size (Conv2D×2)	8×8×128	8×8×2	без функції активації	Нормалізований розмір об’єкта (w, h)

C_3, C_6, C_{10}, C_{13} – це кількість каналів у відповідних блоках MobileNetV2 при значенні alpha=0.35. Grid G=8 для вхідного зображення 128×128.

У ході роботи було загалом було розроблено п’ять архітектурних варіантів моделі розпізнавання БПЛА. Моделі 1-4 використовують описану вище архітектуру з multi-scale fusion та трьома виходами. А модель 5 використовує спрощену архітектуру, яка буде описана далі у наступному підрозділі. Систематизацію всіх варіантів розроблених моделей наведено у таблиці 2.4.

Таблиця 2.4 – Архітектурні варіанти розроблених моделей

Модель	**Вхід**	**Grid**	**Alpha / Архітектура**	**Обґрунтування варіанту вибору моделі**
Модель 1	224×224	14×14	1.0 / multi-scale	Еталонна модель з максимальною точністю для розгортання на потужному обладнанні
Модель 2	224×224	14×14	0.35 / multi-scale	Дослідження впливу значення alpha при збереженні роздільної здатності зображення
Модель 3	96×96	6×6	0.35 / multi-scale	Дослідження деградації INT8 квантизації для невеликої роздільної здатності зображення при малому значенні Grid
Модель 4	128×128	8×8	0.35 / multi-scale	Оптимальна точна модель для розгортання на мікроконтролері ESP32-S3
Модель 5	96×96	12×12	0.35 / block_6 early exit	Спрощена архітектура для максимальної швидкодії на мікроконтролері ESP32-S3

Вибір конкретної роздільної здатності та архітектури для кожної моделі обґрунтований необхідністю систематичного дослідження залежності між

параметрами квантизації та точністю розпізнавання БПЛА. Детальний порівняльний аналіз результатів навчання та квантизації всіх п’яти розроблених моделей наводиться у наступних розділах.

2.3. Спрощена архітектура моделі FOMO для розпізнавання БПЛА

Архітектура з multi-scale feature fusion, описана у попередньому підрозділі 2.2, забезпечує високу точність розпізнавання БПЛА, проте має суттєвий недолік для розгортання на мікроконтролері: значну обчислювальну складність через необхідність виконання операції розпізнавання для чотирьох гілок backbone та операцій UpSampling. У даному підрозділі описано альтернативну спрощену архітектуру моделі FOMO (модель 5), яка була додатково розроблена для досягнення максимальної швидкодії розпізнавання на мікроконтролері ESP32-S3 за рахунок свідомого спрощення архітектури.

Основним обмеженням мікроконтролера ESP32-S3 для задач комп’ютерного зору є відсутність апаратного прискорювача (NPU) та необхідність зберігати Tensor Arena у повільній зовнішній PSRAM. Збільшення кількості операцій в архітектурі нейронної мережі безпосередньо збільшує час самого розпізнавання БПЛА. Аналіз архітектури з multi-scale fusion показав, що основну частину обчислень займають шари backbone після block_6 (block_7-13), які при вхідному зображенні 96×96 не дають суттєвого покращення просторової роздільної здатності feature maps. Тому у моделі 5 застосовано підхід early exit – відсікання backbone одразу після block_6.

Стандартний підхід FOMO від компанії Edge Impulse також використовує відсікання на block_6 як основний варіант розгортання для мікроконтролерів [14]. При вхідному зображенні роздільною здатністю 96×96 пікселів block_6_expand_relu формує feature map розміром 12×12, що є достатньою просторовою роздільною здатністю для виявлення та розпізнавання БПЛА в типовому сценарії спостереження за цими об’єктами. Глибші шари backbone (block_7-13) не

збільшують просторову роздільну здатність (тобто залишається 12×12), а лише підвищують абстрактність ознак за рахунок використання значних обчислювальних ресурсів.

Другим спрощенням є відмова від offset та size голів на користь єдиного виходу – heatmap. Координати центроїда об’єкта визначаються як геометричний центр комірки сітки, де виявлено пік heatmap, а розмір bounding box фіксується рівним одній комірці сітки. Це суттєво зменшує кількість параметрів моделі та час post-processing на мікроконтролері. Обмеження точності локалізації БПЛА є прийнятним для задачі швидкого виявлення наявності та розпізнавання БПЛА в кадрі.

Для компенсації нижчої виразності спрощеної архітектури у моделі 5 застосовано покращену стратегію формування навчальної вибірки – використання гібридного набору даних. Замість простого масштабування (resize) зображень до роздільної здатності 96×96, що призводить до втрати деталей дрібних об’єктів, навчальна вибірка формується з двох типів патчів: resize-патчі (стандартне масштабування всього зображення) та crop-патчі (вирізання ділянки 96×96 навколо кожного об’єкта зі збереженням оригінального масштабу). Додатково до вибірки додаються hard negative патчі – це ділянки складного фону без об’єктів, що підвищує стійкість моделі до хибнопозитивних спрацювань. Детальний опис методики формування гібридного набору даних наведено у підрозділі 2.4.

Основні відмінності між архітектурою з multi-scale fusion (моделі 1-4) та спрощеною архітектурою (модель 5) систематизовано у таблиці 2.5.

Таблиця 2.5 – Порівняння архітектур розроблених моделей FOMO

Характеристика моделі	**Моделі 1-4 (multi-scale fusion)**	**Модель 5 (early exit)**	**Різниця між моделями**	**Висновок порівняння**
Точка відсікання backbone	block_3, 6, 10, 13	block_6	Використання меншої	Менший час необхідний

			кількості шарів	для розпізнавання
Feature fusion	Multi-scale (4 гілки)	Відсутній	Простіший граф	Менший Tensor Arena
Кількість виходів	3 (heatmap, offset, size)	1 (heatmap)	Використання меншої кількості голів	Простіший post-processing
Роздільна здатність вхідного зображення	Від 96×96 до 224×224	96×96	Використання однієї меншої роздільної здатності	Максимальна швидкість розпізнавання
Розмір Grid (при 96×96)	6×6	12×12	В 2 рази більший розмір Grid	Краща просторова роздільна здатність
Стратегія навчання моделей	Стандартний resize зображень	Використання гібридного набору даних	Більше патчів	Компенсація спрощення архітектури моделі

Детальну пошарову архітектуру спрощеної моделі FOMO наведено у таблиці 2.6 та зображено на рисунку 2.4. Програмний код реалізації даної спрощеної архітектури наведено у Додатку Б.

Таблиця 2.6 – Пошарова архітектура моделі 5 (вхід 96×96×3, alpha=0.35)

Компонент / шар	**Вхідний тензор**	**Вихідний тензор**	**Параметри**	**Призначення**
Input	–	96×96×3	–	Вхідне RGB зображення
MobileNetV2 (block_1-6)	96×96×3	12×12×C_6	alpha=0.35, early exit	Екстракція ознак до block_6
Conv2D(64, 1×1) + BN + ReLU	12×12×C_6	12×12×64	kernel=1×1	Приведення до фіксованої

				кількості каналів
SpatialDropout2D	12×12×64	12×12×64	rate=0.1	Регуляризація під час навчання
Conv2D(64, 3×3) + ReLU	12×12×64	12×12×64	kernel=3×3, padding=same	Просторова обробка ознак
Conv2D(NUM_CLASSES, 1×1) + Sigmoid	12×12×64	12×12×3	bias_init=-4.0	Heatmap: ймовірність наявності об'єкта

C_3 – це кількість каналів block_6 при alpha=0.35. Grid G=12 для вхідного зображення 96×96.

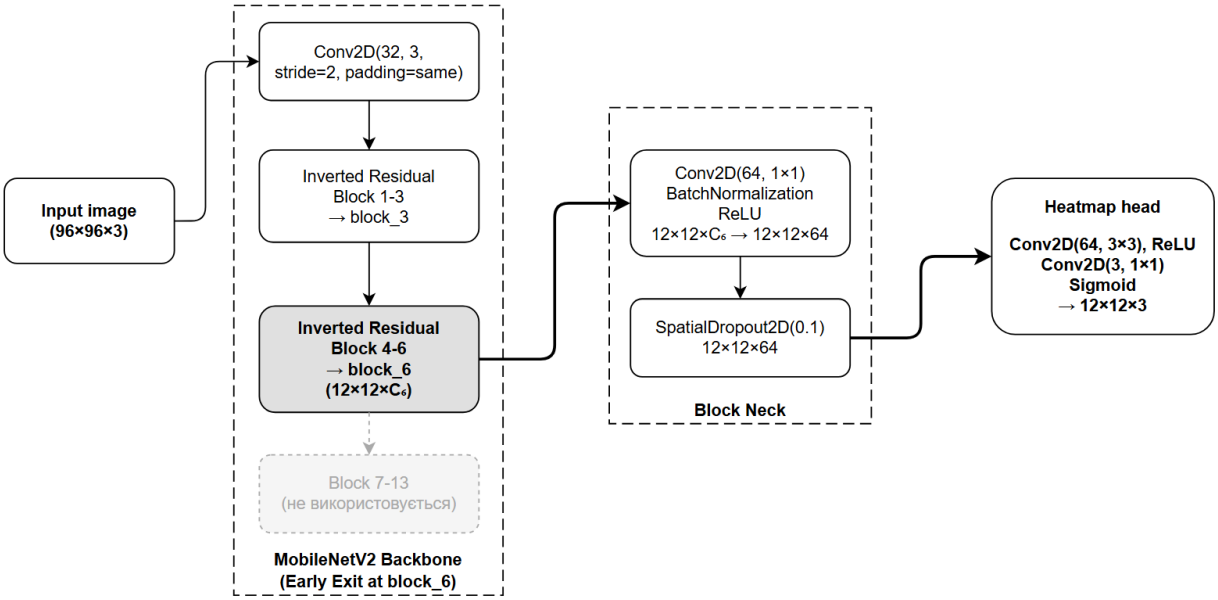

Рисунок 2.4 – Архітектура спрощеної моделі (модель 5)

Важливою перевагою моделі 5 порівняно з іншими моделями при роздільній здатності вхідного зображення 96×96 є більший розмір вихідної сітки. Завдяки відсіканню на block_6 (замість проходження через neck з stride=2 як у моделях 1-4) вихідна сітка має розмір 12×12 замість 6×6. Це означає, що кожна комірка сітки

відповідає ділянці 8×8 пікселів вхідного зображення, що забезпечує кращу просторову роздільну здатність при локалізації БПЛА. Методика навчання моделі та функції втрат описані у наступному підрозділі 2.4.

2.4. Функції втрат та методика навчання моделей

Навчання розроблених моделей розпізнавання БПЛА вимагає спеціалізованих функцій втрат, що відповідають структурі виходів архітектури моделі FOMO. У даному підрозділі описано функції втрат для кожного виходу моделі, методику кодування ground truth, оптимізатор та стратегію аугментації навчальних даних.

Для кодування позицій об’єктів у вигляді теплової карти використовується двовимірний гауссовий розподіл (Gaussian heatmap encoding). Підхід широко застосовується у сучасних системах anchor-free розпізнавання об’єктів [23]. Для кожного об’єкта класу c з центроїдом у позиції (cx, cy) на сітці grid будується двовимірна гауссіана навколо відповідної комірки. Значення гауссіани у позиції (i, j) визначається за формулою 2.3:

$$G(i,j) = \exp(-((i - cy)^2 + (j - cx)^2) \,/\, (2\sigma^2)) \qquad (2.3)$$

де σ – це параметр розмитості гауссіани (sigma), що визначає ширину піку. При σ=2.0 для моделей 1-4 та σ=0.5 для моделі 5 радіус гауссіани становить 3σ комірки. Це забезпечує плавний перехід між позитивними та негативними комірками. Значення heatmap обмежуються діапазоном [0, 1]. Якщо гауссіани двох об’єктів перекриваються, то у відповідну комірку записується максимальне значення з двох гауссіан.

Основною проблемою навчання heatmap детекторів є суттєвий дисбаланс між кількістю позитивних та негативних комірок. Для зображення з одним об’єктом та сіткою 8×8 лише одна комірка є позитивною, тоді як 63 комірки є негативними. Для

вирішення цієї проблеми використовується функція Focal Loss. Ця функція пригнічує вагу так званих легких прикладів та фокусує навчання на важких прикладах [24]. У даній роботі застосовано модифіковану функцію Focal Loss, яка адаптована для heatmap детекцій (формула 2.4):

$$L_{hm} = -(\frac{1}{N}) \sum \begin{matrix} [\, pos_{mask} \cdot (1-\hat{y})^{\alpha} \cdot \log(\hat{y}) + \\ + neg_{mask} \cdot (1-y)^{\beta} \cdot \hat{y}^{\alpha} \cdot \log(1-\hat{y}) \,] \end{matrix} \quad (2.4)$$

де $\hat{y}$ – це передбачене значення heatmap;

y – це ground truth значення;

N – це кількість позитивних комірок;

α=2.0 та β=4.0 – це гіперпараметри;

pos_{mask} = 1 якщо y ≥ 0.99 (центр об'єкта), neg_{mask} = 1-pos_{mask}.

Параметр β для негативних прикладів додатково зменшує вагу комірок, там де ground truth значення близьке до 1 (периферія гауссіани). Зміщення порогу $bias_{initializer}$=Constant(−4.0) у фінальному шарі heatmap head компенсує дисбаланс класів на початку навчання.

Для навчання offset та size голів (для моделей 1-4) використовується функція $L1_{loss}$ з маскуванням. Помилка обчислюється лише для комірок, де знаходяться центроїди об'єктів. Обчислюється за формулами 2.5 та 2.6.

$$L_{off} = \left(\frac{1}{N}\right) \sum_{(i,j) \in positive} off_{pred}[i,j] - off_{true}[i,j] \quad (2.5)$$

$$L_{size} = \left(\frac{1}{N}\right) \sum_{(i,j) \in positive} sz_{pred}[i,j] - sz_{true}[i,j] \quad (2.6)$$

де positive – це множина комірок, що містять центроїди об’єктів;

N – це кількість таких комірок.

Загальна функція втрат для розроблених моделей 1-4 є зваженою сумою трьох компонент (формула 2.7):

$$L = w_{hm} \cdot L_{hm} + w_{off} \cdot L_{off} + w_{size} \cdot L_{size} \quad (2.7)$$

де використовуються ваги зі значеннями w_{hm}=1.0, w_{off}=1.0, w_{size}=0.1. Мале значення w_{size}=0.1 обумовлено тим, що size head є допоміжною частиною. Точність визначення розміру об’єкта є менш критичною ніж точність локалізації та класифікації об’єктів. Для розробленої моделі 5 функція втрат складається лише з компоненти L_{hm}.

Для навчання всіх моделей використовується оптимізатор Adam. Порівняльний аналіз оптимізаторів підтверджує, що оптимізатор Adam залишається найефективнішим вибором для задач глибокого навчання, новіші методи не демонструють стабільної переваги над ним [25]. Початкова швидкість навчання: $\eta_0 = 5 \times 10^{-5}$. Для поступового зменшення швидкості навчання застосовується техніка планування Cosine Decay [26] (формула 2.8):

$$\eta(t) = \eta_{min} + 0.5 \cdot (\eta_0 - \eta_{min}) \cdot (1 + \cos(\pi t \,/\, T)) \quad (2.8)$$

де t – це поточний крок навчання;

T – це загальна кількість кроків навчання;

$\eta_{min} = \eta_0 \cdot 10^{-6}$ – це мінімальна швидкість навчання.

Cosine Decay забезпечує плавне зменшення швидкості навчання без різких стрибків. Це важливо при fine-tuning MobileNetV2 з претренованими вагами ImageNet. Для регуляризації застосовується L2 weight decay з коефіцієнтом $\lambda=10^{-4}$.

Всі моделі використовують backbone MobileNetV2 з претренованими вагами ImageNet. Для fine-tuning застосовується часткове заморожування backbone: перші 60% шарів заморожуються (ваги не оновлюються), решта 40% шарів навчаються з виключенням шарів BatchNormalization (їх ваги залишаються фіксованими для стабільності метрик). Такий підхід дозволяє зберегти низькорівневі ознаки, що добре переносяться між задачами (краї, текстури об'єктів) та адаптувати глибші шари до специфіки зображень БПЛА.

Для підвищення узагальнюючої здатності розроблених моделей та зменшення overfitting застосовується набір аугментацій зображень. Параметри аугментацій наведено у таблиці 2.7.

Таблиця 2.7 – Аугментація навчальних даних

Аугментація	**Параметри**	**Застосування**	**Обґрунтування застосування**
Горизонтальне відображення (flip)	p=0.5	Всі моделі	БПЛА можуть летіти в обох напрямках
Випадкова яскравість	max_delta=0.2	Всі моделі	Різні умови освітлення протягом доби
Випадковий контраст	[0.8, 1.2]	Всі моделі	Варіації атмосферних умов
Випадковий відтінок (hue)	max_delta=0.05	Модель 5	Компенсація колірних відхилень камери
Gaussian noise	$\sigma \in [0.01, 0.04]$	Модель 5	Симуляція шуму матриці камери
Motion blur	kernel 3×3 або 5×5, p=0.2	Модель 5	Симуляція розмиття при русі БПЛА
Grayscale conversion	p=0.07	Модель 5	Підвищення стійкості до монохромності

Для навчання моделі 5 було розроблено спеціальну стратегію формування навчальної вибірки, а саме використання гібридного набору даних, що поєднує три різних типи патчів:

- Resize-патчі – стандартне масштабування всього зображення до 96×96 пікселів. Зберігає загальний контекст зображення, але зменшує деталі дрібних об’єктів.
- Crop-патчі – вирізання ділянки 96×96 пікселів навколо центроїда кожного з об’єктів зі збереженням оригінального масштабу. При навчанні застосовується випадковий зсув центру crop до ±20% розміру патча для аугментації позиції об’єкта в самому кадрі.
- Hard negative патчі – вирізання ділянок 96×96 пікселів зі складним фоном (хмари, дерева, будівлі) без об’єктів БПЛА. Дистанція від центру патча до найближчого об’єкта перевищує дистанцію 0.8×96 пікселів. Частка hard negatives у вибірці становить 25%.

Для валідаційної та тестової вибірок використовується лише resize-патчі без аугментації. Це забезпечує правильну та відтворювану оцінку якості моделі. Навчання зупиняється за правилом early stopping: якщо F1-score на валідаційній вибірці не покращується протягом 15 епох, то навчання припиняється і відновлюються ваги найкращої епохи.

Навчання проводиться з розміром міні-батчу 16 зображень. Кількість епох обмежена 200 епохами для моделей 1-4 та 100 епохами для моделі 5. Попереднє тренування backbone на ImageNet значно прискорює збіжність: більшість моделей досягають оптимального F1-score протягом 60-80 епох. Метрики оцінки якості навчання моделей описані далі у підрозділі 2.5.

2.5. Метрики оцінки якості моделей розпізнавання БПЛА

Для об’єктивного порівняння розроблених архітектурних варіантів та оцінки придатності моделей до розгортання на мікроконтролері ESP32-S3 у даній роботі використовується конкретний набір певних метрик. Вибір метрик обумовлений особливостями архітектури FOMO, а саме виходами у вигляді теплової карти замість класичних bounding boxes.

Основною метрикою для порівняння архітектурних варіантів є F1-score з допуском центроїда. Оскільки модель FOMO виявляє центроїди об’єктів на сітці grid, а не повноцінні bounding boxes, то стандартна метрика IoU (Intersection over Union) не підходить для моделей FOMO. Натомість розпізнавання об’єкта вважається правильним (True Positive), якщо передбачений центроїд знаходиться в межах 1 комірки grid від реального центроїда об’єкта.

Розглянуто алгоритм обчислення F1-score. Спочатку до передбаченої heatmap застосовується операція локального максимуму (local max pooling з вікном 3×3) для пошуку піків, тобто комірок, що є локальними максимумами серед своїх сусідів. Далі відфільтровуються комірки, де значення heatmap перевищує Detection Threshold. Для кожного передбаченого піку (py, px) перевіряється наявність реального об’єкта (ty, tx) (формула 2.9):

$$(py - ty) \leq 1; (px - tx) \leq 1 \tag{2.9}$$

Якщо умова виконується, то розпізнавання об’єкта вважається як True Positive (TP). Якщо передбачений пік не знаходить відповідного реального об’єкта – це False Positive (FP). Реальні об’єкти без відповідного передбаченого піку – False Negative (FN). Кожен реальний об’єкт може бути зіставлений лише з одним передбаченим піком.

На основі підрахованих TP, FP, FN обчислюються метрика точності (Precision), метрика повноти (Recall) та вже сама метрика F1-score (формули 2.10, 2.11, 2.12):

$$Precision = \frac{TP}{(TP + FP)} \tag{2.10}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{2.11}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{(Precision + Recall)} \tag{2.12}$$

Метрика F1-score обчислюється як гармонічне середнє між Precision та Recall та забезпечує збалансовану оцінку якості розробленої моделі, що враховує як хибні спрацювання, так і пропущені об’єкти. Метрика обчислюється глобально по всіх класах та зображеннях тестової вибірки набору даних.

Для порівняння результатів з іншими дослідженнями, що використовують стандартні метрики розпізнавання, застосовується метрика mAP@0.5 – середня точність при порозі IoU=0.5 [27]. На відміну від метрики F1-score, що використовує фіксований Detection Threshold, метрика mAP враховує метрику впевненості передбачень розпізнавань та будує криву Precision-Recall при різних можливих порогах значень.

Для обчислення метрики mAP@0.5 у даній роботі використовується розмір bounding box з size head (для моделей 1-4) або фіксований розмір однієї комірки grid (для моделі 5). Передбачений bounding box вважається правильним, якщо значення IoU з реальним bounding box перевищує 0.5. AP для кожного класу обчислюється

методом 11-точкової інтерполяції (PASCAL VOC). Фінальна метрика mAP@0.5 є середнім значенням AP по всіх активних класах моделі (Other_UAV, Shahed, Bird).

Для того, щоб зменшити розмір моделі та розгорнути її на мікроконтролері ESP32-S3 необхідно виконати операцію квантизації моделі. Для дослідження впливу INT8 квантизації на якість розпізнавання окремих класів навченої моделі використовується метрика деградації квантизації по класах. Метрика обраховується як відносна зміна точності розпізнавання класу c після квантизації порівняно з повноцінною Float32 моделлю (формула 2.13):

$$\Delta_c = \frac{(Accuracy_c(Float32) - Accuracy_c(INT8))}{Accuracy_c(Float32)} \times 100\% \qquad (2.13)$$

де $Accuracy_c(Float32)$ та $Accuracy_c(INT8)$ – це точність розпізнавання класу c для повної Float32 моделі та квантизованої INT8 моделі відповідно. Висока деградація розпізнавання окремого класу ($\Delta_c > 20\%$) свідчить про те, що динамічний діапазон активацій для даного класу є недостатнім для коректного представлення моделі у форматі INT8. Ця метрика є ключовою для дослідження залежності між розміром grid та деградацією розпізнавання окремого класу, наприклад класу з БПЛА типу Shahed.

Для оцінки практичної придатності розробленої системи розпізнавання БПЛА вимірюється час виконання однієї операції розпізнавання (inference) на мікроконтролері ESP32-S3. Вимірювання проводиться функцією millis() з точністю 1 мс. Метрика визначає максимальний FPS системи (формула 2.14):

$$FPS = \frac{1000}{T_{inference}} \qquad (2.14)$$

де $T_{inference}$ – це час розпізнавання (inference), розрахований в мілісекундах.

У таблиці 2.8 наведено зведену характеристику усіх метрик, що використовуються для оцінки моделей у даній роботі.

Таблиця 2.8 – Метрики оцінки якості моделей

Метрика	**Позначення**	**Застосування**	**Призначення**
F1-score (centre-point tolerance)	F1	Всі моделі	Основна метрика порівняння архітектур моделей під час навчання
mAP@0.5 (PASCAL VOC)	mAP@0.5	Моделі 1-4	Порівняння з результатами інших досліджень
Деградація квантизації по класах	Δ_c (%)	Всі моделі	Дослідження впливу INT8 квантизації на точність розпізнавання класів
Час розпізнавання (inference) на ESP32-S3	$T_{inference}$ (мс)	Моделі 4, 5	Оцінка практичного застосування моделей

Отже, комплексне використання описаних метрик дозволяє всебічно оцінити розроблені моделі комп’ютерного зору: F1-score характеризує якість розпізнавання, метрика деградації квантизації оцінює придатність моделі до розгортання на мікроконтролері ESP32-S3, а час розпізнавання (inference) визначає чи можливо використовувати систему на практиці. Результати оцінки всіх розроблених п’яти моделей за наведеними метриками наводяться у наступних розділах.

2.6. Порівняльний аналіз архітектурних варіантів розроблених моделей

Розроблені п’ять варіантів моделей FOMO утворюють систематичний план дослідження, де кожна модель перевіряє окрему наукову гіпотезу. Вибір параметрів кожного з варіантів обґрунтовано результатами аналізу предметної області та теоретичними характеристиками розроблених архітектур, які було описано у попередніх підрозділах. У таблиці 2.9 наведено теоретичні характеристики всіх п’яти моделей.

Таблиця 2.9 – Теоретичні характеристики архітектурних варіантів моделей

Модель	**Вхід (пікс.)**	**Alpha**	**Grid**	**Ви-ходи**	**Param. backbone (approx.)**	**Розмір INT8 (approx.)**	**Наукова гіпотеза**
Модель 1	224×224	1.0	14×14	3	~2.5M	~3-4 MB	Еталонна точність моделі
Модель 2	224×224	0.35	14×14	3	~0.3M	~0.5-1 MB	Вплив зменшення параметра alpha при збереженні роздільної здатності
Модель 3	96×96	0.35	6×6	3	~0.3M	~0.5-1 MB	Перевірка деградації квантизації при малому розмірі Grid (6×6)
Модель 4	128×128	0.35	8×8	3	~0.3M	~0.5-1 MB	Оптимальний Grid для правильної квантизації

Модель 5	96×96	0.35	12×12	1	~0.1M	~0.3 MB	Максимальний FPS при прийнятній точності розпізнавання

Моделі 1 та 2 мають однакову роздільну здатність зображення 224×224 та однакову архітектуру multi-scale fusion, відрізняючись лише значенням alpha (1.0 та 0.35 відповідно). Порівняння цих двох моделей дозволяє точно оцінити вплив ширини backbone на якість розпізнавання БПЛА. Теоретично, зменшення alpha з 1.0 до 0.35 скорочує кількість параметрів backbone приблизно у 8 разів при очікуваній незначній втраті точності. Якщо гіпотеза підтвердиться, то це обґрунтовує вибір alpha=0.35 для всіх наступних моделей як оптимального балансу між точністю розпізнавання та розміром моделі.

Моделі 2 та 3 мають однакове значення параметру alpha=0.35 та архітектуру multi-scale fusion, відрізняючись лише роздільною здатністю зображення (224×224 та 96×96) і відповідно розміром grid (14×14 та 6×6). Зменшення роздільної здатності є необхідним кроком для розгортання на мікроконтролері ESP32-S3, оскільки це суттєво скорочує час розпізнавання (inference). Необхідно перевірити: чи призводить зменшення розміру grid з 14×14 до 6×6 до диференційованої деградації точності розпізнавання різних класів після застосування INT8 квантизації? Класи з вузьким динамічним діапазоном активацій (зокрема Shahed) можуть бути більш чутливими до зменшення кількості комірок grid ніж інші класи.

Якщо модель 3 підтвердить значну деградацію класу Shahed при grid 6×6, то модель 4 з grid 8×8 (вхід 128×128) перевіряє гіпотезу про існування мінімально достатнього розміру grid для правильної INT8 квантизації. Вибір роздільної здатності 128×128 як проміжного розміру між 96×96 та 224×224 обґрунтований тим, що при alpha=0.35 та grid 8×8 очікуваний розмір Tensor Arena залишається в межах

доступної PSRAM мікроконтролера ESP32-S3, тоді як час розпізнавання (inference) є теоретично прийнятним.

Модель 5 перевіряє гіпотезу про можливість досягнення прийнятної точності розпізнавання БПЛА за рахунок спрощення архітектури (early exit, один вихід) та покращення якості навчальних даних (використання гібридного набору даних). На відміну від моделей 3 та 4, де обчислювальна складність визначається глибиною backbone та операціями fusion, у моделі 5 основний акцент зроблено на оптимізацію навчання. Більший розмір grid 12×12 (порівняно з 6×6 у моделі 3) при тому самому вхідному зображенні 96×96 є додатковою перевагою, що теоретично підвищує просторову роздільну здатність розробленої системи. Перевірка сформульованих теоретичних тверджень та гіпотез здійснюється шляхом навчання, квантизації та тестування всіх п’яти моделей. Це детально описано у наступному розділі.

Висновки до розділу 2

Отже, у другому розділі було розроблено та теоретично обґрунтовано п’ять архітектурних варіантів моделі FOMO для розпізнавання БПЛА. Всі моделі базуються на використанні backbone MobileNetV2 з попередньо натренованими вагами ImageNet та використовують heatmap-based підхід до розпізнавання об’єктів.

Моделі 1-4 використовують архітектуру з multi-scale feature fusion з чотирьох точок виходу backbone та трьома виходами (heatmap, offset, size). Дана архітектура розширює стандартний підхід FOMO від Edge Impulse, забезпечуючи точнішу локалізацію об’єктів. Натомість модель 5 використовує спрощену архітектуру з відсіканням на block_6 та одним heatmap виходом для досягнення мінімального часу розпізнавання (inference) та більшого FPS на мікроконтролері ESP32-S3. Також визначено набір метрик оцінювання якості моделі: F1-score, mAP@0.5, метрика деградації квантизації та час розпізнавання (inference) на ESP32-S3. Сформульовано гіпотези щодо впливу архітектурних параметрів на якість розпізнавання БПЛА.

РОЗДІЛ 3. МОДЕЛЮВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ БПЛА

3.1. Інструментальні засоби та середовище розробки

Реалізація системи розпізнавання БПЛА складається з двох основних етапів: навчання та конвертація моделей на персональному комп'ютері з використанням GPU та розгортання квантизованих моделей на мікроконтролері ESP32-S3. Для кожного з етапів використовується окремий набір інструментальних засобів.

Навчання всіх п'яти моделей виконується на персональному комп'ютері під управлінням операційної системи Windows 11 з використанням GPU NVIDIA GeForce GTX 1650 Ti для апаратного прискорення процесу тренування всіх моделей. Основним інструментом розробки є мова програмування Python у поєднанні з фреймворком TensorFlow. Це провідний фреймворк для побудови та навчання нейронних мереж [28]. Keras API, що входить до складу фреймворку TensorFlow, використовується для визначення та опису архітектури моделей. TFLite Converter, що також входить до складу TensorFlow, забезпечує конвертацію навчених Keras моделей у формат TFLite INT8 для подальшого розгортання на мікроконтролері ESP32-S3.

Для роботи з зображеннями набору даних використовується бібліотека Pillow (PIL), що забезпечує завантаження, конвертацію та масштабування зображень. Матричні обчислення при реалізації функцій Gaussian heatmap encoding та метрик оцінювання виконуються з використанням бібліотеки NumPy. Для побудови графіків навчання (loss curves, F1-score, Precision-Recall curves) використовується бібліотека Matplotlib. Зчитування конфігурації класів з файлу data.yaml набору даних виконується за допомогою бібліотеки YAML. Для спостереження за прогресом процесу навчання моделей використовується бібліотека tqdm.

Для розгортання квантизованих TFLite INT8 моделей на мікроконтролері ESP32-S3 використовується програмне забезпечення Arduino IDE. Це інтегроване

середовище розробки для компіляції та прошивки Arduino-сумісних пристроїв. Підтримка мікроконтролерів ESP32/ESP32-S3 забезпечується офіційним board support package від Espressif Systems, що містить драйвери для Wi-Fi та камери. Запуск TFLite INT8 моделей на мікроконтролері реалізовано через бібліотеку esp-tflite-micro [29]. Це офіційна реалізацію TensorFlow Lite Micro від Espressif з оптимізованими ядрами згортки для процесора Xtensa LX7. Бібліотека esp-tflite-micro використовує апаратні SIMD-інструкції процесора ESP32-S3 для прискорення операцій CNN приблизно у 4-8 разів порівняно зі стандартною реалізацією TFLM.

Конвертація навченої TFLite моделі у формат C header файлу (.h) для розгортання програми у Flash пам’яті мікроконтролера виконується спеціально розробленим Python скриптом. Скрипт виконує конвертацію, перевірку розміру моделі та Tensor Arena, тестове розпізнавання (inference) та генерацію header файлу з необхідними константами для моделі (розмір grid, кількість класів, розмір Tensor arena тощо).

Для навчання моделі використано набір зображень БПЛА better_drones_dataset, який було отримано з платформи Roboflow у попередньо розділеному форматі (train/val/test). Навчальна вибірка використовується для оновлення ваг моделей. Валідаційна вибірка слугує для підбору гіперпараметрів та реалізації механізму early stopping. Тестова вибірка використовується виключно для фінальної оцінки якості навчених моделей і не бере безпосередньої участі у процесі навчання. Вихідний набір даних містив досить багато класів БПЛА, проте для вирішення задачі розпізнавання БПЛА у реальних умовах було виконано перемаркування об’єктів до 4 класів: Other_UAV (інші типи БПЛА), Shahed (ударні БПЛА типу Shahed), Sky (фон/небо) та Bird (клас із зображеннями птахів як основний клас хибних спрацювань). Клас Sky у процесі навчання виключається з активних класів та використовується лише як негативний фон. Зображення у наборі даних мають різні роздільні здатності та зображують БПЛА у різних умовах зйомки

та різних ракурсах: різна висота польоту, різне освітлення, різний фон (небо, хмари, земля). Для моделей 1-4 зображення масштабуються до відповідної роздільної здатності операцією bilinear resize. Для моделі 5 додатково застосовується гібридна стратегія формування вибірки з crop-патчами та hard negatives, яка описана у підрозділі 2.4.

Аугментація даних виконується в процесі навчання. Здійснюється горизонтальне та вертикальне відображення зображення з одночасним відображенням відповідної сітки міток (для збереження просторової відповідності між зображенням та мітками), випадкова зміна яскравості та контрастності в межах ±20%, перетворення у відтінки сірого з ймовірністю 15% та додавання гаусівського шуму зі стандартним відхиленням 0.02.

На рисунку 3.1 зображено деякі приклади зображень БПЛА з набору даних.

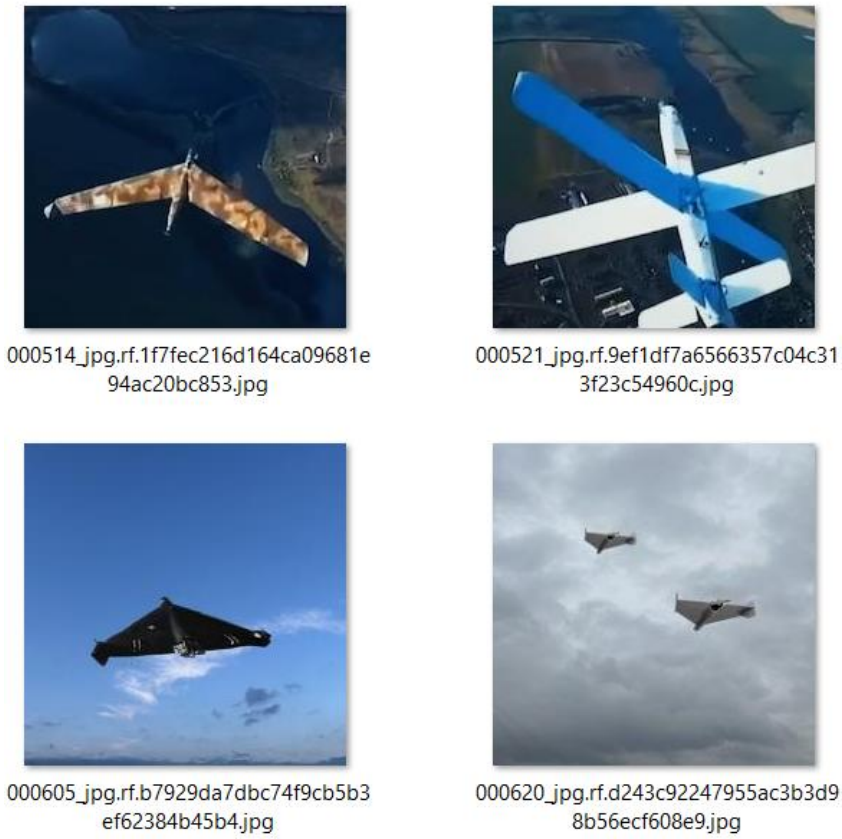

Рисунок 3.1 – Приклади зображень БПЛА з набору даних

3.2. Навчання та порівняння розроблених моделей

У даному підрозділі описано процес навчання всіх п’яти архітектурних варіантів моделі FOMO, наведено результати оцінювання на валідаційній та тестовій вибірках.

Модель 1 є еталонною конфігурацією повної архітектури моделі. На вхід подається зображення роздільною здатністю 224×224 пікселів та використовується backbone MobileNetV2 з alpha=1.0. Навчання тривало 72 епохи до спрацювання early stopping (patience=15). Найкращий результат зафіксовано на 70-й епосі. На рисунках 3.2 і 3.3 наведено графіки результатів навчання моделі 1.

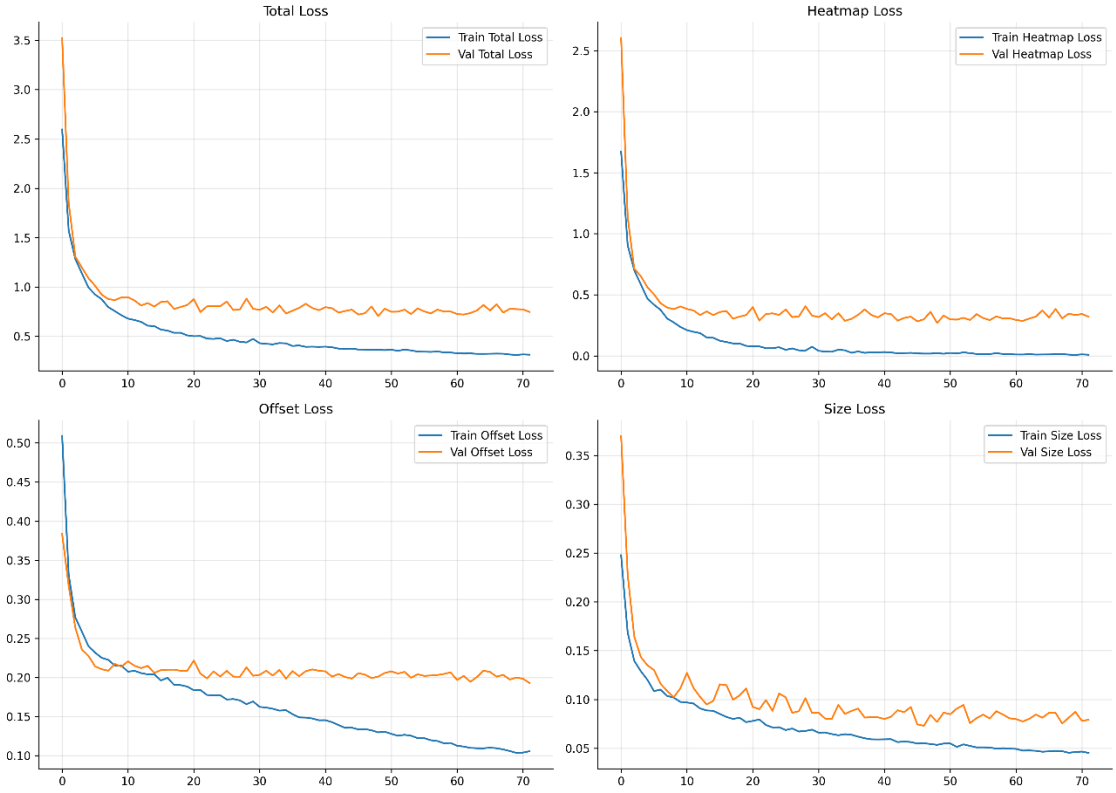

Рисунок 3.2 – Графіки функцій втрат моделі 1

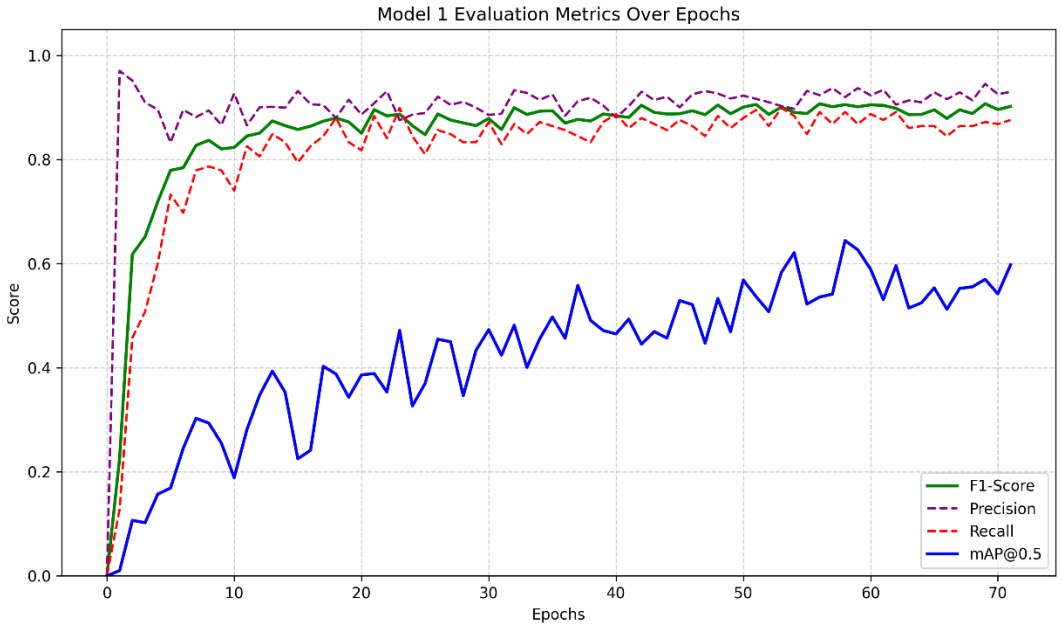

Рисунок 3.3 – Графік метрик моделі 1

Графік функцій втрат (рис. 3.2) демонструє стрімке зниження train loss протягом перших 10 епох з 2.6 до ~0.5, після чого збіжність уповільнюється. Загальна train loss досягає ~0.31 на момент зупинки. Валідаційна loss стабілізується на рівні ~0.75-0.78 починаючи з 15-ї епохи без суттєвих ознак перенавчання, що підтверджує узагальнювальну здатність моделі. Аналогічну картину демонструють окремі компоненти функції втрат: heatmap loss знижується з 2.6 до ~0.02 (train) та ~0.30 (val); offset та size loss також стабілізуються протягом перших 25 епох.

Графік метрик (рис. 3.3) показує швидку збіжність F1-score – вже після 5 епох досягається F1~0.78, після чого метрика поступово зростає до фінального значення 0.9073 (Precision=0.945, Recall=0.872). Метрика mAP@0.5 демонструє більш повільну та нестабільну збіжність. Це характерна особливість IoU-based метрик для heatmap детекторів, де точність bounding box залежить від якості навченого size head. Фінальне значення mAP@0.5 становить 0.5698.

Було перевірено роботу моделі 1 на тестових зображень за допомогою окремої програми predict.py. Ця програма також буде використовуватися для перевірки

інших моделей. На рисунках 3.4 та 3.5 наведено приклади розпізнавання об’єктів моделі 1 на тестових зображеннях.

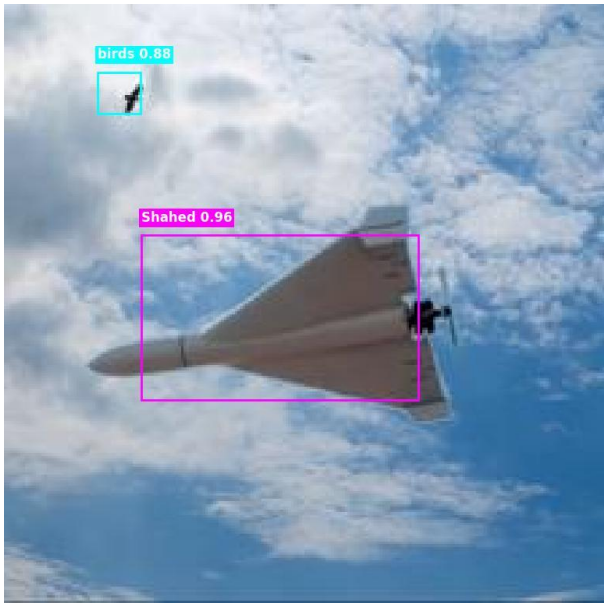

Рисунок 3.4 – Розпізнавання Shahed та одного птаха за допомогою моделі 1

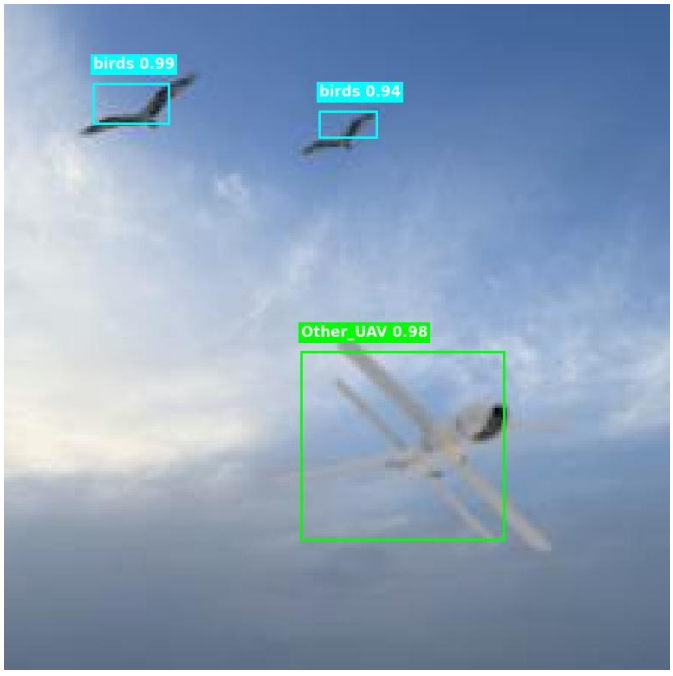

Рисунок 3.5 – Розпізнавання Other_UAV та птахів за допомогою моделі 1

На рисунку 3.4 модель правильно виявляє та розпізнає БПЛА типу Shahed з впевненістю 0.96 та птаха з впевненістю 0.88. Обидва об’єкти правильно класифіковані та локалізовані за допомогою bounding boxes. На рисунку 3.5 модель одночасно виявляє та розпізнає три об’єкти: БПЛА класу Other_UAV (0.98) та двох птахів (0.99 та 0.94). Всі розпізнавання правильні та мають високу впевненість.

Модель 2 відрізняється від моделі 1 виключно значенням параметра alpha=0.35, що теоретично зменшує кількість параметрів backbone у ~8 разів при збереженні роздільної здатності 224×224 та архітектури multi-scale fusion. Навчання тривало 51 епоху до спрацювання early stopping. Це суттєво менше ніж було в моделі 1 (72 епохи). Це пояснюється меншою кількістю параметрів, яка прискорює збіжність моделі. На рисунках 3.6 та 3.7 наведено графіки навчання моделі 2.

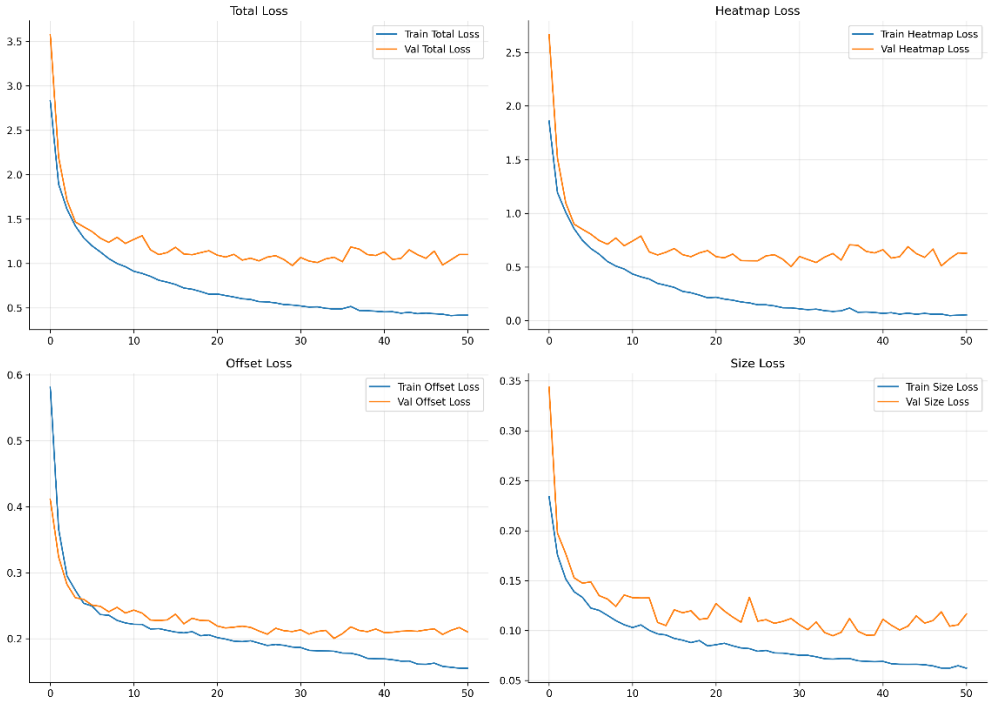

Рисунок 3.6 – Графіки функцій втрат моделі 2

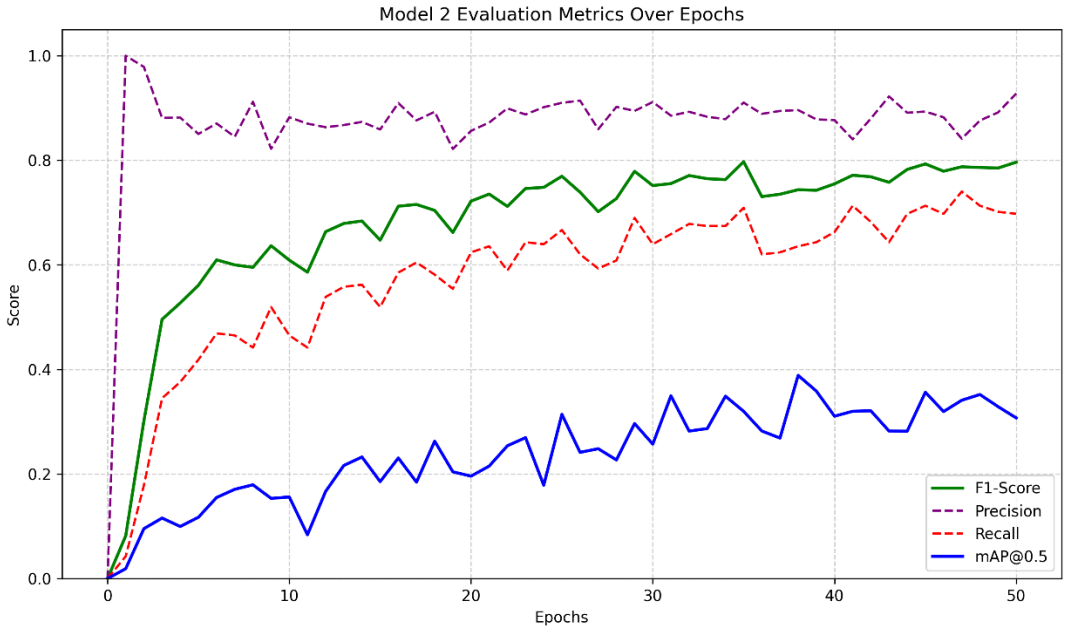

Рисунок 3.7 – Графік метрик моделі 2

Графік функцій втрат (рис. 3.6) демонструє характерну для моделей з alpha=0.35 картину: train loss знижується з 2.8 до ~0.42, тоді як val loss стабілізується на рівні ~1.0-1.1 починаючи з 15-ї епохи. Розрив між train та val loss помітно більший ніж у моделі 1 (~0.65 проти ~0.45). Це свідчить про певний рівень перенавчання, спричинений меншою регуляризуючою здатністю вужчого backbone. Heatmap loss знижується аналогічно до моделі 1, тоді як offset та size loss залишаються стабільними з 10-ї епохи.

Графік метрик (рис. 3.7) показує повільнішу збіжність F1-score порівняно з моделлю 1. Після 10 епох F1~0.61, фінальне значення 0.7974 (Precision=0.893, Recall=0.713). Характерною відмінністю є помітний дисбаланс між Precision (~0.87-0.93) та Recall (~0.70), що свідчить про вищу консервативність моделі при меншій ширині backbone. Модель рідше генерує хибні спрацювання, але частіше пропускає об’єкти. mAP@0.5 досягає лише 0.3564, що майже вдвічі нижче ніж у моделі 1 (0.5698).

На рисунках 3.8 та 3.9 наведено приклади розпізнавання об’єктів моделі 2 на тестових зображеннях.

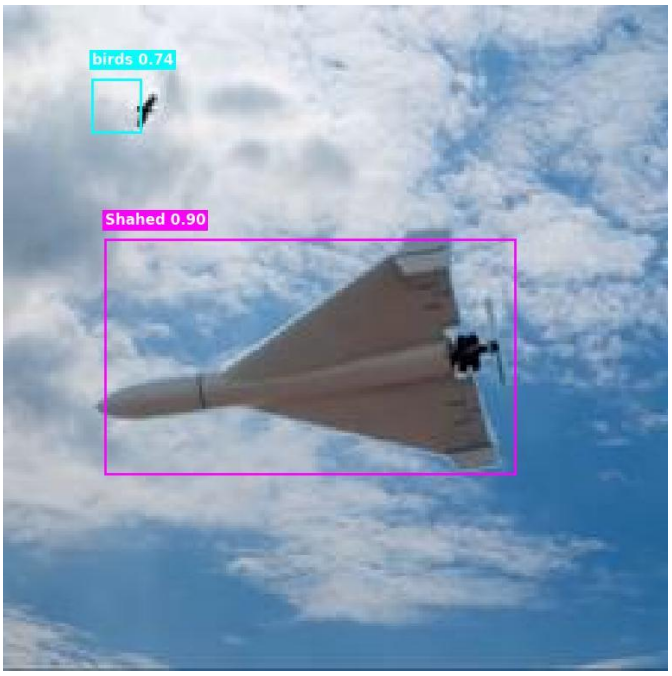

Рисунок 3.8 – Розпізнавання Shahed та одного птаха за допомогою моделі 2

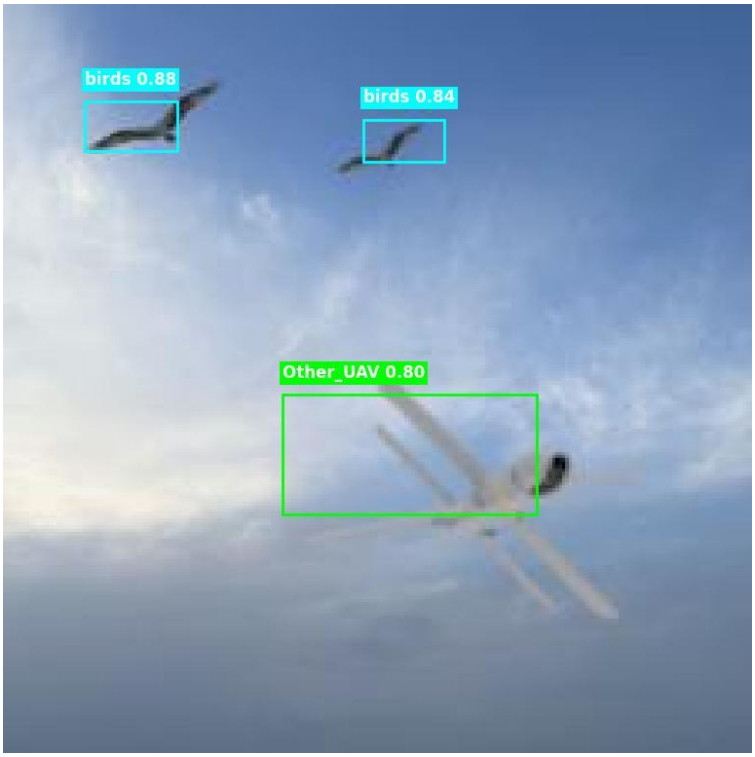

Рисунок 3.9 – Розпізнавання Other_UAV та птахів за допомогою моделі 2

На рисунку 3.8 модель правильно виявляє та розпізнає БПЛА Shahed з впевненістю 0.90 та одного птаха з впевненістю 0.74. Обидва класи правильно розпізнані, проте впевненість для Shahed знизилась з 0.96 (модель 1) до 0.90, а для птаха з 0.88 до 0.74. На рисунку 3.9 модель виявляє Other_UAV з впевненістю 0.80 та двох птахів (0.88 та 0.84). Всі розпізнавання правильні, але впевненість Other_UAV знизилась з 0.98 (модель 1) до 0.80.

Порівняння моделей 1 та 2 показує, що зменшення параметра alpha з 1.0 до 0.35 призводить до зниження F1-score з 0.9073 до 0.7974 (на ~12%) та mAP@0.5 з 0.5698 до 0.3564 (на ~37%). Таким чином, зменшення alpha у 8 разів призводить до помітної втрати точності, що частково спростовує гіпотезу щодо незначного впливу параметра alpha. Разом з тим модель залишається функціональною та забезпечує правильне розпізнавання всіх трьох класів.

Наступна модель 3 відрізняється від моделі 2 зменшеною роздільною здатністю вхідного зображення з 224×224 до 96×96 при збереженні параметра alpha=0.35 та архітектури multi-scale fusion. Теоретично це зменшує обчислювальну складність у ~5.4 рази за рахунок скорочення просторових розмірів feature maps, що робить модель придатною для розгортання на мікроконтролерах класу ESP32-S3. Навчання тривало 65 епох до спрацювання early stopping. Це більше ніж у моделі 2 (51 епоха). Це пояснюється складнішою задачею навчання при суттєво меншій вхідній роздільній здатності зображення. Моделі потрібно більше ітерацій для вироблення стійких ознак із менш деталізованих feature maps. На рисунках 3.10 та 3.11 наведено графіки навчання моделі 3.

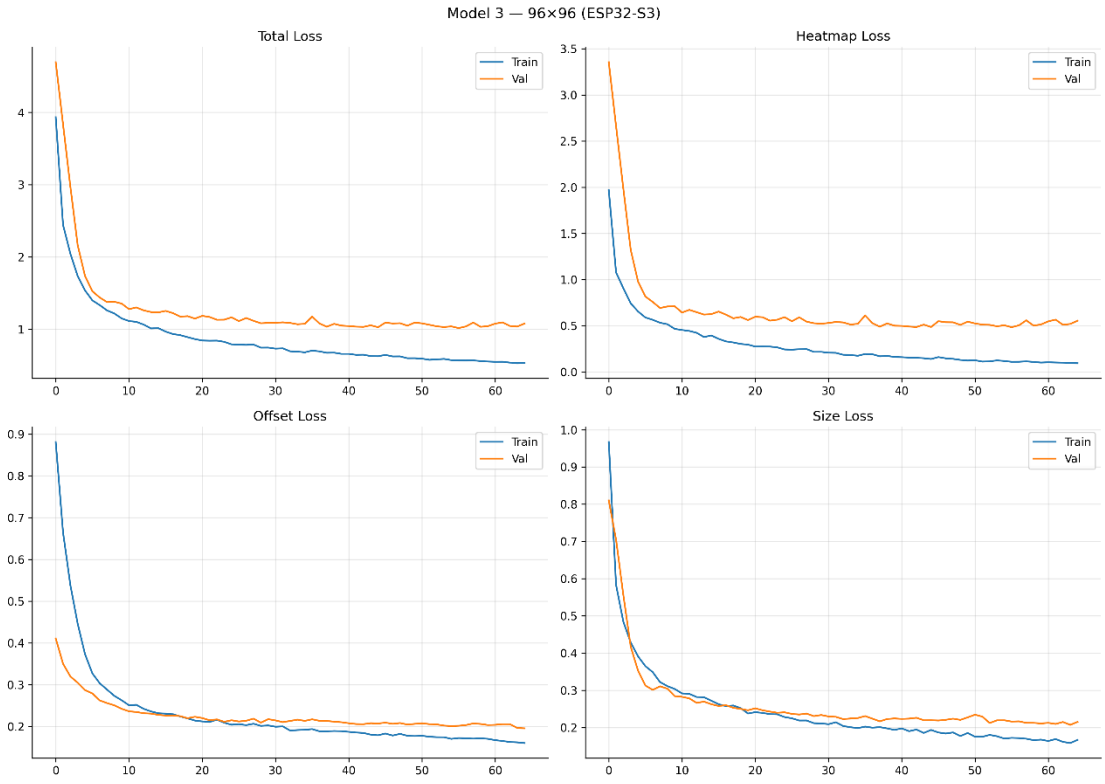

Рисунок 3.10 – Графіки функцій втрат моделі 3

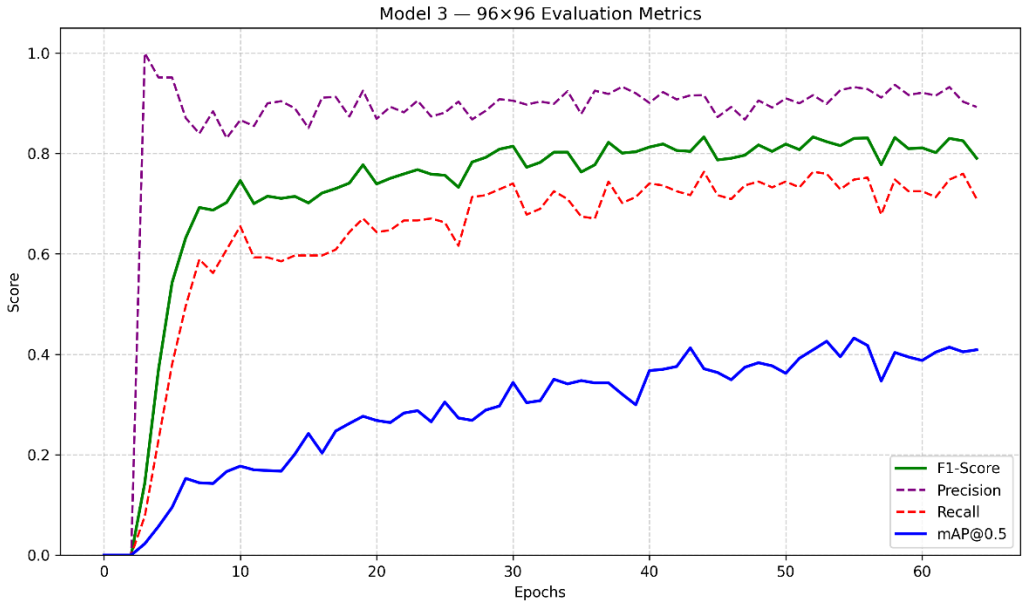

Рисунок 3.11 – Графік метрик моделі 3

Графік функцій втрат (рис. 3.10) демонструє стабільний монотонний спад тренувальної кривої з ~3.9 до ~0.55, тоді як валідаційна крива стабілізується на рівні ~1.0-1.05 починаючи приблизно з 15-ї епохи. Розрив між train та val loss (~0.50) є порівнянним з моделлю 3 (~0.65) і свідчить про аналогічний рівень перенавчання попри значно менший вхідний розмір зображення. Heatmap loss демонструє найбільший дисбаланс між train (~0.10) та val (~0.50). Цк вказує на труднощі узагальнення просторової локалізації при роздільній здатності 96×96. Offset loss та size loss поводяться узгоджено: обидві пари кривих спадають паралельно та стабілізуються на близьких значеннях (~0.15-0.20 для train та ~0.20 для val), що свідчить про задовільну якість регресії координат і розмірів обмежувальних рамок.

Графік метрик (рис. 3.11) демонструє швидку збіжність усіх показників у перших 5-8 епохах із подальшою стабілізацією. Метрика Precision протягом навчання утримувалась на рівні 0.85-0.93. Метрика Recall стабілізувалася в діапазоні 0.70-0.75. Фінальний F1-Score склав 0.8330 (Precision=0.932, Recall=0.748), що на ~4.5% вище ніж у моделі 2 (0.7974) попри восьмикратне зменшення площі вхідного зображення. Це свідчить про ефективність мультимасштабного злиття ознак при малих роздільних здатностях зображення. Натомість метрика mAP@0.5=0.41 перевищує значення моделі 2 на ~15%, проте суттєво поступається моделі 1 (0.5698). Це відображає об’єктивні обмеження точної локалізації при роздільній здатності 96×96.

На рисунках 3.12 та 3.13 наведено приклади розпізнавання об’єктів моделі 3 на тестових зображеннях.

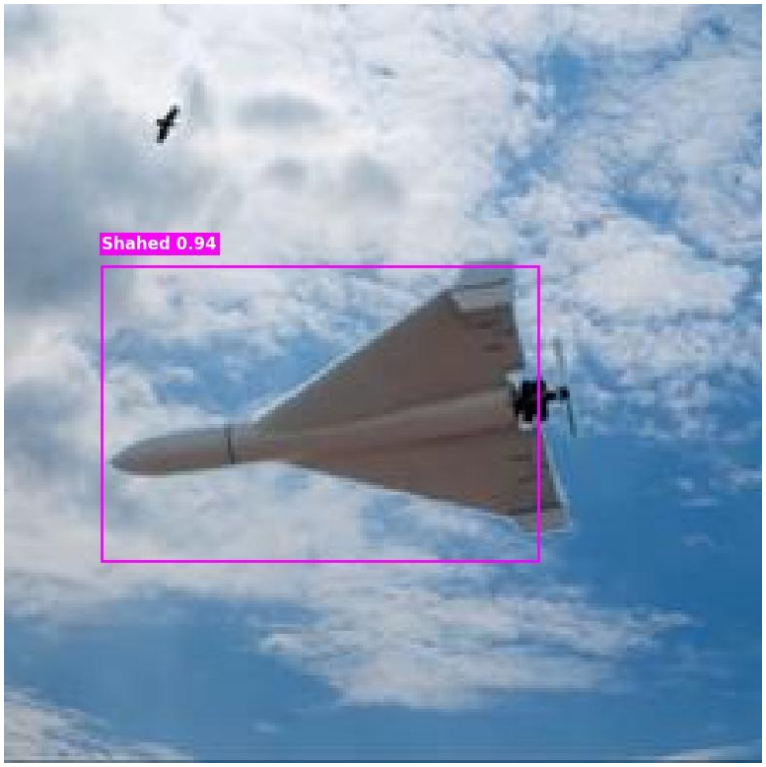

Рисунок 3.12 – Розпізнавання Shahed та одного птаха за допомогою моделі 3

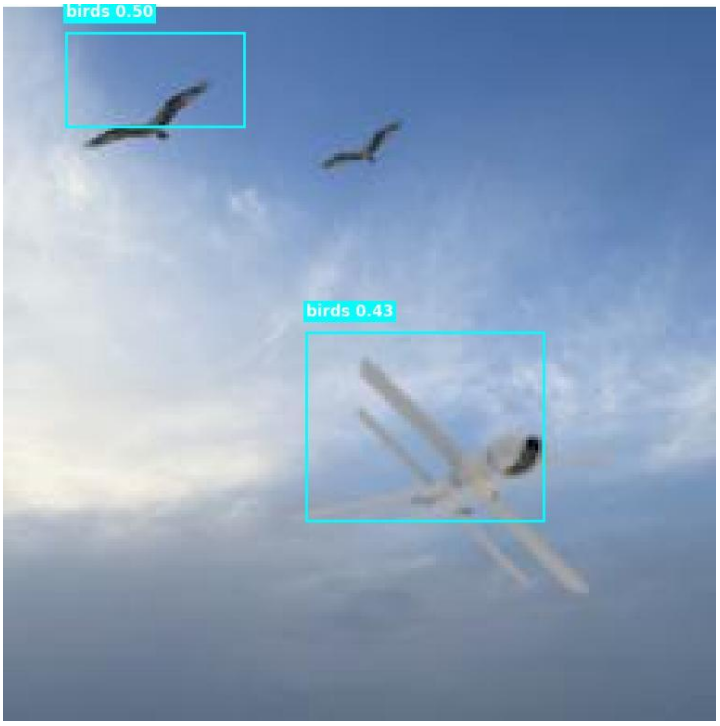

Рисунок 3.13 – Розпізнавання Other_UAV та птахів за допомогою моделі 3

На рисунку 3.12 модель впевнено виявляє та розпізнає БПЛА Shahed з оцінкою впевненості 0.94, формуючи обмежувальну рамку, що майже точно охоплює БПЛА. Малий силует птаха у верхній частині зображення проігноровано. Це означає, що зменшення роздільної здатності вхідного зображення негативно впливає на розпізнавання малих об’єктів.

На рисунку 3.13 за допомогою моделі 3 було виявлено та розпізнано два об’єкти. Обидва об’єкти було класифіковано як birds з оцінками 0.50 та 0.43. При цьому клас birds помилково присвоєно БПЛА у центральній частині зображення, а третій об’єкт (другий птах) залишився невиявленим та нерозпізнаним. Така поведінка моделі 3 вказує на проблему, що загострюється при роздільній здатності зображення 96×96, коли дрібні деталі силуету об’єкта, що дозволяють розрізнити класи, втрачаються через агресивне зменшення просторових розмірів.

Модель 4 відрізняється від моделі 3 збільшеною роздільною здатністю вхідного зображення з 96×96 до 128×128 при збереженні параметра alpha=0.35 та архітектури multi-scale fusion. Збільшення лінійного розміру на ~33% забезпечує приріст площі feature maps у ~1.78 рази, що теоретично покращує здатність моделі до локалізації дрібних об’єктів і розрізнення силуетів схожих класів. Навчання тривало 88 епох до спрацювання early stopping. Це найдовше серед моделей з параметром alpha=0.35. Більша кількість епох пояснюється підвищеною складністю оптимізації при збільшеному просторовому розмірі теплової карти: моделі потрібно більше ітерацій для стабілізації градієнтів від мультимасштабного злиття ознак. На рисунках 3.14 та 3.15 наведено графіки навчання моделі 4.

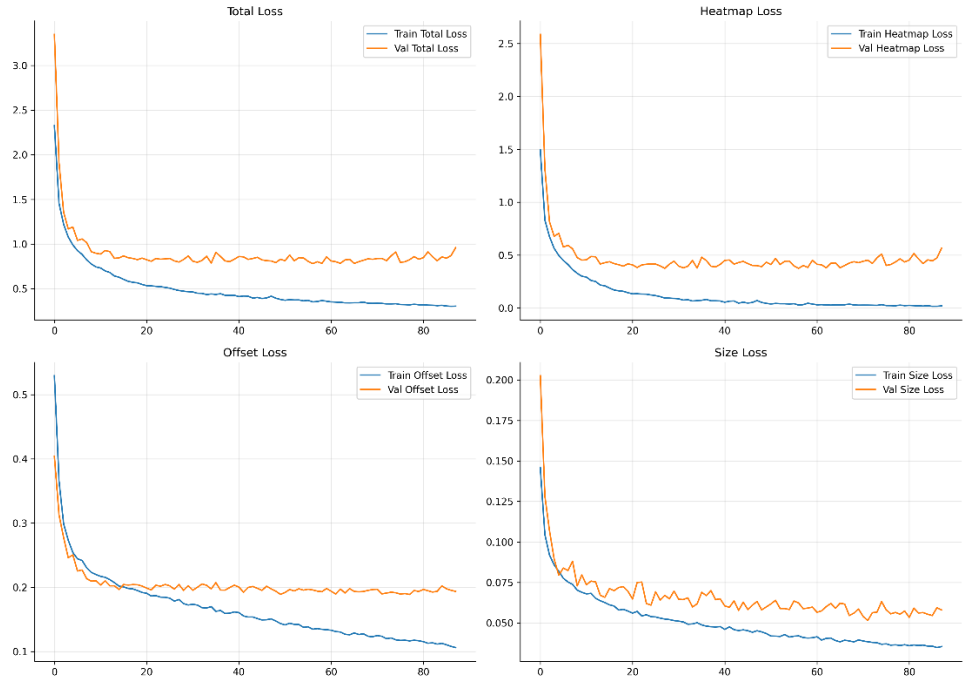

Рисунок 3.14 – Графіки функцій втрат моделі 4

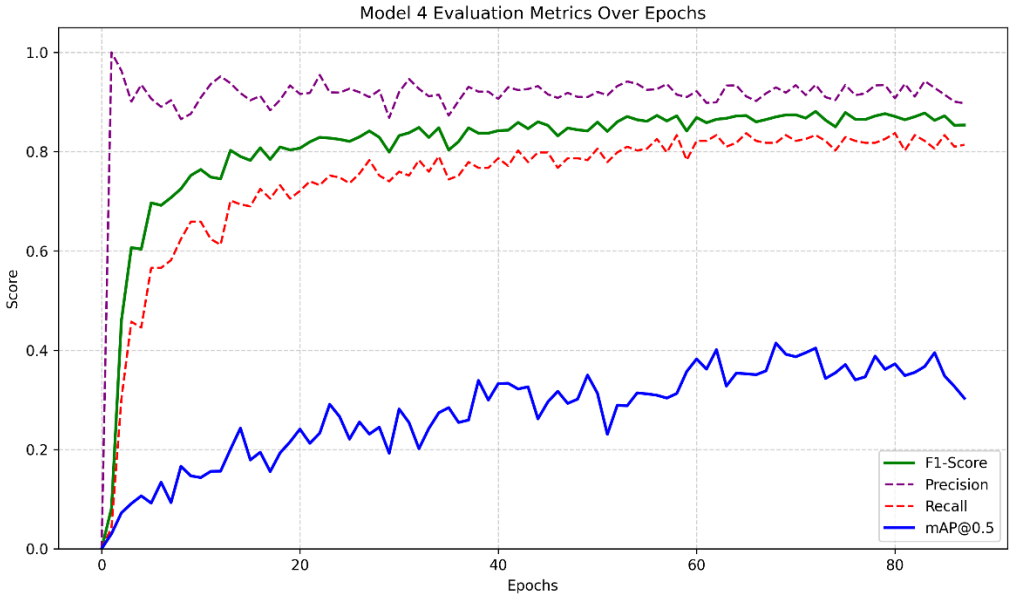

Рисунок 3.15 – Графік метрик моделі 4

Графік функцій втрат (рис. 3.14) демонструє узгоджену поведінку train та val кривих. Загальна тренувальна крива знижується з ~2.3 до ~0.32, тоді як валідаційна стабілізується на рівні ~0.80-0.90, що помітно нижче ніж у моделей 2 і 3 (~1.0-1.05). Розрив між train та val loss (~0.55) є порівнянним з попередніми моделями, однак абсолютний рівень валідаційних втрат є найнижчим, що свідчить про кращу узагальнюючу здатність моделі. Heatmap loss демонструє аналогічну до моделі 3 картину: тренувальна крива спадає майже до нуля, тоді як валідаційна стабілізується близько 0.40-0.50. Найбільш показовими є криві Offset loss та Size loss. Обидві пари train/val практично збігаються протягом усього навчання і стабілізуються на надзвичайно низьких значеннях (~0.10 для train та ~0.20 для val у Offset loss; ~0.03-0.06 для обох кривих Size loss). Це свідчить про суттєво кращу якість регресії координат і якість розмірів обмежувальних рамок порівняно з моделями меншої роздільної здатності.

Графік метрик моделі 4 (рис. 3.15) демонструє найшвидшу збіжність серед усіх моделей з alpha=0.35: вже після 5-7 епох F1-Score перевищує 0.80 і надалі стабільно утримується в діапазоні 0.82-0.88. Метрика Precision протягом навчання коливалась у діапазоні значень 0.88-0.95, метрика Recall – у діапазоні 0.78-0.84. Це свідчить про більш збалансований розподіл між хибнопозитивними та хибнонегативними спрацюваннями порівняно з моделями 2 і 3. Фінальний F1-Score дорівнює 0.8811 (Precision=~0.92, Recall=~0.84), що є найвищим результатом серед усіх моделей, які орієнтовані на розгортання на мікроконтролері ESP32-S3 та на ~6% перевищує показник моделі 3 (0.8330). Метрика mAP@0.5 демонструє характерну нестабільність з помітними коливаннями (~0.20-0.42) протягом усього процесу навчання і з фінальним значенням близько 0.35-0.40. Нестабільність mAP при одночасно стабільному F1 пояснюється чутливістю метрики до точності локалізації обмежувальних рамок, яка залишається обмеженою навіть при використанні роздільної здатності 128×128.

На рисунках 3.16 та 3.17 наведено приклади розпізнавання об’єктів моделі 4 на тестових зображеннях.

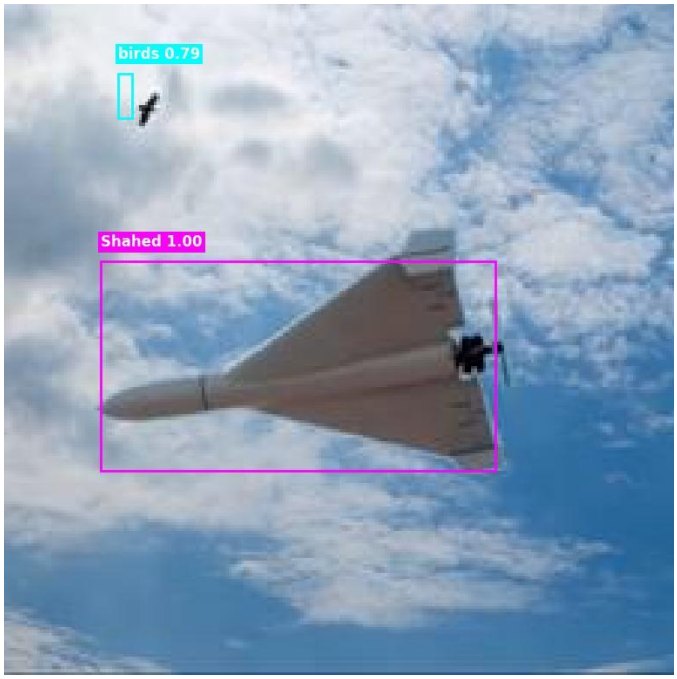

Рисунок 3.16 – Розпізнавання Shahed та одного птаха за допомогою моделі 4

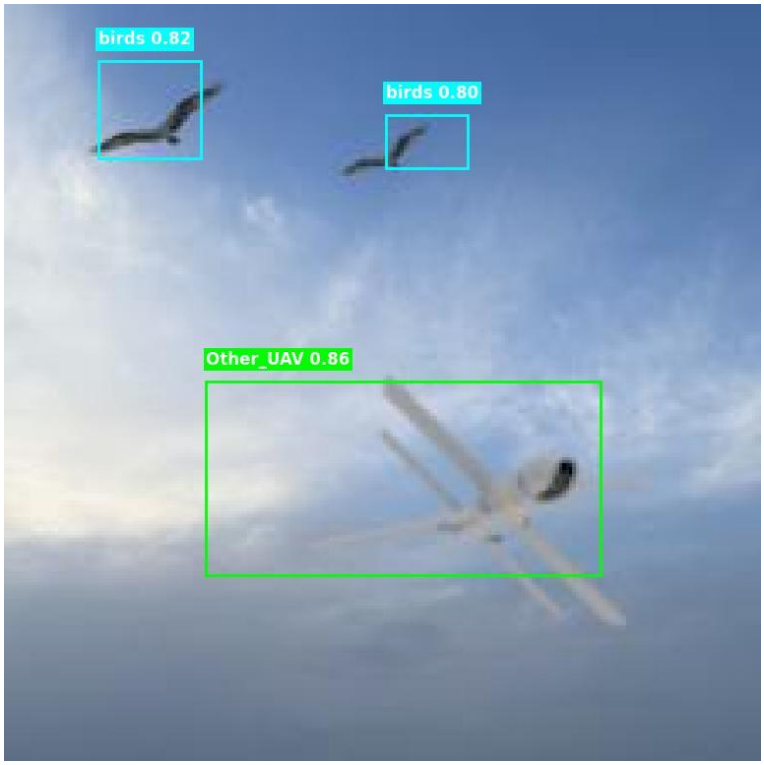

Рисунок 3.17 – Розпізнавання Other_UAV та птахів за допомогою моделі 4

На рисунку 3.16 модель 4 виявила та розпізнала обидва об’єкти і правильно класифікувала їх: Shahed з максимальною оцінкою впевненості 1.00 та птах з оцінкою 0.79. Це є принципово кращим результатом порівняно з моделлю 3, яка на аналогічному зображенні проігнорувала птаха. Оцінка впевненості для Shahed (1.00) є найвищою серед усіх схожих моделей. Це підтверджує той факт, що з використанням роздільної здатності 128×128 можливо формувати більш дискримінативні ознаки для розпізнавання БПЛА.

На рисунку 3.17 модель 4 правильно виявила, розпізнавала та класифікувала всі три об’єкти: двох птахів з оцінками 0.82 та 0.80 відповідно і Other_UAV з оцінкою 0.86. Цей результат кращий за результат моделі 3, яка на аналогічному зображені класифікувала БПЛА як птаха. Збільшення роздільної здатності до 128×128 дозволило моделі 4 зберегти достатньо деталей об’єктів для правильного розпізнавання.

Порівняння моделей 3 та 4 показує, що збільшення роздільної здатності вхідного зображення з 96×96 до 128×128 при незмінному параметрі alpha=0.35 призводить до зростання F1-score з 0.8330 до 0.8811, усунення плутанини класів, а також підвищення впевненості розпізнавання по всіх класах. Разом з тим mAP@0.5 залишається нестабільним і суттєво нижчим за еталонну модель 1 (0.5698), що свідчить про збережені обмеження точності локалізації об’єктів. Для збільшеної роздільної здатності зображення потрібно більше обчислювальних ресурсів. Це варто враховувати при розгортанні на мікроконтролері ESP32-S3. Саме тому була розроблена та проаналізована ще одна модель 5, яка має значне архітектурне спрощення.

Модель 5 принципово архітектурно відрізняється від інших розроблених моделей: замість повної multi-scale fusion архітектури ця модель використовує архітектуру з раннім виходом (early exit) з єдиним виходом на рівні блоку 6 backbone MobileNetV2 при роздільній здатності зображення 96×96 та параметром alpha=0.35.

Це значно зменшує глибину активної частини мережі приблизно вдвічі порівняно з повною архітектурою, зменшуючи вимоги до обчислювальних ресурсів за рахунок відмови від глибоких ознак. Навчання моделі 5 тривало протягом 87 епох до спрацювання механізму early stopping. Оптимальний поріг впевненості для моделі 5 було визначено автоматично за допомогою розрахунку кривої F1 vs Threshold. Поріг впевненості дорівнює 0.31. Це суттєво нижче за типові значення інших моделей та свідчить про загальну знижену впевненість розпізнавань моделі 5. На рисунках 3.18 та 3.19 наведено графіки функції втрат, навчання та PR-криву моделі 5.

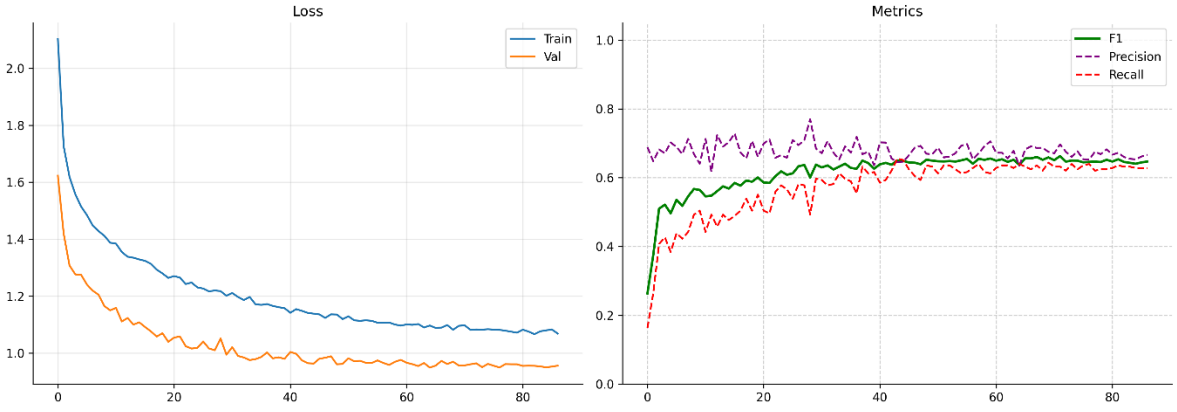

Рисунок 3.18 – Графік функції втрат та метрик моделі 5

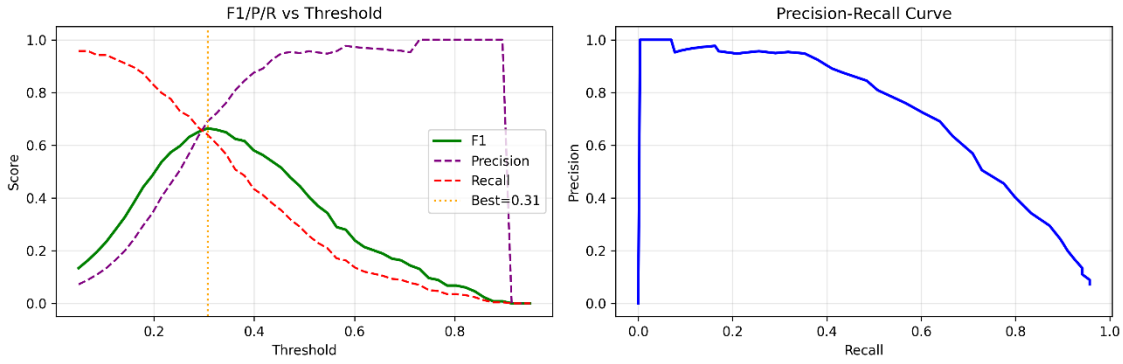

Рисунок 3.19 – Графік PR-кривої моделі 5

Графік функцій втрат та метрик (рис. 3.18) демонструє принципово відмінність у порівнянні з іншими моделями: валідаційна крива спадає нижче тренувальної і стабілізується на рівні ~0.93-0.95, тоді як тренувальна крива зупиняється на рівні ~1.07-1.09. Такий інвертований розрив (val loss < train loss) є нетиповим і може свідчити про надмірно агресивну регуляризацію або про те, що валідаційна вибірка містить більш прості зображення для поточної архітектури. Абсолютний рівень обох кривих (~1.0) є найвищим серед моделей. Це безпосередньо відображає обмежену здатність цієї архітектури з раннім виходом. Ознаки одного блоку 6 не забезпечують достатньої глибини моделі для надійного та впевненого розпізнавання БПЛА. F1-Score стабілізується в діапазоні 0.62-0.66, Precision – 0.63-0.72, Recall – 0.62-0.65. Характерною особливістю є зближення метрик Precision і Recall до практично однакових значень. Це контрастує з виразним дисбалансом на користь Precision у моделей з повною архітектурою. Це перевага використання гібридного підходу до набору даних, який було описано раніше. Фінальний F1-Score 0.6626 є найнижчим серед усіх розроблених моделей та значно поступається моделі 4 (0.8811).

Графіки кривих F1/P/R vs Threshold та графік PR-кривої (рис. 3.19) підтверджують обмеження архітектури моделі 5. На графіку F1 vs Threshold оптимум досягається при threshold=0.31 зі значенням F1=0.664. Це суттєво нижче за аналогічні показники інших моделей при значно меншому пороговому значенні. Різкий спад F1 при підвищенні порогу вище 0.35 свідчить про відсутність впевнених передбачень: більшість розпізнавань мають оцінку в діапазоні 0.30-0.45. PR-крива демонструє прийнятну Precision (~0.95-1.00) при малих значеннях Recall, проте швидко деградує. При Recall=0.5 метрика Precision вже знижується до ~0.75, а при Recall=0.9 падає нижче 0.4.

На рисунках 3.20 та 3.21 наведено приклади розпізнавання об’єктів моделі 5 на тестових зображеннях.

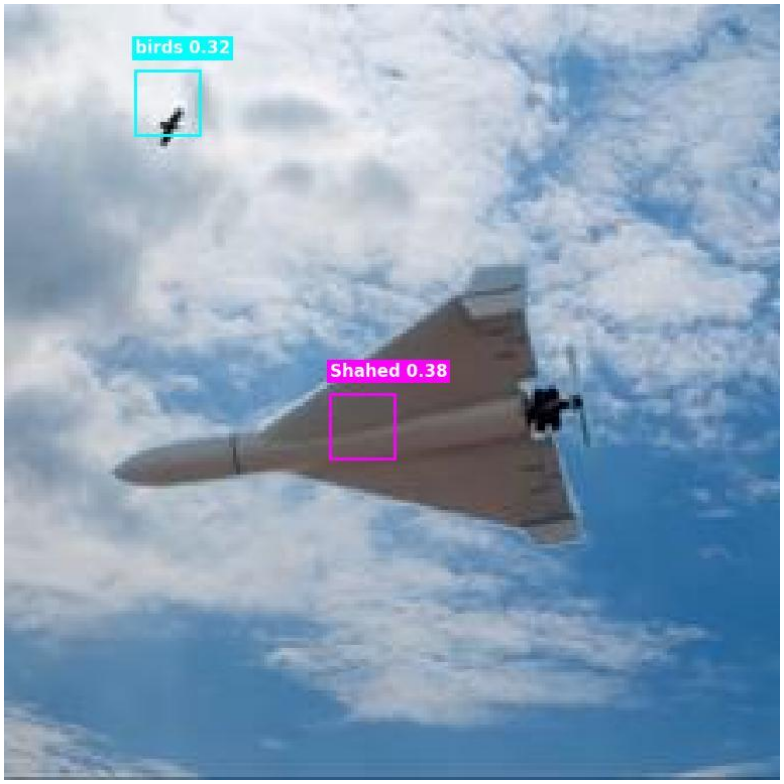

Рисунок 3.20 – Розпізнавання Shahed та одного птаха за допомогою моделі 5

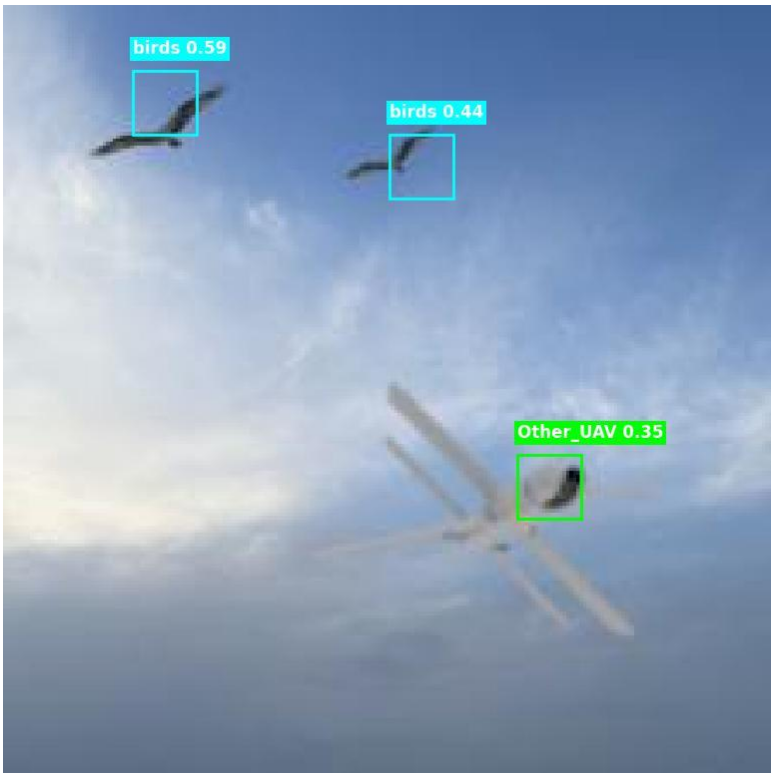

Рисунок 3.21 – Розпізнавання Other_UAV та птахів за допомогою моделі 5

На рисунку 3.20 за допомогою моделі 5 було виявлено та розпізнано обидва об’єкти, проте з критично низькою впевненістю. Обмежувальна рамка для Shahed охоплює лише частину корпусу БПЛА замість повного силуету, що свідчить про неспроможність малих ознак блоку 6 сформувати правильну просторову локалізацію об’єкта.

На рисунку 3.21 за допомогою моделі 5 було виявлено та розпізнано всі три об’єкти. Класифікація об’єктів є правильною, проте обмежувальна рамка для Other_UAV охоплює лише двигун БПЛА. Такі неповні часткові рамки є наслідком обмеженого поля ранніх feature maps, які фіксують лише локальні текстурні ознаки (на рисунку це двигун як найбільш виражена деталь БПЛА).

Порівняння моделі 5 з моделями 3 і 4 однозначно демонструє, що архітектура раннього виходу з блоку 6 при роздільній здатності зображення 96×96 не забезпечує прийнятної якості розпізнавання для практичного застосування. Зниження F1 при одночасній деградації якості локалізації підтверджує, що глибина мультимасштабного злиття є критично важливою для задачі розпізнавання БПЛА. Така архітектура моделі 5 була спеціально розроблена та проаналізована у цій роботі для порівняння з повною архітектурою. Свідоме спрощення архітектури моделі 5 розглядається як потенційний компроміс між якістю розпізнавання та швидкодією розпізнавання на мікроконтролері ESP32-S3. Розгортання моделей на ESP32-S3 та аналіз часу розпізнавання (inference) та кількості FPS є предметом експериментального дослідження у наступному розділі.

3.3. Конвертація та квантизація моделей

Після завершення навчання кожної з моделей було збережено кожну з моделей у форматі Keras Float32. Щоб спробувати розгорнути моделі на мікроконтролері ESP32-S3 виконується конвертація збережених моделей у формат TensorFlow Lite INT8. Моделі 1 і 2 є досить великими за своїм розміром. Квантизація цих моделей

не допоможе отримати модель, яка здатна влізти у пам’ять ESP32-S3. Тому було розглянуто квантизацію моделей 3, 4, 5. Як це було описано раніше, ці моделі використовують параметр alpha=0.35. Значення цього параметра напряму впливає на розмір збереженої моделі.

Конвертація та квантизація виконується за допомогою TFLite Converter з повною INT8 квантизацією, тобто квантизуються як ваги моделі так і активації всіх шарів. Для правильного визначення динамічного діапазону активацій використовується набір з 200 зображень тренувальної вибірки, рівномірно відібраних з усього набору даних. Вхідний тензор квантизується у формат int8, вихідний (heatmap) залишається у форматі float32 для зручності проведення post-processing на мікроконтролері.

Загалом процес конвертації та квантизації складається з таких кроків: завантаження навченої моделі Keras; налаштування TFLiteConverter з параметром оптимізації DEFAULT та цільовим форматом TFLITE_BUILTINS_INT8; запуск representative_dataset_gen для calibration на 200 зображеннях; конвертація та збереження .tflite файлу; перевірка відповідності розміру моделі вимогам Flash пам’яті ESP32-S3; генерація C header файлу (.h) для здійснення прошивки Flash пам’яті мікроконтролера. Для моделей 3 та 4 використовується конвертер з підтримкою трьох виходів (heatmap, offset, size), а для моделі 5 використовується спрощений конвертер для одного виходу (heatmap).

Результати конвертації та квантизації моделей 3, 4 та 5 наведено у таблиці 3.1.

Таблиця 3.1 – Результати конвертації та квантизації моделей

Характеристика	**Модель 3**	**Модель 4**	**Модель 5**
Розмір .tflite файлу	1294 KB	1739 KB	107 KB
Вхідний тип даних	int8	int8	int8
Вихідний тип даних	float32	float32	float32
Розмір Grid	6×6	8×8	12×12
Кількість виходів	3	3	1

Для дослідження впливу INT8 квантизації на точність розпізнавання окремих класів було розроблено скрипт діагностики деградації розпізнавання окремих класів check_quant.py. Скрипт завантажує повну Float32 Keras модель та TFLite INT8 модель, виконує сам процес розпізнавання (inference) на тестовій вибірці (до 30 зображень на клас) та порівнює максимальні значення heatmap для кожного класу. Розпізнавання вважається правильним, якщо максимальний heatmap score для класу перевищує заданий Detection Threshold та є більшим за максимальний score інших класів. Результати оцінювання для моделі 3 наведено на рисунку 3.22, а для моделі 4 – на рисунку 3.23.

```
[Other_UAV]  (30 зображень)
  Keras  FP32 : own=0.688±0.226 | other=0.147 | correct=26/30 (87%)
  TFLite INT8 : own=0.612±0.293 | other=0.081 | correct=23/30 (77%)
  Score drop  : +0.077 після квантизації
  Клас Other_UAV значно деградував після квантизації!

[Shahed]  (30 зображень)
  Keras  FP32 : own=0.700±0.225 | other=0.279 | correct=27/30 (90%)
  TFLite INT8 : own=0.145±0.109 | other=0.310 | correct=2/30 (7%)
  Score drop  : +0.556 після квантизації
  Клас Shahed значно деградував після квантизації!

[Bird]  (24 зображень)
  Keras  FP32 : own=0.619±0.214 | other=0.350 | correct=19/24 (79%)
  TFLite INT8 : own=0.321±0.234 | other=0.304 | correct=8/24 (33%)
  Score drop  : +0.298 після квантизації
  Клас Bird значно деградував після квантизації!
```

Рисунок 3.22 – Діагностика деградації класів для моделі 3

```
[Other_UAV]  (30 зображень)
  Keras  FP32 : own=0.830±0.249 | other=0.079 | correct=28/30 (93%)
  TFLite INT8 : own=0.841±0.259 | other=0.084 | correct=27/30 (90%)
  Score drop  : -0.011 після квантизації

[Shahed]  (30 зображень)
  Keras  FP32 : own=0.753±0.259 | other=0.177 | correct=26/30 (87%)
  TFLite INT8 : own=0.814±0.214 | other=0.155 | correct=26/30 (87%)
  Score drop  : -0.061 після квантизації

[Bird]  (24 зображень)
  Keras  FP32 : own=0.747±0.246 | other=0.294 | correct=18/24 (75%)
  TFLite INT8 : own=0.761±0.233 | other=0.263 | correct=20/24 (83%)
  Score drop  : -0.013 після квантизації
```

Рисунок 3.23 – Діагностика деградації класів для моделі 4

За результатами діагностики моделі 3 було виявлено значну диференційовану деградацію точності розпізнавання після INT8 квантизації. Клас Shahed зазнав катастрофічної деградації: точність знизилась з 90% (Float32) до 7% (INT8), середній heatmap score впав з 0.700 до 0.145 (score drop +0.556). Клас Bird також значно деградував: з 79% до 33% (score drop +0.298). Клас Other_UAV показав меншу, але суттєву деградацію: з 87% до 77% (score drop +0.077). Особливо показовим моментом є те, що для класу Shahed середній INT8 heatmap score власного класу (0.145) став нижчим за середній score інших класів (0.310). Це означає системну інверсію класифікації. Модель після квантизації частіше відносить Shahed до інших класів ніж до правильного класу. Це пояснюється вузьким динамічним діапазоном активацій heatmap для класу Shahed при grid 6×6: при малому розмірі сітки кожна комірка покриває ділянку 16×16 пікселів вхідного зображення. Це призводить до розмиття специфічних ознак БПЛА Shahed у процесі INT8 квантизації активацій.

Результати діагностики моделі 4 демонструють кардинально іншу картину. Клас Shahed зберіг точність розпізнавання на рівні 87% як у форматі Float32, так і в форматі INT8, при цьому середній heatmap score навіть зріс з 0.753 до 0.814 (score drop -0.061). Клас Other_UAV також практично не втратив точності: 93% (Float32) проти 90% (INT8) (score drop -0.011). А клас Bird показав навіть незначне покращення: з 75% до 83%. Відсутність деградації класу Shahed у моделі 4 підтверджує той факт, що збільшення розміру grid з 6×6 до 8×8 достатньо для правильного представлення динамічного діапазону активацій класу Shahed у форматі INT8. Більша абсолютна роздільна здатність вхідного зображення (128×128 проти 96×96) забезпечує збереження характерних специфічних ознак БПЛА Shahed у більшій кількості комірок сітки. Це дозволяє правильно відобразити відповідні активації у 8-бітному діапазоні.

Проблема деградації квантизації є специфічною для моделі 3 з grid 6×6 та повністю усувається у моделі 4 з grid 8×8. Зниження точності розпізнавання класу Shahed з 90% до 7% у моделі 3 робить її непридатною для практичного застосування, тоді як модель 4 забезпечує стабільну точність розпізнавання 87% після INT8 квантизації. Це відповідає одній з технічних вимог (точність розпізнавання не менше 85%), яка була сформульована у підрозділі 1.4. Виявлена залежність між розміром вихідної сітки теплової карти та точністю розпізнавання специфічних класів БПЛА є науковою новизною роботи. Результати розгортання квантизованих моделей на мікроконтролері ESP32-S3 будуть описані у наступному розділі.

3.4. Оцінка отриманих результатів

Було проведено оцінку результатів моделювання системи розпізнавання БПЛА. Найвищу точність серед усіх моделей продемонструвала модель 1 з параметром alpha=1.0 та роздільною здатністю вхідного зображення 224×224 (F1=0.9073, mAP@0.5=0.5698). Ця модель слугує еталоном точності для порівняння. Модель 2 з тією ж роздільною здатністю, але з параметром alpha=0.35 досягла метрик F1=0.7974 та mAP@0.5=0.3564. Таким чином, зменшення alpha у 8 разів призводить до помітної втрати точності, особливо якщо дивитися на метрику mAP@0.5, яка враховує якість побудованих bounding boxes.

Модель 3 з роздільною здатністю вхідного зображення 96×96 та grid 6×6 показала результат F1=0.8330. Це вищий результат ніж у моделі 2 (F1=0.7974) навіть попри меншу роздільну здатність зображення. Це пояснюється адаптованими гіперпараметрами навчання (використання параметра sigma=0.5 для малого розміру grid) та більшою кількістю епох навчання (65 проти 51). Проте, як показало дослідження квантизації, ця модель є непридатною для розгортання на ESP32-S3 через катастрофічну деградацію класу Shahed після INT8 квантизації.

Модель 4 з роздільною здатністю вхідного зображення 128×128 та розміром grid 8×8 досягла результату F1=0.8811. Це найкращий результат серед моделей, які плануються до розгортання на мікроконтролері ESP32-S3. При цьому модель демонструє збалансовані значення метрик Precision (~0.92) та Recall (~0.84). Це свідчить про рівномірну якість розпізнавання об'єктів без хибних спрацювань або пропущених об'єктів. Модель 5 із значно спрощеною архітектурою досягла результату F1=0.6626 при автоматично обрахованому оптимальному значенні threshold=0.31. Ця модель має найгірші метрики з усіх розроблених моделей. Але це нормально та очікувано для такої швидкісної моделі з урахуванням значно менших затребуваних обчислювальних ресурсів для розгортання на ESP32-S3.

Порівняння моделей 1 та 2 дозволяє зробити висновок, що зменшення параметра alpha з 1.0 до 0.35 призводить до суттєвої втрати точності моделі. Зниження значення метрики F1 на 12% та зниження mAP@0.5 на 37% свідчить про те, що вужчий backbone гірше локалізує об'єкти, bounding boxes стають менш точними. Разом з тим модель з alpha=0.35 залишається функціональною: на тестових зображеннях всі три класи розпізнаються правильно, хоча з дещо нижчою впевненістю розпізнавання. Оскільки зменшення alpha у 8 разів дає значно менший розмір моделі при прийнятній точності, вибір параметра alpha=0.35 для варіантів тих моделей, які плануються до розгортання на мікроконтролері, є обґрунтованим.

Одним із ключових науковим результатом даної роботи є виявлена залежність між розміром вихідної сітки grid та стійкістю моделі до INT8 квантизації. Порівняння моделей 3 та 4 показало, що при розмірі grid 6×6 клас Shahed зазнає катастрофічної деградації точності (7% після INT8), тоді як при розмірі grid 8×8 точність класу Shahed залишається стабільною (87%). Ця залежність пояснюється тим, що при малому розмірі grid кожна комірка покриває більшу ділянку зображення і це призводить до усереднення специфічних ознак класу БПЛА Shahed та звуження динамічного діапазону відповідних активацій. Саме це і призводить до втрати

інформації при INT8 квантизації. Збільшення роздільної здатності зображення з 96×96 до 128×128 при збереженні архітектури multi-scale fusion дозволило не лише усунути проблему деградації квантизації, але й підвищити метрику F1-score з 0.8330 до 0.8811 та усунути плутанину класів між Bird та Other_UAV, що спостерігалась у моделі 3. Таким чином, модель 4 є оптимальним варіантом для розгортання точної системи розпізнавання БПЛА на мікроконтролері ESP32-S3.

Технічна вимога щодо точності розпізнавання класу Shahed після INT8 квантизації не менше 85% виконана для моделі 4 (отримано 87%). Вимога щодо метрики F1-score не менше 0.80 для точної моделі також виконана (модель 4 досягла результату F1=0.8811). Для швидкісної моделі 5 з іншою простішою архітектурою вимога метрики F1 не менше 0.60 також виконана (отримано F1=0.6626). Вимога щодо розміру TFLite квантизованої INT8 моделі не більше 3 MB відповідає отриманим результатам конвертації та квантизації. Розміри всіх трьох моделей для розгортання на ESP32-S3 не перевищують 3 MB.

Висновки до розділу 3

Отже, у цьому розділі виконано моделювання системи розпізнавання БПЛА: навчання моделей, їх конвертація та оцінювання деградації квантизації. Навчання п’яти архітектурних варіантів моделі FOMO підтвердило теоретичні прогнози: модель 1 з параметром alpha=1.0 досягла найкращого результату F1=0.9073, модель 4 з розміром grid 8×8 забезпечила оптимальний баланс між точністю та придатністю до розгортання на мікроконтролері, модель 5 із спрощеною архітектурою досягла результату F1=0.6626 при значно меншій обчислювальній складності моделі. Також була виявлена та експериментально підтверджена залежність між розміром вихідної сітки grid та стійкістю моделі до INT8 квантизації. Розгортання квантизованих моделей на мікроконтролері ESP32-S3, тестування системи та техніко-економічне обґрунтування описані у наступному розділі.

РОЗДІЛ 4. РОЗГОРТАННЯ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ БПЛА

4.1. Розгортання системи на мікроконтролері ESP32-S3

У даному підрозділі описано процес розгортання квантизованих TFLite INT8 моделей на мікроконтролері ESP32-S3, архітектуру програмного забезпечення вбудованої системи та налаштування параметрів системи розпізнавання. Розгортання було виконано для двох моделей: моделі 4 (128×128, розмір grid 8×8, висока точність розпізнавання) та моделі 5 (96×96, grid 12×12, максимальна швидкодія розпізнавання).

Цільовою платформою для розгортання було обрано модуль мікроконтролера ESP32-S3 N16R8 WROOM від компанії Espressif Systems. Вибір платформи обґрунтовано у підрозділі 1.3: двоядерний процесор Xtensa LX7 з тактовою частотою 240 МГц, 16 MB Flash та 8 MB OPI PSRAM забезпечують достатні ресурси для виконання TFLite INT8 розпізнавання. Вбудовані модулі Wi-Fi та Bluetooth дозволяють реалізувати веб-інтерфейс системи для завантаження тестових зображень та отримання результатів розпізнавання без використання зовнішніх комунікаційних та обчислювальних модулів. Орієнтовна вартість мікроконтролера ESP32-S3 становить близько 7 USD. Це повністю відповідає вимозі, щоб вартість апаратної платформи системи не перевищувала 10 USD. На рисунку 4.1 зображено модуль мікроконтролера ESP32-S3.

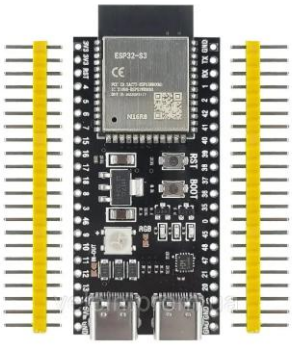

Рисунок 4.1 – Мікроконтролер ESP32-S3

Процес розгортання квантизованої TFLite INT8 моделі на ESP32-S3 здійснюється покроково. Першим кроком є конвертація навченої Keras моделі у формат TFLite INT8 за допомогою розробленого Python скрипту (цей процес було описано у підрозділі 3.3). Результатом є файл .tflite з повністю квантизованими вагами та активаціями. Другим кроком є генерація C header файлу (.h), що містить модель у вигляді масиву байтів для прошивки Flash пам'яті мікроконтролера. C header файл також містить константи конфігурації: розмір grid, кількість класів, розмір вхідного зображення та рекомендований розмір Tensor Arena. Третім кроком є розробка Arduino sketch. Це програма мікроконтролера, що реалізує ініціалізацію TFLite Micro інтерпретатора, Wi-Fi з'єднання, веб-сервер та модуль post-processing. Четвертим кроком є компіляція sketch в Arduino IDE з підключеною бібліотекою esp-tflite-micro та прошивка мікроконтролера через USB порт.

Програмне забезпечення системи було реалізовано у вигляді Arduino sketch та містить три основних компоненти: модуль TFLite Micro inference, веб-сервер та модуль post-processing. Модуль TFLite Micro inference відповідає за ініціалізацію моделі та виконання процесу розпізнавання (inference). Tensor Arena – це статично виділений буфер у пам'яті PSRAM для зберігання проміжних тензорів. Він розміщується у зовнішній пам'яті PSRAM за допомогою функції heap_caps_malloc з прапорцем MALLOC_CAP_SPIRAM. Набір операторів (ops resolver) реєструє лише необхідні оператори CNN: Conv2D, DepthwiseConv2D, Add, Mul, Logistic та інші. Це мінімізує розмір коду прошивки. Після виклику методу interpreter->Invoke() результати розпізнавання зчитуються з вихідних тензорів моделі.

Веб-сервер було реалізовано на основі бібліотеки WebServer, що входить до складу ESP32 Arduino SDK. Сервер обробляє два типи запитів: GET-запити на кореневий маршрут повертає HTML сторінку веб-інтерфейсу з JavaScript кодом для завантаження зображення та відображення результатів розпізнавання об'єктів на зображені; POST-запит за маршрутом /infer приймає бінарні дані зображення у

форматі RGB888, виконує процес розпізнавання (inference) та повертає результати розпізнавання у форматі JSON.

Модуль post-processing обробляє вихідні тензори моделі після процесу розпізнавання (inference). Для моделі 4 (три виходи) виконується зчитування heatmap, offset та size тензорів з подальшим обчисленням координат bounding box для кожного розпізнавання. Для моделі 5 (з одним виходом) обробляється лише тензор heatmap, координати центроїда визначаються як геометричний центр комірки grid. В обох випадках застосовується Non-Maximum Suppression (NMS) з радіусом 1 комірка для усунення дублюючих розпізнавань.

Веб-інтерфейс було реалізовано у вигляді однієї HTML сторінки. Вона повністю вбудована у Flash пам’ять мікроконтролера окремим рядком у коді sketch. Інтерфейс виконує такі функції: завантаження зображення з файлової системи користувача; масштабування зображення до розміру вхідного тензору моделі (128×128 або 96×96) засобами Canvas API браузера; конвертація зображення з формату RGBA у RGB888 та передача бінарних даних на ESP32-S3 через HTTP POST запит; відображення результатів розпізнавання (inference) у вигляді bounding boxes з підписами класів та значеннями впевненості на накладеному Canvas елементі; виведення часу розпізнавання (inference) на мікроконтролері та кількості виявлених та розпізнаних об’єктів. На рисунку 4.2 зображено веб-інтерфейс системи.

Рисунок 4.2 – Веб-інтерфейс системи на мікроконтролері

Після успішної ініціалізації системи мікроконтролер ESP32-S3 підключається до мережі Wi-Fi та запускає веб-сервер. IP-адреса веб-сервера виводиться у Serial Monitor Arduino IDE. Користувач відкриває веб-інтерфейс у браузері за отриманою IP-адресою, завантажує тестове зображення та отримує результати розпізнавання у реальному часі. Результати тестування розгорнутої системи наведено у підрозділі 4.2.

4.2. Тестування системи розпізнавання БПЛА

У даному підрозділі наведено результати практичного тестування розгорнутої системи розпізнавання БПЛА на мікроконтролері ESP32-S3. Тестування виконувалось для двох розгорнутих моделей (модель 4 та 5) з метою оцінки швидкодії, якості розпізнавання та підтвердження працездатності системи в умовах апаратних обмежень.

Тестування проводилось через веб-інтерфейс системи у веб-браузері. На мікроконтролер ESP32-S3 завантажується зображення БПЛА для тестування. Для кожного зображення веб-браузер виконує масштабування зображення до розміру вхідного тензору (128×128 для моделі 4 та 96×96 для моделі 5), конвертацію у

формат RGB888 та передачу на мікроконтролер ESP32-S3 через HTTP POST запит. На мікроконтролері виконується процес розпізнавання та повертається результат у форматі JSON з часом розпізнавання (inference), координатами та впевненістю розпізнавання. Час розпізнавання (inference) вимірюється за допомогою функції millis() безпосередньо на мікроконтролері.

Під час тестування модель 4 показує час розпізнавання в діапазоні 1600 мс на мікроконтролері ESP32-S3. Це значення відповідає частоті обробки кадрів ~0.6-0.65 FPS. Попри невисокий FPS, така швидкодія моделі є прийнятною для задачі дуже точного виявлення та розпізнавання БПЛА. Якщо додати мікроконтролер ESP32-S3 на БПЛА-перехоплювач, то при швидкості цільового для розпізнавання БПЛА ~185 км/год та дальності спостереження від 500 метрів ціль перебуває у полі зору камери мікроконтролера протягом кількох секунд. Це достатньо для отримання кількох послідовних кадрів та підтвердження розпізнавання. На рисунках 4.3, 4.4, 4.5 зображено результати розпізнавання БПЛА за допомогою моделі 4 на мікроконтролері ESP32-S3.

Рисунок 4.3 – Розпізнавання декількох БПЛА Shahed за допомогою моделі 4 на ESP32-S3

Рисунок 4.4 – Розпізнавання БПЛА Other_UAV за допомогою моделі 4 на ESP32-S3

Рисунок 4.5 – Розпізнавання невеликого зображення БПЛА Other_UAV за допомогою моделі 4 на ESP32-S3

Результати тестування моделі 4 підтверджують правильну роботу системи після INT8 квантизації моделі. Клас Shahed розпізнається з високою впевненістю, що відповідає результатам діагностики квантизації у попередньому розділі. Bounding boxes точно локалізують об’єкти завдяки offset та size головам розробленої архітектури. Координати центроїда та розміри рамки передаються у JSON форматі відповіді та відображаються у веб-інтерфейсі. Система правильно обробляє зображення одночасно з кількома об’єктами різних класів. Це підтверджує працездатність алгоритму NMS на мікроконтролері ESP32-S3.

Модель 5 демонструє значно швидший час розпізнавання 113-114 мс на тому ж мікроконтролері ESP32-S3. Цей показник відповідає частоті обробки кадрів ~8.8 FPS. Це у декілька разів швидше ніж було у моделі 4 на тій самій апаратній платформі. Така швидкодія розпізнавання досягається завдяки свідомому спрощенню архітектури та двом архітектурним рішенням: відсіканню backbone на block_6 (замість проходження через всі 13 блоків) та відмові від offset і size голів на користь єдиного виходу heatmap.

На рисунках 4.6, 4.7, 4.8 зображено результати розпізнавання БПЛА за допомогою моделі 5 на мікроконтролері ESP32-S3.

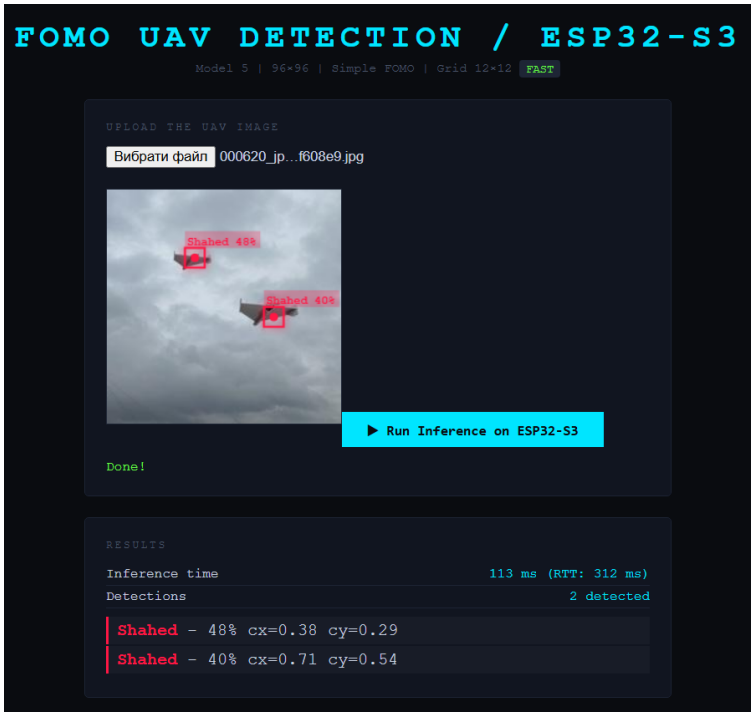

Рисунок 4.6 – Розпізнавання декількох БПЛА Shahed за допомогою моделі 5 на ESP32-S3

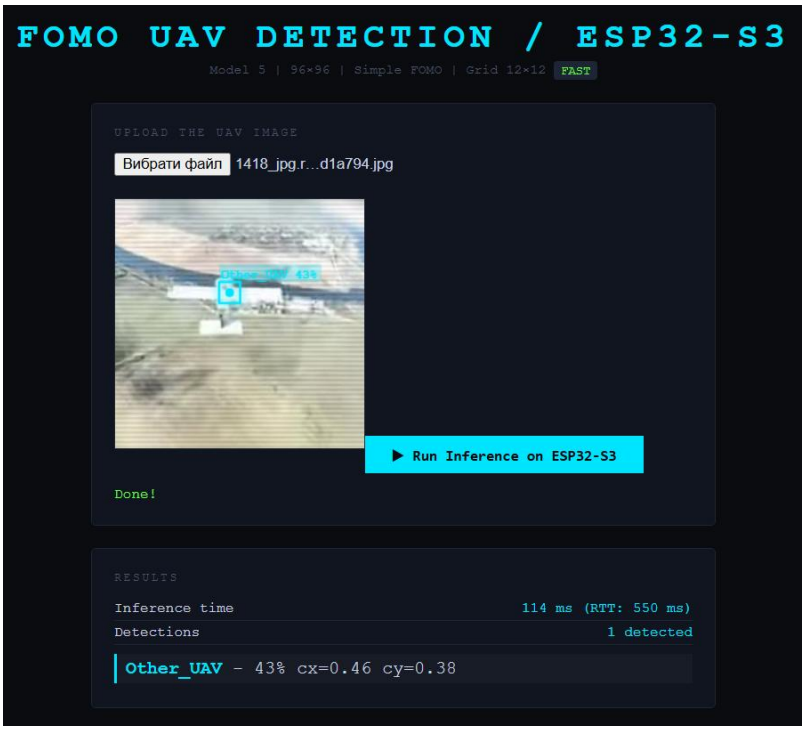

Рисунок 4.7 – Розпізнавання БПЛА Other_UAV за допомогою моделі 5 на ESP32-S3

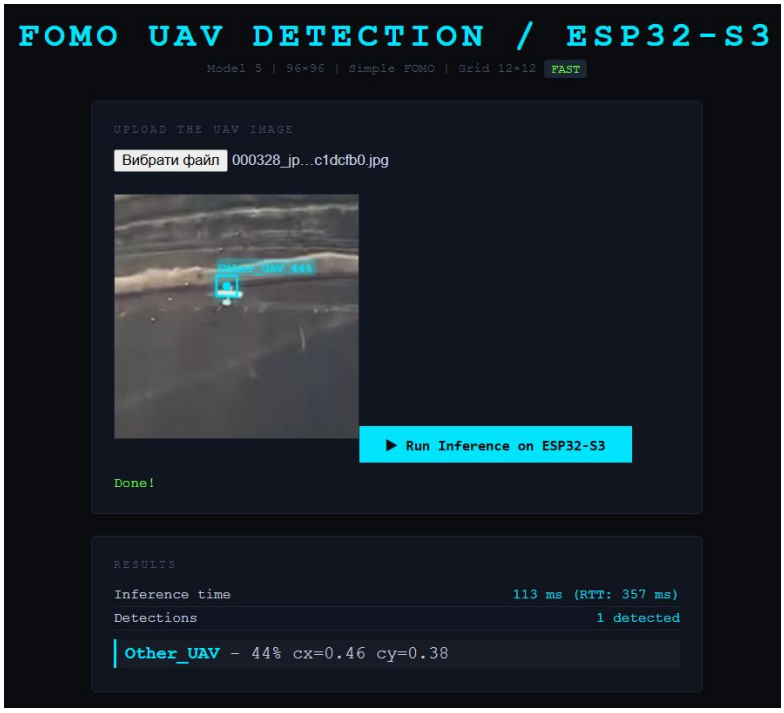

Рисунок 4.8 – Розпізнавання невеликого зображення БПЛА Other_UAV за допомогою моделі 5 на ESP32-S3

Особливість моделі 5: bounding box має фіксований розмір рівний одній комірці grid (8×8 пікселів вхідного зображення у нормалізованих координатах), оскільки архітектура не передбачає offset та size виходів. Це означає, що рамка розпізнавання позначає приблизне положення об’єкта, але не відображає його реальних розмірів. Для задачі первинного виявлення наявності та розпізнавання БПЛА на зображенні таке обмеження є прийнятним.

Порівняння двох розгорнутих моделей демонструє класичний компроміс між точністю та швидкістю розпізнавання на системах Edge AI. Модель 4 забезпечує значно вищу точність розпізнавання (F1=0.8811 проти F1=0.6626) та точнішу локалізацію об’єктів завдяки використанню offset та size голів, але потребує 1600 мс для здійснення процесу одного розпізнавання. Модель 5 забезпечує час розпізнавання 113-114 мс (~8.8 FPS). Це достатньо для виявлення та розпізнавання

рухомих цілей, які рухаються повільно. Ця модель 5 поступається моделі 4 за точністю класифікації та якістю локалізації об’єктів.

Отримані результати тестування обґрунтовують застосування дворівневої архітектури системи розпізнавання БПЛА. Модель 5 виконує функцію швидкого розпізнавання першого рівня та сигналізує про наявність об’єкта в кадрі з частотою ~8.8 FPS, тоді як модель 4 виконує точну класифікацію та локалізацію виявленого та розпізнаного об’єкта. Така архітектура дозволяє оптимально використовувати дуже обмежені обчислювальні ресурси мікроконтролера ESP32-S3.

Практичне тестування підтвердило, що розроблена система відповідає всім технічним вимогам, які були сформульовані у підрозділі 1.4: точність розпізнавання класу Shahed після INT8 квантизації становить 87% (вимога ≥85%), метрика F1-score моделі 4 становить 0.8811 (вимога ≥0.80), вартість апаратної платформи становить ~7 USD (вимога ≤10 USD).

4.3. Техніко-економічне обґрунтування системи розпізнавання БПЛА

Було зроблено оцінку вартості апаратної платформи, порівняння з наявними аналогами, оцінку трудовитрат на розробку та аналіз економічної ефективності впровадження розробленої системи.

Апаратна платформа розробленої системи складається з мінімального набору компонентів. Основним компонентом є модуль мікроконтролера ESP32-S3 вартістю близько 7 USD. Для живлення системи використовується стандартний USB-кабель вартістю близько 1 USD. Таким чином, загальна вартість апаратного прототипу становить близько 8 USD. Це демонструє високу економічну доступність розробленого рішення.

Розроблена система суттєво відрізняється від наявних рішень для виявлення та розпізнавання БПЛА за критерієм вартості та автономності. Комерційні системи виявлення та розпізнавання БПЛА на базі GPU потребують апаратної платформи

вартістю від кількох сотень до кількох тисяч доларів США. Вартість апаратної платформи розробленої системи є набагато нижчою порівняно з іншими варіантами.

Ключовою перевагою розробленої системи є повна автономність. Вона не потребує підключення до зовнішньої обчислювальної інфраструктури. Це є критично важливою характеристикою для застосування в реальних бойових умовах. Відсутність мережевої затримки через те, що операція розпізнавання виконується безпосередньо на самому пристрої, також є важливою перевагою розробленої системи.

Споживання електроенергії системи становить близько 200 мА при напрузі 5V (~1 Вт). Це дозволяє живити системи від стандартного павербанку ємністю 10 000 мАг протягом 25 годин безперервної роботи. Досить компактні розміри модуля ESP32-S3 дозволяють інтегрувати систему на фізичний БПЛА-перехоплювач.

Розробка системи складалася з кількох етапів. Це збір зображень та формування набору даних, проєктування та навчання п’яти різних архітектурних варіантів моделей, конвертація, квантизація та діагностика квантизації, розробка програмного забезпечення для ESP32-S3, тестування розробленої системи. Загальний обсяг роботи складає приблизно 3-4 місяці праці інженера. При середній зарплаті інженера з машинного навчання в Україні ~1500 USD/місяць, загальні витрати на розробку складають орієнтовно 4500-6000 USD. Повторне розгортання розробленої системи на новому пристрої потребуватиме лише прошивки мікроконтролера. Цей процес займає декілька хвилин без додаткових витрат на розробку нового програмного забезпечення.

Економічна ефективність системи визначається насамперед її здатністю забезпечити своєчасне виявлення та розпізнавання БПЛА типу Shahed. Вартість одного такого БПЛА за різними оцінками становить від 20 000 до 50 000 USD. Вартість зенітної керованої ракети для його знищення у кілька разів перевищує вартість самого БПЛА. Це зумовлює використання альтернативних засобів протидії

БПЛА. Порівняно з існуючими комерційними системами виявлення та розпізнавання БПЛА вартістю від кількох тисяч до десятків тисяч доларів, розроблене рішення забезпечує суттєву економію коштів при прийнятній точності розпізнавання БПЛА.

Розроблена система має ряд обмежень, що визначають специфічні умови її практичного застосування. Система призначена для роботи у денних умовах з достатнім освітленням. Нічне застосування системи потребує використання інфрачервоної камери. Ефективна дальність виявлення та розпізнавання БПЛА залежить від розміру БПЛА в кадрі. Система є першим рівнем виявлення та розпізнавання, оскільки час розпізнавання БПЛА для розробленої точної моделі є досить великим. Але ця проблема вирішується використанням більш потужних пристроїв для розгортання моделі. Більші обчислювальні ресурси дадуть можливість використання повноцінної, еталонної, точної моделі із затребуваним меншим часом розпізнавання, кращою роздільною здатністю зображень та більшою кількість FPS для відеопотоку з камери, яка може бути розміщена на борту БПЛА-перехоплювача.

4.4. Напрям подальших досліджень

За результатами виконаної роботи було визначено кілька перспективних напрямів подальших досліджень, що можуть суттєво розширити функціональність та точність системи розпізнавання БПЛА.

Найперспективнішим напрямом є розгортання розробленої моделі 1 (зображення 224×224, параметр alpha=1.0, метрика F1=0.9073) на платформах з достатніми обчислювальними ресурсами. Перспективними платформами для розгортання є Raspberry Pi 5 з опційним додатковим модулем NPU Hailo-8L (13 TOPS). Ця конфігурація здатна виконувати розпізнавання (inference) БПЛА за допомогою моделі 1 у режимі реального часу при вищій роздільній здатності

зображення та відеопотоку. Raspberry Pi Zero 2W на базі процесора quad-core ARM Cortex-A53 з 512 MB RAM дозволяє запустити повну версію TensorFlow Lite без застосування INT8 квантизації. Це усуває проблему деградації класів та забезпечує максимальну точність розпізнавання при невеликій вартості (~30 USD). Розгортання на цих платформах потребує лише розробки окремої програми на Python. Сама модель вже навчена та готова.

Поточний набір даних better_drones_dataset, який використовується у даній роботі, містить достатню кількість зображень. Але суттєвим покращенням якості моделі стало б розширення цього набору даних ще більшою кількістю зображень. Збільшення кількості зображень у навчальній вибірці очікувано підвищить здатність моделі до узагальнення.

Особливо перспективним напрямком є використання розробленої системи безпосередньо на борту БПЛА-перехоплювача. Компактні розміри та невелике споживання електроенергії модуля мікроконтролера ESP32-S3 (~1 Вт) дозволяють інтегрувати його у корпус БПЛА-перехоплювача без суттєвого збільшення маси самого БПЛА. У такому сценарії використання БПЛА-перехоплювач за допомогою бортової камери та розробленої моделі розпізнавання БПЛА виявляє, розпізнає та класифікує ціль безпосередньо в повітрі.

Більшість атак БПЛА Shahed відбуваються вночі. Це робить денну систему розпізнавання на основі RGB камери неефективною у нічний час. Перспективним напрямком є дослідження можливості адаптації розроблених моделей до роботи з монохромними зображеннями від інфрачервоних камер. ESP32-S3 підтримує підключення камер, деякі з яких мають режим нічного бачення. Для адаптації моделі достатньо додати до набору даних чорно-білі зображення та зображення, які отримані з інфрачервоних камер. Розглянута архітектура FOMO не залежить від кількості каналів зображення після відповідної зміни першого шару.

Висновки до розділу 4

Отже, у четвертому розділі було описано практичне розгортання розробленої системи розпізнавання БПЛА на мікроконтролері ESP32-S3 та було проведено її практичне тестування. Розгортання виконано для двох моделей . Модель 4 (128×128, grid 8×8, F1=0.8811, час розпізнавання (inference) ~1600 мс) та модель 5 (96×96, grid 12×12, F1=0.6626, час розпізнавання (inference) 113 мс) реалізують дворівневу архітектуру системи виявлення та розпізнавання БПЛА.

Практичне тестування підтвердило повну відповідність системи технічним вимогам: точність розпізнавання Shahed після INT8 квантизації становить 87% (вимога ≥85%), метрика F1-score моделі 4 становить 0.8811 (вимога ≥0.80), вартість апаратної платформи становить ~7 USD (вимога ≤10 USD). Модель 5 забезпечує час розпізнавання 113 мс (~8.8 FPS). Це достатньо для виявлення та розпізнавання рухомих цілей типу БПЛА Shahed.

За допомогою техніко-економічного аналізу було проаналізовано, що вартість апаратної платформи (~7 USD) є набагато нижчою порівняно з іншими варіантами. Визначено перспективні напрями подальших досліджень: розгортання еталонної моделі на Raspberry Pi 5 з NPU Hailo-8L, розширення набору даних більшою кількістю зображень, застосування на борту БПЛА-перехоплювача для автономного наведення на ціль, навчання та тестування системи в нічному режимі з використанням інфрачервоної камери.

ВИСНОВКИ

Поставлені завдання виконані, мета роботи досягнута.

У даній роботі було вирішено актуальну науково-практичну задачу розробки та дослідження методів комп'ютерного зору для розпізнавання безпілотних літальних апаратів з розгортанням на мікроконтролері ESP32-S3. За результатами виконаної роботи зроблено відповідні висновки.

Аналіз сучасних загроз від БПЛА показав, що ударні БПЛА типу Shahed становлять особливу велику загрозу для критичної інфраструктури України. З початку повномасштабного вторгнення було запущено близько 79 000 БПЛА цього типу. Традиційні методи виявлення (радарні, акустичні тощо) мають суттєві обмеження щодо виявлення та розпізнавання малопомітних цілей. Встановлено, що методи комп'ютерного зору на основі методів глибокого навчання є дуже перспективним доповненням до наявних систем виявлення та розпізнавання завдяки їх масштабованості та відносно низькій вартості апаратної платформи розгортання.

Було обґрунтовано вибір архітектури FOMO на основі MobileNetV2 як єдиного практично придатного рішення для розгортання на мікроконтролері ESP32-S3 з огляду на мінімальні вимоги до кількості оперативної пам'яті та підтримки фреймворку TensorFlow Lite Micro. Було розроблено та досліджено п'ять архітектурних варіантів моделей, що відрізняються роздільною здатністю вхідного зображення, значенням параметра alpha та конфігурацією виходів.

Було виявлено та експериментально підтверджено залежність між розміром вихідної сітки grid та стійкістю моделі до INT8 квантизації. Це один із ключових наукових результатів роботи. Модель 3 з розміром grid 6×6 (вхід 96×96) показала катастрофічну деградацію класу Shahed після застосування INT8 квантизації: точність знизилась з 90% до 7%. Це робить її непридатною для практичного застосування. Середній heatmap score класу Shahed після квантизації (0.145)

виявився нижчим за score інших класів (0.310). Це свідчить про системну інверсію класифікації.

Було встановлено, що збільшення розміру grid до 8×8 за рахунок збільшення роздільної здатності вхідного зображення до 128×128 пікселів повністю усуває проблему деградації квантизації. Модель 4 (grid 8×8) забезпечує стабільну точність розпізнавання БПЛА Shahed на рівні 87% після застосування INT8 квантизації, що відповідає встановленій технічній вимозі ≥85%. Таким чином, grid 8×8 є мінімально достатнім розміром для правильної INT8 квантизації класу Shahed.

Також було розроблено спрощену архітектуру FOMO з відсіканням backbone на block_6 та одним heatmap виходом (модель 5). Застосування гібридного набору даних дозволило досягти метрики F1=0.6626 при часу розпізнавання 113 мс (~8.8 FPS) на мікроконтролері ESP32-S3. Це у 14 разів швидше ніж було у моделі 4 (~1600 мс) при розгортанні на тій самій апаратній платформі.

Було розроблено та розгорнуто дворівневу систему розпізнавання БПЛА на мікроконтролері ESP32-S3 з веб-інтерфейсом для завантаження зображень та відображення результатів розпізнавання. Модель 4 виконує функцію точного розпізнавання, а модель 5 виконує функцію швидкого, хоч і не завжди точного розпізнавання. Практичне тестування підтвердило правильну роботу обох моделей на мікроконтролері та відповідність всім технічним вимогам.

За допомогою техніко-економічний аналізу було показано, що вартість апаратної платформи розробленої системи є значно нижчою порівняно з іншими варіантами. Це робить цю систему доступним рішенням для масового розгортання.

Визначено перспективні напрями подальших досліджень: розгортання еталонної моделі (F1=0.9073) на Raspberry Pi 5 з NPU Hailo-8L для розпізнавання БПЛА у реальному часі; розширення набору даних більшою кількістю зображень; застосування системи на борту БПЛА-перехоплювача для автономного наведення на ціль; адаптація моделей до нічного режиму роботи з інфрачервоними камерами.

Результати цієї роботи можуть бути використані при розробці автономних систем виявлення та розпізнавання БПЛА для потреб протиповітряної оборони, а також у подальших дослідженнях у галузі розгортання моделей комп’ютерного зору на малопотужних крайових пристроях.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. O. Zaritskyi, A. Miroshnyk. A combined method for recognizing and intercepting unmanned aerial vehicles using machine learning methods. Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: XII Міжнародна науково-практична конференція. м. Запоріжжя, Україна, 2024. С. 127-131.
2. Institute for Science and International Security. Monthly Analysis of Russian Shahed 136 Deployment Against Ukraine. 2025. URL: https://isis-online.org/isis-reports/monthly-analysis-of-russian-shahed-136-deployment-against-ukraine/ (дата звернення: 05.03.2026).
3. Massive Missile Attacks on Ukraine. 2022–2023 russian strikes against Ukrainian: dataset. URL: https://www.kaggle.com/datasets/piterfm/massive-missile-attacks-on-ukraine/data/ (дата звернення: 12.03.2026).
4. Shahed-136. Army Recognition: веб-сайт. URL: https://www.armyrecognition.com/military-products/army/unmanned-systems/unmanned-aerial-vehicles/shahed-136-loitering-munition-kamikaze-suicide-drone-technical-data/ (дата звернення: 20.04.2026).
5. Shahed-131 & -136 UAVs: a visual guide. Open Source Munitions Portal: веб-сайт. URL: https://osmp.ngo/collection/shahed-131-136-uavs-a-visual-guide/ (дата звернення: 16.03.2026).
6. Rahman M. H., Sejan M. A., Aziz M. A., Tabassum R., Baik J., Song H. A Comprehensive Survey of UAV Detection and Classification Using Machine Learning Approach: Challenges, Solutions, and Future Directions. Remote Sensing. 2024. Vol. 16, № 5. P. 879.
7. Wu X., Li W., Hong D., Tao R., Du Q. Deep Learning for UAV-based Object Detection and Tracking: A survey. IEEE Geoscience and Remote Sensing Magazine. 2021. Vol. 10, № 1. P. 91–124.

8. Tang G., Ni J., Zhao Y., Gu Y., Cao W. A Survey of Object Detection for UAVs Based on Deep Learning. Remote Sensing. 2024. Vol. 16, № 1. P. 149.

9. Na Z., Cheng L., Sun H., Lin B. Survey on UAV detection and identification based on deep learning. Journal of Signal Processing. 2024. Vol. 40, № 4. P. 609–624.

10. YOLOv8 Documentation. Ultralytics: веб-сайт. URL: https://docs.ultralytics.com/ (дата звернення: 23.03.2026).

11. Tan M., Pang R., Le Q. V. EfficientDet: Scalable and Efficient Object Detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, 2020. P. 10781–10790.

12. Alqahtani D. K., Cheema M. A., Rodriguez M. A., Toosi A. N. A Comprehensive Evaluation of Deep Learning Object Detection Models on Heterogeneous Edge Devices. arXiv:2409.16808. 2024. URL: https://arxiv.org/abs/2409.16808/ (дата звернення: 26.03.2026).

13. Announcing FOMO (Faster Objects, More Objects). Edge Impulse: веб-сайт. 2022. URL: https://www.edgeimpulse.com/blog/announcing-fomo-faster-objects-more-objects/ (дата звернення: 30.03.2026).

14. FOMO: Object detection for constrained devices. Edge Impulse Documentation: веб-сайт. URL: https://docs.edgeimpulse.com/studio/projects/learning-blocks/blocks/object-detection/fomo/ (дата звернення: 31.03.2026).

15. TensorFlow Lite Micro. Silicon Labs: веб-сайт. URL: https://docs.silabs.com/machine-learning/2.0.0/machine-learning-tensorflow-lite-for-microcontrollers/ (дата звернення: 02.04.2026).

16. ESP32-S3 TinyML Optimization: TFLM, INT8 & Memory Tuning. Zediot: веб-сайт. 2026. URL: https://zediot.com/blog/esp32-s3-tinyml-optimization/ (дата звернення: 06.04.2026).

17. A deep dive into Raspberry Pi Zero 2 W's power consumption. CNX Software: веб-сайт. 2021. URL: https://www.cnx-software.com/2021/12/09/raspberry-pi-zero-2-w-power-consumption/ (дата звернення: 08.04.2026).

18. Howard A. G., Zhu M., Chen B., Wang W., Weyand T., Andreetto M., Adam H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861. 2017. URL: https://arxiv.org/abs/1704.04861/ (дата звернення: 10.04.2026).

19. Sandler M., Howard A., Zhu M., Chen L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City, 2018. P. 4510–4520.

20. Lin T.-Y., Dollár P., Girshick R., He K., Hariharan B., Belongie S. Feature Pyramid Networks for Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, 2017. P. 2117–2125.

21. Tompson J., Goroshin R., Jain A., LeCun Y., Bregler C. Efficient Object Localization Using Convolutional Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, 2015. P. 648–656.

22. Zhou X., Wang D., Krähenbühl P. Objects as Points. arXiv:1904.07850. 2019. URL: https://arxiv.org/abs/1904.07850/ (дата звернення: 15.04.2026).

23. Huang Z., Li W., Xia X., Tao R. A General Gaussian Heatmap Label Assignment for Arbitrary-Oriented Object Detection. IEEE Transactions on Image Processing. 2021. Vol. 31. P. 1895–1910.

24. Lin T.-Y., Goyal P., Girshick R., He K., Dollár P. Focal Loss for Dense Object Detection. Proceedings of the IEEE International Conference on Computer Vision (ICCV). Venice, 2017. P. 2980–2988.

25. Schmidt R. M., Schneider F., Hennig P. Descending through a Crowded Valley – Benchmarking Deep Learning Optimizers. Proceedings of the 38th International Conference on Machine Learning (ICML). PMLR 139. 2021. P. 9367–9376.

26. Loshchilov I., Hutter F. SGDR: Stochastic Gradient Descent with Warm Restarts. International Conference on Learning Representations (ICLR). 2017. URL: https://arxiv.org/abs/1608.03983/ (дата звернення: 17.04.2026).

27. Padilla R., Passos W. L., Dias T., Netto S. L., Silva A. B. E. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. Electronics. 2021. Vol. 10, № 3. P. 279. DOI: 10.3390/electronics10030279.

28. TensorFlow documentation. TensorFlow: веб-сайт. URL: https://www.tensorflow.org/api_docs/ (дата звернення: 20.04.2026).

29. esp-tflite-micro. Espressif: GitHub-репозиторій. URL: https://github.com/espressif/esp-tflite-micro/ (дата звернення: 22.04.2026).

ДОДАТКИ

ДОДАТОК А

```
def build_full_model(img_size=IMG_SIZE, num_classes=NUM_ACTIVE,
alpha=ALPHA, wd=WEIGHT_DECAY):
    inputs = tf.keras.Input(shape=(img_size, img_size, 3))
    backbone = tf.keras.applications.MobileNetV2(
        input_shape=(img_size, img_size, 3), include_top=False, alpha=alpha,
weights='imagenet'
    )
    for layer in backbone.layers[:-60]:
        layer.trainable = False
    for layer in backbone.layers[-60:]:
        layer.trainable = not isinstance(layer, layers.BatchNormalization)

    f3_layer  = backbone.get_layer('block_3_expand_relu').output
    f6_layer  = backbone.get_layer('block_6_expand_relu').output
    f10_layer = backbone.get_layer('block_10_expand_relu').output
    f13_layer = backbone.get_layer('block_13_expand_relu').output

    feature_extractor = Model(inputs=backbone.input,
                              outputs=[f3_layer, f6_layer, f10_layer, f13_layer])
    feat3, feat6, feat10, feat13 = feature_extractor(inputs)

    # Align all to 28×28, 64 channels
    f3_align  = layers.Conv2D(64, 3, strides=2, padding='same', activation='relu',
                              kernel_regularizer=l2(wd), name='f3_align')(feat3)
```

```
    f6_align  = layers.Conv2D(64, 1, padding='same', activation='relu',
                              kernel_regularizer=l2(wd), name='f6_align')(feat6)
    f10_align = layers.UpSampling2D(size=(2, 2), interpolation='bilinear',
name='f10_up')(feat10)
    f10_align = layers.Conv2D(64, 1, padding='same', activation='relu',
                              kernel_regularizer=l2(wd), name='f10_align')(f10_align)
    f13_align = layers.UpSampling2D(size=(2, 2), interpolation='bilinear',
name='f13_up')(feat13)
    f13_align = layers.Conv2D(64, 1, padding='same', activation='relu',
                              kernel_regularizer=l2(wd), name='f13_align')(f13_align)

    merged = layers.Concatenate(name='multiscale_concat')([f3_align, f6_align,
f10_align, f13_align])
    x = layers.Conv2D(192, 1, padding='same', use_bias=False,
                      kernel_regularizer=l2(wd), name='fusion_conv')(merged)
    x = layers.BatchNormalization(name='fusion_bn')(x)
    x = layers.ReLU(name='fusion_relu')(x)

    # Neck
    x = layers.Conv2D(256, 3, strides=2, padding='same', use_bias=False,
                      kernel_regularizer=l2(wd), name='neck_conv1')(x)
    x = layers.ReLU(name='neck_relu1')(x)
    x = layers.BatchNormalization(name='neck_bn1')(x)
    x = layers.Conv2D(128, 3, padding='same', use_bias=False,
                      kernel_regularizer=l2(wd), name='neck_conv2')(x)
    x = layers.ReLU(name='neck_relu2')(x)
    x = layers.BatchNormalization(name='neck_bn2')(x)
```

```
x = layers.SpatialDropout2D(0.3, name='neck_dropout')(x)

# Heatmap Head
hm = layers.Conv2D(128, 3, padding='same', activation='relu',
                   kernel_regularizer=l2(wd), name='hm_conv1')(x)
hm = layers.Conv2D(64,  3, padding='same', activation='relu',
                   kernel_regularizer=l2(wd), name='hm_conv2')(hm)
hm = layers.Conv2D(num_classes, 1, padding='same',
                   bias_initializer=Constant(-4.0), name='hm_conv3')(hm)
hm = layers.Activation('sigmoid', name='heatmap')(hm)

# Offset Head
off = layers.Conv2D(128, 3, padding='same', activation='relu',
                    kernel_regularizer=l2(wd), name='off_conv1')(x)
off = layers.Conv2D(64,  3, padding='same', activation='relu',
                    kernel_regularizer=l2(wd), name='off_conv2')(off)
off = layers.Conv2D(2,   1, padding='same', name='offset')(off)

# Size Head
sz = layers.Conv2D(128, 3, padding='same', activation='relu',
                   kernel_regularizer=l2(wd), name='sz_conv1')(x)
sz = layers.Conv2D(64,  3, padding='same', activation='relu',
                   kernel_regularizer=l2(wd), name='sz_conv2')(sz)
sz = layers.Conv2D(2,   1, padding='same', name='size')(sz)

return Model(inputs=inputs, outputs=[hm, off, sz])
```

ДОДАТОК Б

```
def build_model(img_size=IMG_SIZE, num_classes=NUM_ACTIVE,
                alpha=ALPHA, wd=WEIGHT_DECAY):
    inputs = tf.keras.Input(shape=(img_size, img_size, 3))
    backbone = tf.keras.applications.MobileNetV2(
        input_shape=(img_size, img_size, 3),
        include_top=False, alpha=alpha, weights='imagenet')
    freeze_until = int(len(backbone.layers) * 0.6)
    for layer in backbone.layers[:freeze_until]:
        layer.trainable = False
    for layer in backbone.layers[freeze_until:]:
        layer.trainable = not isinstance(layer, layers.BatchNormalization)
    cut = backbone.get_layer('block_6_expand_relu').output
    features = Model(inputs=backbone.input, outputs=cut)(inputs)

    x = layers.Conv2D(64, 1, padding='same', use_bias=False,
                      kernel_regularizer=l2(wd))(features)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)
    x = layers.SpatialDropout2D(0.1)(x)
    hm = layers.Conv2D(64, 3, padding='same', activation='relu',
                       kernel_regularizer=l2(wd))(x)
    hm = layers.Conv2D(num_classes, 1, padding='same',
                       bias_initializer=Constant(-4.0))(hm)
    hm = layers.Activation('sigmoid', name='heatmap')(hm)

    return Model(inputs=inputs, outputs=hm)
```